NOVA IMS Information Management School

MDSAA

Master's Degree Program in

Data Science and Advanced Analytics

Machine Learning Operations

Heart Disease Classification Problem

https://github.com/DiogoPimenta26/mlops-heart.git

Beatriz Santos, 20230521

Diogo Pimenta, 20230498

Pedro Malheiros, 20230467

João Maia, 20230746

Tomás Veríssimo, 20230458

Group H

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Index

1. INTRODUCTION

This project aims to simulate the complete lifecycle of a machine learning model deployment, embodying the principles of Machine Learning Operations (*MLOps*) with a problem and dataset of our choice. A robust data pipeline was implemented using *Kedro*, which provided a well-organized and scalable framework for the project. This structure enabled smooth integration of different components and ensured reproducibility. Additionally, MLflow incorporated within *Kedro* facilitated efficient tracking and management of various model versions and combinations, making the comparison and evaluation process more straightforward.

1.1 BUSINESS PROBLEM

Heart disease remains a leading cause of mortality worldwide, making early detection and prevention crucial. This report details our work on developing a predictive model for heart disease, aiming to enhance early diagnosis and intervention. The model is designed to be used by a modern healthcare institution that aims to leverage their extensive data on patient health metrics. By utilizing this data, the institution can make more informed decisions and accurate heart disease predictions can lead to better and faster medical interventions, benefiting both patients and healthcare providers.

The defined success metrics of the project encompass both performance metrics and validation methods to ensure that our model not only performs well on historical data but also generalizes effectively to new, unseen data. In terms of performance, the success of this project is measured by the accuracy of the predicted heart disease outcomes, given the balanced nature of our dataset. This metric measures the proportion of correctly predicted instances out of the total instances. Continuous monitoring of this metric post-deployment will help maintain the model's performance and ensure its long-term assisting healthcare providers. To assure code reliability and correctness, unit tests are implemented using *pytest.* The project's success is also measured by the successful implementation of a pipeline that is reproducible across all environments.

Regarding the data, a search was made to find a reliable dataset where the objective is a classification problem. We chose a dataset from *Kaggle* that is a combination of 5 other datasets about heart diseases and where the goal is to predict if an individual has a heart disease or not. The dataset consists of 1190 instances with 12 features, including various health indicators from demographic information to clinical measurements such as age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, resting ECG results, maximum heart rate, exercise-induced angina, ST depression induced by exercise, ST slope, and the presence of heart disease (target variable). We chose this dataset because it's balanced, ensuring an unbiased model, and it has an appropriate size, providing sufficient data while remaining manageable. Additionally, it includes a meaningful and wide range of features essential for accurately predicting heart disease.

1.2 PROBLEM APPROACH

The project was organized using the principles of agile methodology. This approach ensured a structured and iterative development process, facilitating continuous improvement. The project was divided into the following sprints:

Sprint 1: We focused on data collection and exploration, where we selected and understood the dataset's characteristics;

Sprint 2: we tackled data ingestion and preprocessing, addressing issues such as outliers, inconsistencies, and feature engineering;

Sprint 3: Involved feature selection and model development, where we versioned models using *MLflow* to track experiments and saved model metrics. We trained multiple models and optimized them to evaluate their performance against the champion model;

Sprint 4: Was dedicated to implementing data unit tests to assess data quality and using Pytest for the most changeable pipelines to ensure consistency;

Sprint 5: We checked for data drift to ensure the model's performance remained robust over time. This agile approach facilitated structured development, thorough testing, and reliable model performance monitoring.

2. EXPLORATION AND DATA PREPROCESSING

Throughout the project, several obstacles and constraints were faced regarding the data used. Therefore, a notebook regarding data exploration was created to get insights and build functions and pipelines suitable for the data presented.

To counter these risks, firstly a detailed data understanding was made. The data was checked for missing and duplicated values and duplicate rows to ensure the modeling results quality. During this first phase some data inconsistencies and outliers were also identified, and to have a better understanding of the distribution of each variable histograms and a summary statistics table were developed for each feature. Another important insight to mention is that the classes of the target were checked, and the data was balanced.

In the preprocessing phase, the inconsistencies such as instances with blood pressure and cholesterol with a value of zero were removed. On the other hand, the instances that did not have a max heart rate between 72 bpm and 202 bpm were considered outliers and removed, since anything outside this range is not considered realistic and clinically accepted. Also, to simplify our analysis and understand trends more effectively two variables were binned. In this case, the decision was to group age into age intervals, and group cholesterol values into three groups “normal”, “borderline high”, and “high”.

