

An Automated and Distributed Machine Learning Framework for Telecommunications Risk Management

Luís Ferreira^{1,2}^a, André Pilastrí¹^b, Carlos Martins³^c, Pedro Santos³^d and Paulo Cortez²^e

¹*EPMQ - IT Engineering Maturity and Quality Lab, CCG ZGDV Institute, Guimarães, Portugal*

²*ALGORITMI Centre, Dep. Information Systems, University of Minho, Guimarães, Portugal*

³*WeDo Technologies, Braga, Portugal*

Keywords: Automated Machine Learning, Distributed Machine Learning, Supervised Learning, Risk Management.


Abstract: Automation and scalability are currently two of the main challenges of Machine Learning. This paper proposes an automated and distributed ML framework that automatically trains a supervised learning model and produces predictions independently of the dataset and with minimum human input. The framework was designed for the domain of telecommunications risk management, which often requires supervised learning models that need to be quickly updated by non-ML-experts and trained on vast amounts of data. Thus, the architecture assumes a distributed environment, in order to deal with big data, and Automated Machine Learning (AutoML), to select and tune the ML models. The framework includes several modules: task detection (to detect if classification or regression), data preprocessing, feature selection, model training, and deployment. In this paper, we detail the model training module. In order to select the computational technologies to be used in this module, we first analyzed the capabilities of an initial set of five modern AutoML tools: Auto-Keras, Auto-Sklearn, Auto-Weka, H2O AutoML, and TransmogriAI. Then, we performed a benchmarking of the only two tools that address distributed ML (H2O AutoML and TransmogriAI). Several comparison experiments were held using three real-world datasets from the telecommunications domain (churn, event forecasting, and fraud detection), allowing us to measure the computational effort and predictive capability of the AutoML tools.


1 INTRODUCTION


The rise of data, processing power, and sophisticated learning algorithms opened the way for Machine Learning (ML) applications to deal with large amounts of data and produce useful predictions (Darwiche, 2018). When adopting ML in real-world applications, two particularly useful capabilities are distributed learning and AutoML. Distributed learning is a natural solution to large scale ML problems when there are memory and processing limitations. By using multiple computers or multi-core processors in parallel, each processor can process a different ML algorithm or portion of the data. This allows to overcome memory and time constraints to growing data just by adding new machines or processors (Peteiro-


Barral and Guijarro-Berdiñas, 2013). Also, with the increasing number of non-specialists working with ML (Thornton et al., 2013), it is essential to allow people with limited knowledge in the field to easily select and apply the best ML model. AutoML aims to help solve this issue and it is particularly relevant when constant model updates are required.


This paper proposes a technological architecture that addressed these two ML challenges and that is adjusted to the telecommunications risk management domain. This domain often requires supervised learning tasks (e.g., churn detection) that need to be set and continuously updated by non-ML-experts and that involve big data. Thus, our architecture defines a set of steps that automate the usual workflow of a supervised ML application, including modules for: task detection, data preprocessing, feature selection, model training, and deployment. This work particularly focuses on the model training module, which assumes a distributed AutoML tool. In order to select the computational ML tool, we first analyzed the capabilities of five AutoML tools (Auto-Keras, Auto-Sklearn,

^a  <https://orcid.org/0000-0002-4790-5128>

^b  <https://orcid.org/0000-0002-4380-3220>

^c  <https://orcid.org/0000-0002-0678-4868>

^d  <https://orcid.org/0000-0002-4269-5838>

^e  <https://orcid.org/0000-0002-7991-2090>

Auto-Weka, H2O AutoML, and TransmogriAI). The two tools with distributed ML capabilities (H2O AutoML and TransmogriAI) were then compared experimentally. The comparison used three telecommunications real-world datasets, related to churn (regression), event forecasting (time series) and fraud detection (classification).

