**Лабораторна робота №6**

**Виконав студент групи КН-316**

**Гуня Василь**

(1) Managing/tracking experiments: Logging experiments’ parameters and results with some custom code

Для виконання цього завдання я використав **MLflow.** Для початку необхідно встановити платформу MLflow. Для цього я використав пакетний менеджер pip (перед цим створив віртуальне середовище для роботи з Python використовуючи venv та активуємо його командою).



Далі для перевірки роботи платформи я використав простий код щоб перевірити чи працює UI MLflow.

**Collaboration:** use MLflow web interface to visualize and compare experiments

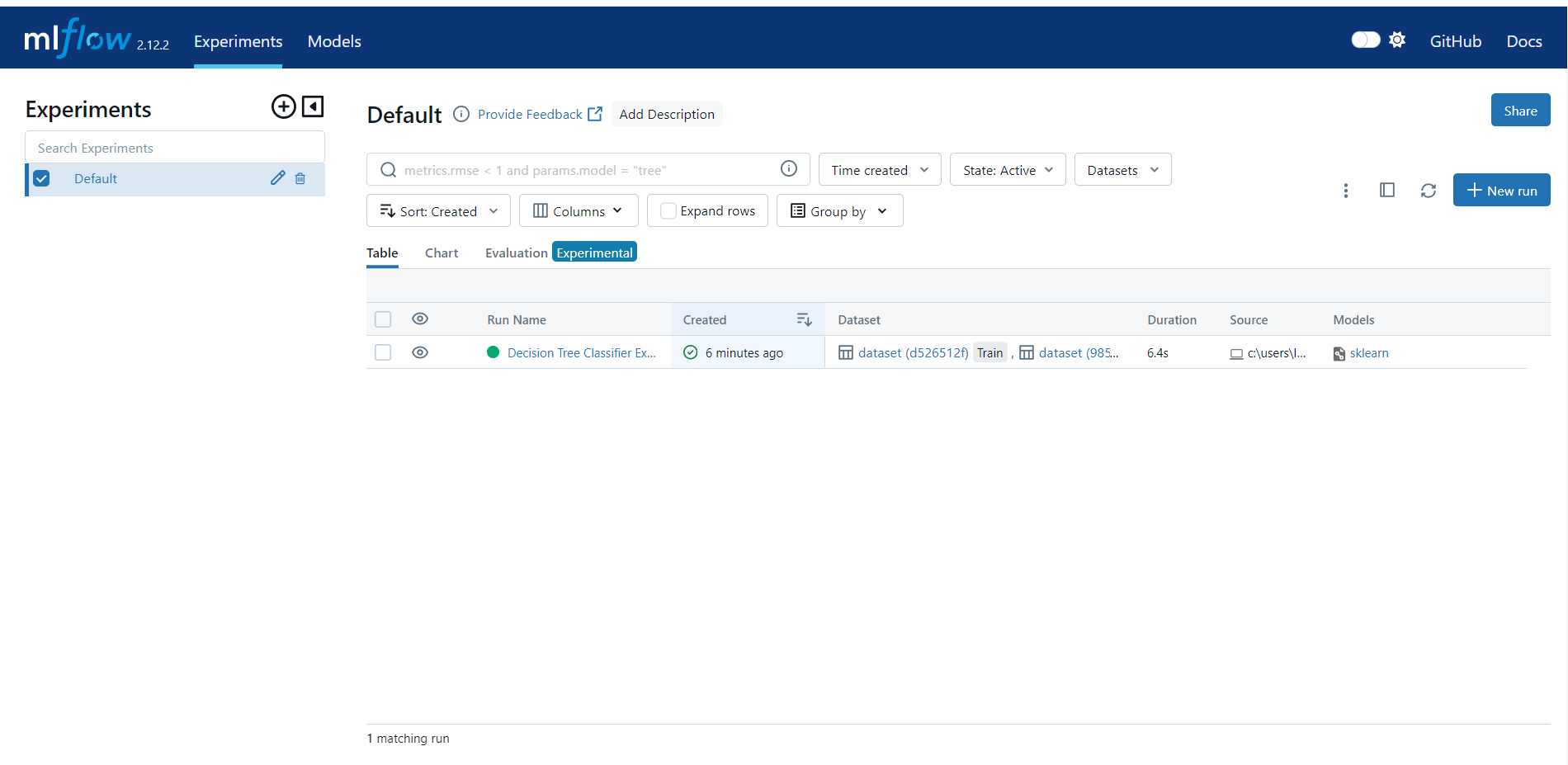
****

Рис. 1 UI MLflow

Тепер натренуємо Dense Neural Network для передбачення серцевих захворювань.

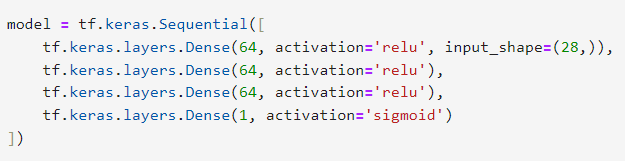


Рис. 2 Архітектура нейромережі

Я провів 5 різних експериментів, змінюючи бач сайз, льорнінг рейт та кількість шарів, використовуючи MLFlow зберіг метрки, криві навчання, назву датасету та посилання на нього та інші логи.

