Deployment and Integration of Machine Learning Methods with Continuous Integration/Continuous Delivery

DREW C. WHAM
PENN STATE UNIVERSITY

ABSTRACT Machine learning methods are increasingly being used to guide decision-making in fundraising. Insights from these methods need to be integrated into the workflow of university advancement processes through a variety of end-user applications. This integration can be a challenge because the systems and technologies used for machine learning model development are generally different from those used to deliver data to advancement staff. Consequently, separate systems need to be designed to support the automation and deployment of machine learning systems, as well as the integration of their insights into end-user systems, including business intelligence (BI) and customer relationship management (CRM) tools. In this paper, I describe the approach used by Penn State University's Prospect Management and Analytics Office (PMA) to deploy a predictive machine learning model that generates prospect scores nightly. This system utilizes continuous integration/continuous delivery (CI/CD) through GitLab (GitLab, n.d.) to provide version control for project code while also supporting code-based automation to the Amazon Web Services (AWS) cloud. The machine learning project code is then run in pre-built Docker container environments on cloud servers. Finally, the output of the processes is automated to endpoints where they can be made available to the CRM and BI tools. I outline the approach, discuss how it has transformed capabilities in the Penn State PMA Office, and then conclude with some best practices that have arisen from utilizing the approach over the last year.

KEYWORDS machine learning, automation, predictive modeling, continuous integration

Across many industries, a paradigm shift from traditional analytics to integrated data science systems is occurring to better inform high-level decision-making. For many years, the primary goal of analytics has been to provide insight on specific reports and dashboards that only required periodic updates. Increasingly, fundraisers are seeking to leverage insights from historical action outcomes to inform day-to-day business processes in near real time, requiring an integrated data science (IDS) approach.

At the heart of the IDS approach is crafting a model specifically tailored to optimize fine-scale decision outcomes. In advancement, examples of these fine-scale decisions include which prospects to reach out to or to solicit. The model is then used to guide subsequent actions, with past results informing present choices, and freshly observed data reshaping the model itself. The aim of this process design is to create a continuous loop of model refinement and improvement. The design is often described as a "flywheel," an analogy popularized by Jim Collins (2001) in *Good to Great*. As with a flywheel, the system gains momentum with each iterative cycle, creating a data-driven virtuous circle of process improvement.

Continuous integration and continuous delivery (CI/CD) (Shahin et al., 2017) allows IDS projects to achieve self-perpetuating improvement by allowing more frequent iteration cycles. CI/CD enables regular model re-training and deployment with fresh data; prioritizes dynamic, code-driven outputs over static project artifacts; and ensures scalability while conserving computational costs. Crucially, CI/CD relies on code-based documentation and automated code execution, reinforcing reliability and repeatability in the analytics process. Such foundational shifts, rooted in automation, pave the way for more sophisticated and agile analytics projects.

In this paper, I explore the challenges and successes within Penn State's Prospect Management and Analytics Office (PMA) as it transitioned from manual updates of reports and dashboards to an automated, IDS-based technical stack. This shift, reflecting our broader goal of moving toward a more modern and sophisticated analytics program, necessitated both practical and philosophical changes. By detailing the specific technology stack and workflow adopted by PMA, this paper offers insights into the creation, delivery, and adoption of an innovative analytics program. It aims to address common questions while contributing to a growing understanding of how these methods can connect insight to action in advancement analytics.

PAST WORKFLOWS OF PENN STATE'S PROSPECT MANAGEMENT AND ANALYTICS OFFICE

Before embracing the CI/CD methodology, Penn State's PMA operated using traditional analytical workflows, mirroring common industry practices. These were primarily designed to aid periodic updates on reports and dashboards. However, the inefficiencies and bottlenecks of these methods became evident when trying to adopt an approach that optimized insight through continuous improvement and nightly delivery of project outputs. Key challenges identified in shifting from this periodic approach to the automated CI/CD methodology included:

1. **Interactive Execution Dependency:** There was a reliance on a manually executed code that required hands-on interaction to run scripts in a sequence outlined by README files as well as to update hardcoded parameters such as date ranges.

