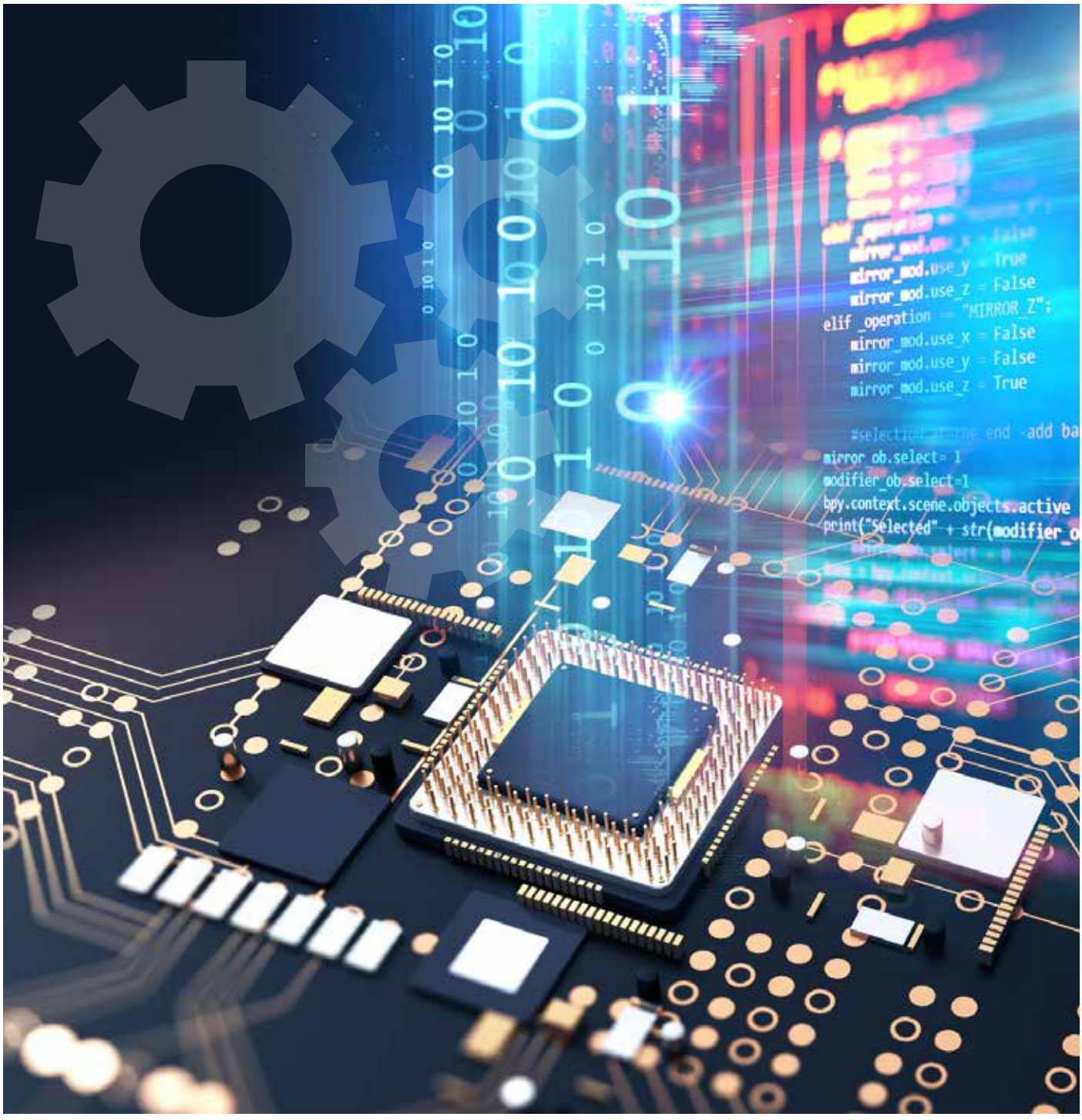
Mphasis
The Next Applied

AutoML: Automating the Machine Learning Pipeline

Whitepaper by Ojasvi Jain, Data Science Intern at Mphasis |
Kaushlesh Kumar, AVP - Applied AI, Mphasis NEXT Labs



Contents

1.
Introduction

As an applied field, Machine Learning (ML) is rapidly maturing from a research- and experimentation- heavy framework (which required significant investments) to a ‘productionalized’ enabler of solutions for mainstream industries.