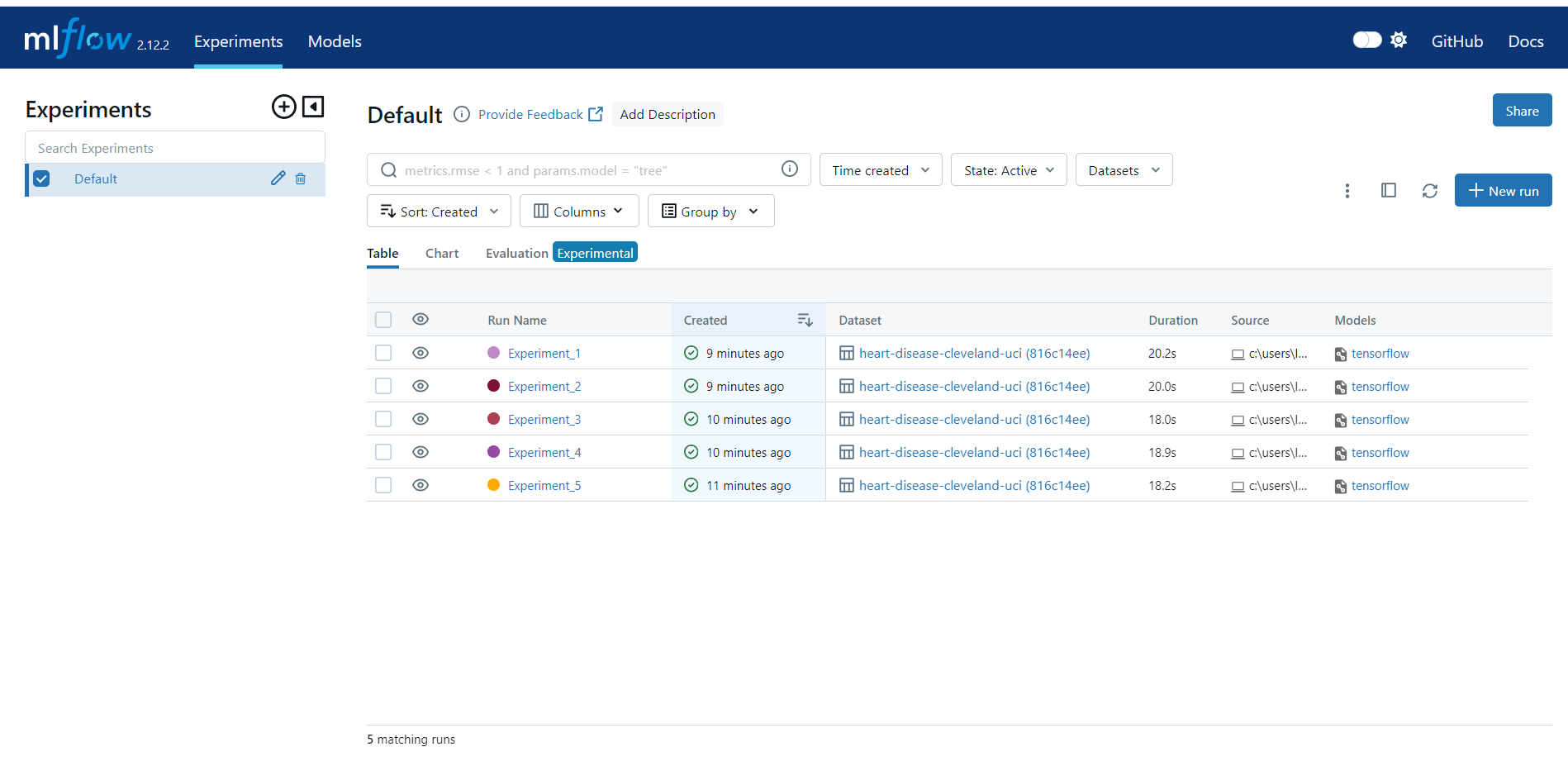


Рис. 3 Логи експериментів в MLflow

Тут ми можемо переглянути рейтинг усіх моделей на основі лосів, точності тощо.