Finally, after binning the variables, we proceeded to transform these categorical variables into a format suitable for the classification algorithms while maintaining the same level of information. To ensure our dataset remained consistent and aligned with the expected feature columns, we implemented an additional function to validate the alignment of features. This function checks whether the number of features after one-hot encoding matches the expected number.

3. FEATURE SELECTION AND MODELING

3.1 Feature Selection

In our *MLOps* workflow, we performed feature selection using *Recursive Feature Elimination* (RFE) to enhance model performance and reduce complexity by identifying the most relevant features from the data. Using a *RandomForestClassifier*, *Recursive Feature Elimination* was applied to the data. This method iteratively evaluates and ranks the importance of each feature by fitting the model and removing the least significant ones until the optimal subset of features is identified. The selected features, which were deemed most predictive, were logged, and a bar plot illustrating their importance was created and saved for reporting purposes, as shown in the figure below. This ensured that our models utilized only the most critical variables, improving efficiency, and reducing the risk of overfitting.

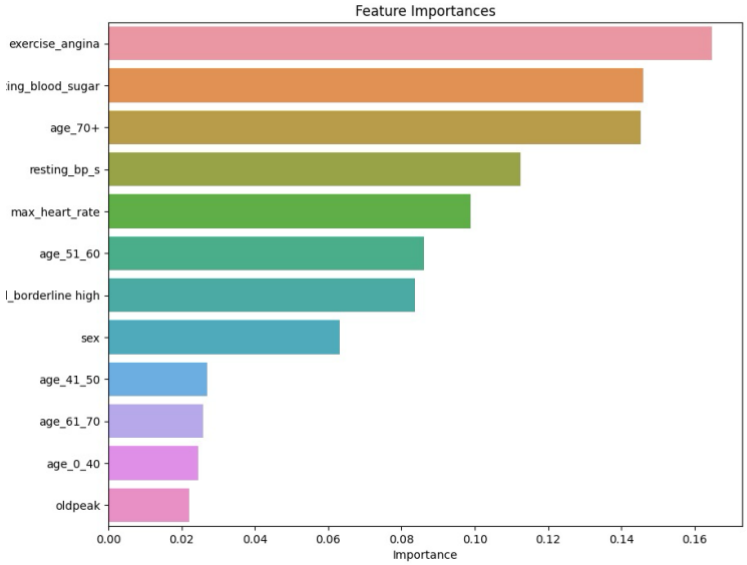


Figure 1: Feature importance RFE

3.2 Modelling

The *model_train* pipeline was designed to train and assess some different classification models. Utilizing *MLflow* for experiment tracking and auto-logging, it captured all relevant training artifacts and metrics. We also performed *SHAP (SHapley Additive exPlanations)* analysis to interpret the model's predictions. The SHAP summary plot illustrated the impact of each feature on the model's output, highlighting the most influential features (Annex 1). Features such as *"st_slope_1"* and *"chest_pain_type_4"* were identified as having the highest impact on model predictions. Following this, the *model_tune* function was employed to fine-tune the hyperparameters of the champion model, which was identified as the *Random Forest classifier*, with an initial accuracy of 0.89. Using *GridSearchCV,* we searched for the optimal hyperparameters, further improving the model's performance. The tuned model registered a training accuracy of 0.9142 and a test accuracy of 0.9390, with the best parameters being a

max_depth of 20 and *n_estimators* of 200. Finally, the *model_predict* function was used to make predictions on the test dataset with the best model.

4. Quality Testing

4.1 Data Unit Tests

To ensure the quality and integrity of our model, several data unit tests were implemented using the Great Expectations library. Firstly, we started by checking the dataset for the expected number of columns, confirming that it contains all features. We also validated the data types of each column, making sure they match the expected types. Critical columns that were chosen based on feature selection such as *"age", "sex", "chest_pain_type", "resting_bp_s", "cholesterol", "max_heart_rate"* and the target value were tested for missing values to ensure that there are no null entries. For categorical columns like *"sex"* and *"chest_pain_type"*, we checked that the values fell within the expected categories, ensuring data consistency. For example, for the sex variable, this should only contain values such as "0" or "1", preventing the inclusion of incorrect values. To provide a comprehensive overview of the dataset's quality, we generated validation reports, including metrics like minimum, maximum, mean, and mode. Also, a list of unexpected values is generated for categorical columns. This report is saved in CSV format for further analysis and auditing. Logging mechanisms are integrated to document each step of the validation process, confirming the successful execution of data checks, and capturing any discrepancies. The comprehensive nature of these data unit tests ensures that our datasets maintain high-quality standards, which is crucial for the accuracy and reliability of our machine learning models. By implementing these tests, we mitigate risks associated with data quality issues and enhance the overall robustness of our MLOps pipeline.

