DataScientest • com

MLOps: Heartbeat Classification

Context and Objectives

1.1 What problem should the application address?

The application should address the problem of automating the classification of ECG signals to differentiate between normal cardiac rhythms and various arrhythmias and myocardial infarctions. This will aid in enhancing diagnostic accuracy, reducing manual analysis, and mitigate the risk of diagnostic errors in the context of cardiovascular diseases. We will develop and deploy a comprehensive Heartbeat Classification API system, focusing on accuracy, scalability, and maintainability. Our objectives for this project include:

1. **Repository Setup:** Establish a well-structured GitHub repository with folder organization in order to achieve final solution architecture.
2. **Database Integration:** Integrate a suitable database onto our API.
3. **API Implementation:** Create the first version of the API incorporating the trained model, with clear usage instructions provided.
4. **Unit Testing Integration:** Integrate unit tests into the GitHub Actions testing pipeline to validate functionality across model training, API endpoints, and database.
5. **API Endpoints:** Implement various API endpoints to support different functionalities.
6. **User Authentication:** Provide secure access to an API to ensure data access/usage is aligned to the current security standards.
7. **Containerization:** Containerize the API, models, and database using MLFlow Web Server to facilitate deployment and scalability.
8. **Evaluation and Monitoring:** Evaluate model performance under production conditions, monitoring effectiveness and efficiency.
9. **Model Maintenance:** Implement mechanisms to re-train the model as necessary to ensure consistent performance.
10. **Automate Processes:** Automate tasks from engineering and model evaluation phases to ensure continuous integration, training, and deployment.

1.2 Who is the app sponsor?

The app sponsor is likely the business or organization that aims to integrate the machine learning model into their ECG machines. This could be a healthcare technology company or a medical device manufacturer focused on automating and improving diagnostic tools for cardiovascular diseases.

Decision-making regarding the model's output will be handled by the medical staff. While we cannot predict their exact approach, we recommend the following:

1. Use the model to expedite decision-making, particularly when analyzing large volumes of data simultaneously.

2. Establish a threshold for flagging patients who require additional review for potential heart failure or arrhythmias. It is important to note that we do not set this threshold ourselves due to ethical considerations.

1.3 Who will be the user of the application?

The primary users of the application will consist of healthcare professionals, including cardiologists, medical technicians, and other healthcare providers responsible for conducting and analyzing ECG tests. Data scientists and engineers may utilize the application for purposes such as model monitoring and enhancement.

End-users encompass individuals employed within the medical field. Their objective is to optimize the accuracy of heartbeat classification and streamline the manual process of classification. These users derive significant advantages from this ML API integration as it provides an initial estimation of the observed heartbeat classification.

In the event that the model identifies heartbeat patterns surpassing a predefined threshold, it triggers an alert to the medical staff, prompting a re-assessment.

1.4 Who will be the application administrator?

The application administrator will typically be an IT professional or a data scientist within the healthcare organization or company that is able to deploy the application. Their responsibilities will include managing the deployment, monitoring performance, handling updates, and ensuring the system's reliability and security.

1.5 In what context will the application have to be integrated (website, software, cloud server, etc.)?

The application will be integrated into API, which may be hosted on local servers within healthcare facilities or on cloud servers for broader access and scalability.

1.6 Via what medium will the application be used (graphical interface, terminal, API, etc.)?

API docs will be used for graphical interface and programmatic access to the model's functionality. If a healthcare provider chooses to integrate this into a website, it would be within their remit to integrate it with their own healthcare systems or software.

———————————————————————————————————

Model

———————————————————————————————————

2.1 The type of model used and a quick explanation of how it works and its general performance

The project employs both machine learning and deep learning models. In terms of deep learning models, convolutional neural networks (CNNs) will be used for ECG signal analysis. CNNs work by automatically learning features from the ECG data to classify different types of heartbeats. The performance of these models is generally evaluated based on their accuracy in correctly classifying heartbeats, robustness to variations in the data, and their ability to generalize to unseen data.

In our previous report our advanced CNN-4 model resulted in an accuracy of 98.57% on the original MIT-BIH dataset, while the advanced CNN-4 model trained on the oversampled dataset attained an accuracy of 98.1%. Our findings in machine learning (ML) versus deep learning (DL) reveal more trends. XGBoost emerged as the leading performer in ML, achieving an accuracy of 98.20% on the original dataset, while Random Forest outperformed other ML models with an accuracy of 98.11% on the oversampled MIT-BIH dataset. Among all traditional machine learning models, the best-performing models tend to be Random Forest and XGB.

Models can be deployed directly to microcontrollers or to an API accessible by the microcontrollers. The training and testing data were sourced from the original dataset, as detailed in the chapter on "Data Collection / Data Sources." If the model predicts false negatives, individuals may suffer from undiagnosed heartbeat failures.

We have not yet implemented a fairness constraint, as this would require extensive collaboration with medical staff to define such constraints and thresholds, which would influence our raw output.

2.2 The definition of evaluation metrics of the model with respect to the constraints of the project (accuracy, robustness, training time, prediction time, etc.)