***Experiment Tracking****:* log and query experiments to track metrics, parameters, and models.

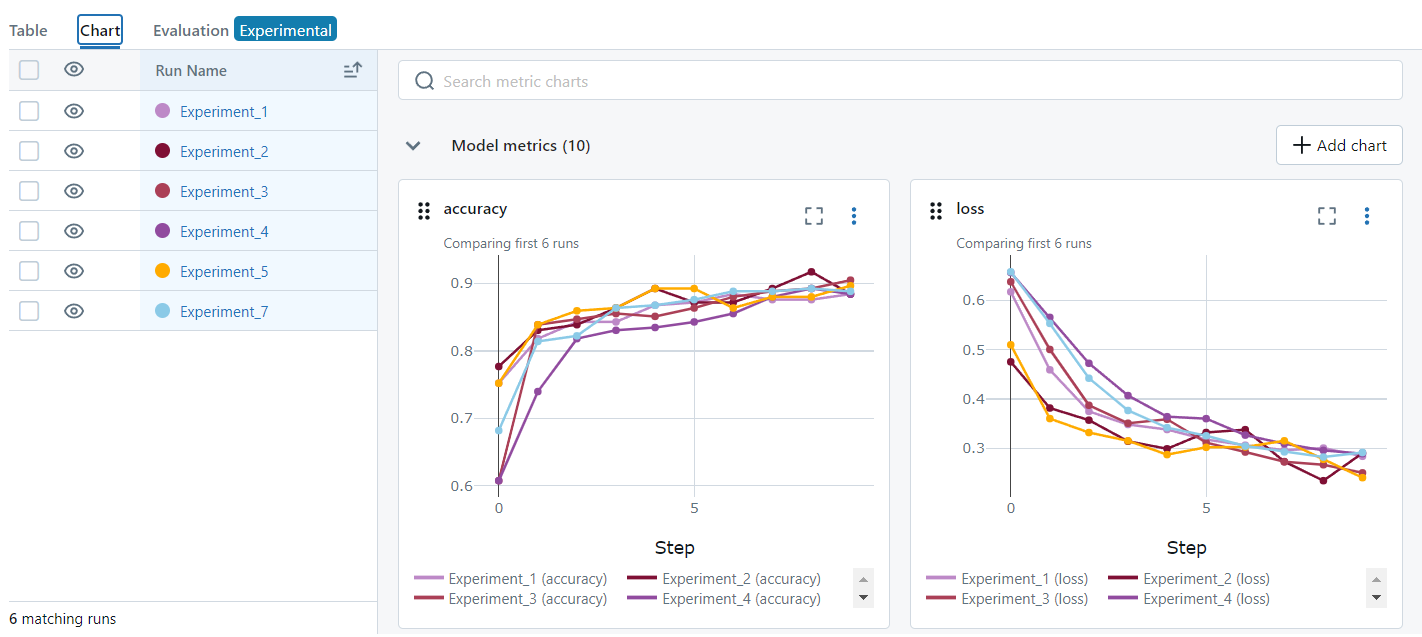


Рис. 4 Рейтинг моделей

Також можна переглянути параметри моделі

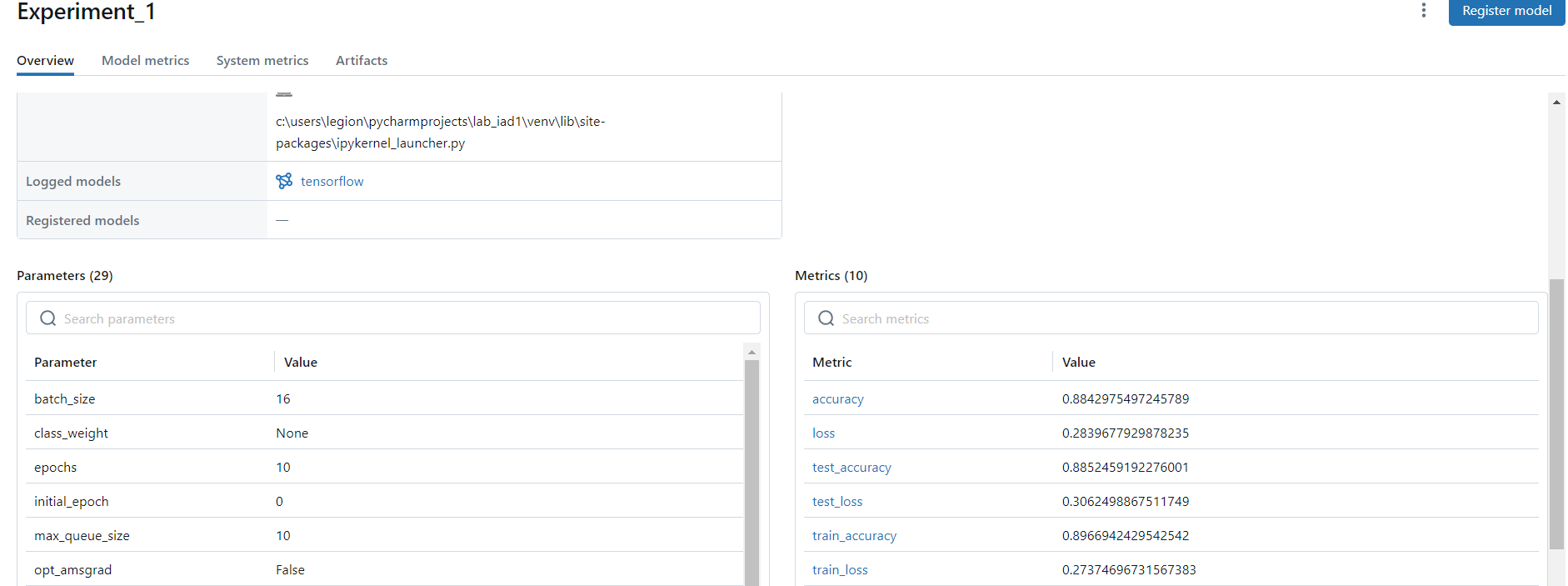


Рис. 5 Параметри моделі

***Artefact Tracking****:* log and store artifacts such as models, datasets, plots, and otheroutput files.

Також можемо переглянути збережені артефакти. Можемо бачити збережений репорт у якому знаходяться такі метрики, як recall, precision, f1, тощо. Також присутні рисунки кривих навчання та тренування та різні метадані моделі.

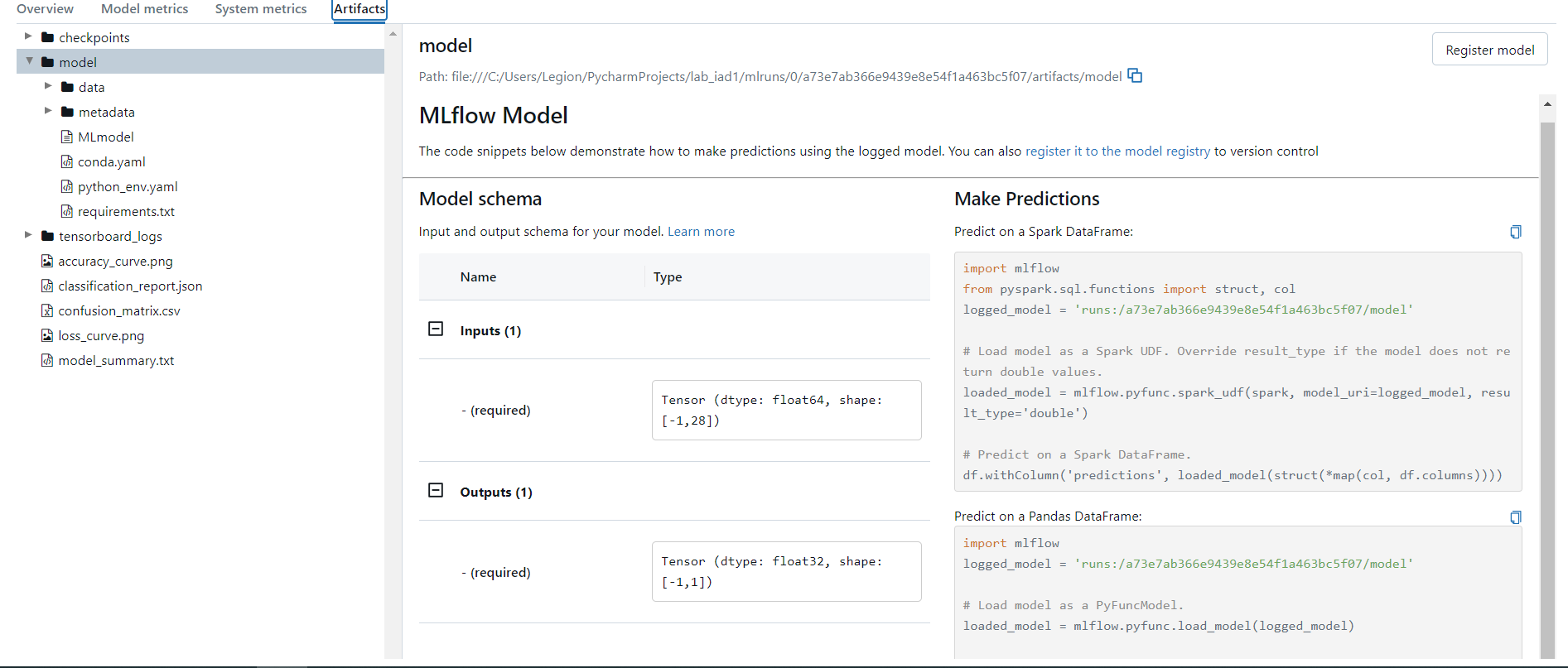


Рис. 6 Артефакти моделі

***Model Registry****:* keep track of model lineage by tracking versions of machine learning models and transiting them between stages (dev, test, prod).

