How consistent are the implementations of machine learning algorithms in different ML libraries?

Albert Millan
Trinity College Dublin
Dublin, Ireland
millana@tcd.ie

Jordan Myers
Trinity College Dublin
Dublin, Ireland
myersjo@tcd.ie

James Sherlock
Trinity College Dublin
Dublin, Ireland
jsherloc@tcd.ie

1 INTRODUCTION

The number of Machine Learning (ML) frameworks/libraries has grown exponentially in recent years. Academics and organisations have more options than ever before when it comes to selecting a framework to implement a given ML algorithm. These frameworks, however, have been implemented using different design paradigms and developers, potentially resulting in performance variations of the algorithm across the frameworks. The aim of this paper is twofold: (a) determine the extent to which an algorithm's performance differs from one framework to the other; (b) asses whether this performance variation is consistent across machine learning algorithms.

2 RELATED WORK

Benchmarking the performance of ML algorithms across different frameworks is an area of the literature widely studied by academics. Most studies have focused on comparing the runtime and optimization of specific ML algorithms across frameworks [5–7], but little attention has been payed to the consistency of the results of the algorithms across the frameworks.

Balaji and Allen conducted a benchmark study of the results of open source automatic machine learning (AML) solutions [2]. Auto-sklearn, TPOT, auto_ml and H2Os AML solutions, all of which use Sci-kit, were tested against a compiled set of 30 regression and 57 classification datasets sourced from OpenML. Mean squared error (MSE) and weighted F1 were the set optimization metrics for regression and classification datasets test cases respectively. They show that auto-sklearn provides more accurate results for classification algorithms, having an average weighted F1 score of 0.75, while TPOT presents the best results for regression algorithms, having an inverted MSE of 0.9. Their experiment also supports the fact that different implementations of algorithms can yield different results even when the same toolkit (Sci-kit) is used.

3 METHODOLOGY

3.1 Dataset

The dataset chosen is from the US Bureaus of Transportation Statistics concerning Airline On-Time Performance Data for July 2018 [1]. It contains data for 701,875 flights and has 119 features, with ARR_DELAY indicating the delay time in minutes. The histogram in **Figure 1** shows the distribution of the arrivals, suggesting that the majority of them arrive on-time. A negative delay indicates an early arrival time. It is common practice in the aviation industry for airlines to overestimate flight times, and so advertise later arrival times than what one might expect. The reasons for this are not relevant to this study and we simply consider early arrivals as 'on-time'.

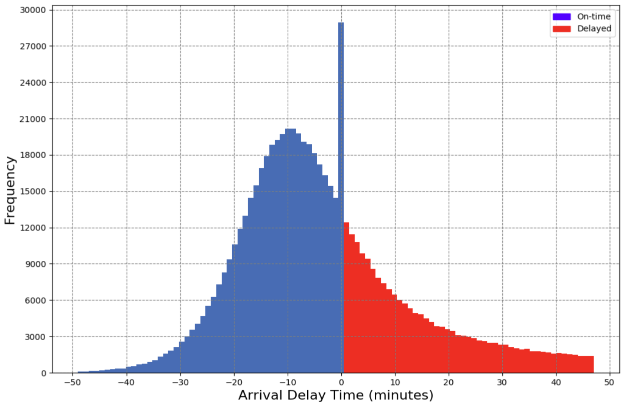

Figure 1: Frequency of Arrival Delays.

3.2 Data Processing

For regression algorithms, we predict the arrival delay of a flight in minutes and for classification algorithms, we predict whether or not a flight would arrive late. We selected features that are strongly correlated with the target and weakly correlated with each other. **Figure 2** shows the correlation between the features of the dataset.

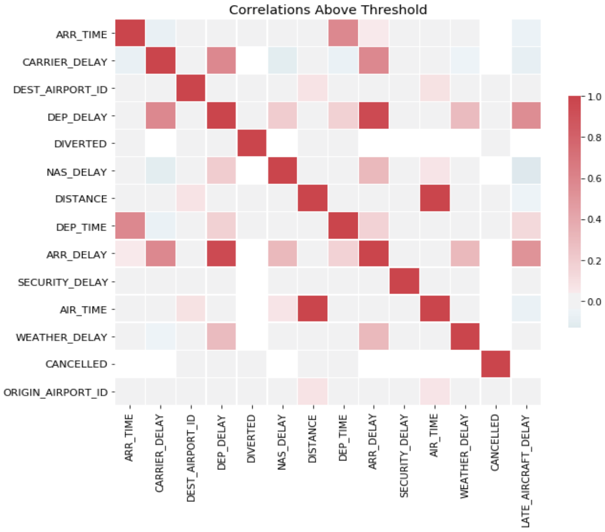

Figure 2: Correlation between the features of the dataset.

