A Viz Recommendation System: ML Lifecycle at Tableau

Kazem Jahanbakhsh
Tableau Inc.
Vancouver, Canada
kjahanbakhsh@tableau.com

Eric Brochu
Tableau Inc.
Vancouver, Canada
ebrochu@tableau.com

Mya Warren
Tableau Inc.
Vancouver, Canada
mwarren@tableau.com

Xiang-Bo Mao
Tableau Inc.
Vancouver, Canada
xmao@tableau.com

Yogesh Sood
Tableau Inc.
Vancouver, Canada
ysood@tableau.com

ABSTRACT

In the recent years, we have seen a rapid growth in the Enterprise space adopting ML models on production to improve the quality of their customers experience. At Tableau, we have released a Viz recommendation feature by which our customers can find the relevant contents (i.e. visualizations) more efficiently. In this paper, we cover our research and development effort for the ML models behind the recommendation especially in the area of model life-cycle management, deployment, and monitoring. The requirements for our ML recommendation system is different from most scenarios described in the literature including [1]. This is mainly because Tableau is an enterprise software company with a large number of customers where 90% of them install Tableau servers on-prem and only 10% run the software in the cloud. This means that we do not have access to our customers data and in most cases cannot collect ML performance metrics from customers sites. Because of these constraints, we had to design our own custom solution for the Hyperparameter optimization to find optimal parameters for each customer. We also had to design a novel monitoring system to collect statistics from the training data and the outcome of the trained models.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; *Modeling and simulation*; Machine learning.

KEYWORDS

machine learning, recommendation systems, collaborative filtering, model management, hyperparameter optimization, monitoring, reliability, best practices

ACM Reference Format:
Kazem Jahanbakhsh, Eric Brochu, Mya Warren, Xiang-Bo Mao, and Yogesh Sood. 2020. A Viz Recommendation System: ML Lifecycle at Tableau. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Tableau [1] is a data visualization software company helping its customers to connect to their specific data. It also allows its customers to analyze and visualize their data and publish/share them. Our team at Tableau has shipped a new feature called "Viz Recommendation" which recommends the relevant visualizations when a user logins to their Tableau server. This feature helps users find the interesting and relevant contents faster.

Over the last several years, we have witnessed a rapid growth in the use of Machine Learning (ML) systems in areas such as recommendation systems, online advertising, fraud detection, and natural language processing. Although there has been a significant research in designing new ML models with higher accuracy, we have not observed an equivalent amount of research on the challenges that companies face when they deploy ML systems on production.

For example, the software industry has matured in development of a number of techniques to verify the correctness of a software product. These techniques include but not limited to unit testing, integration testing, and the use of tools such as Splunk [2] and New Relic [3] to monitor the health of production systems. Despite all the progress in the software space, there is a lack of best practices for getting ML systems to production specifically on the areas such as ML systems verification and monitoring ML systems on production.

In a recent paper by Google, the authors have proposed a list of 28 tests that an ML engineering team should implement for ensuring the production-readiness of their ML system [1]. The suggested tests cover four areas: data, model, pipeline, and monitor. While deploying the Viz recommendation feature, we have taken into account some of the suggestions in [1].

2 HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization (HPO) is a critical step before deploying any ML model to production. The Viz recommendation models consist of nine major hyperparameters. One of the unique challenges specific to Tableau product is that we have to train and deploy thousands of ML models. Specifically, we need to train and deploy one model per customer every 24 hours.

At Tableau, we do not have access to our on-prem customers content usage data. Since running the HPO on customer's site was not an option for us, we decided to optimize the ML parameters in

[1]https://www.tableau.com/
[2]https://www.splunk.com/
[3]https://newrelic.com/

a central fashion. First, we collected a sample of Online customers usage data based on their site's size. A customer site's size is defined as a function of the number of users and contents that the customer has. Our analysis show that these two numbers have a critical role in the distribution of content usage data.

Next, we ran a sweep search on the sampled customers sites data to find parameters' values that optimize the recommendation's *precision at k*. One interesting observation we made was that we could use the same optimal values for most parameters across all customers sites without sacrificing the performance. However, since the matrix factorization is a key component in the recommendation model, we needed to learn the number of latent factors when decomposing the original content usage matrix to two lower dimension user-factor and item-factor matrices.

Our ML experiments have shown that the number of latent factor has a significant impact on the performance of the recommendation systems. We formulated the optimal number of latent factors for a customer's site as a function of the number of users and contents published on the site. As a result, we learned the optimal number of latent factors. We discovered that the number of factors increases with the size of site. We coded the learned function as part of the ML pipeline so that we compute the optimal number of factors per customer site while training the matrix factorization model. While running the HPO experiments, we logged all ML models hyperparameters and metrics to our MLflow [2] server.