2. **Mixture of Code and Manual Processes:** Project execution was a blend of automated and manual steps. Often, the output from one script dictated the manual inputs for subsequent steps, creating dependencies and potential room for error.
3. **Manual Distribution:** End products, such as reports, were manually disseminated to stakeholders via email or shared drives. This mode of delivery was not scalable to meet the real-time or frequent updating needs we aspired to achieve.
4. **Underutilized Version Control:** While Git (Git n.d.) was in use as a rudimentary means to store final project versions, its potent version control capabilities (Bryan 2018) were not fully harnessed. This limited practice made it challenging to monitor changes, collaborate among teams, and maintain a single authoritative version of projects.
5. **Isolated Development Environments:** Code was developed on individual staff computers, leading to discrepancies. What worked seamlessly on one machine might fail on another, introducing inconsistencies and rework.
6. **Dependence on Licensed Software:** Despite the availability of comprehensive open-source tools like Python (Python Software Foundation) and R (R Core Team, 2023), there was a reliance on licensed software. This not only created unnecessary operational costs but also limited flexibility, especially when equivalent or even superior features were available in the open-source ecosystem.
7. **Inefficiencies of GUI-Based Software:** Several tools in our previous workflow required point-and-click interaction through graphical user interfaces (GUIs). These GUI-centric operations made it challenging to establish repeatable and automated processes.

DEFINING THE NEED: PROBLEMS ADDRESSED BY CI/CD TECHNOLOGICAL TOOLS AND PHILOSOPHY

To truly transform the operational efficiency and scalability of our data analytics workflows, it was imperative to re-evaluate not just our technological choices, but also our core philosophies surrounding data project execution and delivery.

Technologically, one major shortcoming in our previous workflows was the inability to consistently leverage prior project work due to a lack of documentation or a lack of consistency in execution environments. Two pivotal facets of our revamped approach are designed to address this gap:

- **Version Control of Project Code:** Every iteration and every modification of a project is checked into the project repository. This practice of meticulous tracking ensures that we can maintain a copy of a record for every project's codebase, and allows us to roll back changes when necessary and to facilitate more efficient collaboration.
- **Server Image Documentation and Containerization:** Every detail associated with building a server suitable to run our projects from a base Linux image is documented in code for both our base images and our Docker containers (Merkel, 2014) where projects are executed. This allows us to regularly re-build images to ensure that operating systems and packages are maintained at the latest version. The documentation also enforces consistency across all development and deployment locations.

Exploring Our Technology Stack

Building on our emphasis of version control and consistent execution, we chose specific technologies to reinforce our workflows. Figure 1 provides a high-level overview of how these technologies link together. Some of the key features of our workflow are that the execution environment and projects' code are version controlled and the code base for each project is in GitLab, which is containerized using Docker, and then deployed in the AWS cloud with GitLab CI/CD. In this section, I explore each component of our tech stack, highlight its role, and mention viable alternatives.

1. **Version Control**
   Role: A platform to track changes, manage versions, and foster collaboration on code.
   Choice: GitLab—With its comprehensive features, GitLab emerged as a reliable and holistic solution for our version control needs.
   Other Considerations: GitHub—Another widely adopted platform known for its community and integrative features (GitHub, n.d.).
2. **CI/CD Process**
   Role: Automates the code integration, testing, and deployment process to ensure consistency and reliability.
   Choice: GitLab—Its integrated CI/CD capabilities make the deployment process streamlined and consistent.
   Other Considerations: GitHub Actions—A robust automation system that caters to CI/CD processes, closely integrated with the GitHub platform (*Understanding GitHub Actions—GitHub Docs*, n.d.).
3. **Containerization**
   Role: Creates isolated environments to run applications, ensuring uniformity across development stages.
   Choice: Docker—Its widespread adoption and the promise of consistent environments made Docker a critical tool in our stack.
4. **Cloud Services**
   Role: Offers secure, scalable, and integrated solutions for computation, storage, and application deployment.

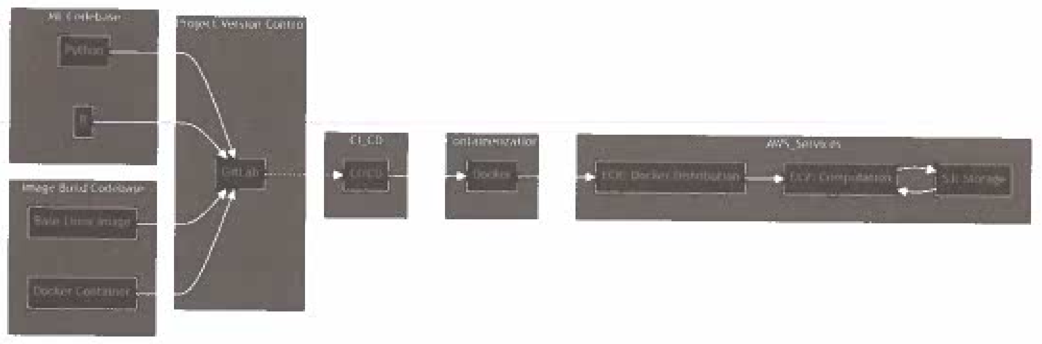

FIGURE 1. Schematic representation of our integrated tech stack. Figure by author.