We reduced the features used to those in Table 1 and 2.

Table 1: Feature Description: Regression

Feature	Value
DEP_DELAY	Difference between departure time and expected departure time in minutes.
CARRIER_DELAY	Delay in minutes attributable to the carrier (late passengers, late loading bags).
WEATHER_DELAY	Delay in minutes attributable to unsuitable weather conditions.
NAS_DELAY	Delay in minutes attributable to National Air System.
LATE_AIRCRAFT_DELAY	Delay in minutes attributable to a delay from an aircraft's previous journey.
ARR_DELAY	Difference between arrival time and expected arrival time in minutes.

Table 2: Feature Description: Classification

Feature	Value
DEP_DELAY	Difference between departure time and expected departure time in minutes.
CARRIER_DELAY	Delay in minutes attributable to the carrier (late passengers, late loading bags).
WEATHER_DELAY	Delay in minutes attributable to unsuitable weather conditions.
NAS_DELAY	Delay in minutes attributable to National Air System.
LATE_AIRCRAFT_DELAY	Delay in minutes attributable to a delay from an aircraft's previous journey.
ARR_DELAY_BIN	Classification as to whether a flight is delayed or not delayed.

The data was further processed by filling the empty values from 'DEP_DELAY' and 'ARR_DELAY' with zeros. These were the occurrences where planes departed/arrived at the right time (neither early nor delayed) and were originally empty in the dataset. Duplicate tuples have been removed. We classified outliers as those data points that did not lie within the lower and upper limits of a boxplot. These were subsequently removed.

$$UpperLimit = Q3 + 1.5 * IQR$$
$$LowerLimit = Q1 - 1.5 * IQR$$

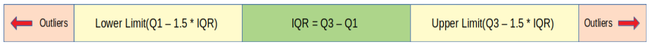

Figure 3: Outlier detection using quartiles Q1 and Q3, and interquartile range IQR.

3.3 Validation

We trained and tested the data using 10-fold cross validation. The average of the results of the folds has been computed.

3.4 Algorithms

The implemented classification algorithms aim to predict whether a given flight will arrive late to destination, whereas regression algorithms predict the arrival delay time.

3.4.1 Linear Regression. Ordinary least squares was used to fit the line.

3.4.2 Neural Network Classification. Min-max normalization was used to scale the data. Table 3 shows the parameters used.

Table 3: Parameters for Neural Network Classification

Parameter	Value
Hidden Layer Size	4
Learning Rate	0.001
Max Iterations	100
Momentum	0.9

3.4.3 Logistic Regression. A c value of 0.001 and an optimisation tolerance of 1E-07 were used with a liblinear solver and an L1 regularisation weight of 1.

3.5 Evaluation Metrics

The algorithms were compared using a variety of evaluation metrics. The combination of these metrics provide a better indication of the behaviour of the algorithm. For instance, a researcher striving to obtain the best performance might conclude that two algorithms have the same performance if they produce equal results for the selected evaluation metric e.g. recall. However, these same algorithms might vary significantly in terms of accuracy, precision, etc., in turn, influencing the overall performance of the algorithm. Therefore, an efficient comparison of the algorithms must evaluate the different metrics that define its behavior/overall performance.

Classification results were evaluated using accuracy, precision and recall. Regression results were evaluated using the Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and the Coefficient of Determination (CoD).

3.5.1 Accuracy: shows how many classifications were correctly predicted. This can be affected when the distribution of classes is skewed. As can be seen in Figure 1, our dataset is split 40/60 between delayed and not delayed respectively. This may affect the accuracy but, as we are comparing metrics across platforms rather than evaluating the models themselves, it will not affect our results.

$$Accuracy = \frac{TruePositives + TrueNegatives}{AllPredictions}$$

3.5.2 Precision: shows how many positive classifications were correct. In our case, "how many flights predicted to be delayed arriving were actually delayed arriving?".

