Performance evaluation of AutoML frameworks vs Conventional strategies

Heetika Gada (hg2532) and Vishruth Nair (vn2286)

Executive Summary

Designing architectures for machine learning networks can be challenging. Our goal is to understand and evaluate the AutoML tools and compare them against the traditional machine learning model building methods.

We have evaluated the efficacy of employing AutoML frameworks in a real world scenario and compared its results against the conventional strategy of a person of Data Science knowledge building the machine learning pipeline himself. We gain knowledge from the metrics and investigate the best method and tool to use for training any machine learning model.

Problem motivation

Ever since the advent of AutoML, the Machine Learning algorithms are developed on it and it is assumed that it outputs the best performance metrics for training. Such algorithms have been used in many integral applications like healthcare and medical. It is important to investigate the best tools and techniques for training machine learning models, specially for such applications. Depending on the application, the user may have limited computation resources such as time, money etc. The methods and tools for running machine learning algorithms are numerous and we question whether each technique is worth investigating in the first place.

Background work

Between 1995 to 2015, many Machine Learning Libraries were developed including Keras, Tensorflow etc. AutoML's initial effort came out of academia and ML practitioners, and then came the start ups. One of the first attempts was Auto-Weka from University of British Columbia in addition to Freiburg which utilizes algorithms provided by Weka. This led to the development of Auto-sklearn. TPOT was developed at UPenn. Finally, auto-ml was released in 2016 as a python open source package. After this breakthrough, the larger companies started to follow it, offering Automated Machine Learning as a Service. Google AutoML Tables, was introduced in 2017. Microsoft Azure followed suit with it's AzureML.

Technical challenges

There can be several challenges while optimizing the performance of the machine learning models: mainly in terms of resources and data. Each dataset is pre processed differently. Additionally, we do not know what is happening in the backend of AutoML. Some of the technical challenges we faced while computation. Some AutoML models accept only categorial, numerical or boolean data types, a few only go further to accept only numerical data type format. Google AutoML Tables does not pre-process the data; it does not recognize certain characters. Need for additional compatibility: H2O uses java, jre etc. H2O has a high RAM requirement which can cause the system to crash.

Approaches

We have looked into 4 autoML systems namely, Auto-sklearn (Feurer et al., 2019), Google Cloud AutoML (Google), H20 (H2O.ai, 2017) and TPot and build the system to tackle Classification problems. At each step of the machine learning pipeline we have focused on the metrics to look into while implementing an AutoML model and compare it against its corresponding counterpart in the conventional or traditional strategy.

Datasets used: Wine dataset, Bank marketing dataset, Spambase dataset

Traditional Methods:

We have built the Machine Learning models by tuning it by GridSearchCV for Random Forest, Gaussian Naive Bayes, Decision Tree and Logistic Regression. The machine learning pipeline follows the following steps: (Also as shown in Figure. 1)

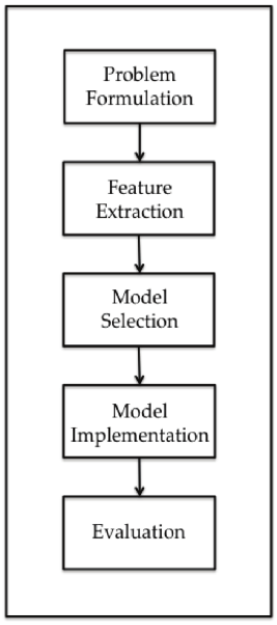

Fig. 1 Pipeline for building the Machine Learning model traditionally

Data Ingestion and Preparation: Manipulation of data is performed before it is ready for ingestion by the algorithms.

Feature Engineering Automation: Estimation of the ability of the system to explore all available features and discover and evaluate features is done.

Machine Learning Algorithm: Decision to decide which ML parameters are to be selected by GridSearchCV is performed.

Evaluation: The algorithm is evaluated on test data and we recorded the performance.

