Take-Home Technical Test: ECG Heartbeat Categorization

Thank you for your interest in the machine learning engineer position. As part of the application process, we would like you to complete the following tasks using the ECG Heartbeat Categorization Dataset obtained from Kaggle.

Introduction

Please note that this is an open-ended test, and we encourage you to demonstrate your expertise and creativity throughout the process. Feel free to use any programming language or machine learning framework of your choice.

Data description

The dataset used in this project is focused on heartbeat classification using deep neural network architectures and explores the capabilities of transfer learning. It consists of electrocardiogram (ECG) signals representing different types of heartbeats. These heartbeats include normal cases as well as those affected by arrhythmias and myocardial infarction. The dataset has been preprocessed and segmented, with each segment corresponding to a single heartbeat.

The Arrhythmia Dataset contains a total of 109,446 samples and is categorized into five classes. The sampling frequency of the ECG signals is 125 Hz, which indicates that the signals were recorded at a rate of 125 samples per second. The dataset is sourced from Physionet's MIT-BIH Arrhythmia Dataset, a well-known resource in the field of biomedical signal processing. The classes in this dataset are represented by the following labels: 'N' (0), 'S' (1), 'V' (2), 'F' (3), and 'Q' (4). These labels correspond to different types of arrhythmias and myocardial infarction cases.

You can download the dataset from kaggle.

Technical details

Task 1: Data Processing

- Perform an Exploratory Data Analysis (EDA) on the ECG Heartbeat Categorization Dataset. Provide insights into the distribution of classes, statistical properties of the features, and any other relevant observations.
- Apply data augmentation techniques to enhance the diversity of the dataset. Explain the rationale behind the chosen techniques and how they contribute to improving the model's performance.
- Perform feature engineering to extract meaningful information from the raw ECG data. Select and justify the features you believe are most relevant for the heartbeat categorization task.

- Handle missing data by applying appropriate data imputation techniques. Discuss the impact of missing data on the model's performance and the reasons behind your chosen imputation method.

Task 2: Model Training and Fine-tuning

- Choose a suitable model architecture for the ECG heartbeat categorization task. Explain your choice and discuss the advantages and potential challenges associated with the selected model.
- Perform hyperparameter tuning to optimize the model's performance. Clearly state the hyperparameters you have selected for tuning and describe the methodology used. You are free to use any tool(s) you deem necessary (MLFlow, Hydra…)
- Address issues related to overfitting and underfitting during the training process. Explain the techniques you have employed to cope with these challenges.
- Implement early stopping and training callbacks to improve the training efficiency and prevent overfitting. Discuss the rationale behind their usage.
- Evaluate the trained model using appropriate evaluation metrics. Provide a detailed analysis of the model's performance and discuss any limitations or potential areas for improvement. You are free to use any tool(s) you deem necessary (MLFlow, DVC, Hydra…)

(Optional) Task 3: Testing Holdout Set

- Create a holdout set from the ECG Heartbeat Categorization Dataset to simulate data shifts and performance degradation. The holdout set is different from the validation and testing set.
- Apply the trained model on the holdout set and evaluate its performance. Analyze any differences in performance compared to the training/validation/test results and discuss potential causes.

Task 4: Deployment Strategies

- Propose deployment strategies for a production-ready solution based on the trained model. Discuss the considerations, challenges, and best practices involved in deploying a machine learning model in a real-world scenario. You are free to use any tool(s) you deem necessary.
- Describe how you would handle model versioning, scalability, and monitoring in the deployed system. Discuss any potential risks and mitigation strategies.

Please document your work in a well-organized manner, including clear explanations, code snippets, visualizations, and any other relevant materials. Feel free to include additional information or experiments to showcase your skills and expertise. You have one week to complete the test.

Submitting your work

To submit your project, please follow the steps below:

- Push your work into a private GitHub repository: Create a **private** repository on GitHub and push all your project files, code, documentation, and any other relevant materials into the repository. Ensure that your repository is well-organized and includes clear instructions on how to run your code and reproduce your results.

- Add the AIM's ML specialist as a viewer by GitHub tag: In order to grant access to the AIM's ML specialist, add them as a viewer to your private repository using his github tag "@I-Akrout".

Once you have completed these steps, please inform AIM's recruitment team about the completion of your project and provide the necessary details for accessing your repository. This will allow the ML specialist to review your work thoroughly.

Suggestions

It is highly recommended to follow the cookiecutter data science structure for organizing machine learning projects due to its numerous benefits. The Cookiecutter data structure provides a standardized and organized framework that promotes reproducibility, collaboration, and maintainability in ML projects. This facilitates collaboration among team members and promotes best practices in version control, enabling efficient code sharing and development. Additionally, the Cookiecutter structure encourages documentation, making it easier to understand the project's purpose, data sources, and model architectures.

The length of explanations and discussions can vary depending on the desired depth of explanation and the level of detail desired. Short explanations provide a concise overview of the concepts, methodologies, or techniques being discussed, highlighting the key points and main ideas. On the other hand, long explanations offer a more comprehensive and detailed analysis, covering various aspects, nuances, and supporting evidence related to the topic. They delve deeper into the subject matter, providing a thorough exploration of concepts, methodologies, or techniques.

Deploying a machine learning model and establishing a clear deployment pipeline hold greater significance than solely focusing on model performance within a Jupyter Notebook. While model performance is crucial for achieving accurate predictions during development, deploying a model involves additional considerations such as scalability, reliability, and integration into a production environment. A well-defined deployment pipeline ensures that the model can be seamlessly incorporated into real-world applications, taking into account factors like data preprocessing, feature extraction, model versioning, scalability, monitoring, and ongoing maintenance.