Along with ‘productionalization’ comes the business requirement to make solutions faster, better and less expensive. Conventional software development and deployment have adopted DevOps practices to shorten the systems development lifecycle and provide continuous delivery with high software quality. The data science practice, on the other hand, has adopted best practices from DevOps in their own development and deployment framework calling it MLOps. The focus here is more on collaboration, tracking and monitoring, deployment, quality of production ML and enhanced automation. One of the areas which is both core to data science and has a significantly high cycle time is model development.

Traditional ML processes are inherently time-consuming and dependent on human intervention and expertise. The theorem of “no free lunches” also implies that the only scenario when an approach outperforms another is when it is customized to the specific problem on hand. This means that the model development invariably has to go through the pains of data preparation, feature engineering, model training, evaluation and selection every time a new dataset is encountered or a new problem arises.

Automated Machine Learning (AutoML) aims to automate and accelerate the process of building ML and deep learning models. It allows us to provide the labelled training data as input and receive an optimized model as output. This functionality enables data scientists to generate accurate insights by leveraging the capabilities of Machine Learning.

AutoML allows professionals from various domains to leverage the benefits of data science and ML. It accelerates processes, reduces errors and costs and provides greater accuracy of results by training multiple high performing models. Most of a data scientist’s time is invested in mundane tasks such as data pre-processing and feature engineering. Introduced to cut down time spent on iterative tasks concerning model development, AutoML channelizes more of their time and resources into model selection and model tracking so that data scientists can deal with more complex problems. These tools have helped developers to build scalable models with pipeline output acceleration.

2.
ML Pipeline and the Need for AutoML

An ML pipeline is used to automate ML workflows and implement AutoML solutions.

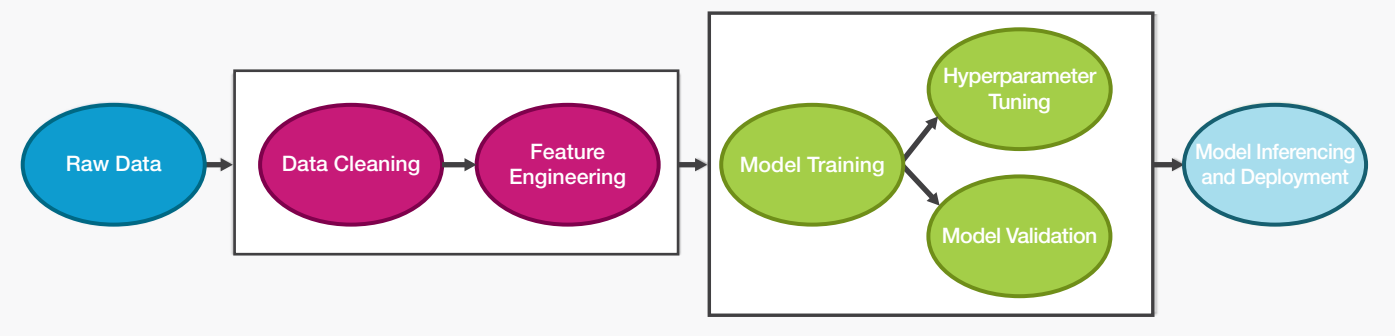


Fig.1: Machine Learning Pipeline

The pipeline starts with ingesting raw data, which is then engineered to suit algorithm and domain requirements by leveraging data cleaning and feature engineering techniques. A ML model is then trained on this data and validated by hyperparameter tuning. The best performing model is then deployed and used for production.

