INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

HEART ATTACK PREDICTION WEB APP

COMPUTER SCIENCE PROJECT

SUBMISSION: PHASE 2

Course Name :	Master of Science
Course of Study :	Computer Science
Course code :	DLMCSPCSP01
Name of Author :	Jean Luc Nsadisa
Matric Number :	9211128
Name of Tutor :	Dr. Oezdemir Cetin

August 18, 2024

LIST OF FIGURES

TABLE OF CONTENT

ABSTRACT

In this work, I propose applying computer science to the medical field with the aim of developing a web application that predicts the likelihood of a heart attack based on information gathered from the patient. Heart attacks are among the leading causes of death from coronary diseases worldwide. According to the World Heart Federation, a heart attack is caused by ruptured plaques forming blood clots that block an artery, impeding blood flow and oxygen to the heart. The New York State Department of Health (New York State Department of Health, 2023) reports that heart attacks cause about 805,000 deaths annually in the United States and have surged globally by 60% over the past 30 years (World Heart Federation, 2023).

According to the Moyo Clinics website (Moyo Clinics, n.d.), symptoms of a heart attack may include tightness or chest pain that feels like pressure, aching, or squeezing; pain spreading to the arm, neck, shoulder, jaw, back, or teeth; high fatigue; cold sweat; and various factors such as age, sex, exercise-induced angina, the number of major vessels, chest pain type, resting blood pressure, fasting blood sugar, resting electrocardiographic results, cholesterol level, and maximum heart rate achieved.

This project will enhance understanding and provide a valuable tool for users by addressing a serious health problem using modern technology. It will take as input the user’s medical report from the lab, which will then be compared against a trained model to make predictions.

INTRODUCTION

Death caused by coronary heart diseases remains a significant challenge in this era of technology, causing numerous deaths globally each day. Nowadays, people of all ages and ethnicities are impacted due to various factors such as the amount and quality of processed food consumed, lifestyle choices that increase stress levels, and sedentary habits leading to a lack of exercise. These are among the many contributors to heart attacks.

In this work, we explore how technology can contribute to the prevention and awareness of heart attack occurrence by developing a web application that allows users to input their lab results and symptoms. The front-end of this application is developed using React.js, and it evaluates user inputs against a model trained on past medical data. The back-end logic is initially developed using Jupyter Notebook for its line-by-line visualization and testing capabilities and then transitioned to Flask (Python) for communication with the front-end.

The dataset used in this project is sourced from a public listing by Cleveland Heart Disease(UCI Repository) (Cleveland Heart Disease dataset , n.d.). A complete data science analysis will be performed to discover hidden patterns in the collected raw data. The data will undergo preprocessing: cleaning, outlier handling, redundancy elimination, error checking, correlation analysis; processing and visualization. The full project will be available on GitHub and the link will be provided at the Appendix.

Problem statement

Coronary heart disease remains a leading cause of death worldwide (World Heart Federation, 2023). Regardless of the advanced technology in the medical arena, early detection and proper measures for the prevention of heart attacks remain challenging. One issue is the lack of proper tools for interpreting lab results that users can constantly monitor based on their medical data. In most cases, patients need to consult professional medical centers for evaluation, which may be a challenge for everyone.

Therefore, we propose a user-friendly technological solution to help individuals predict the likelihood of a heart attack using their medical parameters.

In most cases, patients rely heavily on their doctors to interpret lab results, thereby limiting their ability to proactively manage their health. Most heart attack prediction tools that already exist are not user-friendly and are ill-equipped for continuous or real-time monitoring, relying solely on static data. As a result, patients cannot independently monitor changes in their heart health over time. Our solution fills these gaps by providing an accessible prediction tool that allows individuals to estimate their risk of having a heart attack using existing medical data and to act on this information.

Research question

How can a web application developed using medical data use a machine learning algorithm to predict the likelihood of developing a heart attack in a patient?

Scope of the research

Data Collection: We gathered a comprehensive dataset of medical parameters in a CSV format from Cleveland Heart Disease(UCI Repository) with more than 14 medical parameters related to heart attack risk for a total of 303 patients. Data Analysis: We will perform data processing, cleansing, and correlation analysis on the collected data to prepare the dataset for model training. Model Development: we will explore various machine learning techniques to evaluate and compare their performance metrics. The most feasible one will be selected to train the model to predict heart attack risk. Web Application Development: Using React.js as a frontend tool for data input by the user and Flask for the model logic in the backend. Model Integration: The trained model will then be coupled with our web application for real-time predictions. Testing: Multiple tests will be performed using both the developed web application and Postman to ensure accuracy and reliability. In the next part of the report, we will delve deeper into the technical aspects and implementation details of our Heart Attack Prediction Web Application.

RELATED WORK AND TECHNICAL BACKGROUND

Related work

Data science techniques, including ML and AI, have seen tremendous development across various sectors in recent years, especially in the medical field. By analyzing large datasets from patient trials, researchers have successfully identified patterns and relationships among variables that point to specific health outcomes. Consequently, this advancement is enhancing the prediction of certain medical conditions, such as heart attacks.

One of the more well-known studies is the Framingham Heart Study, which has been researching coronary diseases for over 75 years. Recently, it added a machine learning algorithm to its website for calculating the risk of cardiovascular disease. Variables assessed by this model include age, diabetes, smoking status, systolic blood pressure (both treated and untreated), total cholesterol, HDL cholesterol, and BMI. This approach provides a much more user-friendly and accessible method compared to earlier models that focused primarily on lipids. (Framingham Heart Study, n.d.),

The Cleveland Heart Disease dataset, available in the UCI Machine Learning Repository, has often been used to benchmark predictive models. Logistic regression, decision trees, and neural networks have been applied to this dataset, contributing to numerous improvements in risk prediction for heart disease.