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

3.5.3 Recall: shows how many of the positive classes were correctly predicted. [3] While recall alone has little meaning, we believe it is sufficient to use in our case for comparing algorithm implementations.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

3.5.4 Mean Absolute Error: shows the average error between prediction and classification [3].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

3.5.5 Root Mean Square Error: shows the standard deviation of errors. It is sensitive to outliers, so we have remove outliers as described in section 3.1 previously.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i-1}^{n}e_i^2}$$

3.5.6 Coefficient of Determination: shows the goodness of fit of our model [4]. It uses a worst-case model, which always predicts the mean of the observed responses of the dependent variable Y as the value of Y, as a baseline.

$$CoD = 1 - \frac{SSE}{SST}$$

where SSE is the sum of squared errors of our regression model and SST is the sum of squared error of the baseline model.

4 RESULTS & DISCUSSION

The experiment shows that the results are consistent across the linear regression, logistic regression and neural network classification algorithms. Overall, Tensorflow performs slightly worse than scikit-learn and AzureML in each of the implementations of the aforementioned algorithms. Furthermore, results of the scikit-learn and AzureML implementations are similar across the algorithms.

In linear regression, performance is very similar amongst all the platforms for recall and accuracy, consistent to within approximately 5% as shown in as shown in **Table 4** and **Figures 4 & 5**. The Coefficient of Determination for Tensorflow, however, is significantly lower than Scikit-Learn and Azure ML, as shown in **Figure 6**. One possible explanation for this could be that the baseline model used by Tensorflow is performing better than those used in the other two platforms. As a result, the performance difference between the regression model and the base model in Tensorflow may not be as significant, and so is giving a lower CoD ratio.

Table 4: Comparison of Linear Regression Implementations

Platform	MAE	RMSE	CoD
Tensorflow	7.90	10.28	0.595
Scikit-Learn	7.58	9.89	0.963
Azure ML	7.54	9.86	0.961

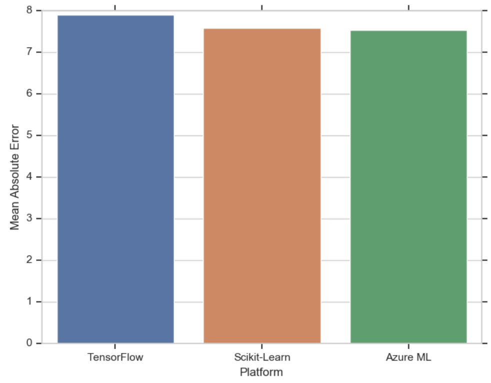

Figure 4: Comparison of Mean Absolute Error for Linear Regression

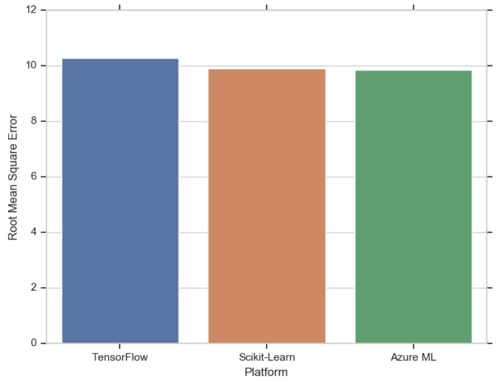

Figure 5: Comparison of Root Mean Square Error for Linear Regression

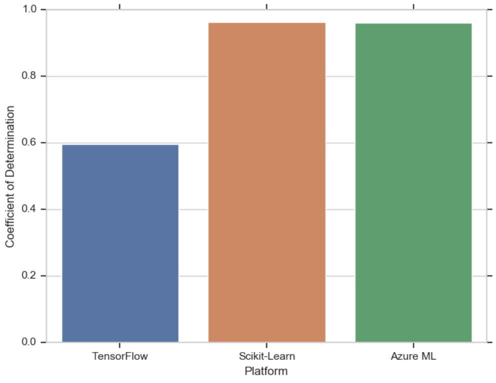

Figure 6: Comparison of Coefficient of Determination for Linear Regression

In the classification algorithms, the results are similar. The accuracy and precision metrics are higher in both scikit-learn and AzureML than in Tensorflow for both logistic regression and neural network classification, as shown in **Tables 5 & 6 and Figures 7 & 8**. It is surprising that Tensorflow poses a better recall rate for the classification algorithms as it is the most appropriate metric for the given dataset i.e number of flights that were delayed that were effectively predicted as delayed. Even though the results for the recall do not support our conclusion, the results for the accuracy and precision outweight these results.