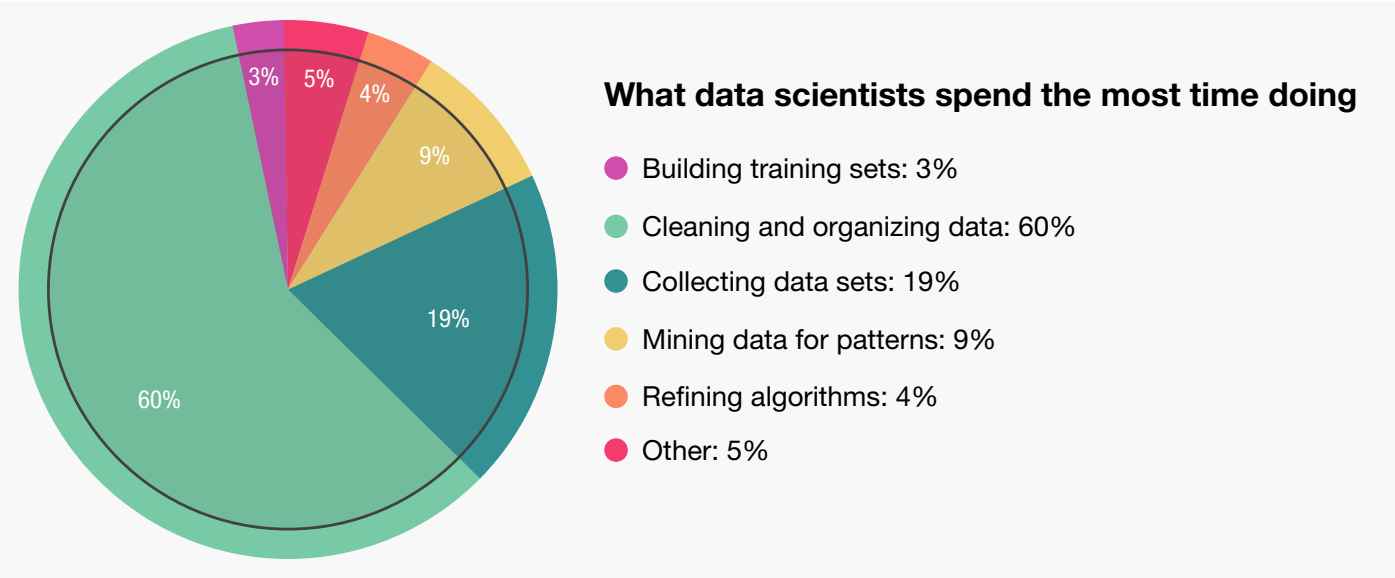


Fig.2: Forbes' Insights

As can be seen from Fig. 2, data cleaning and preparation occupies 60 percent of a data scientist's workload. Tasks such as cleaning the data and exploring multiple models to select the most accurate one can be easily automated using AutoML frameworks.

AutoML automates ML pipeline components in the following ways:

Data Preprocessing

A crucial step in the ML pipeline, data preprocessing enhances data quality and enables the user to draw meaningful insights. It also makes the data suitable for ML algorithms and enables a smooth ML pipeline. As real-world data can be inaccurate and inconsistent, it is

therefore imperative to clean, format and structure it to make it model-ready and gain relevant insights. Most ML practitioners face the challenge of processing data accurately and efficiently. It is believed that around 80 percent of a data scientist's time is spent on preprocessing the data and making it usable.

Data preprocessing involves the following steps:

Handling missing values

Often while data is collected, some fields and values do go missing. AutoML frameworks can detect missing values and apply data imputation techniques to ensure that crucial data points are not ignored by the algorithm.

Depending on the data type, AutoML can impute either the mean, median or mode of the column. Other techniques include imputing by a constant value specified by the user to the nearest neighbour (KNN) value – or, predicting missing values using regression algorithms in the column detected with missing values.

Handling outliers

Outliers are anomalous data points that may negatively affect test results. A few methods for outlier detection and removal can be used in AutoML software.