The National Library of Medicine has also applied machine learning techniques to the risk prediction of heart disease using factors such as age, sex, chest pain, resting blood pressure, cholesterol levels, fasting blood sugar, ECG results, maximum heart rate achieved, exercise-induced angina, ST depression, slope, number of major vessels, and thalassemia. The accuracies for the developed models ranged from 80.32% to 88.5% using Random Forest and Support Vector Machines. (national library of medicine, 2022)

Machine learning has contributed to the development of various heart disease predictive systems. SVM, Random Forests, and K-Nearest Neighbors have proven to be promising

techniques. For example, Kahramanli and Allahverdi integrated fuzzy data with SVM and achieved accuracies of 86.24% on the Pima Indian Diabetes dataset and 86.8% on the Cleveland heart disease dataset. Another researcher, Gudadhe, used SVM for heart disease prediction and obtained an accuracy of 80%.(Turabieh, 2016)

Additionally, modern smartphones play a significant role in monitoring heart diseases, such as heart attacks. Apps like Cardiogram and the Apple Heart Study use wearable technology to monitor cardiovascular health in real time, predicting potential issues before they arise. This demonstrates the synergy between cutting-edge health technology and healthcare management.

The pursuit of excellence in applying AI to diagnostic practices in medicine is a central theme of the article. The enhanced prediction accuracy and increased accessibility of heart disease prediction models are evident benefits of these technologies.

Technical Background

My approach to executing the research and developing the methodology will be illustrated in this section by outlining the techniques and steps used to achieve our objectives. I will explain several important phases:

Understanding the Dataset

The dataset is essential for any machine learning model as it serves as a blueprint for real-world scenarios. Datasets are constructed based on multiple observations and can be gathered in any field, including the medical arena (IU Internationale Hochschule, 2023). To uncover significant patterns hidden in the data, the collected data will go through different preparation stages such as cleaning, curation, correlation analysis, data interpretation, dealing with redundancy, and handling missing data. This process can involve considerable time, money, and effort, leading many companies to outsource this service. Fortunately, for this work, we have a ready-to-use dataset. However, applying domain knowledge to understand the meaning of each parameter is crucial. The proposed dataset has a total of 14 parameters that will be analyzed individually, spread across data from 303 patients, as shown in the following figure:

```
df = pd.read_csv('heart.csv')
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Figure 1 dataset overview

```
]: print("Shape of the Dataset", df.shape)
Shape of the Dataset (303, 14)
```

Figure 2 dataset shape

METHODOLOGY

Referring to the introduction of this work, our goal is to apply computer science principles to a large volume of collected medical data to develop a user-friendly web application that allows patients to predict their likelihood of developing a heart attack. This development necessitates skills in both data science and software engineering. In data science, processes such as pre-processing, processing, and visualization are crucial. In software engineering, we will need full-stack development skills to create a robust backend and frontend application.

Data Science Methods

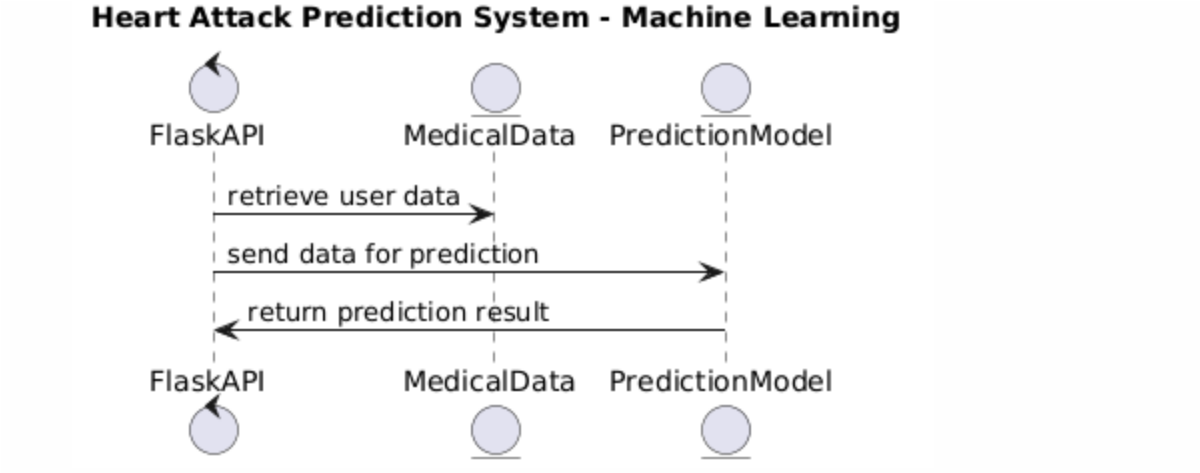

Figure 3 Data science UML model

Preprocessing:

Datasets can be generated automatically, e.g., from a sensor, manually in the case of a lab, or semi-automatically. Either way, we may face issues such as missing data, outliers, or inconsistencies. Techniques such as handling missing values, dealing with outliers, and ensuring data quality are paramount to developing an effective predictive model. This step ensures that the dataset is reliable and free from inconsistencies. (IU Internationale Hochschule, 2023)

Feature Engineering: This can be important in some case where we will need to evaluate the necessity of scaling data within the dataset between a range e.g 0 - 1 so that we can have all features contributing equally to the model for a better performance. We will estimate at the later stage of this process if this is necessary or not. (IU Internationale Hochschule, 2023)

Normalization and Scaling: This can be important in some case where we will need to evaluate the necessity of scaling data within the dataset between a range e.g 0 – 1 so that we can have all features contributing equally to the model for a better performance. We will estimate at the later stage of this process if this is necessary or not. (IU Internationale Hochschule, 2023)

Dataset overview and correlation analysis:

```
]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   rest_ecg  303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Figure 4 data info()