Google Tables:

Cloud AutoML is a suite of machine learning products that enables professionals with limited or

no machine learning expertise to train high-quality models specific to their business needs. AutoML Tables is a supervised learning service. It uses structured data to train ML models for any data that the user gives. We select one column as the target dataset in AutoML. This column can be changed to create any number of models with the analyzed training data. We can also customize the training data by changing different hyperparameters like training/testing data split, node training hours and optimization parameters (AUC ROC, AUC PR etc.)

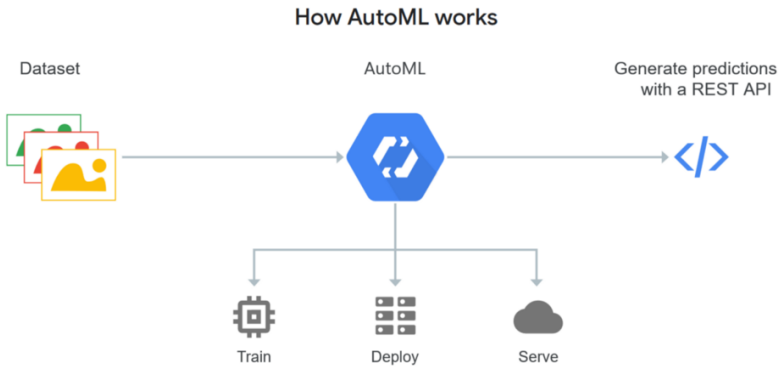

Fig. 2: Figure showing how AutoML works

We implemented AutoML on the 3 datasets mentioned above. In Fig. 3, we can see the results obtained from the Spambase Dataset. We can also check what parameters were used for the best performing model by viewing 'Google Logging' and record these parameters.

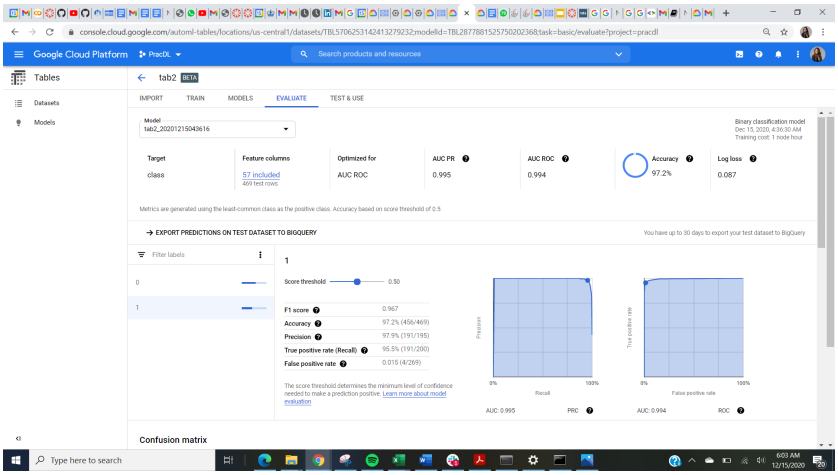

Fig. 3: Screenshot showing the results for Spambase dataset on AutoML

Auto - sklearn:

Auto-Sklearn is an open-source library for performing AutoML in Python. It makes use of the popular Scikit-Learn machine learning library for data transforms and machine learning algorithms. It frees the machine learning user from algorithm selection and hyperparameter tuning leveraging recent advantages in Bayesian optimization, meta-learning and ensemble construction.

Auto-sklearn learns from models that performed well on similar datasets and is able to automatically create an ensemble of top-performing models discovered as part of the optimization process. Another crucial feature is limiting the resources (memory and time) which the scikit-learn algorithms are allowed to use. Setting the resource limits is therefore a tradeoff between optimization time and the number of models that can be tested. Instead of using all available estimators, it is also possible to restrict auto-sklearn's search space.