- *Isolation Forest* identifies and detects anomalies instead of profiling data points. The process can be automated by passing a data frame through an Isolation Forest algorithm.
- *Local Outlier Factor* is extremely useful when the dataset has a small number of features. It is a technique that attempts to harness the idea of nearest neighbours for outlier detection and can be leveraged for automatic detection of outliers.

Encoding

Most ML algorithms only accept numerical data. Categorical variable encoding is a fundamental step in the smooth functioning of a ML pipeline. AutoML encodes categorical variables in two ways:

- **Label Encoding** - Dependent categorical variables/ordinal categorical variables
- **One Hot Encoding** - Usually for nominal independent categorical variables

Though these processes are not completely automated, they are increasingly coming under the scope of AutoML pipelines.

Feature Engineering

This is the next step after data preprocessing. An essential function of any ML algorithm is to accurately and efficiently identify and predict features. For this to happen, the algorithms must receive structured data to enhance their predictive power. Feature engineering often requires domain knowledge to define and manipulate features to serve a business purpose.

Feature transformation

Feature transformation refers to creating new features from existing ones. Although few algorithms transform features internally (such as support vector machines), a few explicit ways of feature transformation include:

- **Feature Scaling and Normalizing** - Scaling and normalizing techniques are not only essential for efficient computing, but also for distance-based algorithms. Based on the data type, AutoML can automate scaling and normalizing techniques.
- **Aggregation** - This combines values in a feature to reduce variability in data.
- **Filtering** - This is mostly used in Time Series to regulate the frequency in the dataset by determining threshold values.

Feature selection

Feature Selection techniques include choosing a specific set of variables that are appropriate for subsequent analysis. The goal is to come up with the smallest set of features that best represent the data characteristics. A few methods used for feature selection are:

- **Filter Method** - This is the fastest method that eliminates highly correlated variables. It includes metrics such as Pearson's Correlation, which identifies variables that are highly correlated and automatically eliminates one of them.
- **Wrapper Method** - This is the most accurate method, but is very computationally intensive. It consists of algorithms such as Forward Selection, Backward Elimination, Recursive Feature Elimination and Bi-directional Elimination.
- **Embedded Method** - Embedded methods include regularization techniques such as Ridge Regression, Lasso Regression and ElasticNet Regression. These techniques are most commonly used in AutoML frameworks and aim to reduce variance while model training.

Dimensionality reduction

This reduces high-dimensional data to a more manageable one to not only improve performance but also improve accuracy of ML algorithms. One of the most common ways to perform Dimensionality Reduction is Principal Component Analysis, which aims to capture the maximum amount of variability in the dataset and map it to a smaller dimension.

However, Feature Engineering can be tricky since it largely depends on the data type of the variable and the application. For instance, ordinal variables can be considered categorical as well as discrete. In the case of a categorical data type, scaling and normalizing reduces the interpretability of the data and is not ideal. Thus, even though scaling and normalizing can be automated for incorporating in the AutoML frameworks, some kind of human intervention is needed.

Model Training and Hyperparameter Tuning

Once the features are ready, the next step is to train a model. Model training encompasses a huge variety of ML algorithms that can be leveraged to get a desired predicted value. Each model has

its benefits and is fit to be used in specific use cases. An important aspect of model training is model validation and hyperparameter tuning. It is essential that the ML model works as accurately and efficiently in the real world as it does on training data. Usually, after training a model, it is also run on a validation set to ensure that the model performs equally well on a different dataset than the training one. Often, it is difficult to find the most optimal model architecture in the beginning. Hyperparameter tuning is widely used for model validation and model tuning, and is the process of finding the most optimal hyperparameters for best accuracy.

A few methods of hyperparameter tuning include:

1. Manual Search with Cartesian Grid.
2. Grid Search Cross Validation, a popular hyperparameter technique leveraged by many data scientists, automates hyperparameter tuning by specifying the number of validations. Random Search can be implemented by AutoML frameworks by specifying a grid search technique such as “Random Discrete”.