4.2 pytests

In the *pytest* implementation, several tests were created to ensure the reliability of data preprocessing and feature extraction functions:

- ***test_feature_extraction***: This checks if "*feature_extraction"* correctly processes input data into the expected format with appropriate feature columns;
- ***test_remove_duplications***: This test verifies the functionality of the *remove_duplicates* function, which removes duplicate rows from the dataset;
- ***test_remove_zero_bpm***: checks the remove_zero_bp function, which removes rows where the 'resting_bp_s' column has a value of zero.
- ***test_remove_zero_cholestrol***: validates the "remove_zero_cholesterol" function, designed to remove rows with zero values in the 'cholesterol' column;
- **test_remove_zero_st_slope:** ensures the remove_zero_st_slope function removes rows where the 'st_slope' column is zero;
- **test_max_heart_ rate_outliers:** test checks the max_heart_rate_outliers function, which filters out extreme outliers in the 'max_heart_rate' column;
- **test_bin_age:** evaluates the bin_age function, which categorizes the 'age' column into predefined bins;
- **test_bin_cholesterol:** ensures the bin_cholesterol function correctly bins 'cholesterol' values into categories.

4.1 Data Drift

The data drift pipeline is crucial because it continuously monitors incoming data for shifts in its distribution. By implementing this pipeline, we can quickly identify and address variations that could lower model performance, thereby maintaining the integrity and accuracy of our predictive models. This pipeline applies various statistical tests and drift reports to detect drift in the distribution of our data, between a reference dataset and an analysis dataset.

Before we started to explore our data, we split the data into “use_data” (crucial for detecting deviations in future data, this is going to be our reference dataset) and “test_data” (crucial for monitoring new incoming data, this is going to be our analysis dataset), by comparing this dataset against the reference one we can detect data drift as soon as it occurs. We started by generating bias in our data for test purposes, we filtered the cholesterol values over 240 which can be a real example where an individual with cholesterol values over that threshold is prone to develop heart disease. Then a constant threshold was defined between 0.1 and 0.5 and the Jensen-Shannon statistical method is applied to detect drift. We initialized a “UnivariateDriftCalculator” by fitting the reference dataset to calculate drift in the analysis data on the columns, “cholesterol”, “age”, “max_heart_rate”, and “resting_bp_s” (Annex 2). These columns are more prone to change their values per individual over time. Next, a data drift report was generated by applying the same statistical test and a significance threshold of 0.1, which is the default value for the Jensen-Shannon method. The drift report showed that in 3 out of 4 columns it detected drift. The variables where the significance threshold was exceeded were,” cholesterol”, “age” and “max_heart_rate” with a drift score of ~0.44, ~0.12, and 0.11, respectively.

Drift is detected for 75.0% of columns (3 out of 4).

Search

	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
>	cholesterol	num			Detected	Jensen-Shannon distance	0.434941
>	age	num			Detected	Jensen-Shannon distance	0.12829
>	max_heart_rate	num			Detected	Jensen-Shannon distance	0.112725
>	resting_bp_s	num			Not Detected	Jensen-Shannon distance	0.098389

Rows per page: 4 rows 1-4 of 4

Figure 2 - Drift report

5. ADVANTAGES, LIMITATIONS, AND NEXT STEPS

We believe our approach was robust and well-executed, as the tools that were used for the development of the project were advantageous for what it was intended. Although we know it is just proof of concept, there are some limitations associated with the packages that can only be overcome whenever it enters production and continuous assessment and when the evaluation takes place.

Starting with *Kedro*, it represented the organization and ongoing flow of the project. Its powerful implementation of the code regarding organization through its modularized code, each having its scope clearly defined, also adding the advantage of troubleshooting more easily when needed, i.e. each part of the project had its objective clearly defined from its foundation. This was important since whenever we've done any of the steps, one could travel back and forward to test and run the code, aligned with the flow of our approach (the separated datasets, pipelines).

In the implementation phase of modeling, MLFlow was advantageous when accessing model tracking. To be able to compare several 'runs', and how the models behaved was crucial to capture the more accurate insights of the results we could have, which proved quite helpful when choosing a champion model. Also*, NannyML*, *Great Expectations*, and *Evidently* played a significant role in dealing with data ingestion, data unit testing, and data drift.