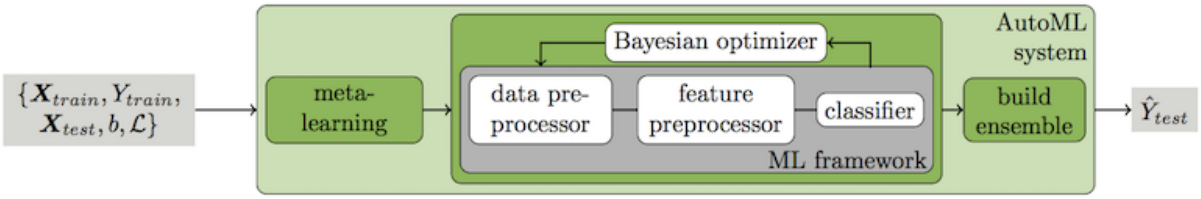

Fig 4: Architectural Diagram

The model takes in training data and the pipeline decides how to handle categorical features. It is to be noted that the only Valid Data types accepted are numerical, categorical or boolean Issue. After training the show_models function return a representation of the final ensemble found by auto-sklearn. sprint_statistics() can also be used to summarize the search and the performance of the final model. This helps alleviate the worry whether most algorithms have been taken into account or not

TPOT

TPOT provides a scikit-learn-like interface for use in Python, but can be called from the command line as well. It constructs machine learning pipelines of arbitrary length using scikit-learn algorithms and, optionally, xgboost. In its search, preprocessing and stacking are both considered. After the search, it is able to export python code so that you may reconstruct the pipeline without dependencies on TPOT.

While technically pipelines can be of any length, TPOT performs multi-objective optimization: it aims to keep the number of components in the pipeline small while optimizing the main metric. TPOT features support for sparse matrices, multiprocessing and custom pipeline components.

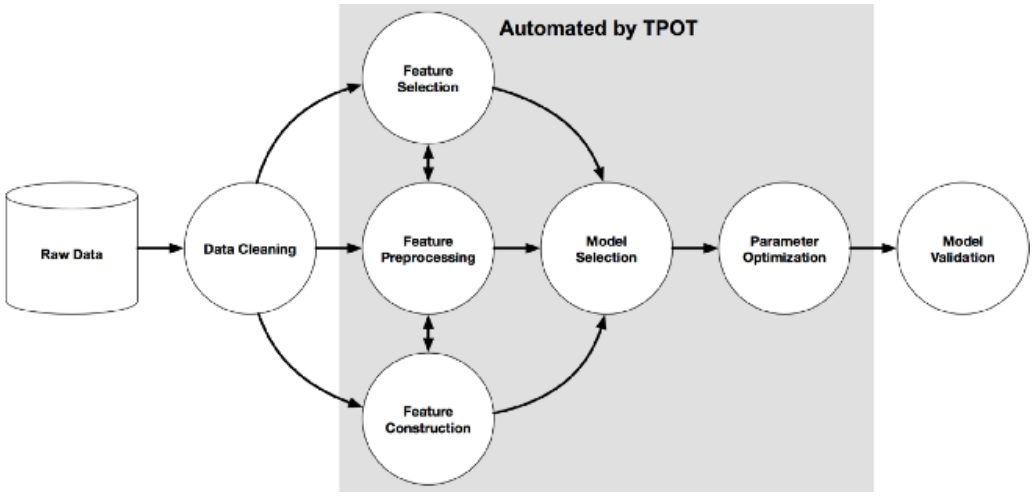

Fig. 4: TPOT architectural diagram

TPOT determines the best pipeline using genetic programming. The essential steps involve:

1. You have a population of possible solutions to a given problem and a fitness function. At every iteration, it evaluates how to fit each solution with your fitness function and selects the fittest ones and performs crossover to create a new population.
2. The mutate models are changed using some random modification and the process is repeated until you get the best solution.

A drawback of TPOT would be that it requires all data to be in numerical format and cannot perform the type conversion by itself.

