No Code ML

Vedant Barbhaya1, Viraj Patel2, Vishal Kundar3, Vidya U4

*Department of Information Science & Engineering,*

*CMR Institute of Technology, Bengaluru.*

1veda17is@cmrit.ac.in,2vipa17is@cmrit.ac.in,

3vish17is@cmrit.ac.in,4vidya.u@cmrit.ac.in

***Abstract*— Data is the oil of the 21st century. Essentially if you are running a service/business, every interaction with a customer generates some kind of valuable data. With the advancements of intelligent systems, it has become possible to extract very valuable information from any given data. But not every business has the ability or means to develop an in-house team to analyse vast amounts of data generated by them.**

**No Code ML is trying to create a platform where a user can upload a data set and get personalized predictions with just one click. This eliminates the need to clean or process the data or create a machine learning model. No code ML helps businesses to analyse their data using ML without actually needing the knowledge of how to code it. Everything will be handled by our platform and the user will get a personalized dashboard to analyse the outcomes.**

***Keywords***— **Machine Learning (ML),**

1. Introduction

With the success of Machine Learning (ML) in recent years, it has started attracting a lot of attention from the research and business communities.

Machine Learning involves the design and development of pipelines for applications and ML systems.

Building such a pipeline requires a team of human experts: data scientists having statistical and ML knowledge; domain experts with years of experience within a specific domain. Together, these human experts can build a sensible ML pipeline containing data preprocessing, meaningful feature engineering, and fine-tuned models leading to great results.

Every machine learning service, at its core, needs to solve the same fundamental problems: deciding which machine learning algorithm to use on a given dataset, whether and how to preprocess its features, how to generate meaningful features, and how to tune all hyperparameters.

This process is a complex task, performed in an iterative manner with trial and error. Building a good ML pipeline is a laborious process and practitioners often use a suboptimal default ML pipeline.

To solve these issues, a novel idea of automating the entire pipeline of machine learning (ML) has emerged, i.e., automated machine learning (AutoML).

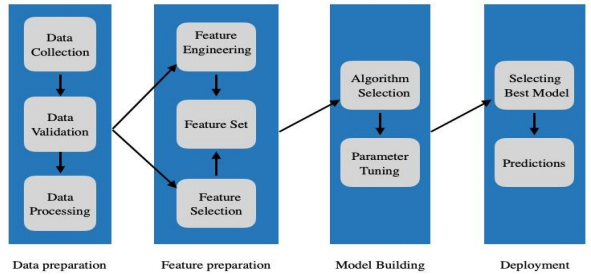
The goal of AutoML is to allow practitioners to build ML applications without much requirement for statistical and ML knowledge.

With the advancements in Cloud and BigData technologies, AutoML has started gaining a lot of attention. A complete AutoML system can dynamically combine various techniques to form an easy-to-use end-to-end ML pipeline system.

In the context of supervised learning, AutoML aims at removing user interaction altogether from all stages of the design and development of supervised learning systems.

As data is being generated at an incredible rate in practically any context and scenario, the number of practitioners available to analyze such data are limited.

AutoML pipeline consists of several processes: data preparation, feature engineering, model generation, and model evaluation. Given below is an illustration of a typical ML pipeline.



1. Related Work

According to the no free lunch theorem (Wolpert and Macready, 1997), it is impossible that a single optimization algorithm is universally superior to any other optimization algorithm. This implies that no universally superior ML pipeline for all ML tasks exists. Consequently, it is not possible to transfer a well-performing ML pipeline to a new domain and still yield outstanding results. Instead, a new ML pipeline has to be constructed for each new data set to obtain optimal results. However, manually building a specialized ML pipeline for each and every problem is very time consuming and therefore expensive. As a consequence, practitioners often use a suboptimal default ML pipeline.

AutoML aims to improve the current way of building ML applications by automation. ML experts can profit from AutoML by automating tedious tasks like Hyperparameter optimization leading to higher efficiency. Domain experts can be enabled to build ML pipelines on their own without having to rely on a data scientist.

