Finding Bottleneck in Machine Learning Model Life Cycle

Chandra Mohan Meena
WalmartLabs

Sarwesh Suman
WalmartLabs

Vijay Agneeswaran
WalmartLabs

Abstract

Our data scientists are adept in using machine learning algorithms and building model out of it, and they are at ease with their local machine to do them. But, when it comes to building the same model from the platform, they find it slightly challenging and need assistance from the platform team. Based on the survey results, the major challenge was platform complexity, but it is hard to deduce actionable items or accurate details to make the system simple. The complexity feedback was very generic, so we decided to break it down into two logical challenges: Education & Training and Simplicity-of-Platform. We have developed a system to find these two challenges in our platform, which we call an Analyzer. In this paper, we explain how it was built and it's impact on the evolution of our machine learning platform. Our work aims to address these challenges and provide guidelines on how to empower machine learning platform team to know the data scientist's bottleneck in building model.

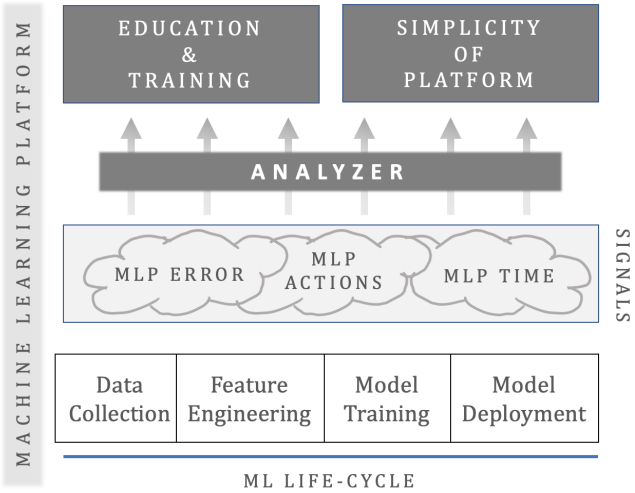

Figure 1: System Architecture

1 Introduction

We have a good strength of data scientists in Walmart. Our system has counted more than 393 frequent users and 998 infrequent users. The data scientists are from various teams in Walmart. We have seen various kinds of challenges in the evolution of our platform to support end to end Machine learning (ML) model life cycle. We have observed that our data scientists are not able to use the platform effectively. Less adoption of our platform is the key challenge for us to solve. It is not sufficient to be completely dependent on surveys to improve our system as people often lie in user surveys, possibly due to a phenomenon known as social desirability bias [2].

Analyzer approach: We have instrumented our machine learning platform (MLP) to collect data for analysing how users interact with the platform. We named the data collected or sent from a product or a feature as signals. Example of signals are clicks, interaction with MLP features, time spent on features, time to complete certain actions, and so on. We have created various classes of signals for our product: MLP-action signals (e.g., key-actions, clicks, and views), MLP-error signals (e.g., error in each component) and MLP-time signal (e.g., time on component, component load time, minutes per session) [3, 4]. We have defined four stages of the ML lifecycle, namely, "Data collection,cleaning,labeling", "Feature Engineering", "Model training" and "Model deployment" [1]. In each stage signals are generated for the Analyzer to capture particular challenges faced by user.. We present two dimensions of challenges: Education-&-Training (ET) and Simplicity-of-Platform (SoP). In ET, the focus is on a data scientist's capability and self-sufficiency to use the MLP and SoP focuses on the technical or design challenges in ML lifecycle. The Fig. 1 shows our system architecture.

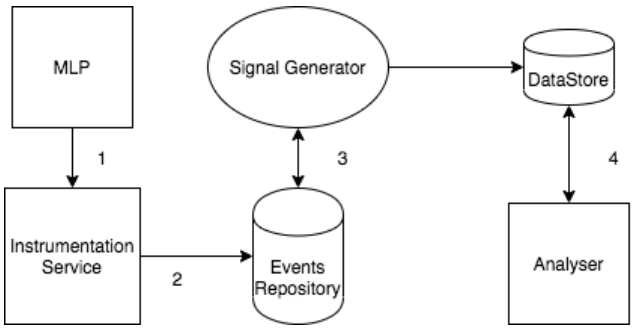

Figure 2: Collection of Signals

2 Signal Collection

Our instrumentation framework collect user actions from platform and sends it to our instrumentation service. We consider all actions as events and store them in our event repository (database). The signal generator reads the data from the repository and classifies it into multiple signals. We name the data collected or sent form product or feature as signals. Example of signals are clicks, interaction with the tools, time spent on the tools, time to complete certain actions. These signals are mapped to multiple classes: MLP-actions, MLP-error and MLP-time. The signal generator annotates these classes of signals with the user and platform related meta-data and stores it into a datastore. Our Analyzer runs the final show. It goes through each new signal and evaluates respective stage wise bottleneck for a user. Fig. 2 describes this flow.

