EARIN Final Project: Heart Attack Prediction

Krszysztof Kotowski and Juan Manuel Aristizabal Henao

Warsaw University of Technology, Plac Politechniki 1, Warsaw, Poland

1 Introduction

This project aims to develop a predictive model that identifies individuals at increased risk of experiencing a heart attack based on clinical and demographic attributes. The primary objective is to build a machine learning model that receives patient data– such as age, cholesterol level, blood pressure, etc.– and outputs a binary classification indicating whether the patient is at high risk of a heart attack.

The data is taken from the **Heart Attack Prediction Dataset** available on Kaggle, which contains 8763 records of patients publicly available. The algorithms used will be: Binary Classification for outputting the prediction of whether or not the patient has a high risk of a heart attack, and Supervised Training algorithms will be used to train the model. The algorithms that can be suitable for this study are: Logistic Regression, Decision Trees, Naive Bayes, Support Vector Machines (SVM), and Random Forests.

2 Dataset Description

The dataset used contains 8763 records each representing a patient's medical profile, with 26 parameters each. A sample of this information can be seen on the Table 2, where the record of Patient 1 is visualized.

The description of each parater of the dataset, according to the information provided on Kaggle is the following:

- **Patient ID**: Unique identifier for each patient
- **Age**: Age of the patient in years
- **Sex**: Gender of the patient (Male/Female)
- **Cholesterol**: Cholesterol levels of the patient
- **Blood Pressure**: Blood pressure of the patient (systolic/diastolic)
- **Heart Rate**: Heart rate of the patient
- **Diabetes**: Whether the patient has diabetes (1 = Yes; 0 = No)
- **Family History**: Family history of heart-related problems (1: Yes, 0: No)
- **Smoking**: Smoking status of the patient (1: Smoker, 0: Non-smoker)
- **Obesity**: Obesity status of the patient (1: Obese, 0: Not obese)
- **Alcohol Consumption**: Level of alcohol consumption by the patient (None/Light/Moderate/Heavy)
- **Exercise Hours Per Week**: Number of exercise hours per week
- **Diet**: Dietary habits of the patient (Healthy/Average/Unhealthy)

- **Previous Heart Problems**: Previous heart problems of the patient (1: Yes, 0: No)
- **Medication Use**: Medication usage by the patient (1: Yes, 0: No)
- **Stress Level**: Stress level reported by the patient (1-10)
- **Sedentary Hours Per Day**: Hours of sedentary activity per day
- **Income**: Income level of the patient
- **BMI**: Body Mass Index (BMI) of the patient
- **Triglycerides**: Triglyceride levels of the patient
- **Physical Activity Days Per Week**: Hours of sleep per day
- **Sleep Hours Per Day**: Target variable (1 = risk of heart attack; 0 = no risk)
- **Country**: Country of the patient
- **Continent**: Continent where the patient resides
- **Hemisphere**: Hemisphere where the patient resides
- **Heart Attack Risk**: Presence of heart attack risk (1: Yes, 0: No)

Data preprocessing For the goal of this model some data preprocesing and cleaning have to be done before the training of the model. The following are the changes to be done to the parameters for training the model:

- **Patient ID**: Will be ignored since it has no relevance for this study since it a parameter that is unique for each patient.
- **Sex**: If the imbalance (70:30 for Male:Female) of that parameter can be addressed properly, its values can be converted to integer values that store 0 for Male and 1 for Female, otherwise this parameter should be ignored.
- **Blood Pressure**: Can be divided into two columns since it has tho values, the first one for *Systolic Pressure* and the second one for *Disatolic Pressure*. Both of these columns will have integer as a datatype, and will be grouped into representative ranges.
- **Diet**: Can be converted to integer values (-1 = Unhealthy, 0 = Average, and 1 = Healthy).
- **Country, Continent and Hemisphere**: Only the *Country* column should be kept since from it, the rest of the information contained on the *Continent* and *Hemisphere* columns can be inferred.
- **Smoking**: Should be ignored since the data imbalance in this parameter is of about 90:10 (Yes:No), which may generate a high bias and poor performance on the minority class (non-smokers).
- **Diabetes**: Can be considered for the study even if it has a significant imbalance (65:35 for Yes:No), but a specific metric can be created to measure the bias of the model and its performance on the minority class (no diabetes)
- **Exercise Hours Per Week, Sedentary Hours Per Day and BMI**: Can be rounded to 2 decimal points, and the data grouped into representative ranges.
- **Age, Income, Cholesterol, Heart Rate, and Triglycerides**: Can be grouped into representative ranges.