It is important to note that AutoML is no new trend. Starting from the 1990s commercial solutions offered automatic Hyperparameter optimization for selected classification algorithms via grid search (Dinsmore, 2016). In 2004, the first efficient strategies for Hyperparameter optimization were proposed. For limited settings, e.g., tuning C and γ of an SVM (Chen et al., 2004), it was proven that guided search strategies yield better results than grid search in less time. Also in 2004, the first approaches for automatic feature selection were published (Samanta, 2004). Full model selection (Escalante et al., 2009) was the first attempt to automatically build a complete ML pipeline by simultaneously selecting a preprocessing, feature selection and classification algorithm while tuning the hyperparameters of each method. Testing this approach on various data sets, the potential of this domain-agnostic method was proven (Guyon et al., 2008). Starting from 2011, many different methods of applying Bayesian optimization for hyperparameter tuning (Bergstra et al., 2011; Snoek et al., 2012) and model selection (Thornton et al., 2013) have been proposed. In 2015, the first method for automatic feature engineering without domain knowledge was proposed (Kanter and Veeramachaneni, 2015). Building arbitrary shaped pipelines have been possible since 2016 (Olson and Moore, 2016). In 2017 and 2018 the topic of AutoML received a lot of attention in media (Google, 2019) with the release of commercial AutoML solutions from various global players (Golovin et al., 2017; Clouder, 2018; Baidu, 2018). Simultaneously, research in the area of AutoML gained significant traction leading to many performance improvements. Recent methods are able to reduce the runtime of AutoML procedures from several hours to mere minutes (Hutter et al., 2018). To further optimize AutoML for deep learning, a new stack called LEAF stack was developed which can optimize parameters, components, and topology of the architecture simultaneously to fit the requirements faster than the current state-of-the-art hand-designed architectures (Jason Liang et al., 2019)

1. Proposed Methodology

Two important problems in AutoML are that (1) no single machine learning method performs best on all datasets and (2) some machine learning methods (e.g., non-linear SVMs) crucially rely on hyperparameter optimization.

The latter problem has been successfully attacked using Bayesian optimization, which nowadays forms a core component of an AutoML system. The former problem is intertwined with the latter since the rankings of algorithms depend on whether their hyperparameters are tuned properly. Another major drawback of current AutoML platforms is their complexity and cost. A person with little to no experience in data science and machine learning will find it difficult to understand and use the platform.

In this paper, we propose a system of integrated modules. These modules split the problem into simpler parts and form a high-level pipeline. The modules are as follows:

-Data validation

-Identification of Problem

-Model Building

-Data preprocessing

-Visualization and Results

-Advanced preprocessing

With these independent modules, our system is able to tackle the CASH problem more effectively. The CASH problem is defined below.

Let A = {A(1), . . . , A(R)} be a set of algorithms, and let the hyperparameters of each algorithm A(j) have domain Λ(j). Further, let

Dtrain = {(x1, y1), . . . ,(xn, yn)} is a training set which is split into K cross-validation folds {Dvalid(1), . . . , Dvalid(K)} and {Dtrain(1), . . . , Dtrain(K)}

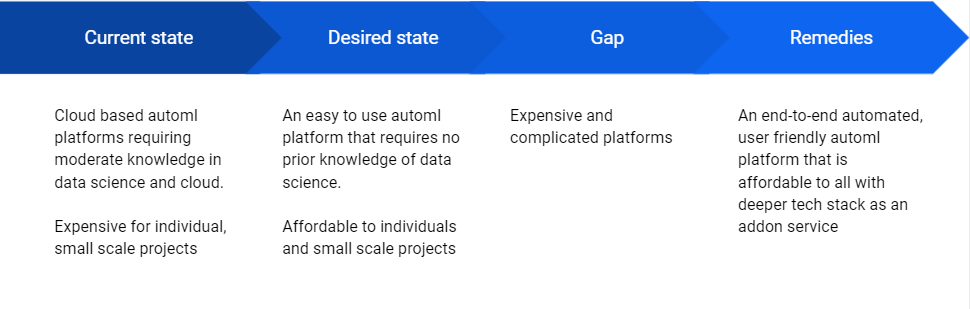
such that Dtrain(i) = Dtrain\Dvalid(i) for i = 1, . . . , K.

Finally, let L(Aλ(j), Dtrain(i), Dvalid(i)) denote the loss that algorithm A(j) achieves on Dvalid(i) when trained on Dtrain(i) with hyperparameters λ. Then,

the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem are to find the joint algorithm and hyperparameter r setting that minimizes this loss:



The user will be able to reap the benefits of AutoML with little to no prior experience in the field. Using optimizations and restrictions the platform is affordable to individuals and small scale businesses. The figure below encompasses the gap between existing systems and what we aim to achieve.