The definition of evaluation metrics of the model with respect to the constraints of the project (accuracy, robustness, training time, prediction time, etc.):

- **Accuracy:** The proportion of correctly classified heartbeats out of the total number of heartbeats. This is crucial as it directly impacts diagnostic reliability.
- **Training Time:** The time required to train the model. While not a primary evaluation metric, it is considered to optimize resource usage and model updates.
- **Prediction Time:** The time taken to classify a heartbeat during real-time analysis. This metric is critical for the application's usability in a clinical setting where timely diagnosis is essential.

In summary, the application aims to automate and enhance the accuracy of ECG signal analysis to aid in the diagnosis of cardiovascular diseases. It will be used primarily by healthcare professionals and administered by IT or data science personnel. The application will be integrated into an API. Evaluation metrics focus on accuracy and the efficiency of the model in terms of training and prediction times, ensuring it meets the clinical demands and provides reliable diagnostic support.

Database

3.1 Data Collection

The initial dataset is obtained from Kaggle: Heartbeat Dataset.

Proposed Workflow Outside of the Microcontroller (MLflow):

- All data is consolidated, including feedback from medical staff when a heartbeat anomaly is detected. A routine is implemented to compare the current version of the model against new data and feedback from medical staff. The model is then retrained, potentially based on a specified number of incorrect predictions or after accumulating a certain amount of new data.

Note: Continuous dataflowto the model cannot be fully simulated at this stage; oversampling and a splitting and feeding in function may be used to approximate this process.

- Real-time prediction is intended for use by medical staff, while batch analysis is used for training the model to improve accuracy. Model training takes approximately 2-5 hours, depending on computing power and the size of the training set. Prediction time for a single heartbeat is negligible. No post-processing is done, as the desired heartbeat classifications are directly produced by the model.

3.2 Data Sources

Due to the lack of expert knowledge required for labeling new data, only oversampling can be used to generate "new data."

Raw data is readily available but requires labeling to be useful as training data. Labeling is time-consuming and costly, requiring medical experts. There is no online library with continuously updated labeled data, nor is non-labeled data readily available online.

API

4.1 API Setup Outline

Step 1: Preparation

- Ensure CSV file containing heartbeat dataset is accessible and loaded.
- Set up a FastAPI environment for API development.

Step 2: User Management

1. **Creation of New User:** Implement functionality to register new users with appropriate credentials.

2. **Credentials Check:** Perform validation of user credentials.
3. **Identification Check:** Verify user identity to ensure access is granted to the right users.

Step 3: Interface Setup

- Integrate FastAPI docs to provide a comprehensive API guide that is easy to follow.

Step 4: Functionality Implementation

- **Real-Time Prediction:**
  - Randomly select a row from the CSV data and predict its outcome using predefined models. Log the outcome aswell as the general routine (including errors).
- **Retraining:**
  - Allow parameter selection as an input criteria for model retraining.
  - Store selected parameters in a file and update model registry accordingly.
- **[Optional] Notification:**
  - Implement functionality to send email notifications.
- **Model Deployment:**
  - Develop a pipeline for model training, transformation, and validation.
  - Conduct integration tests using pytest.
  - Update path to production model in both application logs and model registry.
- **Monitoring:**
  - Track current model metrics.
  - Show the logged values in app.log.
- **[Optional] Comprehensive Monitoring:**
  - Provide an optional feature to monitor all model metrics.

Step 5: Gateway Setup

- **Version 1 (V1):** Gateway API directly executes all endpoints without involving additional APIs.
- **Version 2 (V2):** Gateway API initiates calls to other APIs for endpoint execution. Gateway API stores all databases and log files and is responsible for authentication.

By following this approach, we ensure the seamless setup and functionality of the API, catering to the needs of our users and facilitating efficient management of the heartbeat dataset prediction system.

Testing & Monitoring

———————————————————————————————————

During the deployment of the application, it will be necessary to pay particular attention to the fact that the different parts of the project work correctly individually (unit tests), and that the performance of the application is always in line with the specifications. We will attempt to conduct the following tests:

5.1 Unit Tests Implementation:

1. **Model Training Functionality Test:**
    - Verify that the model trains without errors.
2. **Model Prediction Functionality Test:**
    - Validate that the model accurately predicts heartbeat types.
    - Assess the consistency of predictions across different input data.
3. **API Endpoint Functionality Test:**
    - Test each API endpoint to ensure they handle requests correctly. Ensure appropriate error notifications are raised in case of any issues.
    - Verify that appropriate responses are returned for different scenarios.
4. **New Data Inputs Process Test:**
    - Validate that new data is successfully accepted and stored in the system.
    - Check if the input data aligns with expected formats and structures. Raise errors where necessary.

