Using Microsoft Azure as a Machine Learning Service

Cheng-Yin Eng `chengyineng@umass.edu`
Katie House `khouse@`
Shanu Vashishtha `svashishtha@`
Deeksha Razdan `drazdan@`

Abstract

Azure Machine Learning service provides a cloud-based environment data scientists can use to pre-process data, train, test, and deploy models, and also track various runs of machine learning model experiments. Such service allows more efficient workflow, consistent and scalable computing environment, and also reduces the financial cost of purchasing and maintaining expensive hardware that might not be used often. Since the service currently lacks comprehensive documentation for user tasks, our group's goal is to: (1) be acquainted with cloud computing concepts; and (2) produce user-friendly, end-to-end data science notebook tutorials that are accessible to general data science audience, utilizing the advantages of cloud computing features. In this work, we discuss our experience of using Azure to complete a suite of common data science tasks and present Azure's performance in terms of speed, accuracy and ease of open-source code compatibility. We also present a comparison with other available platforms and libraries used today to perform these tasks.

1 Introduction

Machine Learning as a Service (MLaaS) is an umbrella definition of various cloud-based platforms that cover most infrastructure issues such as data pre-processing, model training, model evaluation, and model prediction, which can be connected to APIs. Amazon Machine Learning services, Azure Machine Learning, Google Cloud AI, and IBM Watson are four leading cloud MLaaS services. MLaaS promises faster model training and deployment, lower upfront cost (since companies only pay for the resources that they use), consistent access to data and models regardless of the platforms data scientists are using, and lastly a scalable computing environment.

Microsoft Azure itself claims to offer several distinctive advantages that no other competitors offer. For instance, Azure has the ability to use mixed compute resources and affords the ease in integrating with existing data science scripts. The former allows flexibility in coordinating multiple pipelines across heterogeneous computing resources. For instance, data scientists can run individual pipeline steps on various compute targets, such as HDInsight, GPU Data Science VMs, and Databricks; this makes efficient use of available compute options. The latter advantage in open-source integration opens up opportunities to use existing scripts readily on the platform, with only the addition of a wrapper to create a workspace and to manage different experiment runs in one portal.

To properly take advantage of and ultimately evaluate Azure features, it is critical that we become proficient in using the Azure platform.

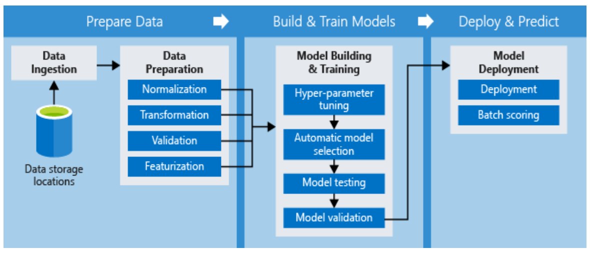

Figure 1: Azure Machine Learning Pipeline

Figure 1 above captures a snapshot of the Azure features that we would like to experiment with (further elaboration can be found in the Methods and Experiments section). To demonstrate the strengths and weaknesses of Azure Machine Learning service, it is essential for us to build different types of machine learning models. In the following sections to come, we will first describe our suite of data science challenges and then provide our evaluations of the Azure platform.

2 Related Work

We came across AWS SageMaker notebooks (Labs, 2019) which demonstrate their platform use to solve data science problems. These notebooks are detailed in their approach to solving a data science problem when a user adapts to their platform. In comparison, the Azure ML platform lacks a sufficiently detailed repository of notebooks that makes it easy for a user to transition to their platform (Learning, 2019). The notebooks reflect their internal organization structure rather than telling a user how to accomplish a given task. In addition, their notebooks cover a small subset of problems that are being tackled on a daily basis by data scientists all over the world.

Our work is novel because we will be explaining the features in depth which even a beginner can quickly adapt to. We will also be providing a demonstration of a wide array of data science problems like model explainability, machine translation, image classification and so on.