H2O

H2O supports the most widely used statistical & machine learning algorithms, including gradient boosted machines, generalized linear models, deep learning, and many more.

It automates the process of building a large number of models, to find the best model without any prior knowledge or effort by the Data Scientist.

The H2O autoML framework performs:

1. Necessary data pre-processing steps(as in all H2O algorithms).
2. Trains a Random grid of algorithms like GBMs, DNNs, GLMs, etc. using a carefully chosen hyper-parameter space.
3. Individual models are tuned using cross-validation.
4. Two Stacked Ensembles are trained. One ensemble contains all the models (optimized for model performance), and the other ensemble provides just the best performing model from each algorithm class/family (optimized for production use).
5. Returns a sorted “Leaderboard” of all models.

The framework doe not have any limitations on pre processing and the dataset can be directly fed

to the framework. The user can choose from the leaderboard of models to use as his final model (By default the leader model is selected during predition).

Experimental Evaluation:

Google Table Results:

	Marketing Dataset	Spambase Dataset	Wine Dataset
Node Hour Used	2	1	N/A
Optimized for	AUC ROC	AUC ROC	N/A
AUC ROC	0.934	0.995	N/A
AUC PR	0.608	0.994	N/A
Accuracy	0.9041	0.972	N/A
Log Loss	0.199	0.087	N/A
F1 score	0.545	0.967	N/A

The detailed experimental evaluation of each framework can be found in the appendix. Table below shows the details on the dataset and its corresponding metric values.

Dataset	Instances	Features	Classes	Framework	Log Loss	Accuracy	Time Taken
Wine Quality	4898	12	10	Auto-sklearn	0.4268	0.6522	3602.58
				TPOT	0.2576	0.6644	1908.73
				H2O	0.3215	0.6588	2645.52
				Google table	~	~	~
Spambase	4601	58	2	Auto-sklearn	0.1882	0.9565	3624.87
				TPOT	0.123	0.9548	1726.09
				H2O	0.1089	0.96003	3086.9
				Google table	0.087	0.9723	3600
Bank-Marketing	45211	17	2	Auto-sklearn	0.2102	0.9181	3601.19
				TPOT	9.99E-16	1	7903.79
				H2O	0.1904	0.909	3048.9
				Google table	0.1999	0.9041	~7200

Conclusion:

All autoML models ease the technical requirements of the user, especially in the case of google tables where no programming experience is required. But it is observed to take a comparatively longer than the other autoML frameworks. TPOT works the best of the autoML frameworks but is still in its experimental phase and work needs to be done for it to accept more data types. H2O and auto-sklearn are more suitable for real world scenarios with auto-sklearn appropriate for quicker and less accurate time constraint results. H2O uses more state of the art machine learning optimisations. Deep Neural Networks in particular is still a very difficult task to automate, especially for a non-expert to tune properly.

Appendix:

The performance results of datasets corresponding to the autoML frameworks

Dataset 1- Wine Quality Dataset:

Auto-Sklearn

```
Framework:  auto-sklearn
log loss:  0.4268143501
accuracy:  0.6522448979591837
Classification report
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.67      0.22      0.33        37
           5       0.70      0.61      0.65       368
           6       0.61      0.82      0.70       544
           7       0.76      0.47      0.58       233
           8       0.83      0.24      0.38        41

    accuracy                           0.65      1225
   macro avg       0.59      0.39      0.44      1225
weighted avg       0.67      0.65      0.64      1225

total time elapsed:  3602.586087703705
```

TPOT

```
Framework:  tpot
log loss:  0.25763234
accuracy:  0.6644897959183673
Classification report
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/
re are ill-defined and being set to 0.0 in labels with
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       1.00      0.03      0.05        37
           5       0.73      0.60      0.66       368
           6       0.61      0.85      0.71       544
           7       0.76      0.50      0.60       233
           8       1.00      0.32      0.48        41

    accuracy                           0.66      1225
   macro avg       0.68      0.38      0.42      1225
weighted avg       0.70      0.66      0.65      1225

total time elapsed:  1908.7302021980286
```