3 Analyzer

In this section, we summarize the design of Analyzer, a system that uses a user-behaviour-segregation-model and set of rules to evaluate challenges of using MLP. The model uses K-means clustering technique, which groups the users based on action and time signals and later by looking at the centroid/clustor representative values. Users in the group are labelled as: Beginner, Intermediate and Expert. Our model has suggested 85 Expert, 118 Intermediate and 190 Beginner users of our platform. We have defined set of rules for each different type of signal. Analyzer applies these rules and smell the challenge from signal. We have defined following set of rules:

1. If more than 50% expert and 65% intermediate and 80% beginner users see the error signal then it is mapped to SoP challenge

2. If less than 10% expert and more than 65% intermediate and 80% beginner users see the error signal then it is mapped to ET challenge

3. if more than 50% expert does redundent actions more than 5 times in a week/month then it is mapped to SoP challenge

4. If less than 10% expert and more than 50% intermediate and 75% does redundent actions more than 5 times in a week/month then it is mapped to ET challenge

Challenge	Count
Education & Training	17
Simplicity-of-Platform	6

Table 1: Platform challenges

Above table depicts different instances of ET and SoP challenge at different stages. Some of critical instances are described below:

a) Model-training stage has a default cut-off time, beyond which jobs are terminated automatically. Analyzer has mapped these signals as ET challenge, based on rule-1, where we found users were unaware of this feature.

b) Data-collection stage provides connectors which enable users to retrieve data from varied data sources. Analyzer has mapped these signals as ET challenge, based on rule-4. We have discovered that users prefer creating their own connectors instead of reusing the existing ones, thus resulting in lot of duplicity.

c) Feature-engineering stage, user see a list of notebooks. Notebook IDE runs in a kubernetes container, if notebook is idle for longer than 6 hours then notebook becomes inactive. User need to activate notebook in order to open it. If there is no enough resource available then display error. Analyzer has mapped these signal as SoP challenge, based on rule-1.

Analyzer has been running in production for last 4 months and these above instances are the outcome of our data analysis. This has helped our product team to build better Education and Training (effective tutorial, documentation and videos) for model-training and data-connectors features. Above SoP instance: 'c' highlighted the complexity which user faces in creating notebook in Feature-engineering stage. This enabled product team to integrate live resource information on the notebook list page as part of UX improvement.

4 Conclusion

The Analyser has identified 17 and 6 instances respectively of Education-&-Training (ET) and Simplicity of Platform (SoP) challenges. ET instances have enabled product team to create effective set of documentation and tutorials. SoP instances helped in the evolution of ML Platform as a whole. Our

Analyzer helps us directly in making data driven decisions to improve user adoption. We believe that our Analyzer can be adopted by other Machine Learning Platform teams facing similar challenges. In future, we will add more challenges (ex: Scale, Performance, Hardware-Resources) and use Deep Learning for deriving more insights from these signals.

Acknowledgments

We would like to acknowledge our colleagues who built ML Platform, Senior Director: Ajay Sharma for the opportunity, Distinguised Architect: Bagavath Subramaniam, Product Manager: Rakshith Uj and Anish Mariyamparampil for helping us in instrumentation in UI and Saiyam Pathak for the design review.

References

[1] Saleema Amershi, Andrew Begel, Christian Bird, Rob DeLine, Harald Gall, Ece Kamar, Nachi Nagappan, Besmira Nushi, and Tom Zimmermann. Software engineering for machine learning: A case study. In *International Conference on Software Engineering (ICSE 2019) - Software Engineering in Practice track*. IEEE Computer Society, May 2019.

[2] Wikipedia contributors. Social desirability bias. https://en.wikipedia.org/wiki/Social_desirability_bias, Jan 2020. Accessed on 2020-01-21.

[3] Aleksander Fabijan, Pavel Dmitriev, Helena Holmstrom Olsson, and Jan Bosch. The evolution of continuous experimentation in software product development: From data to a data-driven organization at scale. *International Conference on Software Engineering*, pages 770–780, 2017.

[4] Kerry Rodden, Hilary Hutchinson, and Xin Fu. Measuring the user experience on a large scale: User-centered metrics for web applications. In *Proceedings of CHI 2010*, 2010.