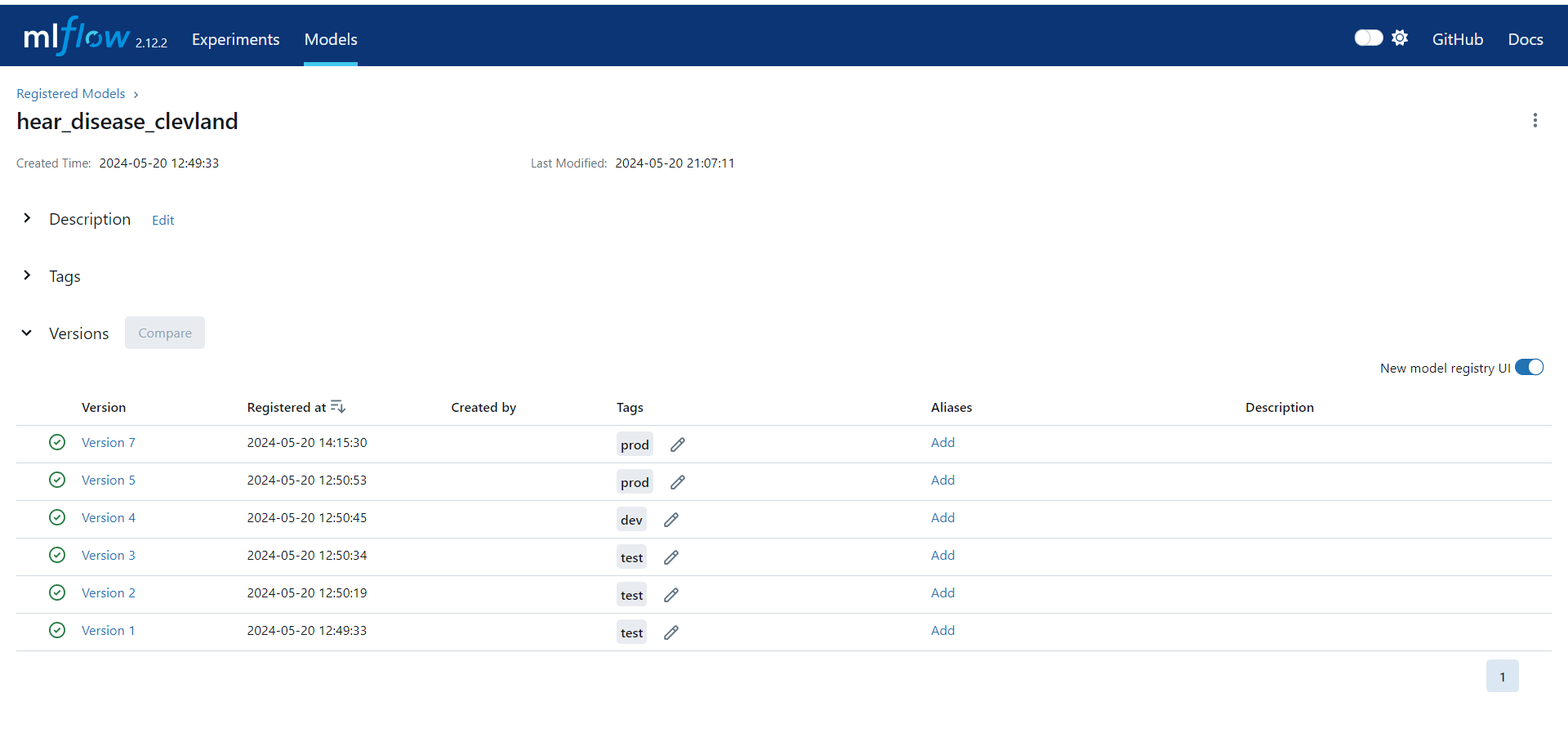


Рис. 7 Зареєстровані моделі та їх версії

***Deployment****:* deploy model as REST API to various environments (GCP, AWS, Azure, local premises)

***Packaging****:* package code into a reproducible project (a directory or a git repository containing code, dependencies, and MLproject file)

Я створив git repository для зберігання проекту та керування версіями. Ось посилання на нього: <https://github.com/Vasyl808/mlops>

Також тут можемо бачити як використовувати ту чи іншу модель. Побудуємо REST API для взаємодії з нашою моделлю, використаємо фреймворк Flask. У ході роботи я також натренував енкодер для ван хот кодування та скейлер, які оброблять вхідні дані від користувача. Нижче подано реалізацію цих функцій.

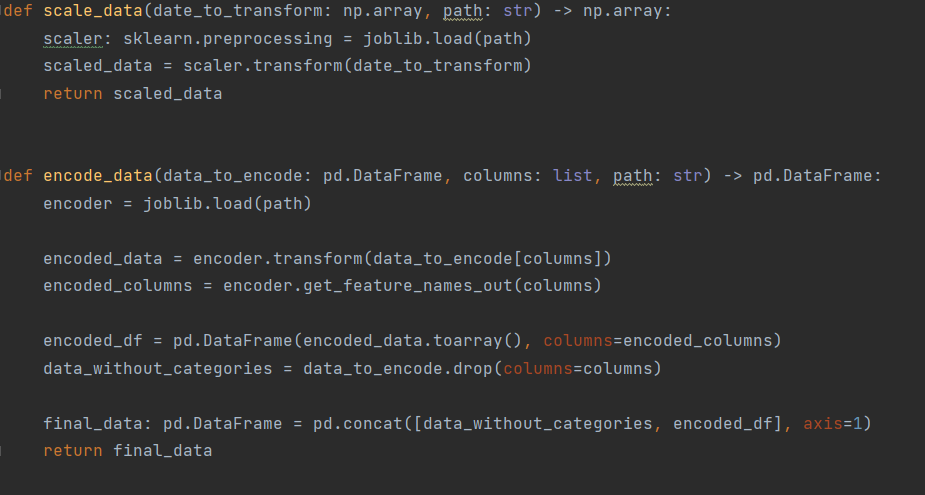


Рис. 8 Використання скелеру та енкодера для препроцесінгу даних

Можемо протестувати наш API, у нас задачі бінарної класифікації, тому модель повертає або true якщо є підозри що людина має серцеві захворювання, відповідно false коли вони відсутні.

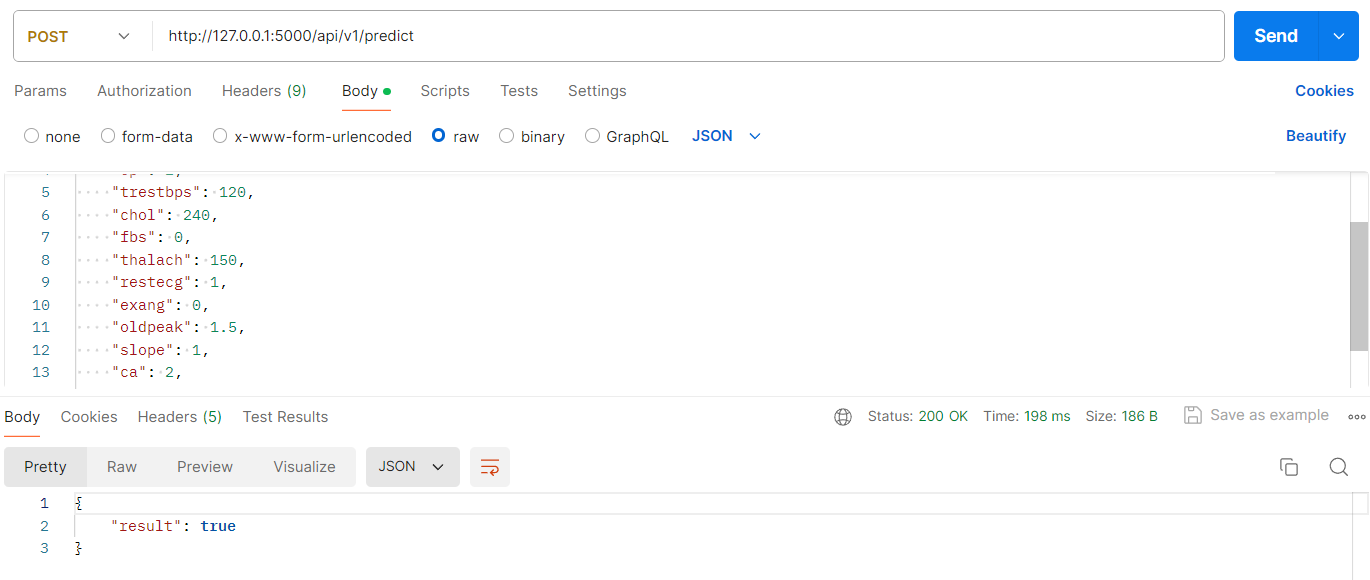


Рис. 9 Приклад роботи API з використанням Postman

Basic CI/CD configuration: model deployment and integration into the development workflow (GitHub Actions or similar).