The paper is organized as follows. In Section 2, we present the related work. Next, Section 3 details the proposed ML architecture. Section 4 describes the analyzed AutoML technologies and the datasets used during the experimental tests. Then, Section 5 discusses the experimental results. Finally, Section 6 presents the main conclusions and future work directions.

2 RELATED WORK

Typically, ML applications operate in-memory. Thus, when the computational complexity of the ML algorithm exceeds the machine memory, it does not scale properly. In this age of big data, there is a need to develop scalable and efficient algorithms that can deal with as many training data as possible under the constraints of memory and time (Peteiro-Barral and Guijarro-Berdiñas, 2013). In this work, we particularly address classical distributed ML, which distributes the computation among distinct processors, each performing an ML task using a static dataset.

Another relevant ML issue is related to the proper selection of the right algorithm to be used and the optimization of its hyperparameters. This model selection and tuning is often performed by using ML expert knowledge, heuristics and several trial-and-error experiments, often performed using ad-hoc setups (Feurer et al., 2015). For non-ML-experts, this is not a trivial task (Gibert et al., 2018). AutoML was specifically developed to solve this issue (He et al., 2019). It can be defined as the automatic (without user input) search for the best algorithm for a given dataset and producing predictions on the test set. This search can be controlled by some user-defined constraints, such as memory usage or execution time (Feurer et al., 2015).

Due to its relevance, several AutoML tools were proposed. Examples of currently available open-source tools include: Auto-Keras (Jin et al., 2018), Auto-Sklearn (Feurer et al., 2015), Auto-Weka (Kotthoff et al., 2017), H2O AutoML (H2O.ai, 2019) and TransmogriAI (Salesforce, 2019). Within our knowledge, there are scarce studies that compare AutoML technologies. Most research works compare a specific AutoML tool with a set of state-of-

the-art ML algorithms, such as presented in (Feurer et al., 2015). Other studies compare the performance of AutoML applications that were submitted to ML automation challenges, such as executed in (Guyon et al., 2019). Recently, a direct AutoML tool comparison was performed using a classification task (Gijsbers et al., 2019). Also in 2019, several AutoML tools were compared by using different types of datasets, grouped by task, number of rows, number of columns, types of columns or balance between classes (Truong et al., 2019). However, none of these recent AutoML comparison studies considered specifically the distributed ML dimension. For instance, as shown in Table 1, TransmogriAI is one of the few AutoML tools that contains a distributed ML capability, yet this tool was not considered in the experimental comparisons. Moreover, the recent comparison studies did not adopt specific telecommunications risk management domain datasets, such as churn or fraud detection.

3 PROPOSED ARCHITECTURE

This paper is part of the R&D project “Intelligent Risk Management for the Digital Age” (IRMDA), which is being developed by leading software and analytics Portuguese company. The main outcome of IRMDA is the development of a scalable and automated ML system to empower the software company Telecommunications clients, allowing them to easily deploy and maintain supervised ML risk management applications. The scalable and automated are fundamental ML requirements of the system since the company has a large number of clients that work with different amounts of data (small or vast). Also, typically these clients are non-ML-experts.

We propose an automated ML technological architecture that tries to combine the full automation of the ML workflow with the ability to be executed in a distributed environment. With the developed pipeline, it is possible to automatically deal with all steps of a typical supervised ML workflow just by identifying a dataset and its target column. Since the architecture is prepared to be executed on a computational cluster, with several processing nodes, this allows users to deal with bigger datasets just by adding new machines to the cluster, if needed. The architecture is illustrated in Fig. 1. The main components of the pipeline are:

- Machine Learning task detection - currently set to detect if the ML pipeline should be considered a classification (binary or multiclass) or a regression (pure regression or time series) task, based on the user-selected dataset and the datatype of

