	test condition:		n - o > 0.1 +/- 0.1	n > 0.5 +/- 0.1
test set size	ϵ	δ	# of runs	# of runs
50k	0.01	0.0001	n/a	2 ($208)
		0.001	n/a	5 ($83)
		0.01	n/a	8 ($52)
	0.025	0.0001	8 ($52)	76 ($5)
		0.001	11 ($37)	80 ($5)
		0.01	14 ($30)	84 ($5)
100k	0.01	0.0001	n/a	16 ($46)
		0.001	n/a	18 ($36)
		0.01	n/a	23 ($5)
	0.025	0.0001	30 ($28)	168 ($5)
		0.001	34 ($24)	170 ($5)
		0.01	38 ($22)	174 ($5)
500k	0.01	0.0001	21 ($198)	130 ($32)
		0.001	25 ($166)	134 ($31)
		0.01	29 ($143)	138 ($30)
	0.025	0.0001	211 ($20)	889 ($5)
		0.001	215 ($19)	891 ($5)
		0.01	217 ($19)	895 ($5)

Table 1: Number of runs and (in braces) a price estimate for a single run for various test set sizes, and (ϵ, δ) values. Cells marked as n/a correspond to insufficient test set sizes.

the test score is above a certain threshold, and another that checks if there is an improvement in the test score when comparing two models. For each test condition, we try out different combinations of the error margin ϵ (taken to be either 0.025 or 0.01) and the error probability δ (taken to be either 1%, 0.1%, or 0.01%).

We associate each test run with a hypothetical "dollar price" as follows. We assume a labelling effort that is conducted at the speed of 5 seconds per label, and at the price of $6 per hour (number taken from `labellbox.com`). At this rate, 720 data examples can be labelled per hour. We do not include the price of acquiring features, though it is not negligible. We argue that the overall price is dominated by the labelling cost since each data example requires human effort. With this setting, we can then compute the price of a test set (and thus the test run). Table 1 presents the results.

6 RELATED WORK

CI has been an industrial standard in practice and the literature on classic CI in software engineering is overwhelming [12]. However, so far little work has been done towards "CI for ML," although there have been emerging discussions in online communities regarding such requirements [8, 17, 18, 26]. The recent effort by Renggli et al. [20, 21] is the first work along this line, as far as we know. It lays out theoretical foundations as well as building a proof-of-concept system to demonstrate the feasibility of "CI for ML." The current paper, meanwhile, takes one step further by addressing the design, implementation, data management, integration challenges for developing an industrial-strength "CI for ML" service.

The "CI for ML" idea also fits well into the broader scope of building AutoML systems and services. Users of AutoML systems only need to provide their data and high-level specifications of their ML tasks (e.g., loss functions to be minimized), and the system will take over the rest of the job, such as automatic pipeline execution, resource allocation, and performance monitoring. Typical AutoML systems include industrial offerings from major cloud service providers such as Amazon SageMaker [1], Microsoft Azure Machine Learning [6], and Google Cloud AutoML [5], as well as ones from academic institutions such as the Northstar system developed at MIT [16], and the `ease.ml` service [15, 19, 27] by ETH Zurich, among others. It remains interesting to see how to incorporate "CI for ML" into these existing AutoML services.

7 CONCLUSION

We have presented our efforts and experiences with building an industrial-strength "CI for ML" service. We discussed the details of its design, implementation, data management, and integration, as well as evaluation over real datasets. We showcased the risk of overfitting a static test set in the context of "CI for ML" that motivated us to come up with the "staged test set" solution, and demonstrated its affordable cost in practice.

REFERENCES

[1] Amazon sage maker. https://aws.amazon.com/sagemaker/.
[2] Aws codepipeline. https://aws.amazon.com/codepipeline/.
[3] Azure devops services. https://azure.microsoft.com/en-us/services/devops/.
[4] Git LFS. https://git-lfs.github.com/.
[5] Google cloud automl. https://cloud.google.com/automl/.
[6] Microsoft azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning/.
[7] `mltest` tool open-source repository on github. https://aka.ms/gsl-ml-test.
[8] Continuous integration for machine learning. https://medium.com/@rstojnic/continuous-integration-for-machine-learning-6893aa867002, April 2018.
[9] S. Ackermann et al. Using transfer learning to detect galaxy mergers. *MNRAS*, 2018.
[10] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.
[11] A. Blum and M. Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *ICML*, pages 1006–1014, 2015.
[12] P. M. Duvall, S. Matyas, and A. Glover. *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.
[13] C. Dwork et al. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.
[14] I. Girardi et al. Patient risk assessment and warning symptom detection using deep attention-based neural networks. *LOUHI*, 2018.
[15] B. Karlas, J. Liu, W. Wu, and C. Zhang. Ease.ml in action: Towards multi-tenant declarative learning services. *PVLDB*, 11(12):2054–2057, 2018.
[16] T. Kraska. Northstar: An interactive data science system. *PVLDB*, 11(12):2150–2164, 2018.
[17] A. F. Lara. Continuous integration for ml projects. https://medium.com/onfido-tech/continuous-integration-for-ml-projects-e11bc1a4d34f, October 2017.
[18] A. F. Lara. Continuous delivery for ml models. https://medium.com/onfido-tech/continuous-delivery-for-ml-models-c1f9283aa971, July 2018.
[19] T. Li, J. Zhong, J. Liu, W. Wu, and C. Zhang. Ease.ml: Towards multi-tenant resource sharing for machine learning workloads. *PVLDB*, 11(5):607–620, 2018.
[20] C. Renggli et al. Continuous integration of machine learning models with ease.ml/ci: Towards a rigorous yet practical treatment. In *SysML*, 2019.
[21] C. Renggli, F. A. Hubis, B. Karlas, K. Schawinski, W. Wu, and C. Zhang. Ease.ml/ci and ease.ml/meter in action: Towards data management for statistical generalization. *PVLDB*, 12(12):1962–1965, 2019.
[22] K. Schawinski et al. Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. *MNRAS*, 2017.
[23] K. Schawinski et al. Exploring galaxy evolution with generative models. *Astronomy & Astrophysics*, 2018.
[24] D. Stark et al. PSFGAN: a generative adversarial network system for separating quasar point sources and host galaxy light. *MNRAS*, 2018.
[25] M. Su et al. Generative adversarial networks as a tool to recover structural information from cryo-electron microscopy data. *BioRxiv*, 2018.
[26] D. Tran. Continuous integration for data science. http://engineering.pivotal.io/post/continuous-integration-for-data-science/, February 2017.
[27] C. Zhang, W. Wu, and T. Li. An overreaction to the broken machine learning abstraction: The ease.ml vision. In *HILDA@SIGMOD 2017*, pages 3:1–3:6, 2017.