To keep developing our proof of concept there are many suggestions one can give, from exploring more data drift tests, more unit tests, etc.

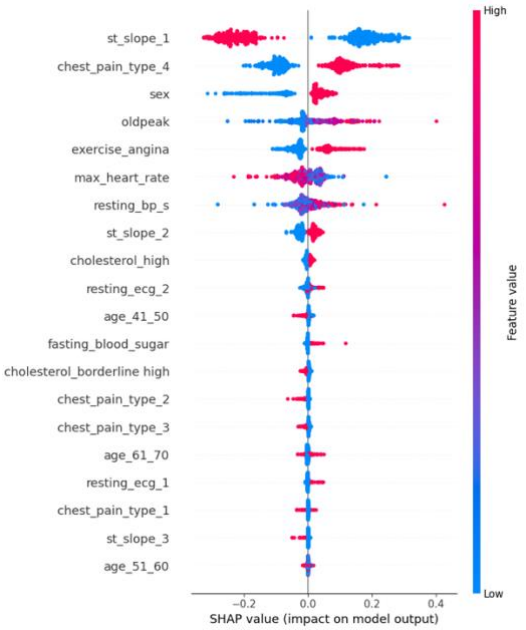
Finally, to summarize the tools used in the project, the most relevant libraries that we used were, *Kedro, Hopsworks, Great Expectations, NannyMl, Scikit-learn, Pytest, Evidently, Numpy, Pandas, Scipy, Seaborn, Shap, xgboost* and for more packages information there is a *requirements.txt* in the project folder of our GitHub.

6. CONCLUSION

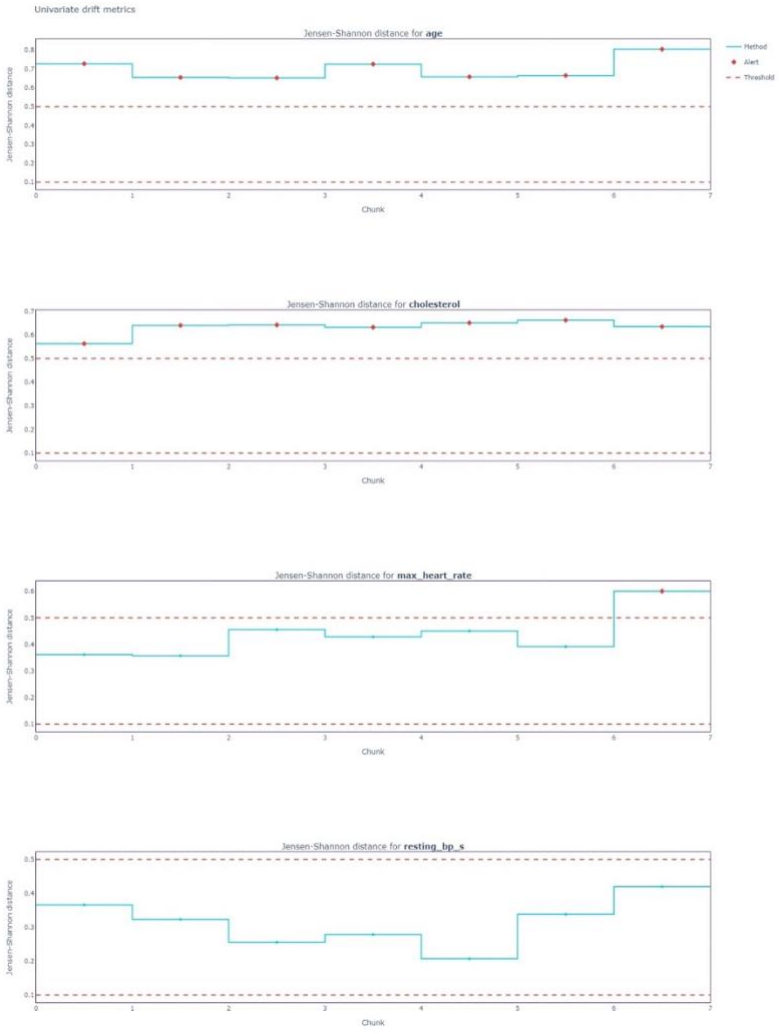
In conclusion, our project successfully demonstrated a structured approach to deploying a machine learning model, showcasing the power of *MLOps*. We developed a reliable, scalable (CI/CD), and interpretable workflow of operations that culminated in a robust combination of pipelines by empowering powerful methods such as the Kedro organization tool, great expectations for the data unit testing that ensures the quality of data throughout time, or *MLFlow* for the model experimentation and comparison between them. This ensured optimal performance in our classification problem. Our predictive modeling in the healthcare industry highlighted the critical importance of incorporating machine learning solutions to evolve and leverage technology in one of the most vital areas of human life. Specifically, our research achieved a high level of accuracy, validating our performance metric given the well-balanced nature of our dataset. As a last detail, after the predictions of the trained model, we got a distribution of 48% of the people not having heart disease and consequently the rest having it. Thus, complementing the well-distributed nature of the target identified in the data analysis and exploration.