Model Inferencing and Deployment

After building and training the ML model, we must now test the model and evaluate its performance in the real world. A testing dataset is provided to predict and evaluate model performance using various metrics of accuracy and the model is sent to production for deployment and day-to-day use.

3.
Prominent Open-source AutoML Frameworks

H2O.ai

H2O is an open-source machine learning platform developed by H2O.ai. H2O is available for all major preferred languages like R, Python, Java, Scala, etc., and has customized versions of the machine learning and deep learning algorithms for the platform leading to enhanced performance. The H2O AutoML automates tasks like feature engineering, model validation, model tuning, model selection and model deployment. Some of the main features of H2O AutoML are:

- Available through scripting in multiple languages as well as through a Web GUI (Flow)
- Customization of AutoML tasks available through passing of parameters
- Native capability of many data preparation tasks like missing value imputation, transformations, etc.
- Has an excellent model selection framework including forming stacked ensembles
- Good integration with big data platforms like Hadoop, Spark, etc.
- Large and active user and contributor base
- Difficult to use with other frameworks like scikit learn

PyCaret

PyCaret is available as an open-source library in Python and claims to be a low code framework reducing the model development time. This is actually the Python version of “caret” package in R. The operations performed using PyCaret become part of a custom pipeline that is ready for deployment without any additional processing. It provides a Python based wrapper around most of the popular machine learning libraries such as scikit-learn, XGBoost, LightGBM and even natural language processing and deep learning frameworks, etc.

- It allows for defining the various data preprocessing and transformation operations to perform using its setup() function
- It also allows to compare, evaluate and tune many standard machine learning algorithms through its compare_models() and tune_model() functions
- Ensembling models, saving pipelines and deploying the pipelines that are available
- Supports supervised learning algorithms
- No GUI interface is available

Auto-Sklearn

Auto-Sklearn is built around the widely used scikit-learn. It extends the concept of configuring a general MI framework with efficient goal optimization by building an ensemble of all models tested during the global optimization. It also uses Bayesian Hyperparameter Optimization via Meta-Learning wherein it uses knowledge from previous optimization that runs on various datasets to identify the best algorithm-hyperparameter combination.

- Available with 15 classification algorithms and 14 feature engineering pipelines
- Natively handles scaling, encoding and missing values
- Sometimes ensembling fails to converge
- Meta-feature computation process can be compute intensive and slow

TPOT (Tree-Based Pipeline Optimization Tool)

One of the initial AutoML methods was the TPOT. It creates a combination of a flexible expression tree representation of ML pipelines with stochastic search algorithms while using algorithms in the scikit-learn library. While TPOT uses the scikit-learn framework, it has its own regressor and classifier methods.

- Training time can be restricted by setting a time limit or population size. Its search space can be restricted by a configuration file.
- The optimization process can be paused and resumed
- The biggest feature of TPOT is that it can port the optimized pipeline to code to be further modified manually

- TPOT cannot automatically process natural language inputs and categorical inputs must be integer encoded before feeding the data
- Since it uses genetic programming, running times can be long before a high accuracy is attained, but given time, it will find the best parameters

4.
Case Study

This project was executed for one of the largest pharmaceutical companies of the world. The manufacturing process in the industry has very high standards of quality controls and has to adhere to extremely stringent regulations. If any issue is detected during the manufacturing process, it undergoes a rigorous process of root cause analysis with corrective and preventive actions (CAPA) to avoid issue recurrence. Once an issue has been identified and logged, a Quality Assurance Review (QAR) process is initiated. In this process, the issue needs to be analyzed and actioned to place controls in an established timeframe. Any delay in the QAR process leads to loss of productivity and revenue along with compliance issues.

The client requested for a ML-based solution to identify causes that could delay their QAR, so that more resources could be allocated with greater management focus.

An AutoML solution was implemented for efficient execution of tasks such as data preprocessing and feature engineering and increased accuracy of the predictive model. Since the dependent column is categorical, this is a classification problem. The proposed solution was implemented by leveraging PyCaret and H2O.ai frameworks.

