Monitoring Machine Learning Models
DS440 Fall 2022
Section 002 Group 6
Progress Report 3

Vincent Gan(wxg39@psu.edu)
Glenn Hubbard(gch5121@psu.edu)
Dequan Ma(djm7012@psu.edu)
Yuming Sun(yfs5150@psu.edu)

October 9, 2022

1 Team Members

Vincent Gan is a senior with a major in Computational Data Sciences and a minor in statistics. He is familiar with programming languages, including Python, C++, R, SQL, and Scala. Regarding machine learning, he is skillful in data preprocessing, hyper-parameter tuning, and algorithm/model evaluation.

Glenn Hubbard is a senior in Applied Data Sciences. He has programming experience in Python and Java. He is interested in the ethical implications of artificial intelligence and the prolific use of machine learning. Glenn's expertise rests mostly in technical writing, algorithm development, and front-end design. Glenn is a senior in Applied Data Sciences.

Dequan Ma is a senior with a major in Computational Data Sciences and a minor in mathematics. He has programming experience in Python, R, and SQL. He is interested in transforming real-world problems into machine learning problems and exploratory data analysis.

Yuming Sun is a senior in Statistical Modeling Data Sciences. Most of his work is in Python. He is interested in physics informed networks, unsupervised anomaly detection algorithms and practical application of the algorithm of machine learning. He also has experience in basic techniques for building intelligent computer systems.

2 Topic Introduction

Model monitoring is the practice of measuring model performance. As a field, model monitoring is broadly defined. **A model monitoring system can monitor features including but not limited to predictive correctness, hardware usage, the integrity of input data, and data drifts**. Monitoring these features allows model users to **spot model malfunctions as early as possible and prevent loss**.

The team will craft a model monitoring system that can inform users of key metrics related to their trained model and input data. **It is assumed that users will perform necessary measures specific to their scenarios to provide a satisfactory baseline model before monitoring**. From this, the team hope to learn the viability of model monitoring in everyday machine learning tasks, and will apply this knowledge to compare the built monitoring system against existing open source systems.

3 Existing Efforts

3.1 Why Monitor Models

As gathered from a paper on managing machine learning models (Vartak and Madden 2018), model monitoring is a tool that helps improve performance, reliability, and accountability when implementing a machine learning model. Model monitoring remains present in all stages of the data science pipeline–from data intake, to model training, and to parameter tuning. Advanced monitoring systems can be deployed as a standalone system that operates with a live model–leading to an automatic change/alert system of events within the model. For instance, if new data being fed into the model included null values, then the monitoring system could flag that data. If a shift in the data values–also known as a data drift–were to occur drastically enough, it would lead to limited reliability from the predicted result, then the monitoring system could flag this. Monitoring can clearly serve a purpose in the day-to-day operations of firms looking to maintain a solid up-time and reliable services.

Monitoring also plays a role in preventing malfeasance. An advanced cyber threat, known as an adversarial attack, can manipulate the input into a machine learning model, hoping to alter the predicted result. In advanced settings, such as autonomous vehicles, this threat poses a clear and present danger–a danger that could lead to loss of life and property damage. However, advanced monitoring systems may detect these types of cyber attacks, and thus work to correct the prediction–or at the very least, alert the relevant authorities.

The team is interested in developing our knowledge and skill set surrounding model monitoring because of the growing presence of these tools in the workplace and in government. Machine Learning Operations, otherwise known as MLOPS, is a growing practice in firms with vast model deployment. MLOPS describes the field of work individuals who labor on model monitoring problems. MLOPS jobs are abundant–with varying degrees of opportunity in a variety of industries. Despite the opportunity for career growth, model monitoring offers exploration–new technologies will be needed to ensure machine learning models can stay relevant–and the best way to ensure this is to continue to improve the reliability of older models.

3.2 Challenges

(Schröder and Schulz 2022) points out that monitoring machine learning models comes with many challenges. Before we dive into various metrics of monitoring model performance, the quality of data input needs to be assessed. It is crucial to monitor the quality, format, and distribution of data continuously while operating an MLM.

Then, model performance can be measured by several metrics, including performance, robustness, confidence, economy, and more. Performance is usually measured by predictive

correctness metrics, such as accuracy, precision, recall, roc-curves, and more. Robustness is defined as the degree to which a component continues to function correctly in the presence of invalid inputs or challenging environmental conditions (Committee 1983). Confidence describes the probability with which a model believes its given prediction is correct (Machin et al. 2013). The economy of a model relates to the economic consequences of deploying a model.