3 Problem Definition

Our overall goals are as such:

(a) Become proficient in Azure Machine Learning features

(b) Address gaps in currently existing Azure documentation, through identifying bugs and giving feedback to the Azure team

(c) Create sample end-to-end data science notebooks that encompass a wide array of tasks, including cleaning the data, training models, and deploying models

(d) Evaluate Azure Machine Learning Services against existing tools or packages

4 Methods and Experiments

To demonstrate the strengths and weaknesses of Azure Machine Learning service, it is essential for us to build different types of machine learning models. (House, 2019)

4.1 Experiment 1: A Simple End-to-end Classification Notebook

We initiated our investigation of Microsoft Azure's features by creating a simple end-to-end classification notebook that implemented the AML functionality. A classification task was performed to distinguish between commercial blocks and TV news. The commercial block data is available online through the UCI Machine Learning Repository (A. Vyas and Guha, 2017). The feature set comprises 124 quantitative video broadcast statistics such as shot length and motion distribution. The binary training and testing labels are commercial (+1) and non-commercial (-1).

Automated Machine Learning iterates through various high-performing machine learning models and automatically selects the best model based on a user-defined loss function. We compared the performance of the best AML model with three simple classification models, K-Nearest Neighbors, Neural Networks, and Random Forest, using default parameters in the Scikit Learn API (Buitinck et al., 2013) in Python.

4.2 Experiment 2: Training Neural Network on CIFAR-10 Dataset

We further performed classification task on a more complex dataset. In this experiment, we ran a two layer fully connected network on the CIFAR-10 dataset. The base *learning rate* gave a classification accuracy of 42%. We performed a hyperparameter optimization for this simple model using the platform's inbuilt feature. Also, we investigated the training time for our model on the platform and our local machine while trying different values on the hyperparameter *hidden layer size*. To investigate whether there is a difference in the final accuracy values while using different hyperparameter optimization libraries, we compared the platform's feature with the existing *hyperopt* library. We also collaborated with an in-class team working on a separate project. The idea was to perform hyperparameter optimization for their model and see if we are able to help them derive benefits from the platform as well.

4.3 Experiment 3a: Incorporating Model Explainability using Contrib library

We explore the realm of customer retention. Here we have Telecommunication Churn Dataset. The dataset has around 3333 records with 21 columns one of which corresponds to whether the person has churned (left the plan) or not. The aim is to predict whether or not a customer is going to churn, and to use model explanation to understand why a customer left. All of this is done improve customer retention for the company.

Table 1: Summary Results of Experiments

Metric	Commercial Block Classification	CIFAR-10	Online News Popularity
Accuracy on Azure	91.7	58.69	68.9% (for 5 classes)
Local Accuracy	88.4	57.85	- [a]
Accuracy Changes	3.3	0.84	- [a]

[a]Not applicable, since the analysis is done on only Azure

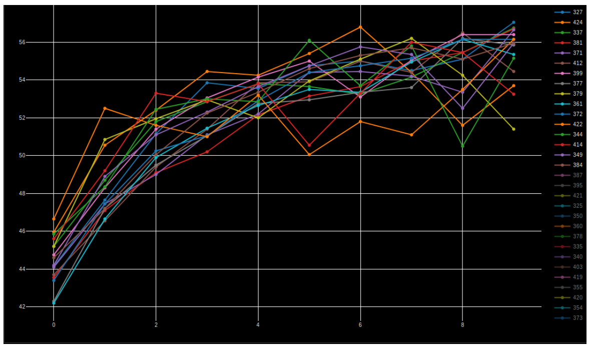

Figure 2: Hyperdrive run visualization on the Azure platform. Horizontal axis displays the Epochs while Vertical axis displays the validation accuracy during the training for best runs

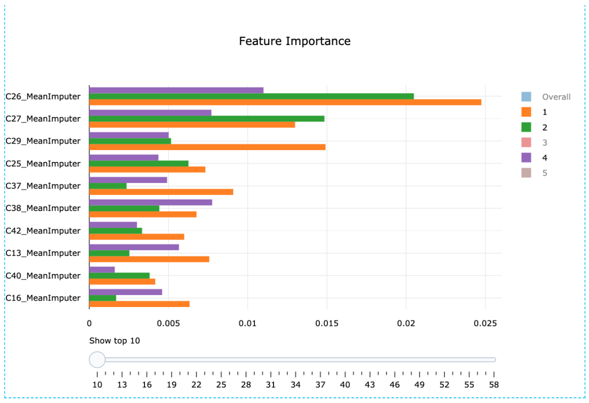

Figure 3: Model explainability graph generated automaticallyon the portal

The idea is to first analyse the data, see what story it tells you. Then using autoML we used the best model to predict whether or not a customer has churned. After which we use the explanation module to understand why the model thinks that a particular customer will leave/not leave.