Table 1. Sample of the dataset to show the parameters present and the data format and types.

Parameter	**Patient 1**
Patient ID	BMW7812
Age	67
Sex	Male
Cholesterol	208
Blood Pressure	158/88
Heart Rate	72
Diabetes	0
Family History	0
Smoking	1
Obesity	0
Alcohol Consumption	0
Exercise Hours Per Week	4.168188835442079
Diet	Average
Previous Heart Problems	0
Medication Use	0
Stress Level	9
Sedentary Hours Per Day	6.61500145291406
Income	261404
BMI	31.2512327252954
Triglycerides	286
Physical Activity Days Per Week	0
Sleep Hours Per Day	6
Country	Argentina
Continent	South America
Hemisphere	Southern Hemisphere
Heart Attack Risk	0

For the predicted parameter (**Heart Attack Risk**) there is a considerable imbalance of 64:36 for No: Yes. This imbalance can be treated by implementing an imbalance treatment technique that can bring closer both values, so that to minimize the risk of poor performance over people that have high risk of heart attack. A proposed solution to this issue can be using an oversampling approach that will duplicate the entries that have high risk, so that the ratio may look more like 50:50.

If the above-mentioned approach happens to yield poor results, more data could be scraped from other websites that also publicly offer access to this data.

Metrics Our model's results will be analysed using:

- Accuracy - Percentage of correct predictions in general
- Confusion matrix - Plotting model's results against actual results
- Precision - How many positive classifications (results) are actually correct?
- Recall - How many actually positive cases were detected by the model?

3 Algorithms used

Common methods used for Binary Classification problems are Logistic Regression, Decision Trees, Naive Bayes, Support Vector Machines (SVM) and Random Forests.

3.1 Logistic Regression

Logistic regression is a supervised machine learning algorithm that performs binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers probabilities mapped to classes (typically 0 or 1).

This is one of the methods which will be implemented. A sigmoid function will be used to map the probabilities. For the data preprocessing, The previously explained data preprocessing will be used in addition to measure the linearity of the relationships between the parameters, to determine if they are suitable to train the model in this way.

3.2 Decision Trees

A decision tree is a flowchart-like model that maps out the possible outcomes of a series of related choices, helping individuals or organizations evaluate potential actions based on their costs, probabilities, and benefits. It's a visual tool that can also be used to build predictive models in machine learning.

3.3 Random Forests

A Random Forest is a machine learning algorithm that combines multiple decision trees to improve prediction accuracy and robustness. It's an ensemble method that utilizes bagging, where each tree is trained on a random subset of the data and considers only a random subset of features at each node split. By averaging or voting the predictions of these trees, Random Forests produce more reliable results than a single decision tree.

This is one of the methods that will be used. The previously mentioned data processing will be done, with a focus on some data augmentation if the results are of poor quality.

3.4 Support Vector Machines

Support Vector Machines (SVMs) are supervised machine learning algorithms used for classification and regression tasks, particularly effective in high-dimensional spaces. They work by finding a hyperplane that best separates different classes of data, maximizing the margin between them. This "margin" represents the distance between the hyperplane and the closest data points from each class, known as support vectors.

3.5 Naive Bayes

Naive Bayes is a simple yet powerful classification algorithm based on Bayes' Theorem, assuming that features are independent of each other. It calculates the posterior probability of each class given the input features and selects the one with the highest probability. It's widely used for tasks like text classification, spam filtering, and sentiment analysis due to its speed and efficiency, even with small datasets. Despite the simplifying "naive" assumption of feature independence, it often performs surprisingly well in practice.