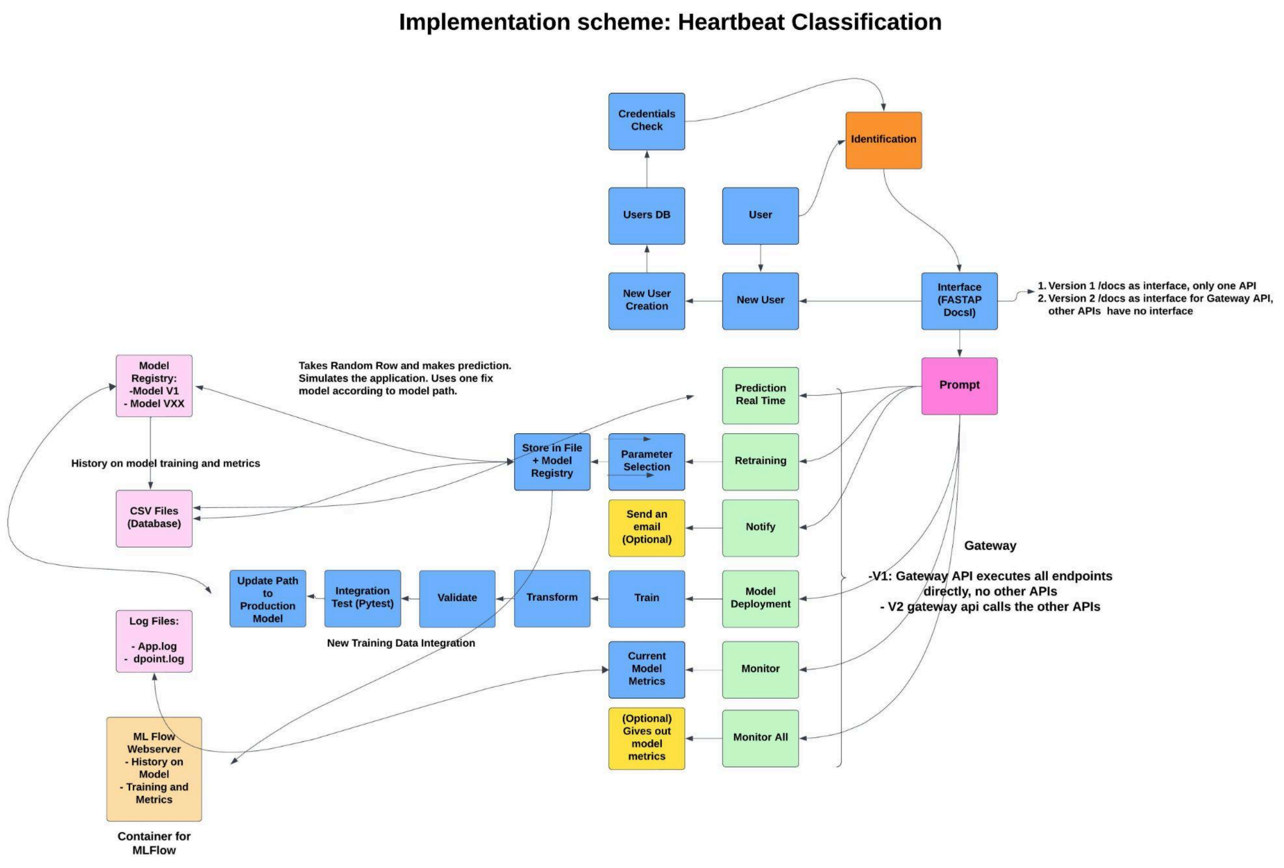
5.2 Model Monitoring and Decision Making:

1. **Model Performance Evaluation:**
    - Evaluate the model's performance periodically using metrics such as accuracy, precision and recall.
    - Assess performance on the entire test set and compare it with performance on the most recent data to identify any deviations.
2. **Retraining Triggers:**
    - Retrain when model performance falls below a certain threshold, indicating degradation in predictive accuracy.
    - Determine whether to retrain the model on the entire dataset or only on a sample of the most recent data.
3. **Response to Model Performance Issues:**
    - When the model fails to meet the required performance threshold, trigger actions such as sending email alerts to relevant individuals.
    - Consider implementing automated measures to temporarily suspend application functionality until the model's performance is restored to an acceptable level.

By incorporating these tests and monitoring procedures, we ensure the reliability, robustness, and continuous improvement of the heartbeat prediction system, thereby enhancing its effectiveness in healthcare applications.

Implementation scheme

Link


Implementation scheme: Heartbeat Classification
Credentials Check
Identification
Users DB
User
New User Creation
New User
Interface (FASTAP Docsl)
1. Version 1 /docs as interface, only one API
2. Version 2 /docs as interface for Gateway API, other APIs have no interface
Prompt
Model Registry: -Model V1 - Model VXX
Takes Random Row and makes prediction. Simulates the application. Uses one fix model according to model path.
Prediction Real Time
History on model training and metrics
Store in File + Model Registry
Parameter Selection
Retraining
CSV Files (Database)
Send an email (Optional)
Notify
Gateway
-V1: Gateway API executes all endpoints directly, no other APIs
- V2 gateway api calls the other APIs
Update Path to Production Model
Integration Test (Pytest)
Validate
Transform
Train
Model Deployment
Log Files:
- App.log
- dpoint.log
New Training Data Integration
Current Model Metrics
Monitor
(Optional) Gives out model metrics
Monitor All
ML Flow Webserver - History on Model - Training and Metrics
Container for MLFlow