This coupled with the data insights, gives a complete, end-to-end analysis of the problem statement and how to improve the customer retention.

4.4 Experiment 3b: Incorporating Model Explainability using AutoML Explainer

In this online news popularity dataset on Mashable, there are 60 features in total and slightly under 40,000 records. The goal of the model is to predict five different levels of popularity of an article, specifically defined by the number of shares in social networks. It was extremely easy to use the $AutoMLExplainer$ package that is built within the $AutoML$ package. An array of suitable models was selected automatically during different experiment iterations. For evaluation, various graph outputs with relevant metrics and model explanability graph were also generated automatically (Figure 3) and are available to be retrieved from the portal.

4.5 Experiment 4: Machine Translation

This machine translation experiment is a course project in CS690D: Deep Learning for Natural Language Processing. The goal is to translate 200k German sentences into English. The current translation method is to use a BERT tokenizer with Transformer model. This experiment will also make use of Azure's hyperparameter optimization functionality to improve BLEU (bilingual evaluation understudy) score. This experiment is a testament to the ease of incorporating with existing codes. Little changes were needed to run the experiment on Azure. The actual transformer model was not able to finish due to memory issues, but it is worth noting that Google Colab crashed right away when the model training process started.

4.6 Experiment 5: DeepCheck

In order to solve a real-world problem with Microsoft Azure, the team combated a real-world issue: gun violence in the United States. In a hackathon called TechTogether Boston, our team member implemented an end-to-end machine learning model that aimed to improve gun background checks, called DeepCheck. The purpose of DeepCheck was to better inform the FBI and gun owners of the risk of individuals purchasing firearm by analyzing their public Twitter accounts for hate speech and high frequencies of offensive language.

To create this model, the team implemented a paper that formed a labeled dataset of tweets that could be classified as either hatespeech, offensive language, or neutral (Davidson et al., 2017). This is a randomly sampled dataset of 25,000 from over 85.4 million tweets. Then, the team used the paper's feature engineering approach to build a classifier with Azure Automated Machine Learning feature. Additionally, the team was able to deploy a web app on Azure.

As a result, the hackathon team won three awards at the hackathon, including: "Best Use of Human-Centered AI", "Azure Champ Challenge", and "Policy Makers and Citizens Political Polarization Challenge." The DeepCheck repository can be found at `www.github.com/katiehouse3/deep-check` and the hackathon submission is public at `https://devpost.com/software/deepcheck-dphtv6`.

5 Evaluation

Our final deliverable is a series of user-friendly notebooks that use Azure Machine Learning Services to solve real-world data science problems. To evaluate the usability of Microsoft Azure, we collaborated with other data science teams and compared Azure with existing platforms.

5.1 Collaborating with another Data Science Team

With the idea of making our classmates undertaking other Data Science projects use the advantages that Azure has to offer, we collaborated with the team undertaking the project *Probabilistic Embeddings on Taxonomies* in collaboration with Google. They had come up with a model where they had to manually perform the hyperparameter tuning for five different values - learning_rate, w1, w2, r1 and embedding_dimension.

We took their existing code and were able to run their entire script on the Azure platform. In the process, we had to modify three lines of code where they were naming the files to be written relative to the root directory. We also had to add the logging part in the code which is 2 more lines. Compared to the overall code, this was less than 1% of the code lines modified. The team appreciated the overall hyperparameter tuning visualizations. However, when it came to final results, we weren't able to perform better than their KL loss score of 0.0017. The best value that Azure's automated search provided was 0.0038 which was close but, could have been better.

5.2 Comparison with Google Cloud Platform

One method of evaluating the performance of Microsoft Azure is comparing it to other platforms. Although our notebooks showcase benefits of using Azure Machine Learning Services, these benefits may be similar or better in other platforms. We decided to use Google Cloud Platform as our comparison technology due to it's prevalence in industry and $300 free trial credits.