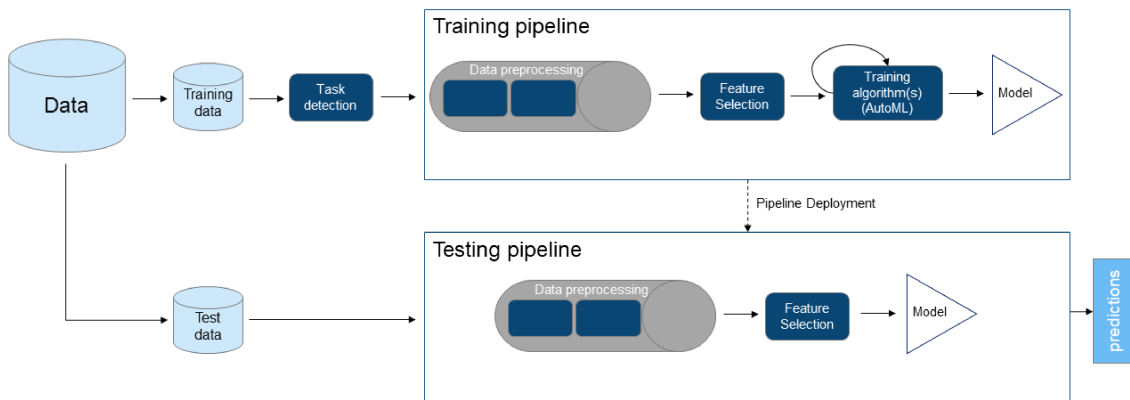


Figure 1: The proposed automated and scalable ML architecture.

the target column. The type of supervised tasks handled will be expanded according to feedback provided by software company clients and the AutoML tool capabilities. Interesting future possibilities of tasks to be addressed are an ordinal classification or multi-target regression. This component also automatically infers the dataset schema.

- **Data Preprocessing** - deals with missing data (e.g., average imputation), the encoding of categorical features (one-hot transform), a transformation of univariate time series into a dataset with time-lagged inputs and the standardization of numerical features.
- **Feature Selection** - removes irrelevant features from the dataset that can reduce the performance of the ML models. Currently, this component is implemented by using filtering methods (e.g., correlation values, information gain). In future, more sophisticated automatic feature selection approaches, such as provided by the TransmogriAI tool (Salesforce, 2019) or Apache Spark (which can be used with the H2O Sparkling Water tool (H2O, 2019)), will be addressed.
- **Model training using AutoML** - automatically trains and tunes a set of ML models using a set of constraints (e.g., time limit). This component also identifies the best model to be used on the test set.
- **Pipeline Deployment** - saves the pipeline that will be used on a test set. The pipeline ensures that the test data goes through the same transformations as the training data and that it uses the best model obtained during the training to make predictions.

The proposed architecture assumes two main phases (Fig. 1): a training phase and a test phase. The training phase includes the creation of the pipeline and the definition of the pipeline stages. In order

to process the pipeline, the user needs to select the dataset that will serve as training data and identify the target column. Then, the dataset is loaded and its schema is inferred. The task detection module analyzes the output target type, assuming a classification goal (binary or multiclass, depending on the number of target labels) or a regression task (pure or univariate time series, depending on if the dataset contains several variables or just the output target). The next steps consist of the definition of the pipeline stages based on the selected ML task. Each stage either transforms the data (e.g., dealing with missing data, categorical one-hot encoding) or creates a model based on the training data which will be used on the test phase to transform the data.

This paper focuses on the AutoML component of the architecture, which assumes a distributed ML and that is tested using real-world datasets from the telecommunications domain. In particular, we analyze and benchmark open-source AutoML tools, in order to technologically instantiate this component. The AutoML goal is to select the best ML algorithm for a given supervised learning task, tuning its hyperparameters and fitting the model to the training data. After this phase, the best model is added to the pipeline and this object is ready to be exported. When necessary, the pipeline may be imported and executed on test data. The new data will pass through every stage and produce predictions.

We highlight that current version of the overall architecture, which received a positive feedback from the Portuguese software company of the IRMDA project, is expected to be incrementally improved in future research. In particular, we intend to evolve and test the non AutoML components by using more real-world datasets and feedback from the software company clients.