Best pipeline: KNeighborsClassifier(RobustScaler(input_matrix), n_neighbors=70, p=1,

weights=distance)

H2O

Leaderboard

model_id	mean_per_class_error	logloss	rmse	mse	training_time_ms	predict_time_per_row_ms
GBM_grid__1_AutoML_20201216_020906_model_15	0.629892	0.974609	0.536878	0.288238	1341	0.145039
XGBoost_grid__1_AutoML_20201216_020906_model_5	0.630002	1.02758	0.540866	0.292536	927	0.026143
GBM_3_AutoML_20201216_020906	0.640258	0.936908	0.53719	0.288573	982	0.130451
GBM_grid__1_AutoML_20201216_020906_model_22	0.640654	0.966226	0.52975	0.280635	1623	0.221353
StackedEnsemble_BestOfFamily_AutoML_20201216_020906	0.641232	0.869254	0.536349	0.28767	21755	0.148778
XRT_1_AutoML_20201216_020906	0.642959	1.29546	0.544668	0.296664	1193	0.098868
DRF_1_AutoML_20201216_020906	0.643066	1.32428	0.544484	0.296463	735	0.086306
GBM_4_AutoML_20201216_020906	0.649275	0.934342	0.533898	0.285047	1887	0.131289
GBM_grid__1_AutoML_20201216_020906_model_17	0.649514	0.954645	0.541141	0.292834	1633	0.189937
GBM_grid__1_AutoML_20201216_020906_model_24	0.650336	0.956071	0.536499	0.287831	1413	0.181271

```
Framework:  H2O
log loss:  0.32154732
accuracy:  0.6587755102040816
Classification report
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.53      0.22      0.31        37
           5       0.70      0.65      0.67       368
           6       0.64      0.78      0.70       544
           7       0.68      0.51      0.58       233
           8       0.64      0.34      0.44        41

    accuracy                           0.66      1225
   macro avg       0.53      0.42      0.45      1225
weighted avg       0.66      0.66      0.65      1225

total time elapsed:  2645.5268199443817
```

Dataset 2 - Spambase Dataset

Auto-sklearn

```
Framework:  auto-sklearn
log loss:  0.18817840144885561
accuracy:  0.9565595134665508
Classification report
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       701
           1       0.97      0.92      0.94       450

    accuracy                           0.96      1151
   macro avg       0.96      0.95      0.95      1151
weighted avg       0.96      0.96      0.96      1151

total time elapsed:  3624.878770828247
```

TPOT

```
Framework:  tpot
log loss:  0.12303342451402823
accuracy:  0.9548218940052129
Classification report
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       701
           1       0.95      0.94      0.94       450

    accuracy                           0.95      1151
   macro avg       0.95      0.95      0.95      1151
weighted avg       0.95      0.95      0.95      1151

total time elapsed:  1726.0997183322906
```

Best pipeline: GradientBoostingClassifier (BernoulliNB(input_matrix, alpha=0.01, fit_prior=True), learning_rate=0.1, max_depth=6, max_features=0.5, min_samples_leaf=15, min_samples_split=5, n_estimators=100, subsample=0.7500000000000001)

H2O

model_id	auc	logloss	aucpr	mean_per_class_error	rmse	mse	training_time_ms	predict_time_per_row_ms
GBM_grid__1_AutoML_20201216_033155_model_32	0.9883	0.130907	0.981757	0.0502501	0.190995	0.0364791	1880	0.04003

```
Framework:  H2O
log loss:  0.10893043400624643
accuracy:  0.9600347523892268
Classification report
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       701
           1       0.94      0.96      0.95       450

    accuracy                           0.96      1151
   macro avg       0.96      0.96      0.96      1151
weighted avg       0.96      0.96      0.96      1151

total time elapsed:  3086.9009573459625
```