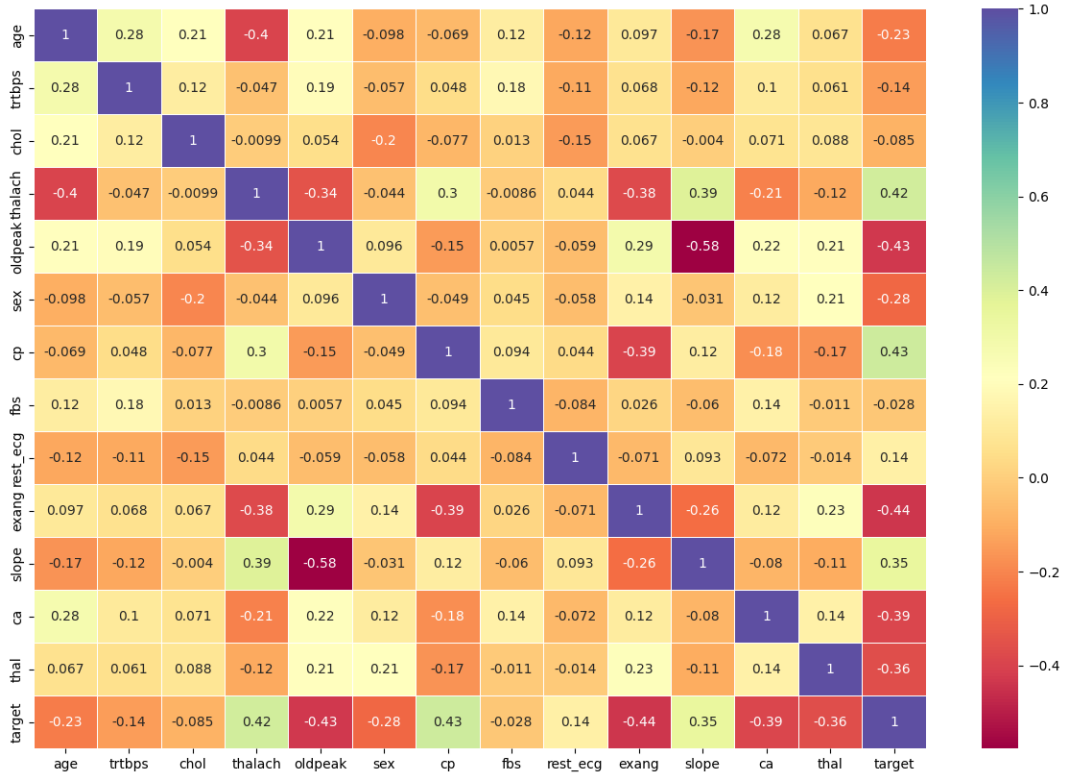

Figure 5 dataset correlation analysis with target as 1

```
graph = sns.pairplot(df[numeric_var], diag_kind = 'kde')
graph.map_lower(sns.kdeplot, levels = 4, color = '.2')
plt.show()
```

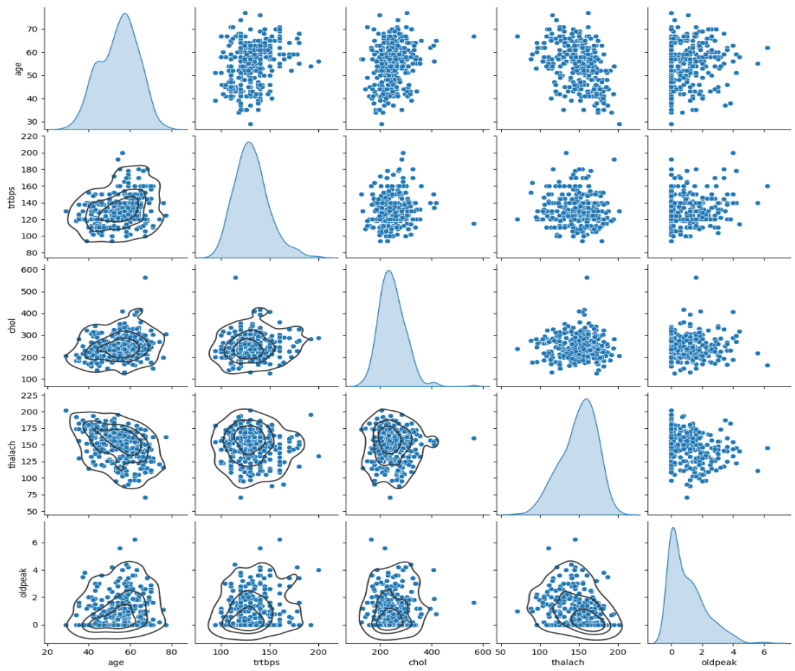

Figure 6 Pairplot of variables

Data Cleaning:

- Missing Values: Missing data were detected and treated correctly to prevent biases in the model. Depending on the nature of the missing data, either mean or median imputation was used. In cases where missing entries made up a negligible proportion of the dataset, such entries were removed. The function *df.isnull().sum()* was used then we inputted the missing values with the median : *df.fillna(df.median(), inplace=True)*
- Consistency Checks: The consistency of the data entries was checked. For categorical variables, it was ensured that all possible categories were properly labeled and no incorrect values were introduced. We used the key word *unique()* to check in categorical variables. Then correcting any inconsistencies found with the replace().

Outlier Detection and Treatment:

Identification of Outliers: Outliers in numerical features like age, trtbps, thalach, and oldpeak were identified using statistical methods like Z-score and Interquartile Range (IQR). The below is an example of the z_scores technique applied on the 'trtbps'. The same function was used for across numeric categorical variables. I used the boxplot to determine the outliers in numerical values with high correlation based on the Figure above. The following code

```
from scipy import stats
from scipy.stats import zscore
from scipy.stats.mstats import winsorize
```

```
# we can determnine the number of outliers above the threshold
z_scores_trtbps = zscore(df['trtbps'])
for threshold in range(1, 4):
    print('Threshold Value:  {}'.format(threshold))
    print('Number of Outliers: {}'.format(len(np.where(z_scores_trtbps    > threshold)[0])))
    print('-------------------')
```

```
Threshold Value:  1
Number of Outliers: 51
-------------------
Threshold Value:  2
Number of Outliers: 13
-------------------
Threshold Value:  3
Number of Outliers: 2
-------------------
```

Figure 7 outlier detection using Z-score

Then it was also verified used the IQR method using the below functions:

```python
def iqr(df, var):
    q1 = np.quantile(df[var], 0.25)
    q3 = np.quantile(df[var], 0.75)
    diff = q3 - q1
    lower_v = q1 - (1.5 * diff)
    upper_v = q3 + (1.5 * diff)
    return df[(df[var] < lower_v) | (df[var] > upper_v)]
```

Figure 8 outliers detection using the iqr function

Winsorization and Transformation:

- Winsorization: This process involved capping the data at certain percentiles to reduce the impact of extreme values. For example, variables like trtbps and oldpeak were winsorized to set outliers at a more representative range.
- Transformation Techniques: A square root transformation was applied to the variable oldpeak, which helps to dampen the effects of skewed data distributions, making the data more symmetric for the modeling process.

```python
from scipy.stats import mstats

# Winsorization
df['trtbps_winsorize'] = mstats.winsorize(df['trtbps'], limits=[0.05, 0.05])

# Square root transformation
df['oldpeak_winsorize_sqrt'] = np.sqrt(mstats.winsorize(df['oldpeak'], limits=[0.05, 0.05]))

```

Figure 9 Winsorization and square root transformation

```python
winsorize_percentile_trtbps = (stats.percentileofscore(df['trtbps'], 165)) / 100
print(winsorize_percentile_trtbps)
```

```
0.957095709570957
```

After analyzing the 4 numerical variable with valuable correlation: age, trtbps, thalach and oldpeak. We can visualize the distribution of data using histogram tool:

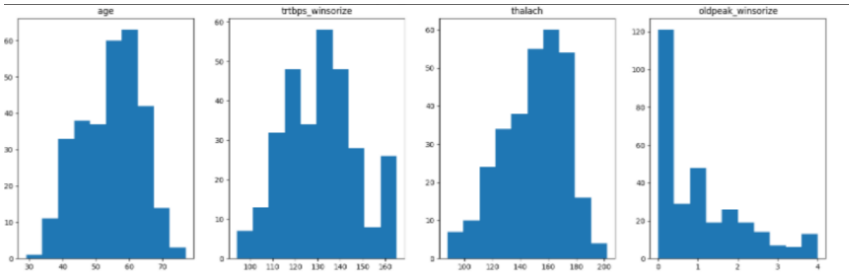

Figure 10 histogram plot after winsorization

Normalization and Standardization:

- **Normalization**: Numerical features were scaled to a common range, particularly when features were measured in different units or had different orders of magnitude, to ensure all features contributed equally to the model's learning process.
- **Standardization**: This step was applied to some algorithms to ensure features had a mean of zero and a standard deviation of one. Standardization often stabilizes the learning process and can improve model performance.
- **Addressing Skewed Distributions**: Skewness is important to consider in this work, as it can affect the accuracy of the model. If the data is skewed, it might distort statistical assumptions and result in biased predictions. The script below was used to determine the skewness.

```
df[['age', 'trtbps_winsorize', 'thalach', 'oldpeak_winsorize']].agg(['skew']).transpose()
```

	skew
age	-0.199209
trtbps_winsorize	0.251969
thalach	-0.461611
oldpeak_winsorize	0.996036

Figure 11 skewness detection on variables

To address the above skewness, we used the transformations method to help normalizing the distribution of data to improve the model performance and ensuring that the predictions are accurate and reliable. The following script was used:

```
df['oldpeak_winsorize_log'] = np.log(df['oldpeak_winsorize'])
df['oldpeak_winsorize_sqrt'] = np.sqrt(df['oldpeak_winsorize'])
```

```
df.head()
```

	age	sex	cp	thalach	exang	slope	ca	thal	target	trtbps_winsorize	oldpeak_winsorize	oldpeak_winsorize_log	oldpeak_winsorize_sqrt
0	63	1	3	150	0	0	0	1	1	145	2.3	0.832909	1.516575
1	37	1	2	187	0	0	0	2	1	130	3.5	1.252763	1.870829
2	41	0	1	172	0	2	0	2	1	130	1.4	0.336472	1.183216
3	56	1	1	178	0	2	0	2	1	120	0.8	-0.223144	0.894427
4	57	0	0	163	1	2	0	2	1	120	0.6	-0.510826	0.774597

Figure 12 addressing skewness using log and square root

To verify if it worked, we used the above Figure again, as we can see no skewness found on the oldpeak_winsorize_log.

```
df[['oldpeak_winsorize', 'oldpeak_winsorize_log', 'oldpeak_winsorize_sqrt']].agg(['skew']).transpose()
```

	skew
oldpeak_winsorize	0.996036
oldpeak_winsorize_log	NaN
oldpeak_winsorize_sqrt	0.108926

Figure 13 skewness detection confirmation

We decided to drop oldpeak_winsorize_log :

```
df.drop(['oldpeak_winsorize', 'oldpeak_winsorize_log'], axis =1, inplace = True)
```

At this stage, our data has been cleared of missing data, outliers, skewness and it is ready for the next step which is processing

Processing:

After ridding data of impurities such as missing values, outliers, and skewness, we are good to go with building our model using different techniques. To ensure that the model performs as expected, metrics are used to evaluate its performance. Some of these metrics are discussed below.

Model Evaluation and Comparison:

To assess the performance, we will consider accuracy, precision, recall, F1 score, and ROC-AUC. This set of metrics provides a comprehensive view of a model's strengths and weaknesses, particularly regarding different types of errors.
Class Imbalance Handling: In the case of class imbalance in the dataset, techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or class weight adjustment will be applied.

Overfitting Prevention: we used cross-validation technique. However, the updated version of this work will cover more options.

Accuracy: Indicates the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances. It provides a general idea of how often the model is correct and can be mathematically expressed as follows:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Instances}}$$

Precision: Represents the proportion of true positive predictions relative to all positive predictions made by the model. It shows how many of the predicted positives are actually positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall: Also known as Sensitivity or True Positive Rate, it indicates how well the model identifies actual positives. It shows the proportion of actual positives that are correctly identified by the model.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score: The F1 score is the harmonic mean of Precision and Recall. It balances the concern of Precision and Recall and is useful when you need a single metric to evaluate performance.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC-AUC: The Area Under the Receiver Operating Characteristic Curve (ROC-AUC) quantifies the model's ability to distinguish between positive and negative classes across different threshold values. A higher AUC indicates better model performance.
Formula: The ROC-AUC does not have a simple formula. It is calculated as the area under the ROC curve, which plots the True Positive Rate (Recall) against the False Positive Rate for various threshold settings.
These metrics provide a comprehensive view of the model's performance, highlighting its strengths and weaknesses.