Для CI/CD я використав GitHub Actions

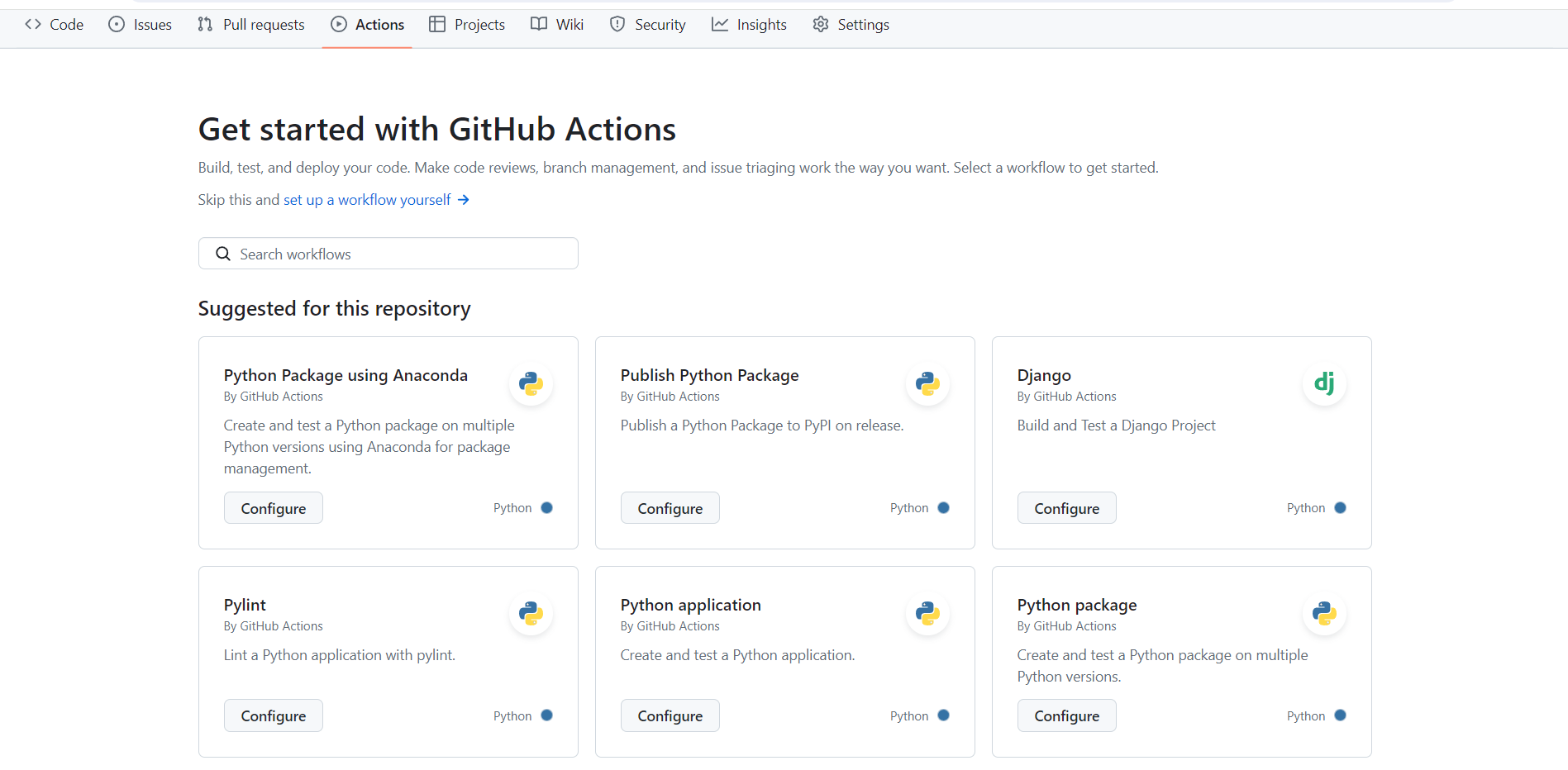


Рис. 10 GitHub Actions

Automated model retrain, validation, and selection of the best model.

Я додав скрипт, який автоматично перетреновує мою модель та зберігає їх історію, його я запускаю при пайплайні що розгортає мій проект.

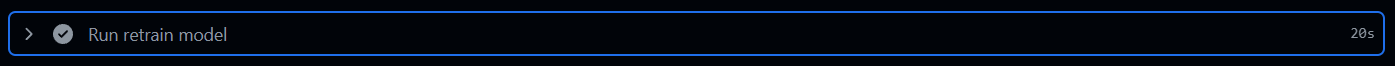


Рис. 11 Успішне виконання скрипта

Ось тригер для виконання пайплану.

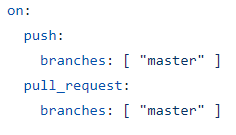


Рис. 12 Тригер для виконання пайплайну

Тут подано кроки, які виконується перед запуском API.