The leaderboards generated by both the frameworks are given below:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.6748	0.8252	0.6431	0.6747	0.6687	0.4692	0.4738	2.3840
gbc	Gradient Boosting Classifier	0.6683	0.8253	0.6109	0.6670	0.6490	0.4435	0.4553	1.2280
et	Extra Trees Classifier	0.6667	0.8211	0.6165	0.6624	0.6536	0.4436	0.4522	0.2760
ridge	Ridge Classifier	0.6593	0.0000	0.6221	0.6587	0.6514	0.4435	0.4485	0.0570
catboost	CatBoost Classifier	0.6512	0.8160	0.6015	0.6444	0.6363	0.4214	0.4297	33.6120
rf	Random Forest Classifier	0.6496	0.8166	0.5714	0.6579	0.6216	0.3934	0.4161	0.2570
xgboost	Extreme Gradient Boosting	0.6407	0.7987	0.5937	0.6303	0.6284	0.4057	0.4110	4.0380
lda	Linear Discriminant Analysis	0.6309	0.7740	0.6037	0.6330	0.6283	0.4055	0.4080	0.2570
lightgbm	Light Gradient Boosting Machine	0.6301	0.7885	0.5830	0.6246	0.6165	0.3864	0.3939	0.2800
ada	Ada Boost Classifier	0.6268	0.7356	0.5848	0.6152	0.6152	0.3886	0.3925	0.1260
dt	Decision Tree Classifier	0.6187	0.6872	0.5856	0.6200	0.6170	0.3822	0.3840	0.0630
svm	SVM - Linear Kernel	0.6163	0.0000	0.5666	0.6473	0.5790	0.3646	0.4040	0.0670
knn	K Neighbors Classifier	0.6114	0.7431	0.5638	0.6039	0.5957	0.3595	0.3658	0.1570
nb	Naive Bayes	0.4325	0.6533	0.5267	0.5007	0.3793	0.2072	0.2437	0.0390
qda	Quadratic Discriminant Analysis	0.3610	0.5277	0.3754	0.5004	0.3310	0.0525	0.0672	0.2070

Fig.3: PyCaret Leaderboard

model_id	mean_per_class_error	logloss	rmse	mse	accuracy_score
GBM_4_AutoML_20201120_154046	0.364967	0.730957	0.501678	0.251681	0.708955
GBM_1_AutoML_20201120_154046	0.366122	0.710949	0.495178	0.245201	0.69403
GBM_3_AutoML_20201120_154046	0.371626	0.734427	0.504922	0.254946	0.705224
GBM_2_AutoML_20201120_154046	0.372156	0.723502	0.499386	0.249387	0.708955
GBM_5_AutoML_20201120_154046	0.373944	0.73497	0.509343	0.25943	0.682836
DRF_1_AutoML_20201120_154046	0.404334	0.757848	0.517709	0.268022	0.701493
GBM_grid__1_AutoML_20201120_154046_model_1	0.410331	0.744887	0.515659	0.265904	0.679104
XRT_1_AutoML_20201120_154046	0.41596	0.759434	0.520022	0.270423	0.697761
DeepLearning_1_AutoML_20201120_154046	0.511869	1.18038	0.612473	0.375123	0.507463
GLM_1_AutoML_20201120_154046	0.666667	1.01985	0.631025	0.398193	0.5

Fig.4: H2O Leaderboard

Pipeline Step	PyCaret	H2O.ai
Preprocessing		
Missing Values	Mean/Constant value	Mean/Constant value
Outliers	Single Value Decomposition	Isolation Forest
Fixing Imbalance	SMOTE	Oversampling and Undersampling based on specified proportion value
Feature Engineering		
Feature Scaling	Normalization; Box-Cox for target transformation	Standardization, Normalization
Feature Aggregation	Creates new columns through interaction of existing features	Limits Categorical levels to most relevant
Feature Selection	Removes Multicollinearity/PCA	PCA
Model Training	Machine Learning models	Machine Learning models, Deep Learning Models, Stacked Ensembles
Model Validation	Hyperparameter Tuning using RandomSearch, GridSearch, Bayesian Search	Hyperparameter Tuning using RandomSearch and GridSearch
Model Deployment	Model saved as Pickle file	Model saved as MOJO/POJO file
Use Cases	Classification, Regression, Clustering, Anomaly Detection, NLP	Classification, Regression, Clustering
Best Performing Model	Logistic Regression	Gradient Boosting Machine
Best Accuracy Score	0.6748	0.7089
Total Training Time	9.48 minutes	4.54 minutes