Machine Learning Algorithms: Variou s algorithms can be employed to predict heart attack risk, including:

Logistic Regression: Ideal for binary classification problems like predicting the presence or absence of a heart attack. It provides clear probabilistic outputs. We will be using the python package name sklearn.linear_model.LogisticRegression. The Logic regression can be expressed with the following mathematical function. (IU Internationale Hochschule, 2023)

$\sigma(z) = \frac{1}{1+e^{-z}}$

where:

- $\sigma(z)$ is the logistic function.
- $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$ is the linear combination of the predictors $x_1, x_2, \ldots, x_n$ weighted by coefficients $\beta_1, \beta_2, \ldots, \beta_n$.
- e is the base of the natural logarithm, approximately equal to 2.71828.

For this work, we used the sklearn.model_selection, sklearn.linear_model and sklearn.metrics to calculate the performance metrics. And we got the following results: Test accuracy: 0.8709, Cross validation accuracy Scores: 0.8666, Precision : 0.875, recall : 0.9545 and F1 Score: 0.9130 and the ROC Curve and AUC :

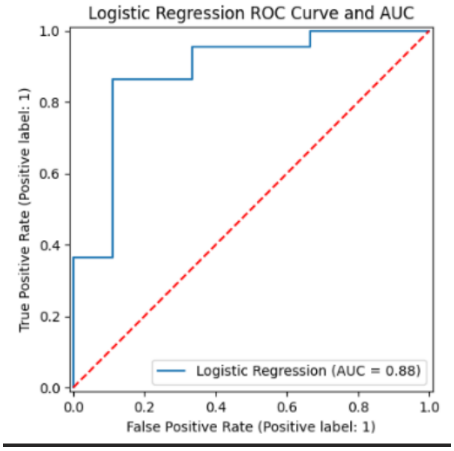

Figure 14 Logic regression ROC curve and AUC

Decision Trees and Random Forests: These models provide interpretable outputs and are excellent at handling feature importance, making them suitable for medical data. We will be using the package randomForest. It can also be expressed mathematically as follows:

Gini Impurity:

$I_G(p) = 1 - \sum_{i=1}^{m} p_i^2$ where p_i is the proportion of samples of class iii in a particular node. (IU Internationale Hochschule, 2023)

Entropy: $I_H(p) = -\sum_{i=1}^{m} p_i \log(p_i)$ where p_i is the proportion of samples of class iii in a particular node. (IU Internationale Hochschule, 2023)

For the Random Forest we used the package sklearn.ensemble and we imported the RandomForestClassifier class. We go an accuracy score of 0.8064, a precision of 0.9, a recall of 0.818181 and F1 Score of 0.857142 and the ROC curve and AUC :

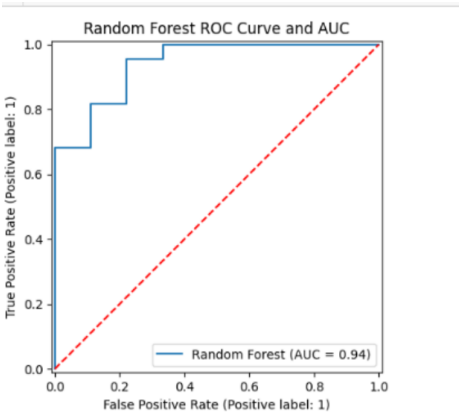

Figure 15 Random Forest ROC cuver ad AUC

Support Vector Machines (SVM): Effective in high-dimensional spaces and particularly useful for classification tasks. SVMs are known for their robustness in complex datasets. We can use the package name sklearn.svm. The mathematic expression of a SVM is a bit complex. For the SVM metrics calculation, used the package sklearn.svm and found the following results:

Test accuracy score of 0.87096, cross validation accuracy score of 0.8333, precision of 0.9, recall of 0.8181, F1 Score of 0.8571 and the ROC Curve and AUC :

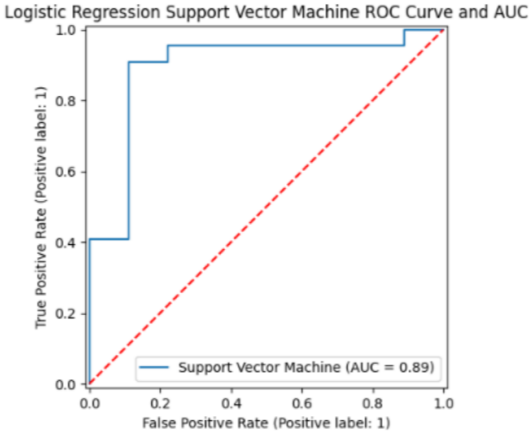

Figure 16 SVM ROC curve and AUC

Data Visualization Techniques: Utilizing charts, graphs, and other visualization tools to interpret and present the data findings effectively. This helps in understanding the

underlying patterns and trends within the dataset. We will the following packages Plotly, Seaborn and Matplotlib.

Looking at the performance metrics above, logistic regression shows better performance and will therefore be the choice for this project. The backend will be developed based on the logistic regression model. The model is trained on 90% of the data and tested on the remaining 10% to evaluate its performance.

Software Engineering

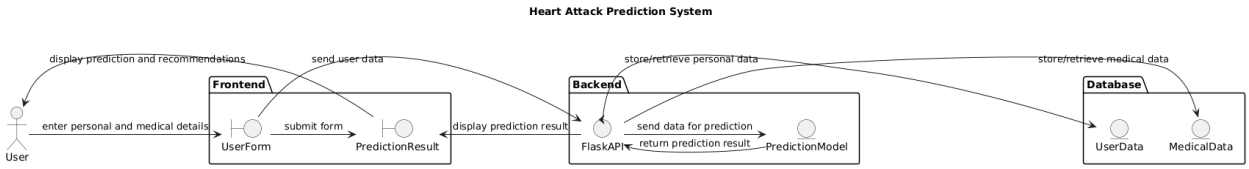

Figure 17 UML model system architecture

Backend Technology:

Flask: The following UML show the overview of the backend using Flask. Flask is a lightweight WSGI web application framework in Python, ideal for building APIs and integrating machine learning models.