The Azure team performed the same data science task as discussed in Section 4.1 using the Google Cloud Automated Machine Learning framework. Our first impression of the Google Cloud Platform (GCP) user interface was positive. The team liked the straight-forward and clean user interface compared to the Azure workspace.

However, one large downside with GCP was the time to complete the data import and modeling process. It took over 1 hour to upload 16.9 MB of training data. Furthermore, it took over 2 hours to train the automated machine learning model on GCP. Doing this modeling process on Azure only took 20 minutes from end-to-end. Overall, the initial comparison of GCP indicated that Azure was a more efficient platform for this data science task.

6 Results

Overall, we are pleased with the ease of open-source integration. Azure platform integrates with existing data science scripts very well and we only needed to make minimal changes to run the scripts on the platform.

Table 1 provides metric results associated with each experiment. We further break down the benefits and limitations of Azure below.

6.1 Benefits

6.1.1 *Auto Machine Learning improves accuracy*

By comparing accuracy results of standard Scikit Learn classification techniques with the AML model, we were better able to understand the benefits of using Microsoft Azure AML. According to our models, the maximum F1 Accuracy score was 88%. However, using the Microsoft Azure AML feature, we instantly brought our accuracy up to 92%. In future work, we will experiment with more AML models, standardize the format of this notebook and add more visuals in order to publish it to the Microsoft Azure Machine Learning Services repository.

6.1.2 *Hyperparameter optimization*

The portal's hyperparameter optimization feature was another benefit we came across. Fig. 2 represents the hyperdrive experiment which resulted in an almost 16% classification accuracy improvement from the original model. The parameter was sampled randomly from a uniform distribution. A total of 100 runs were created with a bandit termination policy that ended 14 runs which were performing poorly to save resources. The key take away from this study was that apart from obtaining the best hyperparameters for a model, one also gets a visualization of all the runs to perform an epoch wise comparison with few code lines being added to the original script.

6.1.3 *Improves training time*

To compare the training time, we performed a hyperparameter optimization (for the hidden layer size) on the portal and locally on a Dell XPS 15 with Intel Core i7-8750H CPU @ 2.20GHz 12 processor, 16 GB RAM and GeForce GTX 1050 Ti with Max-Q Design/PCIe/SSE2 Graphics card. Our observations are reported in Tab. 2. We could not run the 5 hyperparameter tuning runs parallely because of insufficient CUDA memory on our local machine. However, we didn't have any issues in running them at the same time on the portal. All of them ran parallely. By CPU for the azure platform, we mean the cpucluster and by GPU we mean the gpucluster which have been allotted to us for this project. Usually, these resources are dependent on the type of subscription availed. We observe that when just CPU is used for training, Azure consistently takes less time compared to our laptop. It is worthwhile noting here that the laptop being used is a high-end one. In case of GPU, the training time observed is lower for the laptop but, the limitation is of the GPU memory. We could run only 4 threads simultaneously while the platform didn't have an issue running all 5. It must be noted here that in our previous hyperdrive run to find optimal learning rate, 10 parallel runs were spawned on the platform at any given instant. Hence, we can say that although local laptops might be a good option for trying few initial runs, the Azure platform is a better choice when one is planning to scale up.

6.1.4 *Model explainability searches through different explanation algorithms*

Using the MLI SDK, researchers, data scientists, and machine learning practitioners can explain machine learning models using the state-of-art methods in an easy-to-use and scalable fashion.

Azure leverages proven third-party libraries (SHAP, LIME, GA2M, Mimic, etc.) along with our own improvements and unique algorithms (as needed), creating common APIs and data structures across the integrated libraries, and interfaces seamlessly with AML services. It wraps all the explainers so they have a common API and output format.

We only have to use one module to perform explanation. It automatically selects a suitable direct explainer and generates the explanation info based on the given model and data sets.

After using their model explanation package, one can upload the explanation to the cloud and visualise it!

6.2 *Improves task efficiency*

The $AutoML$ package that iterates through different model types was immensely helpful. The graph outputs provided on the portal are also typical graphs that data scientists would use to evaluate their models. In addition, $AutoML$ records different types of metrics during each experiment run, saving much time and effort and preventing code replication.