3 RECOMMENDATIONS MONITORING ON PRODUCTION

For deploying thousands of recommendation models where we do not have direct access to most of the customer sites, we had to design a new monitoring system. The designed system should respect the customers data privacy while providing useful insights for our ML engineers to debug the ML pipeline in case of any failures on a customer's site. For achieving this, we decomposed our monitoring system into three components: (I) data monitoring part which detects any unexpected pattern in the input data, (II) monitoring ML models' scores to identify any unexpected values, and (III) monitoring the overall performance of the recommendation system from the customers point of view. In this section, we describe how we designed each monitoring component.

3.1 Content Usage and Hybrid Models Monitoring

One key difference between deploying an ML model with deploying a regular software code on production is that an ML model is a composition of the model's code and the training data. Therefore, to evaluate the correctness of an ML-based recommendation feature, we need to monitor any unexpected pattern in the input data. Any noise in the data could generate a faulty ML model with an undesirable customer impact. In the case of the recommendation feature, we needed to monitor any undesirable entries in the content usage data that we use for training our ML models.

Our training data contains four main features: *user id*, *item id*, *nviews* (i.e. number of times a user has clicked on an item), and *time* indicating the last time a user has viewed an item. We have defined the expected range for every input data feature. So, while running the ML pipeline, we scan the training data for any feature value that falls outside of the expected range. We also compute some high-level statistics from the training data for each customer site in order to provide more visibility on the site's data.

One of the models in the recommendation system is the collaborative filtering with implicit feedback [3] (i.e. Implicit CF). The Implicit CF is based on the matrix factorization technique where we decompose the user-item usage matrix into two lower-dimensional matrices. With Implicit we learn the users tastes and the underlying topics for contents by learning users' and items' factors. Before training Implicit CF, we normalize the input usage matrix with BM25 weighting function. Due to the underlying mathematics of BM25 [4] and the matrix factorization, we have to monitor and log any patterns in the input matrix that could cause an unexpected behavior during the training. Specifically, we scan the content usage matrix and log any populated row or column. We also log the sparsity ratio of the matrix. Logging these metrics allows us to detect and debug any numerical instability caused by the Implicit CF model. We log all of these violations and statistics in Splunk.

Our recommendation model is a hybrid of the Implicit CF model and a popularity model. The popularity component is responsible for modeling the timeliness of contents in the future. We have the mathematical specification for each model. After running the ML pipeline on production, we check if the scores of each model fall in the expected range. Specifically, the scores of Implicit CF model should always be finite. Therefore, we log any unexpected score value along with its user/content id. This helps our on-call engineers debug the system while finding the root cause of production issues. We also log any score from the popularity model which falls outside of its expected mathematical range.

3.2 Predictions Quality Monitoring

Finally, we have instrumented our front-end code so that we measure how our users are engaged with the recommended visualizations. When a user visits their hosted Tableau website, we serve them a personalized list of contents generated by the recommendation system. Because of our UI instrumentation, if a user takes an action on one of the recommended visualization, we log all the necessary meta-data for their click event. We use Google BigQuery to store all of the collected telemetries along with meta-data such as user id, view id, and timestamp. We also collect other actions that a user might take on a recommended visualization such as: "favourite", "share", or "dismiss". Collecting and storing this telemetry data allows our product managers and data scientists to monitor and detect any regression in prediction quality over time.

4 CONCLUSIONS

In this paper we shed some light on the unique challenges that an ML engineering team could face while deploying ML models in an Enterprise company. We discussed some of the novel techniques that we have developed especially for optimizing hyperparameters. We also have described the design of our monitoring system by which we can troubleshoot potential ML production issues. We

[4]https://en.wikipedia.org/wiki/Okapi_BM25

believe that our findings presented in this paper could be valuable to other teams who manage the lifecycle of ML within their organizations.

REFERENCES

[1] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley. 2017. The ML test score: A rubric for ML production readiness and technical debt reduction. In *2017 IEEE International Conference on Big Data (Big Data)*. 1123–1132.

[2] A. Davidson A. Ghodsi S.A. Hong A. Konwinski S. Murching T. Nykodym P. Ogilvie M. Parkhe F. Xie M. Zaharia, A. Chen and C. Zumar. 2018. Accelerating the Machine Learning Lifecycle with MLflow. In *IEEE Data Engineering Bulletin*, Vol. 4.

[3] Yehuda Koren Yifan Hu and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. 263–272.