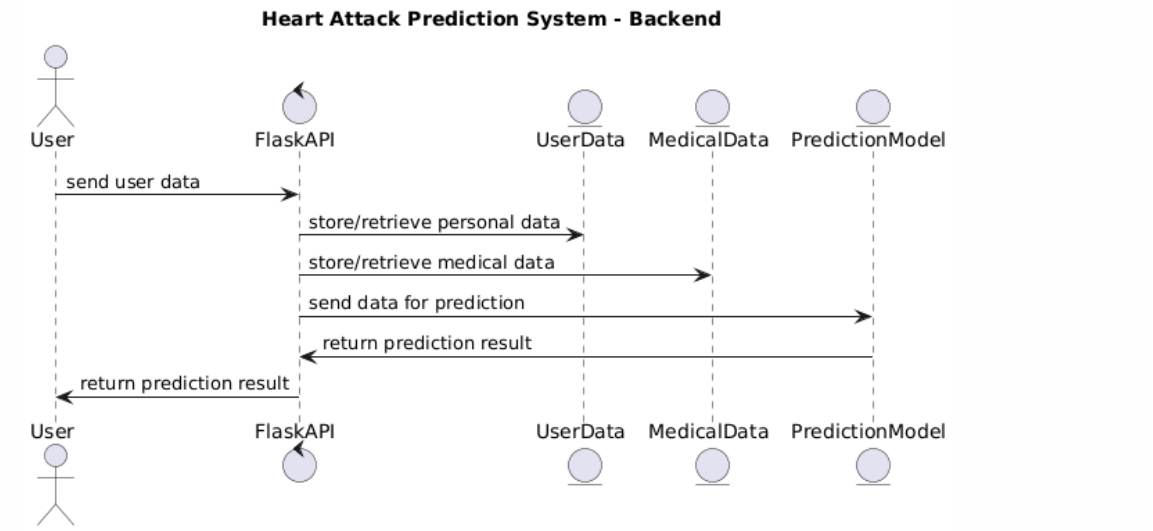

Figure 18 UML backend architecture

The backend is linked to the frontend used the following function.

```python
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    # Convert JSON to DataFrame
    df = pd.DataFrame(data)
    # Ensure the DataFrame has the correct columns
    expected_columns = ['age', 'thalach', 'trtbps_winsorize', 'oldpeak_winsorize_sqrt', 'sex_1
                        'cp_1', 'cp_2', 'cp_3', 'exang_1', 'slope_1', 'slope_2', 'ca_1', 'ca_2
                        'ca_3', 'ca_4', 'thal_2', 'thal_3']
    df = df[expected_columns]
    # Make predictions
    predictions = model.predict(df)
    # Return the predictions as JSON
    return jsonify(predictions.tolist())

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 19 backend predict function

Frontend Technology:

React.js: This choice is made because it is a widely used JavaScript library used for building user interfaces. It is good for single-page applications that require real-time interactivity, providing a seamless and responsive user experience. The following is a UML overview of the front-end system. In the appendix of this work, we have provided results of the user experience using dummies data to predict the likelihood of developing a heart attack.

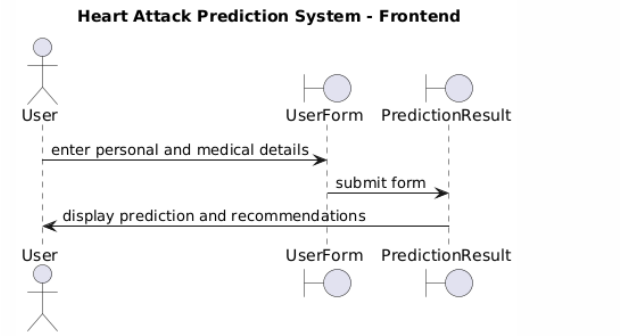

Figure 20 frontend system architecture

Testing

The testing will be done using Postman to test the application. This is the testing result using Postman with a negative result, meaning a low risk of developing a heart attack. Please note that additional information has been added to make it more realistic but is irrelevant to the dataframe, such as first name, last name, height, weight, etc.

Figure 21 postman testing - with negative result

This is the result of a positive test using Postman, indicating that the patient has a higher chance of developing a heart attack.

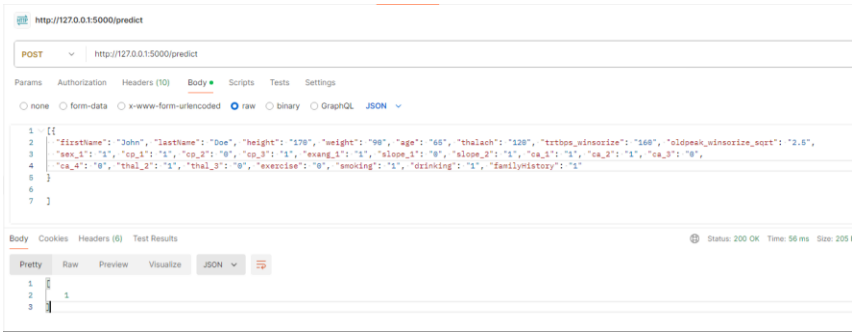

Figure 22 postman testing - with positive result

The appendix includes a demo of the front-end along with user experience results using dummy data.

User Interface Design

The user will interact with a series of pages, each containing relevant forms and informational content:

Personal Data Page: users will see the fields: First name, last name, height, weight. And additional information such as: "What is a heart attack?" with a short description to educate the user.

Medical Data Page 1:Fields: Age, maximum heart rate achieved, resting blood pressure, ST depression induced by exercise, sex. Information: "What causes a heart attack?" providing insight into the risk factors.

Medical Data Page 2: Fields: Chest pain types 1, 2, and 3. Information: "How does a heart attack occur?" offering a detailed explanation of the physiological process.

Medical Data Page 3: Fields: Exercise-induced angina 1, slope of peak exercise ST segment 1 and 2. Information: "What are the symptoms of a heart attack?" to help users recognize warning signs.

Medical Data Page 4: Fields: Number of major vessels 1, 2, 3, and 4. Information: "Who is at risk for a heart attack?" highlighting the risk factors associated with cardiovascular diseases.

Medical Data Page 5: Fields: Thalassemia 1 and 2.Information: "How can heart attacks be prevented?" offering preventive measures and lifestyle tips.

Additional Page: Fields: Exercise, smoking, drinking, family history of heart attack. Information: "What should you do if you suspect a heart attack?" providing immediate steps and first aid advice.

Each page will have 'Back' and 'Next' buttons to navigate through the stages. At the end, a 'Predict' button will be available to calculate the likelihood of a heart attack and provide personalized recommendations. This design ensures a comprehensive user experience, allowing individuals to update and review their information anytime they receive new reports from the laboratory.

By integrating these detailed steps and methodologies, the web application will not only be a powerful tool for predicting heart attack risk but also serve as an educational resource for users, promoting better understanding and proactive health management.

IMPLEMENTATION

Implementing the Heart Attack Prediction Web Application involved a methodical approach, combining data science techniques with software development practices. This section outlines the detailed steps and procedures we followed to bring our project to life.

Getting the Data Ready