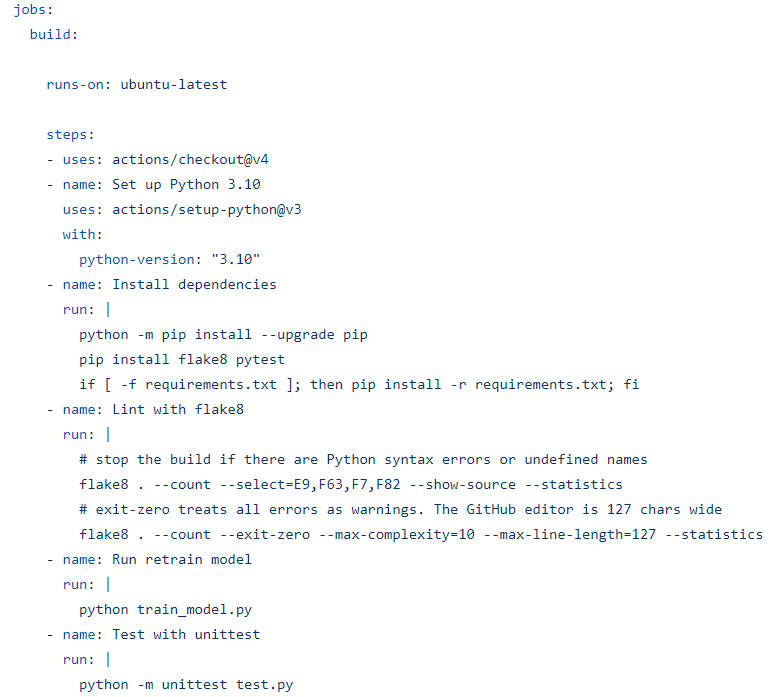


Рис. 13 Кроки розгортання проекту

Extra points

Advanced CI/CD: automated code style checks, testing, deployment, A/B testing, and rollbacks.

По перше, ми встановлюємо усі залежності, далі перевіряємо код стиль, використовуючи flake8. flake8 — це інструмент для перевірки якості коду на Python. Потім запускаємо юніт тестування нашого API. Можемо бачити що білд пройшов успішно та всі кроки були виконані.

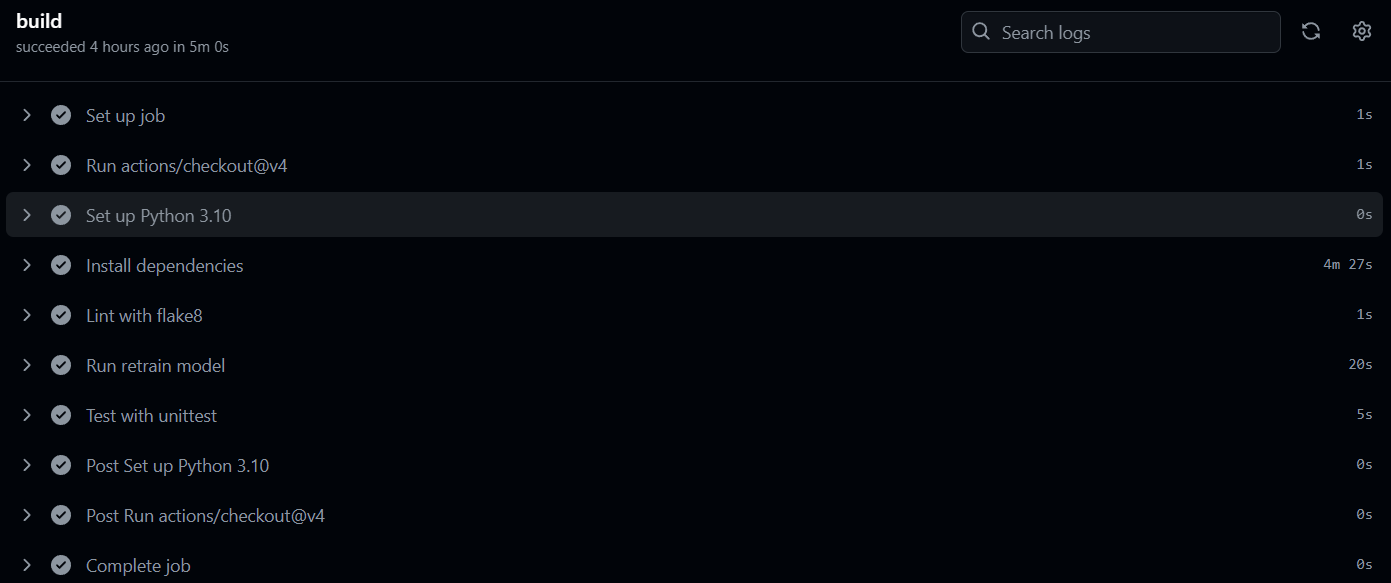


Рис. 14 Звіт про успішне виконання

Knowledge of compliance, data privacy, requirements and ability to implement them in the code.

Я використав SonarQube. Це платформа для перевірки якості коду, яка також включає правила для безпеки та конфіденційності даних.