Dataset 3 - Bank Marketing

Auto-sklearn

```
Framework:  auto-sklearn
log loss:  0.2102055772435655
accuracy:  0.9181314946100806
Classification report
              precision    recall  f1-score   support

          no       0.94      0.97      0.95      9132
         yes       0.68      0.52      0.59      1165

    accuracy                           0.92     10297
   macro avg       0.81      0.75      0.77     10297
weighted avg       0.91      0.92      0.91     10297
```

TPOT

```
log loss:  9.992007221626413e-16
accuracy:  1.0
Classification report
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      8233
         1.0       1.00      1.00      1.00      2064

    accuracy                           1.00     10297
   macro avg       1.00      1.00      1.00     10297
weighted avg       1.00      1.00      1.00     10297

total time elapsed:  7903.794376850128
```

Best pipeline: DecisionTreeClassifier(input_matrix, criterion=gini, max_depth=6, min_samples_leaf=20, min_samples_split=14)

H2O

model_id	auc	logloss	aucpr	mean_per_class_error	rmse	mse	training_time_ms	predict_time_per_row_ms
ckedEnsemble_BestOfFamily_AutoML_20201216_070656	0.948524	0.192412	0.664305	0.141878	0.241443	0.0582947	1592	0.065912

```
Framework:  H2O
log loss:  0.1904871474818192
accuracy:  0.9090026221229485
Classification report
              precision    recall  f1-score   support

           0       0.97      0.92      0.95      9132
           1       0.57      0.80      0.66      1165

    accuracy                           0.91     10297
   macro avg       0.77      0.86      0.81     10297
weighted avg       0.93      0.91      0.92     10297

total time elapsed:  3048.9236533641815
```

Traditional Methods Results:

Wine quality Dataset:

	Accuracy Score	Precision Score	Recall Score	ROC AUC score
Gaussian Naive Bayes model on training set	0.9805	0.9969	0.9828	0.9562
Gaussian Naive Bayes model on test set	0.9685	0.9806	0.9776	0.9591
Decision Tree Classifier on training set	1	1	1	1
Decision Tree Classifier on test set	0.9854	0.9868	0.9939	0.9766
Logistic Regression on training set	0.9581	0.9581	1	0.5
Logistic Regression on testing set	0.7538	0.7538	1	0.5
Random Forest on Training set	1	1	1	1
Random Forest on test set	0.9185	0.9065	0.996	0.8319

Bank Marketing Dataset:

	Accuracy Score	Precision Score	Recall Score	ROC AUC score
Gaussian Naive Bayes model on training set	0.7176	0.7388	0.673	0.7176
Gaussian Naive Bayes model on test set	0.7586	0.2714	0.6789	0.7238
Decision Tree Classifier on training set	0.7483	0.8234	0.6321	0.7483
Decision Tree Classifier on test set	0.8383	0.3692	0.6142	0.7405
Logistic Regression on training set	0.7273	0.7535	0.6755	0.7273
Logistic Regression on testing set	0.7686	0.2823	0.6832	0.7313
Random Forest on Training set	0.7427	0.7928	0.657	0.7427
Random Forest on test set	0.8178	0.3391	0.6509	0.7449

Spambase Dataset:

	Accuracy Score	Precision Score	Recall Score	ROC AUC score
Gaussian Naive Bayes model on training set	0.9428	0.9794	0.9587	0.468
Gaussian Naive Bayes model on test set	0.8132	0.6962	0.9697	0.8406
Decision Tree Classifier on training set	0.9827	0.9818	1	0.8783
Decision Tree Classifier on test set	0.8534	0.7327	0.989	0.8771
Logistic Regression on training set	0.9291	0.9291	1	0.5
Logistic Regression on testing set	0.3941	0.3941	1	0.5
Random Forest on Training set	0.962	0.9607	1	0.7321
Random Forest on test set	0.6775	0.5502	0.9972	0.7334