Collecting and Preparing Data: We started by gathering a rich dataset from the Cleveland Heart Disease dataset available in the UCI Machine Learning Repository. This dataset contained detailed medical information from 303 patients, including factors known to influence heart attack risk.

Cleaning and Organizing Data: Before diving into predictions, we carefully cleaned the dataset:

- **Fixing Missing Data:** We filled in missing information using statistical methods or removed incomplete entries.
- **Spotting and Fixing Outliers:** We identified unusual data points that could skew results and corrected them to ensure our predictions were reliable.
- **Making Data Comparable:** To make sure all data played nicely together, we standardized measurements and adjusted scales where needed.
- **Finding New Insights:** We dug deep into the data, looking for patterns and connections that could help predict heart attacks more accurately.

Understanding the Data Better: We used charts, graphs, and statistical tests to explore and visualize the dataset. This helped us see which factors—like blood pressure, cholesterol levels, and exercise habits—might have the most impact on heart attack risk.

Building and Training Models

Choosing the Right Models: We tested several different machine learning techniques to see which ones could predict heart attack risk most effectively:

- **Logistic Regression:** This method was great for predicting yes-or-no outcomes, like whether someone might have a heart attack.
- **Decision Trees and Random Forests:** These models helped us understand which factors were most important in predicting risk, giving us more insight into our predictions.
- **Support Vector Machines (SVM):** SVMs were perfect for handling complex data relationships and making accurate predictions based on various medical factors.

Making Models Smarter: We didn't stop at just building models—we fine-tuned them to be as accurate as possible. By adjusting their settings and parameters, we ensured they could predict heart attack risk with high confidence.

Creating a User-Friendly App

Designing the Interface with You in Mind: Our web app was designed to be easy and intuitive to use. We used React.js to create a frontend that guided users step-by-step through entering their medical information.

Building a Reliable Backend: Behind the scenes, we used Flask, a lightweight framework in Python, to connect our frontend to our powerful machine learning models. This setup allowed us to provide real-time predictions based on the user's input.

Testing and Making Sure It Works

Checking Every Detail: We ran thorough tests to make sure everything worked smoothly. From checking how the app handled different inputs to ensuring predictions were accurate, every detail was scrutinized.

Getting Your Feedback: Before finalizing our app, we asked for feedback from users like you. Your input helped us improve the app's usability and ensure it met your needs effectively.

TESTING

The Testing our Heart Attack Prediction Web Application was crucial to ensure it worked flawlessly and provided accurate predictions. We used a combination of automated testing with Postman and manual testing to cover all aspects of the application.

Automated Testing with Postman

Automated Checks: Using Postman, we will automate the testing process to ensure the web application handles various inputs and scenarios correctly e.g. with the positive or negative results or high risk of heart attack or low risk of heart attack.

Verification Process: The Postman testing will replicate user inputs and assess the application's performance with different types of medical data. This approach will help to ensure the accuracy of the application.

Manual Testing

User-Centric Evaluation: We will conduct multiple rounds of manual testing on the web application, asking different users to test it using dummy data or data from the dataset. This will evaluate the overall user experience and functionality of our web application.

User Journey Evaluation: We will assess how user-friendly the web application is, including the process of inputting dummy data or data from the dataset and comparing this with our expectations.

Prediction Validation: We input various sets of medical data into the app manually to confirm the accuracy of the predictions generated by our machine learning models.

User Feedback Integration: During manual testing, we will be collecting feedback from users to identify any usability issues and areas for improvement. This will improve the user experience.

Ensuring Reliability