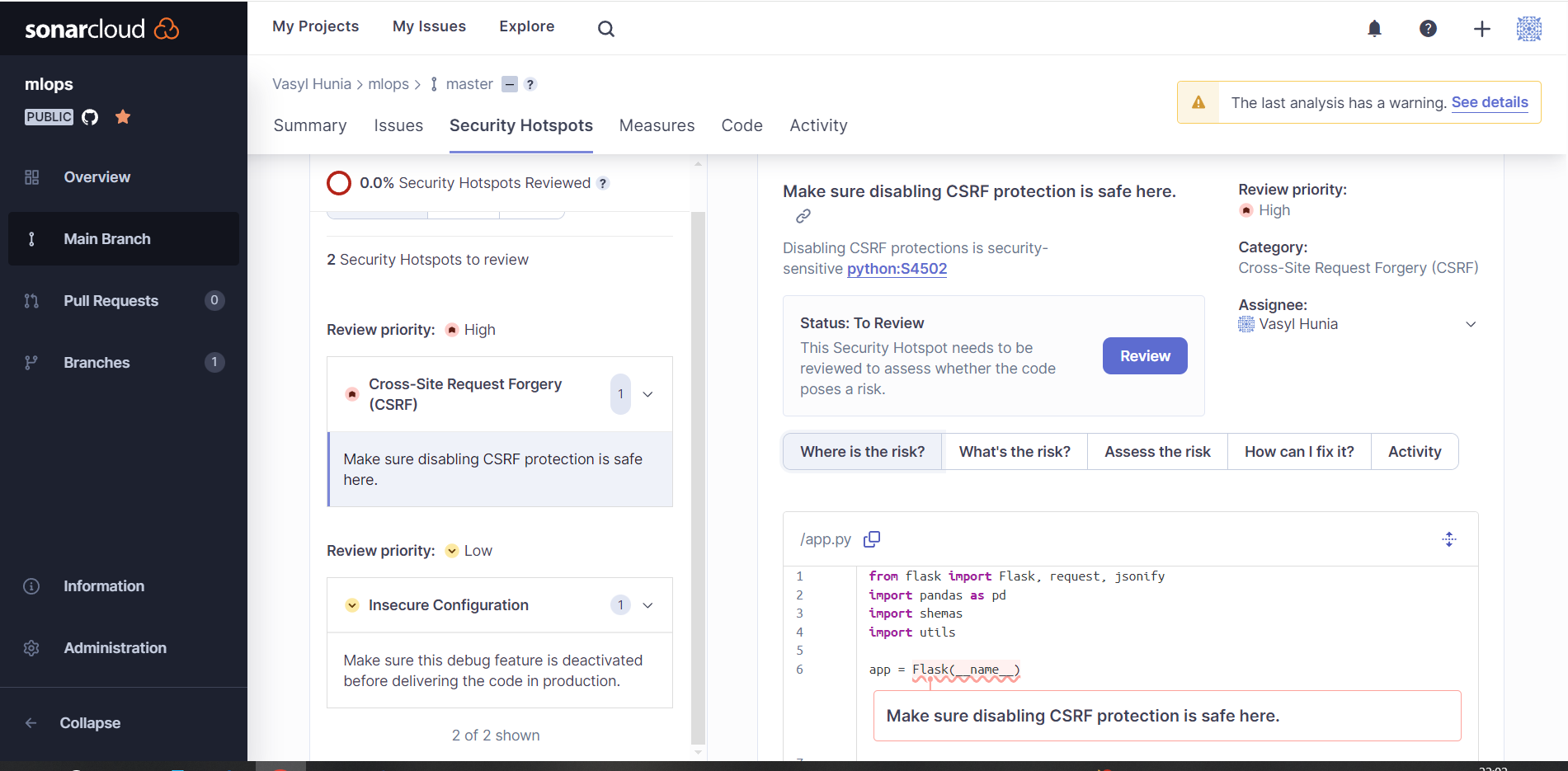


Рис. 14 Приклад роботи SonarQube

Можемо бачити повідомлення про проблему CSRF. Тут для вирішення проблеми потрібно додати CSRF унікальний токен до кожного чутливого запиту, який перевіряється сервером. Також сам інструмент надає приклад як вирішити цю вразливість.

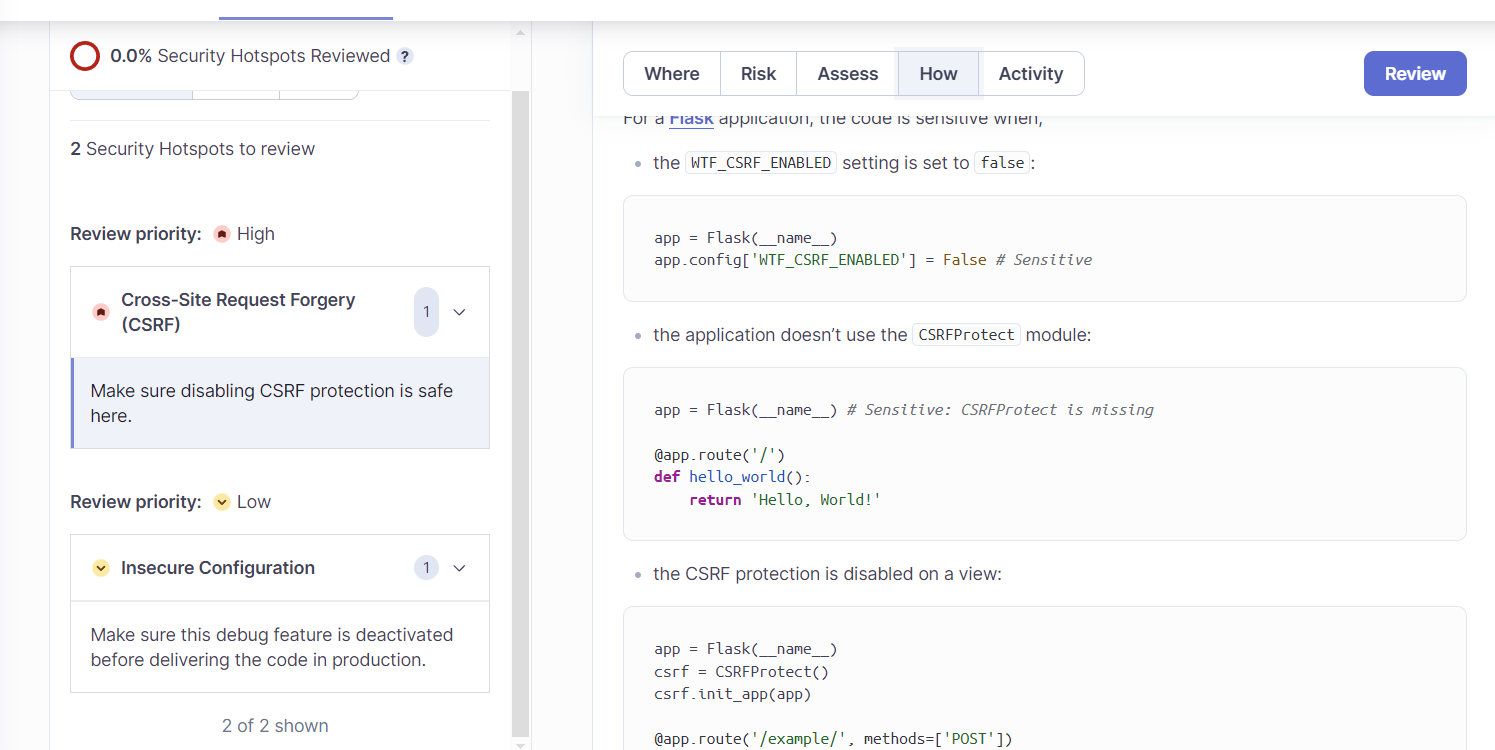


Рис. 15 Fix від SonarQube

Нижче подано детальний звіт по проекту.