4 MATERIALS AND METHODS

4.1 AutoML Tools

We first analyzed an initial set of five modern open-source AutoML tools, to check if they could fulfil the desired IRMDA project requirements.

4.1.1 Auto-Keras

Auto-Keras is an AutoML Python library. The focus of Auto-Keras is Deep Learning and the automatic search for architectures and hyperparameters of Deep Learning models. The goal of Auto-Keras is to allow domain experts with limited ML background to easily build and tune Deep Learning models (Jin et al., 2018). Auto-Keras allows the user to specify a time limit for the executions and the possibility of perform data augmentation.

4.1.2 Auto-Sklearn

Auto-Sklearn uses the the ML scikit-learn library (Pedregosa et al., 2011) to perform algorithm selection and hyperparameter tuning. It is a Python module that makes use of recent advances in Bayesian optimization, metalearning and ensemble learning to increase the efficiency (Feurer et al., 2015). Auto-Sklearn includes 15 ML algorithms, 14 feature preprocessing methods and 4 data preprocessing methods.

4.1.3 Auto-Weka

Auto-Weka is a module of the WEKA ML tool (Witten et al., 2016). The Auto-Weka module is the WEKA solution to the Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem on classification tasks (Thornton et al., 2013). In its most recent version, Auto-Weka also supports regression tasks (Kotthoff et al., 2017). Auto-Weka uses 27 WEKA learners, 10 metalearners and 2 ensemble methods.

4.1.4 H2O AutoML

H2O is a ML and predictive analytics platform. It works with in-memory data, using a distributed and scalable architecture that allows the users to build ML models on big data environments. H2O was built to facilitate the production of ML models when working in an enterprise environment (Cook, 2016). The H2O goal is to help non-ML-expert users to produce models with high performance (H2O.ai, 2019).

The result of an AutoML execution is an object that includes the leaderboard of all the models trained

and validated during the ML model search process. The number of searched models is dependant of the optional parameters selected by the user (e.g., time limit, exclusion of algorithms). The models on the leaderboard are sorted by an evaluation metric, which by default is the Area Under Curve (AUC) for binary classification, Mean Per Class Error for multi-class classification and Deviance for regression. H2O AutoML applies most of H2O families of algorithms, such as Gradient Boosting Machine (GBM), Generalized Linear Model (GLM), XGBoost, Random Forest and Deep Learning algorithms. It also allows very personalized executions for more advanced users. The user optional parameters include the specification of a time limit, evaluation metric, stopping metric, the possibility of excluding algorithms and automatic balancing of the training example classes.

4.1.5 TransmogriAI

TransmogriAI is a tool written in Scala that runs on Apache Spark. It focuses on the automation of ML applications, allowing the users to rapidly obtain ML models with few manual settings. The automation addresses the ML model selection and also feature selection and engineering phases (Salesforce, 2019).

The user only has to specify the dataset, the schema and the target column. TransmogriAI uses these parameters to infer more precise attribute types (e.g., phone numbers, zip codes) and automatically discards input features that do not present a predictive value. Depending on the type of problem that is being modeled (classification or regression), TransmogriAI trains a set of algorithms and a predefined set of hyperparameters. The list of trained algorithms includes Random Forest, Logistic Regression, Linear Regression and Naive Bayes, among others.

4.1.6 AutoML Tool Comparison

Table 1 presents the main characteristics of each AutoML tool in terms of: interface language, associated platforms, current version and if it contains a Graphical User Interface and distributed ML mode.

For the experimental comparison study, two tools were selected, H2O AutoML and TransmogriAI, since these are the only tools from Table 1 that include a distributed ML capability. The list of ML algorithms provided by the two computational tools are presented in Table 2. The last two H2O algorithms are related with stacking ensembles that reuse the searched individual ML models: all, which combines all model predictions; and best, which considers only best predictive results per ML algorithm.

Table 1: Summary of the characteristics of the AutoML technologies.