4 Implementation

Project consists of a single source code file "main.py". Is is supposed to be run using "python main.py" command and program automatically aquires training data from "heart.csv" file and outputs results to "/New results/" directory. It creates 8 model result output files names "results_<model_name>.csv"for each model and trivial models with paramters and result metrics for trainings and tests. Additionally, one more file is created named "results_feature_importances _from_fr_clsf.csv" that contains feature importances of input dataset, aquired from training a single random forest classifier model.

Table 2. Intermediate results from the experiments.

Name	Best accuracy	Best precision	Best recall
Logistic regression	0,641756988	0,356334842	0,506369427
Decision tree	0,641186537	0,407185629	0,751592357
SVM	0,641756988	0,5	0,535031847
Random forest	0,64346834	1	0,324840764
Gaussian NB	0,641756988	0	0
Random	0,494010268	0,355307263	0,506369427
Constant 0	0,641756988	0	0
Constant 1	0,358243012	0,358243012	1

5 Intermediate results

The results obtained after training the models with different paramters are shown on the Table 2.

From these results it can be seen that the accuracy of the models is around 64%, which is not a totally bad result but it still has more space to improve. The most underperforming method implemented was the Gaussian NB, with a precision and recall of 0%, indicating that the parameters need additional tuning.

In terms of precision, the Random Forest model outperformed the rest with a precision of 100% in some cases, but overall speaking from the experiments, the average precision of the models was between 30% and 40%, which is still suboptimal, and raises the need to both tune the parameters more and/or add more data to the dataset (e.g. oversampling).

In terms of recall, the best performing model was the Decision Tree with around 75%, which aids in identifying the set of parameters that make the model perfom overall well in the predictions. The other models had a recall of around 50% or less, which shows that the models are struggling to understand the patterns of people with an actual high risk of heart attack.

One significant feature that we have learned from this is that the class weight of the Logistic Regression needs to be set to balanced, otherwise the predictions were always 0.

5.1 Change of dataset

In addition, it was discovered that the dataset used for this had comments on Kaggle mentioning that there were no patterns discovered in the data, causing the under-performance of the models trained on this dataset. Because of that the dataset used was changed to one that was provided to us by the professor, also used for predicting Heart Attack Risk.

5.2 Description of new dataset

The new dataset consists of 303 rows with the following patient information:

- Age: the patient's age, represented as an integer
- Sex: the patient's sex, encoded as binary (1 or 0, with the specific gender representation for these values
- Chest Pain Type (cp): the type of chest pain experienced by the patient, categorized into four types: 0 for typical angina, 1 for atypical angina, 2 for non-anginal pain, and 3 for asymptomatic
- Resting Blood Pressure (trtbps): the patient's resting blood pressure in millimeters of mercury (mmHg)
- Cholesterol (chol): the level of cholesterol in milligrams per deciliter (mg/dl).
- Fasting Blood Sugar (fbs): indicates if the fasting blood sugar level is greater than 120 mg/dl (1 = true, 0 = false)
- Resting Electrocardiographic Results (restecg): results expressed as 0 for normal, 1 for having ST-T wave abnormality, and 2 for probable or definite left ventricular hypertrophy.
- Maximum Heart Rate Achieved (thalachh): the maximum heart rate the patient has achieved.
- Exercise Induced Angina (exng): whether the angina was induced by exercise (1 = yes, 0 = no).
- Previous Peak (oldpeak): the previous peak, measured as a float.
- ST Segment Slope (slp): the slope of the peak exercise ST segment, with values of 0, 1, or 2
- Number of Major Vessels (caa): the number of major vessels seen in fluoroscopy, represented as an integer.
- Thallium Stress Test Result (thall): results from the Thallium stress test, categorized as 0, 1, 2, or 3.

The predicted variable: Outcome (output) is the target variable indicating if the patient is at risk of a heart attack (1 = yes, 0 = no).