Table 5: Comparison of Neural Network Classification Implementations

Platform	Accuracy	Precision	Recall
Tensorflow	0.792	0.764	0.78
Scikit-Learn	0.86	0.92	0.713
Azure ML	0.864	0.939	0.697

Table 6: Comparison of Logistic Regression Implementations

Platform	Accuracy	Precision	Recall
Tensorflow	0.812	0.695	0.729
Scikit-Learn	0.86	0.93	0.701
Azure ML	0.848	0.983	0.691

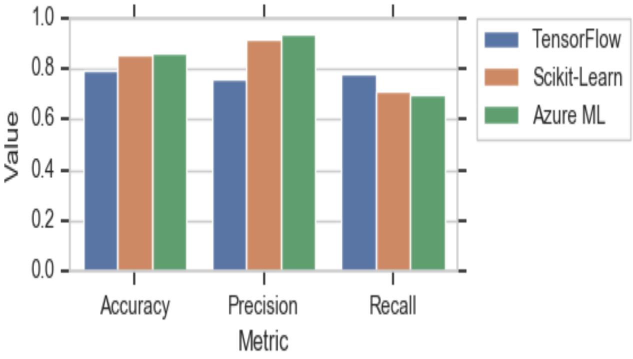

Figure 7: Comparison of Results for Neural Network Classification

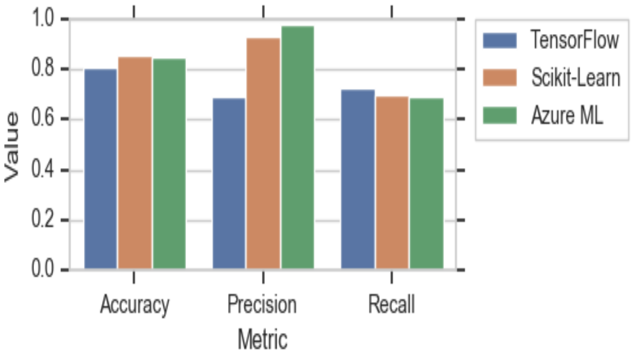

Figure 8: Comparison of Results for Logistic Regression

5 LIMITATIONS & OUTLOOK

Only a single dataset was used in our comparison. Future work could use additional datasets to reinforce the validity of the results by ensuring the dataset itself is not influencing the result.

We only used evaluation metrics to compare the implementations of algorithms. Future work comparing platforms could look at other aspects, such as efficiency/run time and flexibility/configurability of the implementations.

REFERENCES

[1] Airline On-Time Performance Data 2018. Airline On-Time Performance Data (Marketing Carrier). Retrieved October 28, 2018 from https://transtats.bts.gov/Tables.asp?DB_ID=120&DB_Name=Airline%20On-Time%20Performance%20Data&DB_Short_Name=On-Time

[2] A. Balaji and A. Allen. 2018. Benchmarking Automatic Machine Learning Frameworks. *ArXiv e-prints* (Aug. 2018). arXiv:1808.06492

[3] J. Beel. 2018. CS4404 Week 05 and 06: Machine Learning Training and Evaluation.

[4] Coefficient of Determination (R-squared) Explained 2018. Coefficient of Determination (R-squared) Explained. Retrieved December 17, 2018 from https://towardsdatascience.com/coefficient-of-determination-r-squared-explained-db32700d924e

[5] DL Framework Speed Comparison 2018. Deep Learning Frameworks Speed Comparison. Retrieved Octorber 26, 2018 from https://wrosinski.github.io/deep-learning-frameworks/

[6] L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin, and Q. Zhang. 2018. Benchmarking Deep Learning Frameworks: Design Considerations, Metrics and Beyond. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS).* 1258–1269. https://doi.org/10.1109/ICDCS.2018.00125

[7] S. Shi, Q. Wang, P. Xu, and X. Chu. 2016. Benchmarking State-of-the-Art Deep Learning Software Tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD).* 99–104. https://doi.org/10.1109/CCBD.2016.029