	Auto-Keras	Auto-Sklean	Auto-Weka	H2O AutoML	TransmogriAI
Interface language	Python	Python	Python R	Python R Scala AWS	Scala
Associated platforms	-	-	WEKA	Azure Google Cloud Spark	Spark
Current version	pre-release	0.6.0	2.6.1	3.26.0.6	0.6.1
Graphical User Interface	-	-	✓	✓	-
Distributed mode	-	-	-	✓	✓

Table 2: List of algorithms implemented by H2O AutoML and TransmogriAI.

Algorithm	H2O AutoML	TransmogriAI
Decision Trees	-	✓
Deep Learning	✓	-
Extremely Randomized Forest	✓	-
Gradient-Boosted Trees (GBT)	-	✓
Gradient Boosting Machine (GBM)	✓	-
Generalized Linear Model (GLM)	✓	-
Linear Regression	-	✓
Linear Support Vector Machine	-	✓
Logistic Regression	-	✓
Naive Bayes	-	✓
Random Forest (RF)	✓	✓
XGBoost	✓	-
Stacking All (SA)	✓	-
Stacking Best (SB)	✓	-

4.2 Data

For the benchmark comparison study, we consider three real-world datasets related with the telecommunications domain. The datasets were provided by the IRMDA project software company, which selected them as representative of common risk management ML tasks executed by the company. The datasets are related with customer churn (binary classification), event forecasting (univariate time series) and telecommunications fraud detection (regression).

4.2.1 Customer Churn

The customer churn dataset contains records from the software company clients. Each row includes the features that characterize each client and the associated probability for canceling the company analytics service (churn), as defined by the software company. The dataset is rather small, with 189 rows and 21 attributes (Table 3). In this work, the dataset is modeled as a pure regression task, where the ML models attempt to predict the numeric churn probability.

4.2.2 Event Forecasting

The event forecasting dataset contains the number of telecommunication events of a certain type (e.g., phone calls, short messages) that occurred from February to April of the year 2019. The total number of events was aggregated on an hourly basis, resulting in 1,418 records related with 59 days of hourly data. Each record contains the respective timestamp and the number of events in that interval, as described in Table 4. The number of hourly events ranges from 3,747 to 56,320. Considering the dataset temporal characteristics, it is modeled as a univariate time series task.

4.2.3 Fraud Detection

The fraud detection dataset includes data records about phone calls performed between a sender (A) and a receiver (B). Each row includes also the labeled classification of the call (“fraud” or “normal”), as identified by the software company using their expert fraud knowledge and analytic tools. The dataset much larger than the previously described datasets, it consists of more than 1 million data examples, which corresponds to a day of phone calls that were performed using the same client telecommunications company. The data attributes are described in Table 5. This dataset is modeled as a binary classification task.

Table 3: Summary of the attributes of the churn dataset.

Attribute	Description
tenure	Time since the beginning of the contract
streaming_quality	Resolution of the contract
prime_video	If prime video is contractualized or not
contract	Duration of the contract
payment_method	Specification of the method of payment
product_name	Identification of the product
platform	The type of connectivity
financial_status	If the payment is late or regularized
service_latency	Latency of the service
dropped_frames	Number of dropped frames
volume	Information about volume
duration	Information about duration
account_number	Account identification number
service_latency_category	Category of the service latency attribute
dropped_frames_category	Category of the dropped frames attribute
volume_category	Category of the volume attribute
duration_category	Category of the duration attribute
tenure_category	Category of the tenure attribute
account_segment	Age segment of the client
equipment	Equipment used by the client
churn_probability	Probability of canceling the service ($\in [0, 1]$)

Table 4: Summary of the attributes of the event forecasting dataset.

Attribute	Description
Time	Timestamp (yyyy-mm-dd hh:mm)
datapoints	Number of events in that interval

Table 5: Summary of the attributes of the fraud dataset.