The table shows that H2O.ai outperforms PyCaret in not only accuracy, but also in training time. H2O's best model, Gradient Boosting Machine, gave an accuracy of 70.89 percent while PyCaret's best performing model, Logistic Regression, gave an accuracy of 67.48 percent. PyCaret also trained 15 models in 9.48 minutes while H2O trained 10 models in 4.54 minutes.

5.
Benefits of AutoML

As we see, the key tenets of ML involve various processes. Some of these processes such as data preprocessing and model tuning are mundane and require little to no human intervention. With a significant rise in ML applications over the past few years, AutoML has gained much traction. It not only minimizes the workload on data science professionals, but also enables citizen data scientists to leverage the benefits of ML solutions.

AutoML provides distinct advantages of:

Speed

The process of training multiple models and choosing one with the best accuracy is an important aspect of any ML solution. Finding the model with the most optimal hyperparameter can take several iterations. Apart from this, some complex ML algorithms can take up to several hours to train and may still not provide satisfactory results. Most AutoML frameworks aim to mitigate this problem by training several algorithms concurrently along with hyperparameter tuning, thus drastically reducing time.

Accuracy

Most ML algorithms fail when exposed to test datasets - either due to high variance or high bias during the feature engineering and model training processes. AutoML reduces bias by limiting human intervention and automating most processes in the ML pipeline, and also reduces variance by evaluating the model against different validation datasets. AutoML frameworks often perform validation techniques such as GridSearch Cross-Validation or RandomizedSearch against different validation datasets to choose the algorithm that gives the highest accuracy and is best suited for the problem statement.

Scalability

AutoML aims to automate the entire ML pipeline - right from data preprocessing to model deployment. For many problem statements such as Time Series Analysis, with a constant influx of new data, AutoML provides a framework that not only monitors the data, but also constantly trains and improves the algorithms based on the new incoming data. Most AutoML frameworks also provide GPU accelerators to handle large amounts of data and train multiple complex algorithms simultaneously. These frameworks also involve distributed systems which enable multiple ML models to be trained simultaneously.

Optimization of ML Processes

One of the key benefits of an AutoML framework is its ability to explore multiple algorithms with ideal hyperparameters resulting in an extremely accurate system.

Reduced Resource Expenditure

Simultaneous processing of multiple algorithms optimizes time and cost. AutoML not only speeds up redundant processes, but also enables efficient processing of more complex steps in the ML pipeline.

Democratization of the Power of ML-based Solutions

ML is embedded in most business decisions today - and rightfully so. AutoML enables citizen ata scientists to perform complex ML tasks with just a few simple steps such as data ingestion and inferring the results. This enables individuals to leverage ML capabilities without advanced data science knowledge.

Elimination of Errors due to Human Intervention

Often with human intervention, excessive bias creeps into several ML tasks. AutoML minimizes human interference, thus reducing the scope of its errors.

6.
Challenges

While AutoML systems can speedily generate predictive models to achieve near-optimal performances, their coverage is still narrow and raise few challenges such as lack of high-quality data and fixed optimization objectives. There exists a need for business and domain expertise to implement customized solutions. Few other challenges faced by an automated machine learning workflow include:

Data Inconsistency

Inconsistent data poses a huge challenge for an AutoML pipeline. Dealing with semi-structured and unstructured data can be tedious and often impossible to deal with by many AutoML frameworks.

Explainability

When the entire ML pipeline is automated, it may be difficult to explain certain aspects of the result, especially if it is not what the user wants. A key challenge in most AutoML frameworks is model explainability which arises due to minimal interference from the user.