IV. Architecture

To design a robust AutoML system, as our underlying ML framework we chose

scikit-learn [7], one of the best known and most widely used machine learning libraries. It offers a wide range of well established and efficiently implemented ML algorithms and is easy to use for both experts and beginners.

The system is designed in a modular manner to make the individual components work individually in an efficient manner and also are designed to be cohesive with each other to create the core functional part of the system.

The main components of the core functional stack are:

* Data validator
* Data preprocessor
* Model engine
* Data profiler
* Prediction mechanism

1. Data validation and preprocessing

The client end of the system is realized and implemented using web based technologies which allows the user to upload any dataset from its local computer. The uploaded dataset is passed using a data validator to check if the dataset meets certain criteria which are the base requirements for a dataset in our system for performance and design purposes.It is at this time that data is identified as a regression problem or a classification problem.

1. Data preprocessor

Once a dataset has been accepted by the validator, it is passed on to the data preprocessor of the type of machine learning problem the dataset belongs to. Certain common data preprocessing as well as data preprocessing required for each type of problem is applied on the dataset. The goal of this component is to transform the data into a model ready dataset. It is an essential prerequisite of fitting an algorithm in the best possible way on the dataset.

1. Model engine

Each type of problem - classification and regression, has its own model engine. The model engine in its essence creates a hypothesis space of all possible pipelines of model fitting on the dataset, runs all the candidates and selects the best three candidates from baseline model building. The top three candidates are then subjected to optimization using a Randomized search for hyperparameter tuning. The goal of this component is to identify the best possible candidate from the search space of possible model pipelines and optimise them for the best possible results.

* Evaluation criteria

The criteria to evaluate a model is as important as the model itself. Different metrics convey different information about the model performance and each metric has its own merit and demerit. For regression analysis, we concluded on using root mean squared error and adjusted R2 and for classification problems, we will be using F1score and accuracy as the metrics of choice.

1. Data profiler

Data profiler is responsible for giving insights to the data through visualizations. Some of the data insights are following:

* Correlation
* Univariate analysis of features
* Chi square analysis
* Statistical analysis of the data
* Outlier analysis

1. Prediction mechanism

Prediction mechanism is the last component of the system. Once the model is fitted to the dataset, the model can be utilized by the prediction mechanism to predict new data.

V. Future Work

A truly intelligent Automl system should be able to reduce the search space of the possible pipelines even before running all the pipelines on the data. To achieve this we propose a meta-learning model that sits on top of the existing model engine. This meta-learning model will be a pre-trained model using a set of datasets. This meta-learning model will be able to identify certain features of the data and based on those features will be able to predict the best preprocessing techniques and predictive algorithms that will work on the data beforehand hence eliminating the need to run all possible versions of the pipeline on the data.

Our system at present only works for numerical and categorical data. We plan to extend the system's capabilities to textual data as well.

To extend the capabilities of the system, we plan to create API endpoints to allow users to integrate trained models into their systems.

VI. Conclusions

We demonstrated that our new AutoML system performs favourably against the previous state of the art in AutoML and that our meta-learning and ensemble improvements for AutoML yield further efficiency and robustness. In this paper, we did not evaluate the use of No code ML for interactive machine learning with an expert in the loop and weeks of CPU power, but we hope in the future to run this utilizing the power of cloud computing. As such, we believe that No Code ML is a promising system for use by both machine learning novices and experts.

Our system also has some shortcomings, which we would like to remove in future work. As one

For example, we have not yet tackled Natural language or semi-supervised problems. Most importantly, though, the focus on regression and classification implied a focus on small to medium-sized datasets, and an obvious direction for future work will be to apply our methods to modern deep learning systems that yield state-of-the-art performance on large datasets; we expect that in that domain especially automated ensemble construction will lead to tangible performance improvements over Bayesian optimization.

At the end of this project, we see the potential of an easy to use no-code application in the business intelligence field for future predictions.

VII. References

[1]  *Randal S. Olson and Jason H. Moore* Identifying and Harnessing the Building Blocks of Machine Learning Pipelines for Sensible Initialization of a Data Science Automation Tool

[2] *Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, Frank Hutter* Efficient and Robust Automated Machine Learning

[3] *Chi Wang, Qingyun Wu, Markus Weimer, Erkang Zhu* FLAML: A Fast And Lightweight AutoML Library