Attribute	Description
A	Number of the sender of the call
B	Number of the receiver of the call
Result	Classification of the call (“fraud” or “normal”)

5 RESULTS

5.1 Experimental Setup

To benchmark the AutoML tools, we executed several computational experiments using the three real-world datasets. The tools were compared under the same experimental setup, which was run on a machine with an i7-8700 Intel processor with 6 cores. A holdout split was used to divide the datasets into training (with 3/4 of the data) and test (1/4) sets. For churn and fraud dataset, the split was randomly selected, while for the event forecasting data a time order division was used (since the data is ordered in time). Using the training data, each AutoML tool optimizes a single performance measure, which was set as the Mean Absolute Error (MAE) for the regression tasks (churn and event forecasting) and the AUC for the classification data (fraud detection). An internal 10-fold cross-validation was used by both AutoML tools in order to get a validation set where the selected performance measure is computed. For comparison purposes, for the test data we also computed the Normalized MAE (NMAE, in %, which is equal to the MAE divided by the target range) and Root Mean Squared Error (RMSE) values for regression tasks, and the Precision and Recall measures for the binary classification.

We tested all ML algorithms from Table 2, except for the Deep Learning, which was disabled from the H2O due to two main reasons. First, it required a huge computational effort, particularly for the large fraud detection dataset. Second, to achieve a more fair comparison, since TransmogriFAI AutoML tool does not include a Deep Learning algorithm. In order to allow the execution of all ML algorithms, no computational time execution limitation was used.

5.2 Churn

Two scenarios were designed to test the performance of the AutoML tools. The first scenario (1) assumes all the attributes of the dataset as input features of the ML models. The second scenario (2) uses an initial feature selection phase before training the models. The goal was to test the automatic feature selection option provided by TransmogriFAI. Under this scenario, and for H2O, we used the features that were considered more relevant by the best H2O performing ML model for the first scenario. The obtained results are presented in Table 6, where the computational execution time is presented in the *minutes:seconds* notation.

Table 6: Results for the churn data (best values in **bold**).

Tool	Scenario	Execution time	Best algorithm	Test data		
				MAE	NMAE	RMSE
H2O AutoML	1	00:28	SA	0.112	11.2%	0.189
	2	00:26	GLM	0.126	12.6%	0.186
TransmogriAI	1	03:44	RF	0.210	21.0%	0.283
	2	03:36	GBT	0.109	10.9%	0.136

5.3 Event Forecasting

Since both H2O AutoML and TransmogriAI do not have native univariate time series forecasting algorithms (e.g., ARIMA, Holt-Winters), we performed a transformation on the dataset by using a set of time lags to create the regression inputs. These lags were created using the CaseSeries function of the rminer R package (Cortez, 2010), under three input, lagged scenarios: 1 – with time lags $t - 1$, $t - 24$ and $t - 25$, where t is the current time (corresponding to the previous hour, day and hour before that day); 2 – with all time lags from the last 24 hours (from $t - 1$ to $t - 24$); and 3 – with the time lags $t - 12$, $t - 24$, $t - 36$ and $t - 48$. The results for each scenario are presented in Table 7.

5.4 Fraud Detection

The dataset is highly unbalanced, with only around 0.01% of the calls being illegitimate. In order to test the effect of balancing methods, three scenarios were analyzed during the training phase. The first scenario (1) used a simple oversampling which used “fraud” records and a random selection (with replacement) of “normal” cases. The second (2) and third (3) scenarios use the Synthetic Minority Oversampling Technique (SMOTE), which is a more sophisticated balancing method that generates synthetic examples for the minority class, such that the training data gets more balanced (Chawla et al., 2002). The SMOTE was used to generate 100% of new fraud cases in the second scenario and 200% of extra fraud examples in the third scenario. The test phase also considers three scenarios of unseen data, with different normal to fraud ratios: A – 50%/50%, thus balanced; B – 75%/24%; and C – 80%/20%. Table 8 shows the obtained results.