3.3 Current Monitoring Systems

Building a monitoring system from scratch is not always possible and workable. Fortunately, there are many monitoring services available on the market. Major businesses usually consider services from Arize, Amazon SageMaker, Censius, and more (Czakon 2022). Among them, Amazon SageMaker seems to be the most prominent. Amazon SageMaker is a fully inclusive MLOPS system, and includes the ability to "build, train, and deploy machine learning (ML) for any use case with fully managed infrastructure, tools, and workflows" (Mishra 2019). Despite these broad abilities, SageMaker's monitoring component offers continuous model management that can track bias and drift within the data. Amazon takes model monitoring even further and even offers the ability to re-train the model once the bias and drift are found. Because of SageMaker's integration within Amazon Web Services and its broad viability, this product is widely used across industry.

While developing a system comparable to Amazon's SageMaker is improbable within one semester, the team believes that some model monitoring projects on GitHub serve as a solid starting point and will provide guidance. One of these guiding projects will be Evidently (Czakon 2022). Evidently offers a direct library that can be used within a Jupyter notebook to monitor models.

3.4 Useful Libraries

Since the project is mostly concerned about the post-training model monitoring stages, the team will use existing libraries to minimize the time spent on training models. FLAML (A fast library for AutoML) is a helpful Python library to facilitate the process (*Getting started* n.d.). It automates finding the best fitting models for various machine learning tasks, including classification, Natural Language Processing, Rank, regression, Time Series Forecasting, and many more. With this library, the team can create different models to develop and thoroughly assess the model monitoring system. If time permits, the team would also like to enable our model monitoring system to monitor neural networks, which can be trained with the TensorFlow library (*Introduction to tensorflow* n.d.).

In order to test the monitoring system, a continuous stream of testing data of different types is required. In addition, it is necessary to make sure that the testing data follows a similar distribution with minor shifts, so that the team can observe how well our system can monitor those shifts. There is a Python library called SDV (Synthetic Data Vault) that solves this problem (Patki, Wedge, and Veeramachaneni 2016). This library allows users to generate tabular data that has similar statistical properties as the training data easily.

4 Proposed Work

4.1 Deliverables

The team would like to explore the creation of **a model monitoring system that can provide insights on performing most machine learning models and the data stream flowing into the models**. The team tentatively decided that the system will be delivered as a one-file python library and there will be a sample Jupyter notebook provided for each use case.

4.2 Roles

Vincent will serve as the model developer, which is based on his expertise in tuning, evaluation, and data processing. Glenn will serve as a model tester, which is based on his knowledge of algorithm development and front-end design. Dequan will serve as a visualization engineer, which will provide a user-friendly way to view the monitoring environment. Yuming's deep knowledge of machine learning will help him serve as a technical writer, since he can succinctly convey advanced concepts into words. While these roles will get the most value out of each individual, they are not set in stone. The team will work together on most problems. If help is needed in one area, the team will translate their labor into that field.

4.3 Milestones

4.3.1 Model Training

The first milestone is model training with a due date of September 23rd. Each member of the team will work on at least one type of model. This milestone gains insights on requirements of monitoring different types of models, since each type of model will have different input data format and different evaluation metrics.

4.3.2 Model Performance Plotting

The second milestone is model performance plotting with a tentative due date of October 13th. The team members will continue to work with the trained models and training data from the previous milestone. They will write functions to visualize the performance of our models with appropriate evaluation metrics.

4.3.3 Data Plotting

The third milestone is data plotting with a tentative due date of November 1st. Now that there are functions to visualize model performance, the team will then visualize essential statistical properties of the input data, which will help users prevent or troubleshoot model performance declines.

4.3.4 Comparing with Existing Libraries

The final milestone is comparing with existing libraries with a tentative due date of November 10th. The team will seek open-source libraries that share similar functionalities and compare the outputs to see how well the system works and how it can be improved.

4.3.5 Optional: Build A Complete Python Library

An optional milestone will be to compile the Python file into a Python library. Then, it will be published online and hopefully be continually improved with the help of the online community.

5 Implementation

5.1 Model Training

For this milestone, the team members each built a type of machine learning model and gained some insights on essential functionalities of a model monitoring system. Currently, the investigated models include time series forecasting, classification, regression, and image classification. The team mainly used FLAML and TensorFlow to train the models and achieved satisfactory performance, thanks to extensive use case demonstrations on respective documentations. Then, each of the team members saved their model to a file for future testing.

