SYSTEM CENTRIC APPROACH TO ML PROJECT MANAGEMENT: WHY WE INVESTED IN SIMULATION INSTEAD OF IN OPTIMIZING THE MODEL MORE

Hagit Grushka-Cohen, Eliya Habba, Jeremy Samama, Yaniv Lavi and Rachel Lemberg

Introduction

When developing a new ML based solution, we all agree that any model we use should be trained and tested to meet the highest standards of performance: accuracy, f1 score or any other model KPI. In applied data science, deployed models rarely serve as a standalone system [1]. Often the developed solution (system) is made of layers and combinations of different models, components, and decision layers. It is on us as managers of the developed solution to invest also in the evaluation measurement and methods of the solution or system provided to our users. We must make sure that the results of activating the different models and components working as one system align with the desired business KPIs and to validate the impact of our solution on other components affected by our model's recommendation.

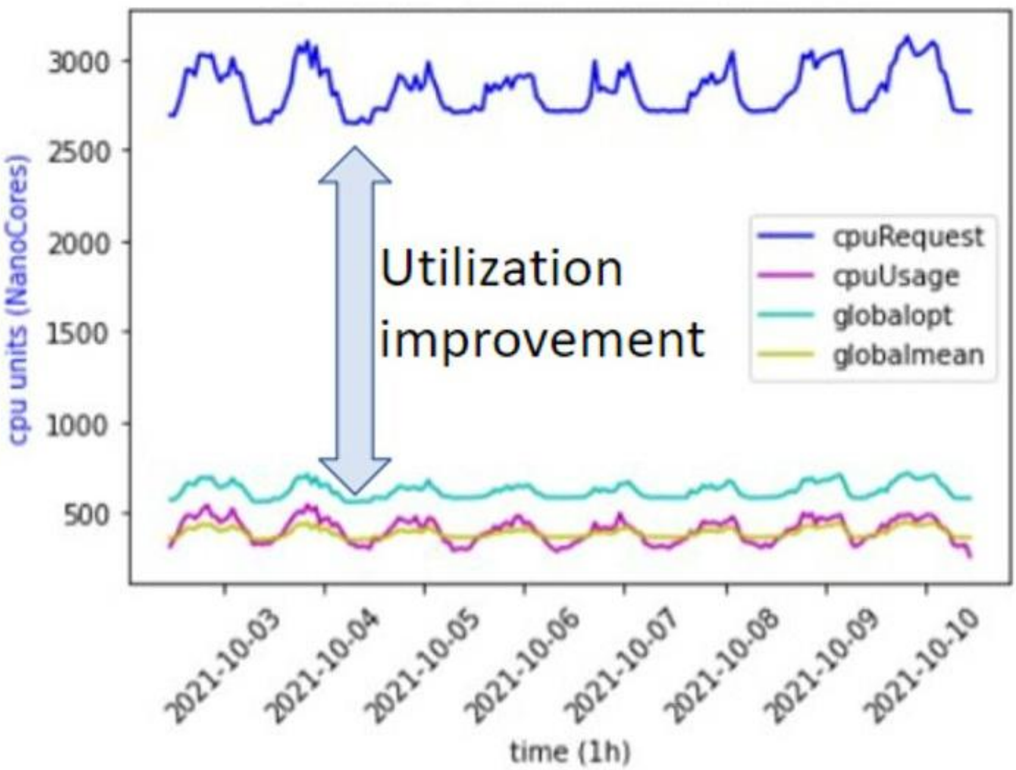
We must acknowledge that a data science project has different layers of KPIs, some of them unique to a specific model, while others are related to the results of using our system and define how we are going to measure and evaluate each of them. When applying this methodology, we witness that sometimes evaluating the model and working towards increasing the target score (such as accuracy) has limited impact on the end users experience or satisfaction from our system. In this paper we will present the following use case where we prioritized the development of an evaluation framework to predict the impact of following a model's recommendation on the users' KPI over further modelling. This allowed us to maximize the impact of our system and as the key to drive user adoption.

Business use case

background

Our goal was to build a recommendation system to help Azure Kubernetes users optimize their resource allocation. Efficient resource allocation improves their systems' ability to accommodate workloads and minimizes cost.

Kubernetes [2] clusters are made of pods: one or more containers packaged together. Pods are assigned to nodes (VMs). Resource allocation is managed by defining for each container the resources it needs in terms of containers request and limit. Request is used as the minimal resources set aside (or guaranteed) for the service while Limit serves as the maximal resources allowed to be consumed by the container. Kubernetes uses these definitions to assign containers to nodes. Requests directly effect on the pods and containers scheduling. Changing the values of request changes the future possible pods allocations. Managing large Kubernetes clusters is hard and requires investing a lot of time regularly or wasting resources, automating this process in a data driven fashion presents a great opportunity [cite other k8s resource allocation papers).



Project management

At the beginning of the project during the data exploration stage we analyzed many Kubernetes clusters in terms of actual usage vs resource and request and limit values. We found that many of the cluster suffer from low utilization. Low utilization translates directly to high costs and waste. We also learned that one of the reasons the utilization is low is because users find it hard to estimate the resource requirements as they are fluctuating over time. Preparing for "worst case" they tend to over-request resources [graph?]. Increasing the utilization of the clusters and reducing the size of these clusters will both save our customers money and reduce our environmental footprint. At the same time, we must keep in mind that our customers' desire is to minimize risks of low availability of their services, impacted by too-frequent scaling events, and cases of insufficient resources to perform a task.

To increase utilization, we predict the future needs based on the historical usage telemetry and derive the recommendation from these predictions. Several models and approaches for this right sizing problem have been suggested and implemented.

Since we had to decide between multiple models and architectures, we wanted a method for comparing the different recommendations on our end users KPI's from the system . [3] deployed a cluster and experiment the deployment of different request and limits settings combined with different (controlled) loading of the system. However, we were looking for a evaluation method that would generate and scale to any system deployed in an AKS cluster of any size, to validate the robustness of the recommendation system. Especially in our use case where the request and limit values used in the deployment of the cluster changes the feasible scheduling of pods to nodes. In other words when users apply the systems' recommendations, they are changing the "reality" in a way that is hard to estimate in advance. To overcome this obstacle, we built a simulation gym, inspired by the open AI gym [4]. This simulation gym allowed us to perform a robust evaluation of applying models' recommendations, comparing models impact by generating thousands of scenarios based on data collected from real life clusters.

This evaluation framework allows us to provide incentives and explanation for our end users by adding information regarding the predicted savings and probability of facing performance degradation. Investing in development of a framework that allows comparison of models in terms of both probability of risk and utilization has a great impact on the adoption of recommendations by end users.

Bibliography

1. Lecunn Yann, "Actually, AI R&D has *3* flavors" https://twitter.com/ylecun/status/1525540832027234304, May 14 2022
2. "What is Kubernetes" , https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
3. E. R. Rodrigues, R. L. F. Cunha, M. A. S. Netto, and M. Spriggs, "Helping HPC users specify job memory requirements via machine learning," Proceedings of the Third International Workshop on HPC User Support Tools, pp. 6–13, 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=3018836
4. 2 Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym." *arXiv preprint arXiv:1606.01540* (2016).