5.5 Discussion

In terms of computational processing time, the results show that in general a small effort is needed by the AutoML tools. The highest execution time is related

with the fraud detection data for the H2O tool and it corresponds to just around 9 minutes.

This small computational effort is explained by several factors, including: usage of a distributed ML and multi-core machine; usage of benchmark telecommunications datasets that are either have a small number of examples (churn, event forecasting) or inputs (fraud detection); and the disabling of the Deep Learning algorithm in the H2O tool. Nevertheless, the execution time results confirm that the AutoML can perform an automatic ML selection in a reasonable time. And if larger datasets were analyzed, the computational effort could be reduced by adding more processing elements to the computational cluster. As for the tool comparison, H2O AutoML required less time to process the regression tasks (churn and event forecasting), while TransmogriAI was faster for the classification task.

In terms of the predictive performance, H2O AutoML obtained better results for three regression comparisons (for both MAE and RMSE: scenario 1 for churn and scenarios 1 and 3 for event forecasting), and seven classification comparisons (when using the AUC measure). TransmogriAI obtained the best results in two regression scenarios and two classification ones. Overall, the AutoML predictive results are of high quality and the tools do not present substantial predictive differences. For example, all H2O AUC test results are equal to or higher than 95%, which corresponds to an excellent discrimination level. And the largest AUC classification difference when compared with TransmogriAI is just 3 percentage points. Similarly, the best churn prediction models present a NMAE value that corresponds to an interesting value of around 10%. The tool NMAE differences are small for scenario 2 (1.7 percentage points) but larger for scenario 1 (9.8 percentage points). As for the event forecasting, the predictions are of high quality, with NMAE values ranging from 4.0% to 6.6%. The tool NMAE differences are very small for scenarios 1 and 2, with percentage point differences of 0.10 and 0.07, while the difference is larger for scenario 3 (1.68 percentage points).

The predictive results confirm the potential of the distributed AutoML technologies, which are capable

Table 7: Results for the event forecasting data (best values in **bold**).

Tool	Scenario	Execution time	Best algorithm	Test data		
				MAE	NMAE	RMSE
H2O AutoML	1	02:32	GBM	2673	5.08%	4032
	2	02:53	GBM	2138	4.07%	3535
	3	01:50	GBM	2589	4.92%	4079
TransmogrifAI	1	05:16	GBT	2725	5.18%	4332
	2	05:00	RF	2101	4.00%	3441
	3	03:47	RF	3468	6.60%	5212

Table 8: Results for the fraud detection data (best values in **bold**).

Tool	Train Scenario	Execution time	Best algorithm	Test Scenario	AUC	Precision	Recall
H2O AutoML	1	05:20	GBM	A	0.97	0.99	0.98
				B	0.95	0.99	0.97
				C	0.95	0.98	0.99
	2	07:18	GBM	A	0.99	1	0.99
				B	0.97	0.99	0.98
				C	0.97	0.99	0.96
	3	08:54	GBM	A	0.99	1	0.99
				B	0.99	0.99	0.98
				C	0.98	0.99	0.98
TransmogrifAI	1	01:19	RF	A	0.98	0.99	0.96
				B	0.93	0.98	0.97
				C	0.92	0.98	0.93
	2	01:45	RF	A	0.98	1	0.99
				B	0.96	0.99	0.98
				C	0.96	0.97	0.94
	3	02:13	GBT	A	0.99	1	0.99
				B	0.98	1	0.98
				C	0.97	0.99	0.96

of achieving high-quality predictive results in a reasonable amount of time and with a minimum human intervention. The obtained results were shown to the risk management software and analytics company, which opted to select the H2O AutoML tool for several reasons. First, it provided better predictive results for the majority of the tested scenarios. In particular, when the AutoML tools presented the largest metric differences, the best results were achieved by H2O. Second, the company classified the tool as “more mature” software, since as shown in Table 1, it is available in different programming languages and it can be integrated with more platforms (other than Spark). Also, the H2O provided an easy to use Graphical User Interface.