In addition, the insights from each member's model training experience helped the team with setting concrete plans for developing the model monitoring system. First, preprocessing is fundamental and specific to a type of dataset, because every dataset will a have different format and require different preprocessing strategies. Therefore, it is necessary to have separate plotting functions for different machine learning tasks. Also, the team realized that the focus of the project is not to train a perfect model. Rather, the focus should be to monitor how the model performance changes as the input data changes in its statistical properties, such as the distribution and joint distribution of individual features in a dataset. For model performance, it is necessary to use different sets of evaluation metrics. For example, the definition of accuracy would be different between a classification and a regression model. The former would mean the number of correct predictions out of all predictions, while the latter would typically mean the mean error between true values and predicted values.

5.2 Model Performance Plotting

With the insights from the previous milestone, the team figured out a clear road map from now on. Each of the team members will write functions to visualize model performance with various evaluation metrics. To calculate the scores from different metrics, the team will use Scikit-Learn libraries, such as mean square error, r squared, accuracy, precision, and confusion matrix. To visualize the scores, the team will use Matplotlib's Pyplot and Animation libraries with Jupyter notebook's widget mode. This configuration allows the functions to generate real-time interactive plots, as inspired by Amazon's SageMaker.

While the outputs of the functions will depend on a specific model, the workflow of the functions will be roughly the same. For example, the plotting function for a time series model will take in predicted values, true values, batch size, and a list of metrics. The predicted and true values can be arrays of label values. The batch size parameter allows users to define the number of data points in one evaluation.

The list of metrics allows users to define a list of one or more supported metrics. Each metric will output one subplot. Currently, the supported metrics include mean squared error, r squared, and mean absolute percentage error. If the batch size is set to one, then the function will evaluate defined metrics and update the plots for one data point at a time; if

it is set to the length of the predicted values, then the function will evaluate defined metrics for the entire dataset.

In order to test the plotting functions, it is necessary to prepare a pool of test data for each with controlled statistical properties. One idea is to reserve more test data from original datasets and perform upsampling when necessary. This idea is a good starting point, but the team would still need to find more data to test the functions thoroughly. Ideally, the team would like to write a data simulation function from scratch that fits a dataset to its closest distribution, performs a controlled distribution shift, and resamples from the shifted distribution. Such a simulation function would allow the team to have as many data as necessary for testing. However, the related statistical calculations and resampling procedures are currently beyond the team's capabilities.

Fortunately, the team found an alternative solution. There are many Python libraries that allow users to generate synthetic data that emulates the statistical properties of a given dataset. Among them, the team finds the library SDV, as mentioned in section 3.4, the most suitable for existing models. SDV aims to generate synthetic data that looks close to original data, but not perfectly. As a result, the generated data will come with minor distribution shifts. Then, the team can observe the effectiveness of the functions in responding to those shifts.

Most of the progress for this milestone so far has been conceptual. The team is currently working to turn the concepts into codes. The latest codes can be found here.

References

Committee, IEEE Computer Society. Software Engineering Technical (1983). *IEEE Standard Glossary of Software Engineering Terminology*. Vol. 729. 1983. IEEE.

Czakon, Jakub (2022). *Best tools to do ML model monitoring*. URL: https://neptune.ai/blog/ml-model-monitoring-best-tools.

Getting started (n.d.). URL: https://microsoft.github.io/FLAML/docs/Getting-Started/.

Introduction to tensorflow (n.d.). URL: https://www.tensorflow.org/learn.

Machin, David et al. (2013). *Statistics with confidence: confidence intervals and statistical guidelines*. John Wiley & Sons.

Mishra, Abhishek (2019). *Machine learning in the AWS cloud: Add intelligence to applications with Amazon Sagemaker and Amazon Rekognition*. URL: https://aws.amazon.com/sagemaker/.

Patki, N., R. Wedge, and K. Veeramachaneni (Oct. 2016). "The Synthetic Data Vault". In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410. DOI: 10.1109/DSAA.2016.49.

Schröder, Tim and Michael Schulz (2022). "Monitoring machine learning models: A categorization of challenges and methods". In: *Data Science and Management*. ISSN: 2666-7649. DOI: https://doi.org/10.1016/j.dsm.2022.07.004. URL: https://www.sciencedirect.com/science/article/pii/S2666764922000303.

Vartak, Manasi and Samuel Madden (2018). "MODELDB: Opportunities and Challenges in Managing Machine Learning Models". In: *IEEE Data Eng. Bull.* 41, pp. 16–25.