6.2.1 *Parallel runs and version tracking*

For the machine translation project, Azure's ability of being able to parallelize runs and keeping track of versions is hugely beneficial. We were able to submit different hyperparameter combinations and make changes to the scripts and send

Table 2: Training time (in min)for a two layer fully connected model on CIFAR-10 dataset

# Hidden units in the fc-layer	4000		5000		6000		7000		8000	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
Azure platform	11.31	3.12	20.44	1.31	23.02	1.27	25.18	3.27	27.22	3.31
Dell XPS 15	20.26	2.45	23.10	2.82	25.33	3.14	26.87	3.37	27.67	1.75

them to compute targets separately. It was also helpful that we did not have to track changes manually, since Azure's platform keeps a record of the script that is submitted to run.

7 Limitations

Azure provides a $dprep$ module that helps with data cleaning and imputation. The module is not meant to be a re-invention of $pandas$ and returns a $dflow$ object, rather than a regular dataframe, therefore lacking many of the functionalities, including returning columns of a data frame. A neat feature introduced by $dprep$ was $autoreadfile()$, which can infer file type automatically, however the limitations of $dprep$ module far outweigh the convenience of $autoreadfile()$.

8 Conclusion

Overall, we are pleased with Azure's usability, especially its ability to integrate with open-source codes. The portal interface is a little convoluted, but the machine learning features themselves are easy to use. We found that Google Cloud Platform had longer uploading data and training times than Microsoft Azure. Refer to Table 3 for summary evaluation across different platforms. Link to our GitHub repo is provided at `http://www.tiny.cc/azureml`. All in all, despite the initial learning curve, using Azure has been a good experience and we encourage all hesitant watchers to try out Azure.

9 Future Work

We are collaborating with Professor Eric E. Poehler from Classics department, UMass Amherst who is undertaking a project to describe the landscape of Pompeii's architectural decor. He needs data rich environments that can serve as workflow templates for a cadre of undergraduate students who will deeply describe those thousands of artworks at Pompeii. As of now, we are focusing on recognizing common elements within those artworks using standard object recognition

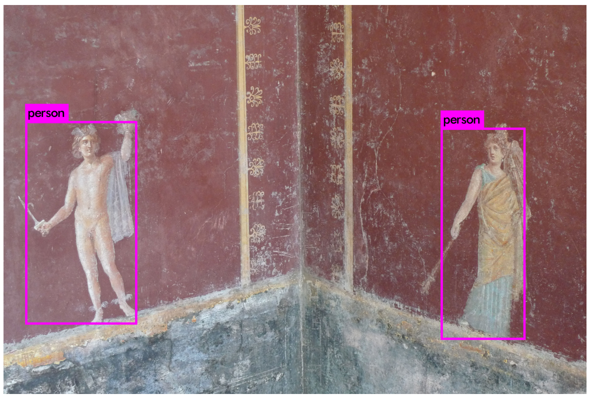

Figure 4: YOLO prediction on one of the Pompeii Images

architectures on the Azure platform. Fig. 4 presents one of our early experiment results with further investigation underway.

Our teammate, Katie House, will also add to the DeepCheck web application and notebook throughout the summer as a personal project. Next steps for DeepCheck involve adding more model documentation and tuning the model to more accurately detect hate speech. The DeepCheck team plans to compete in Microsoft's 2020 Imagine Cup.

References

A. Vyas, R. Kannao, V. B. and Guha, P. (2017). Commercial block detection in broadcast news videos, in proc. ninth indian conference on computer vision, graphics and image processing. *CVPR*.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

House, K. (2019). Github repository of our notebooks. Available at

Table 3: Summary of Evaluation across Platforms

Feature	Azure	Google	Local
Free Advanced GPU	-	+	NA
Uploading Data	+	-	-
Preprocessing	-	-	+
Training Time	+	-	-
Hyperparameter Sweep	+	NA	-
Model explainability	+	-	-
Usability	+	+	-

+ : good; - : unsatisfactory; NA : not applicable

https://github.com/katiehouse3/microsoft-azure-ml-notebooks.

Labs, A. (2019). Amazon sagemaker examples. Available at https://github.com/awslabs/amazon-sagemaker-examples.

Learning, A. M. (2019). Machine learning notebooks. Available at https://github.com/Azure/MachineLearningNotebooks.