6 CONCLUSIONS

In this paper, we proposed a ML framework that can handle the full ML workflow, including several modules to identify the ML goal, preprocess the data, select features, train models and deploy testing pipelines with a minimum human input. The framework was specifically designed within a R&D project that includes a major Portuguese software and analytics company that provides risk management services for the telecommunications domain. The telecommunications clients are typically non-ML-experts. Moreover, they work with varying sized datasets (small or large). Thus, the proposed ML framework works on a cluster that uses distributed ML to ensure scalability and makes use of AutoML capabilities to facilitate the development and maintenance of the ML models.

This work particularly focus on one of the core components of the architecture: the model training module. To instantiate technologically such model, we first analyzed the characteristics of five open-source AutoML tools (Auto-Keras, Auto-Sklearn, Auto-Weka, H2O AutoML and TransmogriAI). Then, we performed a benchmark experimental study with the two tools that presented a distributed ML capability: H2O AutoML and TransmogriAI. The experiments were conducted using three real-world datasets provided by the software company (churn, event forecasting and fraud detection). The obtained results allowed us to evaluate the potential of both AutoML technologies for the model training module of the proposed architecture.

Overall, the proposed framework received a positive feedback from the software company, which opted to select the H2O AutoML tool for its model training module. In future work, additional telecommunications datasets will be addressed, in order to further benchmark the AutoML tools. In particular, we wish to extend the framework ML capabilities to handle more ML tasks (e.g., ordinal classification, multi-target regression). Moreover, we intend to focus the development on the remaining components of the architecture, in order to select the best technologies to be used (e.g., for handling missing data).

ACKNOWLEDGEMENTS

This work was executed under the project IRMDA - Intelligent Risk Management for the Digital Age, Individual Project, NUP: POCI-01-0247-FEDER-038526, co-funded by the Incentive System for Research and Technological Development, from the Thematic Operational Program Competitiveness of the national framework program - Portugal2020.

REFERENCES

- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- Cook, D. (2016). *Practical machine learning with H2O: powerful, scalable techniques for deep learning and AI*. ” O’Reilly Media, Inc.”.
- Cortez, P. (2010). Data mining with neural networks and support vector machines using the r/rminer tool. In *Industrial Conference on Data Mining*, pages 572–583. Springer.
- Darwiche, A. (2018). Human-level intelligence or animal-like abilities? *Commun. ACM*, 61(10):56–67.
- Feurer, M., Springenberg, J. T., and Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.
- Gibert, K., Izquierdo, J., Sánchez-Marrè, M., Hamilton, S. H., Rodríguez-Roda, I., and Holmes, G. (2018). Which method to use? an assessment of data mining methods in environmental data science. *Environmental modelling & software*, 110:3–27.
- Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., and Vanschoren, J. (2019). An open source automl benchmark. *arXiv preprint arXiv:1907.00909*.
- Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., et al. (2019). Analysis of the automl challenge series 2015–2018. In *Automated Machine Learning*, pages 177–219. Springer.
- H2O (2019). Sparkling water. <http://docs.h2o.ai/sparkling-water/2.4/latest-stable/doc/index.html>.
- H2O.ai (2019). Automl: Automatic machine learning. <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html>.
- He, X., Zhao, K., and Chu, X. (2019). Automl: A survey of the state-of-the-art. *arXiv preprint arXiv:1908.00709*.
- Jin, H., Song, Q., and Hu, X. (2018). Auto-keras: Efficient neural architecture search with network morphism. *arXiv preprint arXiv:1806.10282*.
- Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1):826–830.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Peteiro-Barral, D. and Guijarro-Berdiñas, B. (2013). A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11.
- Salesforce (2019). TransmogriAI. <https://docs.transmogriai/en/stable/>.
- Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM.
- Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, B., and Farivar, R. (2019). Towards automated machine learning: Evaluation and comparison of automl approaches and tools. *arXiv preprint arXiv:1908.05557*.
- Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.