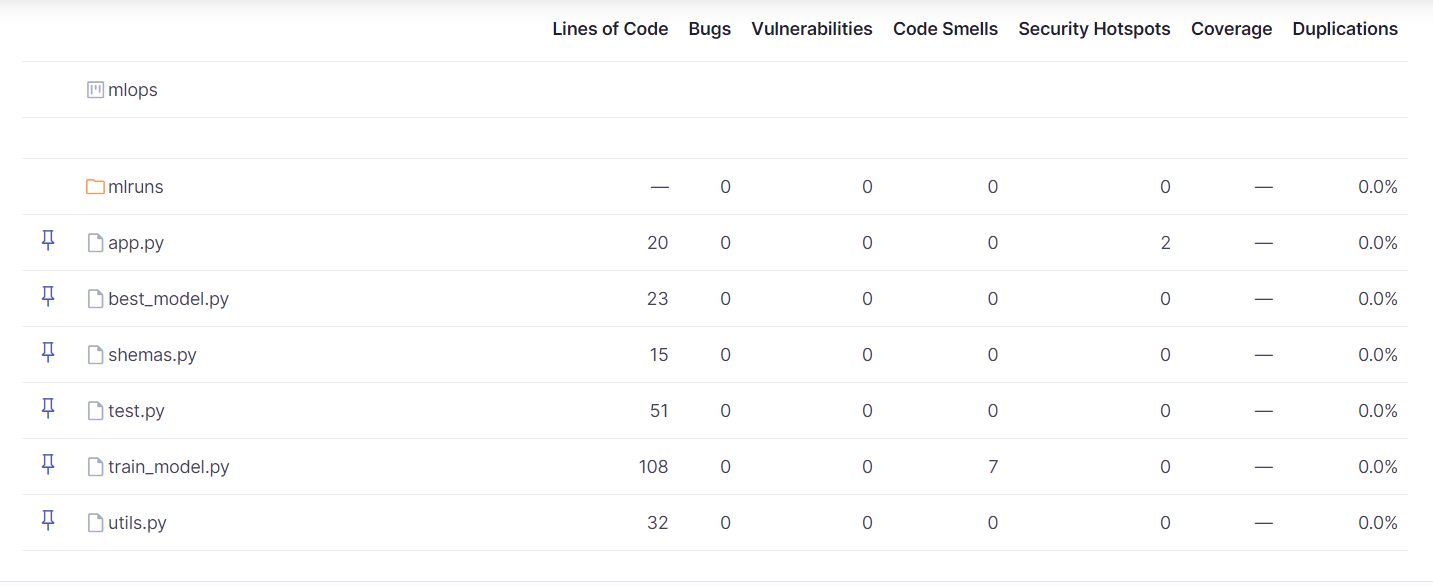
****

Рис. 16 Детальний звіт

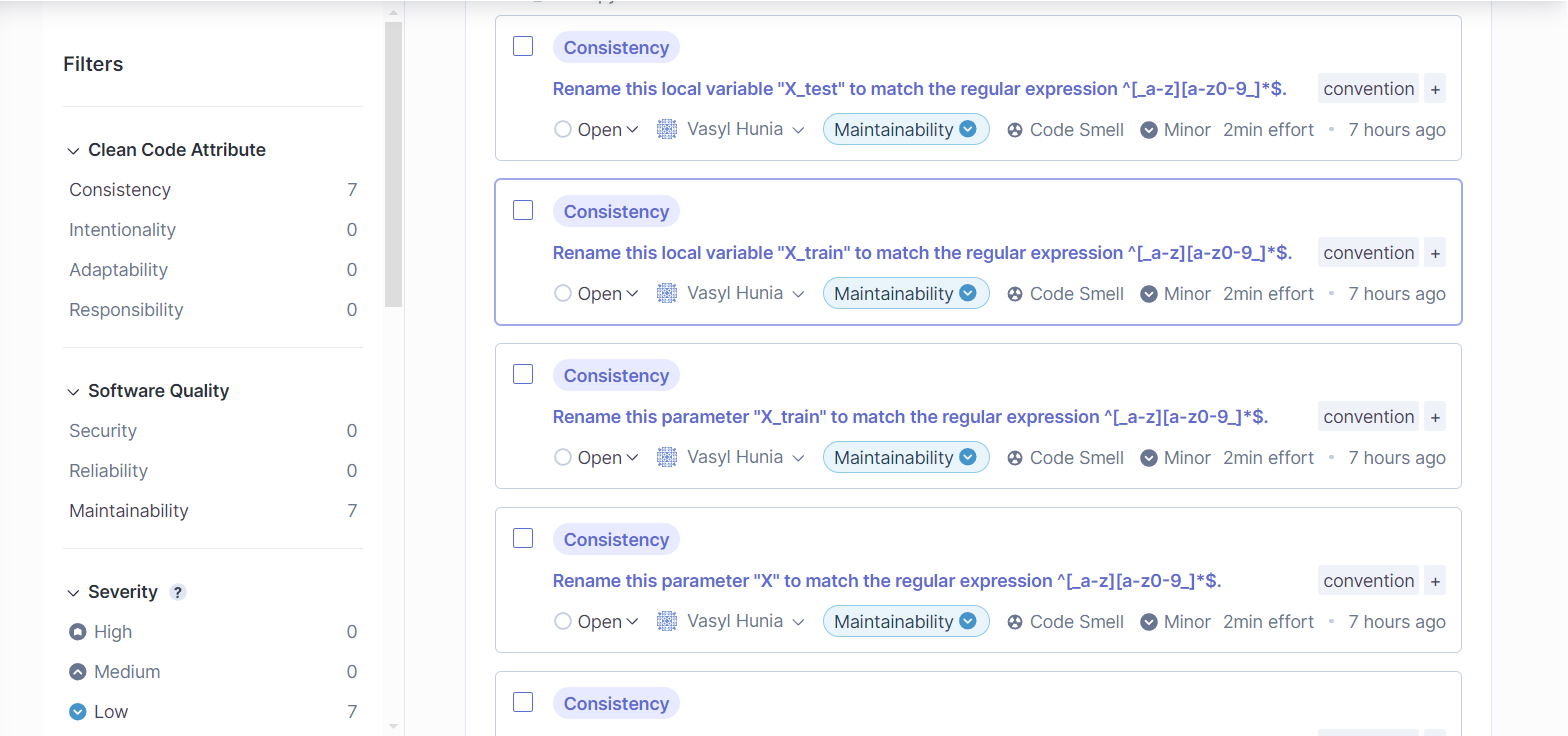
****

Рис. 17 Якість коду

Помітно що у одному з файлів не було дотримано коректного неймінгу згідно стандартів. Також є пояснення як це вирішити і що це підлягає простому вирішенню. Також цей інстремент можна використовувати в пайплані CI/CD до прикладу Github Actions. Нижче наведено приклад використання.

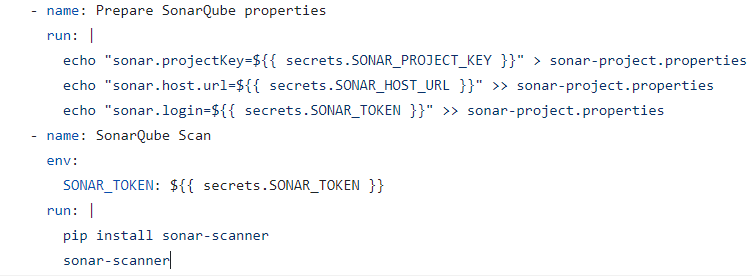


Рис. 18 SonarQube в Github Actions

**Висновок:** У даній роботі я використав MLflow для керування та відстеження експериментів у процесі машинного навчання. Зокрема, було показано, як за допомогою MLflow можна логувати параметри, результати та артефакти експериментів, а також порівнювати моделі за різними метриками. Було використано реєстр моделей для відстеження версій та їх станів, що дозволяє легко керувати моделями у різних середовищах, включаючи розгортання як REST API за допомогою Flask. Також було виконано створення git репозиторію для зберігання проекту та керування версіями, налаштування CI/CD за допомогою GitHub Actions для автоматизації тестування, перевірки стилю коду, розгортання та ретренування моделей. Використання інструментів для забезпечення якості коду, таких як flake8 та SonarQube, дозволило підвищити безпеку та відповідність коду стандартам.