Need for Domain Expertise

Often realistic objectives are lost behind the set AutoML framework and the results can be irrelevant in a business decision-making process. Sometimes, it may be necessary to customize a few aspects of the ML pipeline - for instance, removing a variable that is not important for a business decision.

7.
Conclusion

ML as an element of data science will continue the practice of hypothesis creation and experimentation. What AutoML strives to bring to the table is automating these aspects to identify the best performing algorithm from the available universe of features, algorithms and hyperparameters. AutoML promises to facilitate intelligent automation of repetitive tasks in the ML workflow. This enables high-value resources to move away from repetitive tasks to value-adding analysis and evaluation of best performing models. This will result in significant improvement in time-to-production of models and solutions built on these models.

While AutoML systems can speedily generate predictive models to achieve near-optimal performances, their coverage is still narrow and their true potential still untapped. While we have seen increasing influence of AutoML in feature engineering and data preprocessing, there still exist areas of heavy domain-dependence, where it is more of an art than engineering. As an active area of research that is making rapid strides (with many players tackling existing issues in automating the entire model development process), AutoML will be a major capability to accelerate the adoption of ML-based solutions.

8.
References

- https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#983c1d86f637
- https://www.jeremyjordan.me/hyperparameter-tuning/
- https://www.infoworld.com/article/3430788/automated-machine-learning-or-automl-explained.html
- https://research.aimultiple.com/auto-ml/
- https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf
- https://medium.com/riselab/scalable-automl-for-time-series-prediction-using-ray-and-analytics-zoo-b79a6fd08139
- https://www.getapp.com/resources/citizen-data-scientist-vs-expert-data-scientist/
- https://www.datarobot.com/wiki/citizen-data-scientist/#:~:text=%E2%80%9CCitizen%20Data%20Scientist%2C%E2%80%9D%20a,to%20implement%20machine%20learning%20technology.
- https://analyticsindiamag.com/what-are-the-limitations-of-automl/
- https://www.h2o.ai/
- https://pycaret.org/

Authors

Ojasvi Jain

Data Science Intern at Mphasis

Ojasvi Jain is a Data Science Intern at Mphasis. She has a keen interest in various ML applications and has built AutoML, Explainable AI and NLP solutions during her internship. She is pursuing her Bachelor of Technology in Data Science from NMIMS University and wants to further explore her interests in the field of Responsible AI.

Kaushlesh Kumar

AVP - Applied AI, Mphasis NEXT Labs

Kaushlesh Kumar, in his career spanning more than a decade, has helped solve business problems for clients using innovative and structured approaches. In his current role, Kaushlesh designs, develops and executes solutions for transforming business services leveraging machine learning, data science and process. These initiatives deliver super normal value to the clients. He has a PG Diploma in Management and is a graduate engineer.

About Mphasis

Mphasis (BSE: 526299; NSE: MPHASIS) applies next-generation technology to help enterprises transform businesses globally. Customer centricity is foundational to Mphasis and is reflected in the Mphasis' Front2Back™ Transformation approach. Front2Back™ uses the exponential power of cloud and cognitive to provide hyper-personalized ($C = X2C^{2}_{TM} = 1$) digital experience to clients and their end customers. Mphasis' Service Transformation approach helps 'shrink the core' through the application of digital technologies across legacy environments within an enterprise, enabling businesses to stay ahead in a changing world. Mphasis' core reference architectures and tools, speed and innovation with domain expertise and specialization are key to building strong relationships with marquee clients. To know more, please visit www.mphasis.com

For more information, contact: marketinginfo.m@mphasis.com

USA
460 Park Avenue South
Suite #1101
New York, NY 10016, USA
Tel.: +1 212 686 6655

UK
1 Ropemaker Street, London
EC2Y 9HT, United Kingdom
T : +44 020 7153 1327

INDIA
Bagmane World Technology Center
Marathahalli Ring Road
Doddanakundhi Village
Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

www.mphasis.com

NR 12/01/21 US LETTER 0A51L6624