This project underscores the significant benefits of *MLOps* in developing and deploying effective machine learning models in healthcare, ultimately contributing to better patient outcomes and more efficient healthcare practices.

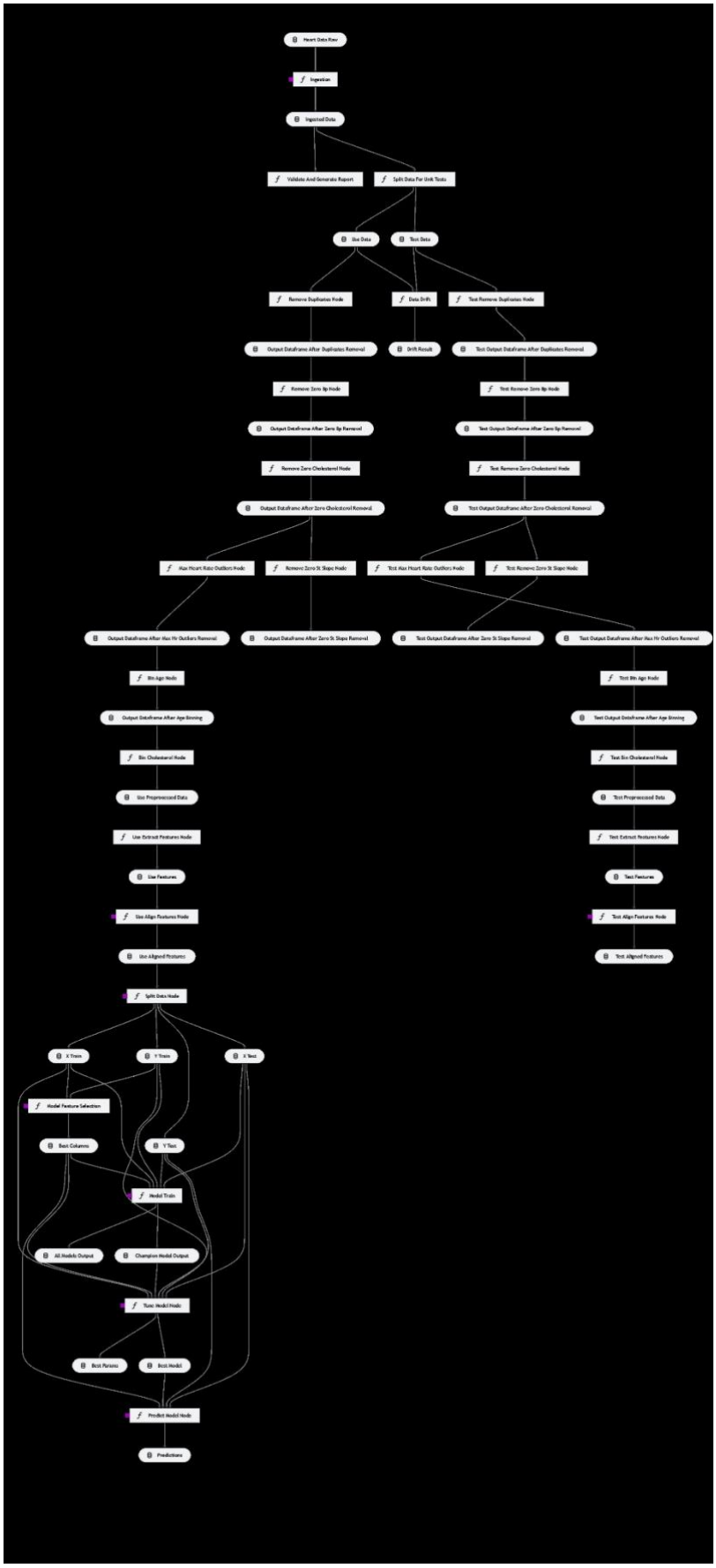
7. ANNEXES



Annex 1: Shap values



Annex 2: Data drift



Annex 3 - Kedro Viz