Data processing of the new dataset The data provided had no missing or incorrect values. In addition, none of the values of the columns needed changed since the data present on them was already in a numerical format. Finally, the data did not present any significant imbalances on the predictor variables and neither on the predicted variable. Hence, no major changes were required to be made to the data before training the models.

6 Final results

The following are the final results obtained after training the models with the new dataset provided.

6.1 Compilation of best metrics

The best results are depicted on the Table 3 for each of the evaluated models according to their accuracy, precision, and recall.

Table 3. Compilation of best metric values for each model

Name	Best accuracy	Best precision	Best recall
Logistic regression	86,9%	87,5%	87,5%
Decision tree	83,6%	82,4%	97,0%
SVM	85,2%	80,6%	100,0%
Random forest	88,5%	84,2%	97,0%
Gaussian NB	82,0%	78,9%	90,9%
Random	42,6%	47,1%	48,5%
Constant 0	45,9%	0,0%	0,0%
Constant 1	54,1%	54,1%	100,0%

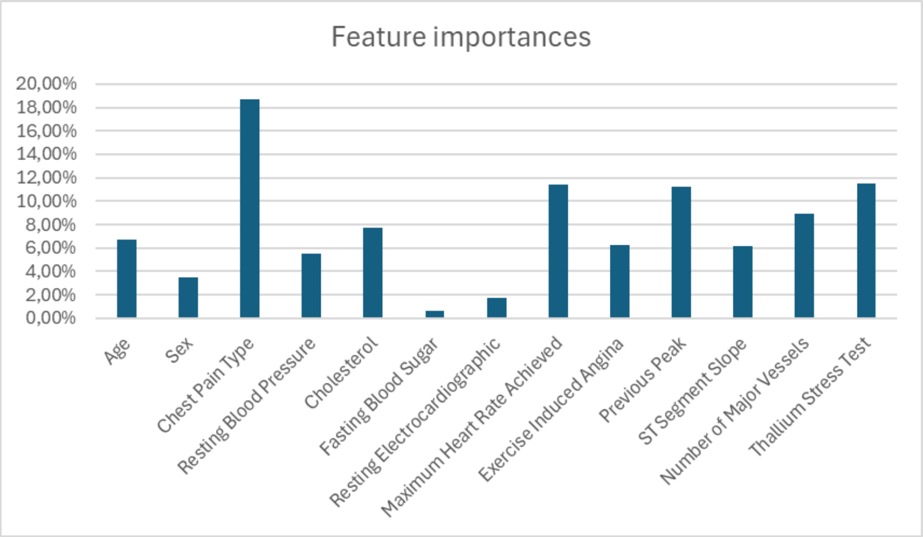

Fig. 1. Figure 1. Input dataset feature importance in percentage

6.2 Dataset feature importances

Feature importances were assessed from random forest classifier model trained with parameters:

- n_estimators=100
- criterion='gini'
- max_depth=10
- min_samples_split=5
- min_samples_leaf=2
- class_weight='balanced'
- random_state=42

6.3 Final experiment results and Conclusions

All of trained models proved to predict heart attack for the given dataset better than random or trivial answering. Out of those five, Random Forest Classifier proved to have the highest accuracy, while Gaussian Naive Bayes the lowest one, although the diffence betwee them is not as signifcant as with the previous datasets. It is also important to mention that the accuracy of all model raised with this new dataset to 82% to 86% which is much better than the results previously achieved.

In the case of precision, the results were again much better than with the previous dataset, achieving precisions ranging from 78% to 87%, being Gaussian Naive Bayes the one with the lowest precision and Logistic regression the one withe the highest presicion.

Finally, in the case of recall, the SVM model achieved 100% of recall, which made it the highest one, and for the lowest, the Logistic regression achieved 87%.

6.4 Problems encountered

There were two critical problems that were encountered during model training. The first was the wrong initial dataset that contained data that was probably randomly generated and was unusable, but this was solved by purchasing the second valid dataset. The second problem was with the "class weight" parameter in python model class. This was set to None by default and it yielded much worse results than models with it being set to 'balanced.'