Choices: ECR (Docker Distribution): To manage and deploy Docker container images. EC2 (Computation): To execute our containerized applications. Amazon Web Services (AWS) S3 (Storage): For archiving project artifacts (Amazon Web Services, n.d.)

Other Considerations: Azure—Microsoft's cloud platform is a strong contender, with services like Azure Kubernetes Service, Azure Compute VMs, and Azure Blob Storage offering equivalent functionalities to AWS counterparts (Azure, n.d.).

5. **Programming for Data Science**

   Role: Languages that provide comprehensive libraries and tools for data analysis and modeling.

   Choices: Python: Renowned for its extensive libraries and versatility in data science. R: Offers advanced statistical tools and is widely used in data science and analysis projects.

6. **Base Operating System**

   Role: Serves as a reliable and cost-effective foundation for all applications and tools.

   Choice: Linux—Recognized for its extensive support of data science libraries and other tech components while also bypassing the licensing costs associated with other operating systems.

Exploring Our Philosophy

Philosophically, there was a radical transformation in what was valued as a project outcome. We shifted from our historical focus of producing tangible project artifacts like reports and spreadsheets to creating robust, reliable code that would, in turn, generate these project artifacts repeatedly. The shift toward prioritizing code and process over artifacts is enabled by the CI/CD process, which binds the code-based documentation of a project to its actual execution on a server with automation. This shift means that, in addition to the scheduled execution of a project's code base, every time a change is made and checked in, the project is run. These checks ensure that every change is tested and checked with automation before it becomes the version of record.

Overall, this process enabled us to adopt a new norm: only artifacts created through CI/CD would be considered fit for dissemination. Every file, report, or value delivered via integration to another system must be backed by end-to-end documentation because artifacts and documentation are now inexorably linked.

CASE STUDY: ASSEMBLING OUR FIRST DATA FLYWHEEL FOR PENN STATE'S PROSPECT MANAGEMENT AND ANALYTICS OFFICE

Historically, Penn State's PMA office designated prospects to particular donor pipelines with updates happening only twice a year, often leading to outdated insights. To enhance our university's fundraising endeavors, it was necessary to revolutionize our data-driven decision-making process. Therefore, the goal was to develop a model that could be run nightly to estimate the probability of a donor contributing a major gift over the next 5 years and to forecast their total giving over that same time period.

To meet this goal, we developed an IDS system, changing both our workflow and technology stack. First, using XGBoost (Chen & Guestrin, 2016), we developed a gradient boosted tree ensemble model, which allowed us to consider myriad variables, many more than our traditional models could handle. Being able to use a significantly larger set of variables resulted in more granular and adaptive ratings.

Following the general workflow schema in Figure 1, every morning our data warehouse transfers raw data into the AWS S3 storage service. GitLab CI/CD then initiates an Amazon Elastic Compute Cloud (EC2) instance, which receives the updated data and runs the project code inside the Docker container. This process results in updated prospect scores, which are transferred back to AWS S3, where we capitalize on the synergistic capabilities of AWS S3 and Athena to make the new data available to Power BI, thus making the new scores available to end users, including our research teams and gift officers across fundraising departments via an analytical dashboard.

The recursive nature of this system allows for an adaptive modeling process. With the capacity to retrain our predictive model on a monthly basis and the ability to adjust individual ratings on a daily timeline, our system remains perpetually aligned with the most recent information for an individual donor while also adapting the model to changes in giving patterns across donors over time. For instance, any modifications in a prospect's giving behavior are rapidly assimilated and reflected within a 24-hour window.

This operational shift holds significant implications for our frontline fundraising strategies. Instead of adhering to the previously established semiannual update protocol, our fundraisers have access to real-time analytical insights, which facilitates timely outreach to potential donors, optimizing engagement strategies. Such immediacy not only bolsters donor relations but also instills an ethos of data-driven proactivity.

