Final Project Reflection

Alex G. McCorriston

Northwestern University School of Professional Studies

MSDS 434: Data Science and Cloud Computing

Dr. David Alfred Ostrowski

March 16, 2025

1.0 Application Overview

The Pulmonary Embolism (PE) Prediction application is a machine learning-based system designed to assess PE risk using structured electronic health record data. It is built using Amazon Web Services (AWS) SageMaker for model training, FastAPI for application programming interface (API) deployment, Docker for containerization, and GitHub Actions for continuous integration and continuous delivery/deployment (CI/CD) automation. The system integrates Prometheus and Grafana to monitor API request counts and latencies. The model is trained on historical clinical data from the Beth Israel Deaconess Medical Center and is stored in AWS S3, from which it is accessed by the API for inference. The API allows users to input clinical data in JavaScript Object Notation (JSON) format and receive a prediction of whether a patient is at risk for PE. GitHub Actions automates code linting, formatting, and pushing code updates, ensuring that changes to the repository are applied consistently. Docker ensures environment consistency, simplifying deployment across different systems.

2. Application Advantages & Disadvantages

One of the key strengths of the application is its cloud-based scalability. AWS SageMaker enables efficient model training and inference, while FastAPI provides a lightweight, low-latency backend. The use of GitHub Actions for CI/CD automation ensures that code quality is maintained through linting and formatting. The inclusion of Prometheus and Grafana allows real-time monitoring of API performance metrics.

However, the system also has several limitations. The model includes too many features, increasing complexity and potentially introducing noise, so reducing the number of features could improve the model's efficiency and interpretability. Additionally, the API does not

currently provide users with a list of expected feature names, making it difficult to format input data correctly. Another issue is that the model produces a binary prediction rather than a probability score. This limits the ability of application users to interpret the level of risk and make informed decisions. Also, since the model is based on static historical data without periodic retraining, it may become outdated if clinical practices and patient populations shift over time. Without an automated retraining mechanism, the model's predictions could lose accuracy if newer clinical data deviates from the training distribution.

3. Areas for Improvement

To enhance usability, the system should expose feature names to API users as this would guide input data formatting. Feature reduction should also be implemented to improve model interpretability. Additionally, the model's output should be adjusted to provide probability scores instead of binary predictions. The system's monitoring should be expanded to include error rate tracking. Currently, Prometheus tracks API request counts and latencies, but adding error metrics would help detect failures more efficiently. Automating model updates is another critical improvement. Without periodic retraining, the model's predictions may become less reliable over time. Implementing a machine learning operations (MLOps) pipeline that retrains and evaluates the model at specified intervals would help ensure long-term accuracy. Furthermore, while GitHub Actions currently automates code linting, formatting, and pushing updates, the CI/CD pipeline could be expanded to automate the Docker deployment process. Automating the rebuilding of Docker images and running Docker Compose when new code changes are pushed would streamline deployment and reduce manual intervention.

4. New Technology Considerations

Several new technologies could improve the application's performance and ease of maintenance. Amazon Elastic Beanstalk is a managed service that simplifies deploying and running applications by handling infrastructure tasks like server setup, scaling, and monitoring. Instead of manually configuring an EC2 instance, Elastic Beanstalk automates these processes, making it easier to manage and scale the application as more users access it. For even greater scalability and reliability, Kubernetes could be used instead of Docker Compose to manage containers. Kubernetes automates key tasks such as distributing workloads across multiple machines, handling traffic spikes, and restarting failed processes. This would make the system more resilient, ensuring that it stays online even if some parts fail. By implementing these technologies, the application could run more efficiently with less manual effort, making it better suited for long-term use in production environments.