Testing Environment Variability: Testing the application under different conditions such as the speed of the internet or devices. However, some functionalities can be updated of the improved version of this application.

CONCLUSION

In this project, I have designed a Heart Attack Prediction Web Application using computer science techniques, specifically machine learning algorithms, to predict the likelihood of a heart attack based on user input data. Heart attacks are one of the most critical health problems worldwide and cause thousands of deaths annually (World Heart Federation, 2023). The application will be beneficial and accessible for individuals to identify risk factors and remain vigilant.

The application utilizes a dataset from the Cleveland Heart Disease dataset in the UCI Machine Learning Repository, which includes 14 medical parameters related to heart attack risk from 303 patients (Cleveland Heart Disease dataset , n.d.). After appropriate preprocessing, the data will be cleaned, normalized, and engineered, making it ready for training with various machine learning models.

In this project, several machine learning algorithms were evaluated: Logistic Regression, Decision Trees, Random Forests, and Support Vector Machines (SVM). Logistic Regression performed best, with a test accuracy of 0.8709, a cross-validation accuracy score of 0.8666, precision of 0.875, recall of 0.9545, and an F1 Score of 0.9130. Random Forest achieved a test accuracy of 0.8064, precision of 0.9, recall of 0.8182, and an F1 Score of 0.8571. SVM had a test accuracy of 0.87096, precision of 0.9, recall of 0.8181, and an F1 Score of 0.8571. Logistic Regression's superior performance metrics make it the chosen model for predicting the risk of heart attacks.

The development of the web application will involve using React.js for the front-end to facilitate user interaction and Flask, written in Python, for the back-end to handle model inference and predictions as discussed in the Methodology section above. The user interface will guide users through inputting their medical data step by step and provide educational content about heart attacks, risk factors, symptoms, and preventive measures.

This project combines advanced technology with medical knowledge to enhance early detection capabilities and offer an educational tool about cardiovascular health. The web application supports proactive health management by providing a well-designed structure for usability, responsiveness, and accuracy in predicting heart attack risks.

In the next version, real-time data from wearable health devices could be integrated into the application, allowing for continuous monitoring and analysis of personal risk. Moreover, more complex machine learning models, including deep learning, could be explored for even higher predictive accuracy.

This means that ethical considerations regarding medical data and other sensitive information are crucial. Compliance with data protection regulations, such as GDPR, will be necessary. During development, great emphasis will be placed on secure data management and transparency to ensure user trust and the protection of privacy.

REFERENCES:

Cleveland Heart Disease dataset . (n.d.). Retrieved from https://archive.ics.uci.edu/dataset/45/heart+disease

Federation, W. H. (2023, may 20). *Deaths from cardiovascular disease*. Retrieved from World Heart Federation: https://world-heart-federation.org/

IU Internationale Hochschule, G. (2023). *Data Science Course.*

medicine, n. l. (2022, june 17). *PMC9206502*. Retrieved from national library of medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9206502/

New York State, D. o. (2023). *Heart Disease and Stroke Prevention*. Retrieved from New York State, Department of Health: https://www.health.ny.gov/diseases/cardiovascular/heart_disease

Study, F. H. (n.d.). *FraminghamHeartStudy*. Retrieved from framingham-heart-study-fhs: https://www.framinghamheartstudy.org/

Turabieh, H. (2016). A Hybrid ANN-GWO Algorithm for Prediction of Heart Disease. *American Journal of Operations Research*.

APPENDIX A

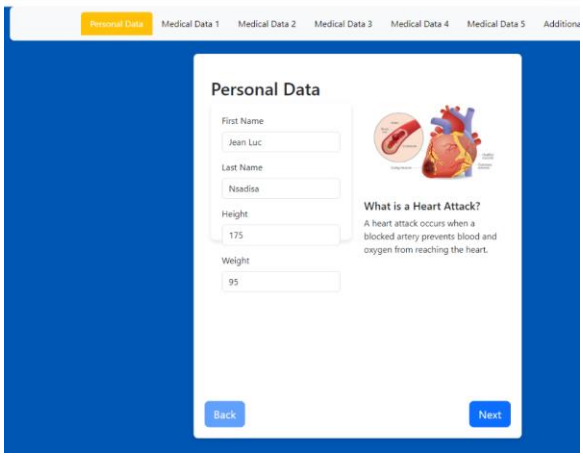

Figure 23 webApp personal data page

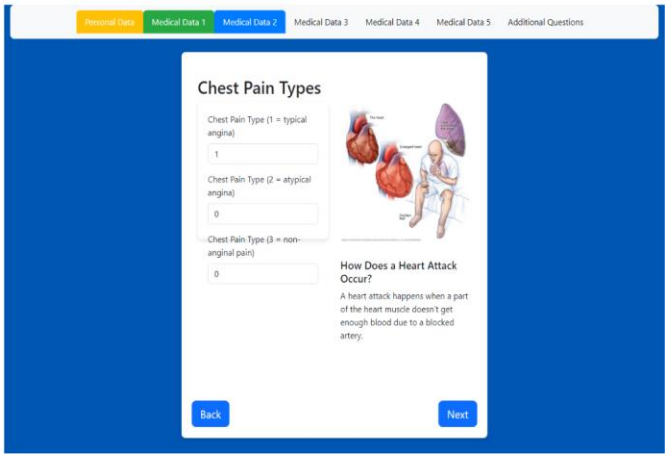

Figure 25 WebApp Medical page 2

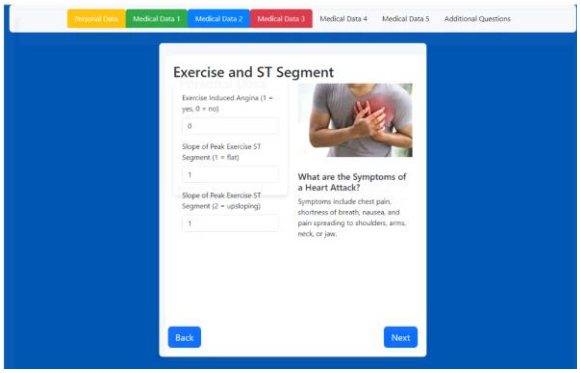

Figure 24 WebApp Medical page 3

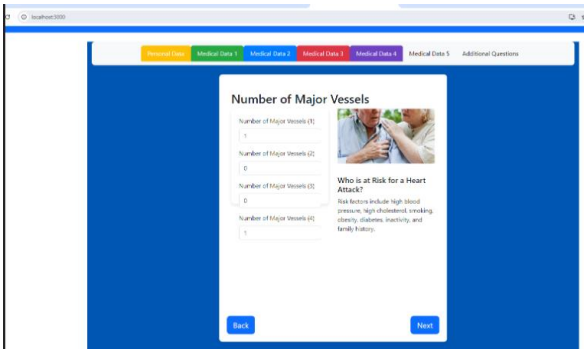

Figure 26 WebApp Medical page 4

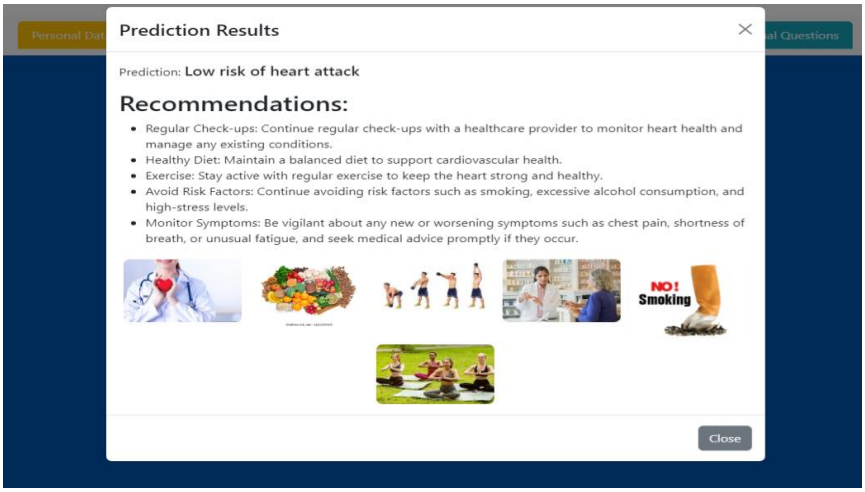

Figure 27 WebApp - Result of prediction

https://github.com/Claveur12/NSADISA_JEAN_LUC_9211128_DLMCSPCSP01/tree/master