BEST PRACTICES AND LESSONS LEARNED

Throughout our process of refining our data analytics workflows, we developed several best practices that underpin our success. By applying these strategies, we have managed to reduce errors, increase efficiency, and streamline operations. Below are some of the most useful practices we have developed:

1. **Cloud-Based Development Servers for Code Development**

   Our adoption of cloud-based development servers revolutionized the way we handle code development. By running Docker containers in the cloud and initiating RStudio Server, we can SSH (Secure Shell) tunnel to these servers directly from our work laptops. Crucially, this adoption allows us to develop within the same Docker containers as those used during deployment. The primary benefit of this practice is consistency: If a process functions correctly during development, it is guaranteed to function during deployment. This setup preemptively addresses potential issues, including missing packages, driver discrepancies, or permission errors.

2. **Coherent Setup of Development and Production Resources**

   Both our deployed code and development systems reference AWS S3 buckets to access file dependencies essential for projects and to archive resultant files. We designed a twin structure for development and production buckets. Their architecture is nearly identical, distinguished solely by the "-dev" suffix on development buckets. This mirroring technique facilitates a seamless transition between development and production stages. By manipulating a straightforward environmental variable, we can dictate whether a CI/CD deployment writes project files to development or production.

3. **Output File Compression**

   To optimize storage efficiency and enhance data transfer rates, we consistently maintain all files in the ".csv.gz" format, which not only reduces our storage footprint and associated costs but also accelerates data transfer processes. An added advantage is that these compressed files can often be worked with directly, obviating the need for decompression.

4. **Robust File Versioning for Seamless Recovery**

   Leveraging AWS's capabilities, we instituted folder-level versioning for our files. Given our automation's design principle to generate new file versions daily, this versioning results in the daily update becoming the version of record while maintaining previous versions. It allows us to resolve any data inconsistency or problems that escape the automatic testing facilitated by our CI/CD process because using AWS's versioning ensures that reverting data to a previous state is straightforward. Thus, this feature is a safeguard against unexpected issues, negating the need for downtime while resolving such problems.

5. **Uniform Project Folder Structure for Consistent Development**

   A core element of our development process is the uniformity we maintain across projects, irrespective of project scale or complexity. We embraced the project folder structure advocated by Cookiecutter Data Science (Cookiecutter Data Science). This methodology offers a "logical, reasonably standardized, but flexible project structure for doing and sharing data science work." The structure ensures that every team member knows precisely where to locate specific project components, enhancing collaboration and project handovers.

REFERENCES

Amazon Web Services (n.d.). Available at https://aws.amazon.com/

Azure (n.d.). Available at https://azure.microsoft.com/

Bryan, J. (2018). Excuse me, do you have a moment to talk about version control? *The American Statistician*, *72*(1), 20–27.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). (Association for Computing Machinery).

Collins, J. (2001). *Good to great* (Harper Collins).

Cookiecutter Data Science. Cookiecutter Data Science. Retrieved July 20, 2023, from http://drivendata.github.io/cookiecutter-data-science/

Git (n.d.). Retrieved July 20, 2023, from https://git-scm.com GitHub (n.d.). Available at https://github.com/

GitLab (n.d.). Retrieved July 20, 2023, from https://git-scm.com

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *239*(2), 2.

Python Software Foundation. Retrieved July 20, 2023, from https://www.python.org/

R Core Team (2023). *R: A language and environment for statistical computing* (Vienna: R Foundation for Statistical Computing).

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943.

Understanding GitHub Actions—GitHub Docs. (n.d.). GitHub Docs. Retrieved July 20, 2023, from https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

ABOUT THE AUTHOR

DREW C. WHAM currently serves as the Director of Data Science and Analytics in the Office of Prospect Management and Analytics at the Pennsylvania State University. Leveraging his expertise in advanced machine-learning techniques and modern technical infrastructure and deployment practices, he is working to transform the organization's approach to prospect management. Prior to his current role, he worked with a team to develop data science-based applications that support student success by providing near real-time data and alerts to instructors and advisors. In addition to his current role, Wham helps shape the next generation of data scientists as an adjunct professor at Penn State, where he teaches "Statistical Learning through Computation." Drew Wham earned his PhD in biology from Penn State, specializing in statistical genetics.