Predicting Heart Disease

Diogo Cerveira
up201907427

Diogo Mendes
up202005393

João Amorim
up202009074

Abstract—**Detecting heart diseases is of paramount importance due to their widespread prevalence and significant impact on public health. Cardiovascular diseases, including heart attacks and strokes, are leading causes of mortality globally. Early detection plays a crucial role in implementing timely interventions and preventive measures, potentially saving lives and reducing the burden on healthcare systems. Machine Learning (ML) emerges as a powerful tool in this context, offering the ability to analyze vast datasets, identify intricate patterns, and make accurate predictions. The integration of ML in heart disease detection signifies a paradigm shift towards proactive healthcare, enabling more efficient and precise strategies for managing cardiovascular health on a large scale. This report explores the application of machine learning algorithms to predict heart disease using the Cleveland Heart Disease Database. The study employs the Naive Bayes, Logistic Regression, Decision Trees, and K-Nearest Neighbors models. The dataset, containing various patient features, undergoes preprocessing, including handling outliers, scaling, and encoding categorical variables. Feature analysis involves heatmap visualization and Principal Component Analysis. The optimization of hyperparameters and the assessment of model performance were conducted through the utilization of GridSearchCV from the scikit-learn library. The objective was to fine-tune the parameters of various models while employing a 5-fold cross-validation strategy. Among the models explored, Logistic Regression exhibited superior performance, boasting a Recall of 0.792, an F1-Score of 0.826, and a ROC AUC of 0.902.**

Key words — Heart Disease, Naive Bayes, Logistic Regression, KN-Neighbors.

I. Introduction

Heart diseases belong to a group of conditions affecting the heart such as blood vessel disease, arrhythmia, congenital heart defects and disease of the heart muscle and valves. [6] This range of conditions are currently the main cause of death in the world, causing around 20.5 million of deaths around the world, every year. [12] In Portugal, 35 thousand people die annually due to these kind of diseases.[7]

In the context of machine learning, assessing the risk of a patient developing heart disease means creating models to predict the healthiness of the heart's patient, from a set of data features collected from a large sample of patients.

In this project was chosen a dataset containg 13 different features, referring to 270 observations, and provided by the *Cleveland Heart Disease Database*. [1]. Below are shown the features used, abbreviations and the respective types:

- Age (*age*) - Real
- Sex (*sex*) - Binary
- Chest pain type (*cp*) - Nominal
  Patients are qualitatively categorized according to 4 possible types of chest pain.
- Resting blood pressure (*trestbps*) - Real
- Serum cholesterol in mg/dl (*chol*) - Real
  A person's serum cholesterol level comprises the amount of high-density lipoprotein (HDL), low-density lipoprotein (LDL), and triglycerides in the blood. [13]
- Fasting blood sugar $>$ 120 mg/dl (*fbs*) - Binary
  This attribute indicates whether or not the patient has blood glucose (sugar) levels above 120 mg/dl. [20]
- Resting electrocardiographic results (*restecg*) - Nominal
  A person's resting electrocardiographic results refer the electrical activity of his heart during the rest. In the used data set, patients are classified according 3 possible levels of resting electrocardiographic activity. [2]
- Maximum heart rate achieved (*thalach*) - Real
- Exercise induced angina (*exang*) - Binary
  This attribute indicates whether or not the patient has chest pain or tightness (angina) during exercise or emotional stress, due to inefficient pumping of blood resulting from narrowing of blood vessels. [8]
- Oldpeak = ST depression induced by exercise relative to rest (*oldpeak*) - Real
  The ST segment represents the period in the cardiac cycle when the myocardium maintains contraction to expel blood from the ventricles. The ST depression occurs when this segment is below baseline in ECG. This attribute indicates quantitatively the size of this depression, induced by exercise. [19]
- The slope of the peak exercise ST segment (*slope*) - Ordered
  This attribute refers to the angle or orientation of the ST segment during exercise.
- Number of major vessels (0-3) colored by flourosopy (*ca*) - Real
- Thal (3 = normal; 6 = fixed defect; 7 = reversable defect) (*thal*) - Nominal
  This attribute indicates the results of thallium stress test. This test involves a small amount of a radioactive substance injected intravenously and measures the blood flow to the heart during rest and during exercise. [17] In the used data set, patients are classified according 3 possible levels.

Binary attributes represent two possible categorical states. *real* attributes are numerical measurable features represented in real values. *nominal* attributes are values that represent a category without an order between them. *ordered* (or ordinal) attributes are categorical values organised in a ranking without a meaningful magnitude between them. [10]

The target variable is coded as 1 to indicate the absence of heart disease and 2 to signify the presence of heart disease.

II. Methods

A. Data Preprocessing

The initial step in data preprocessing involves identifying and dealing with any **missing or duplicated values** in the dataset. Missing data indicates that certain information is not available, potentially undermining the statistical robustness of the model analysis, and compromising the accuracy and reliability of the overall analysis results.

The presence of **outliers**, observation that lies an abnormal distance from other values in a random sample from a population, can deprecate the fitting of the models. Often, they reflect poor data points, associated with errors in the data collection, but they can also provide important information about the feature. Knowing that the Interquartil Range (IQ) is the difference between the upper quartile (Q3) and the lower quartil (Q1), an extreme data point is considered an mild outlier if farther than 1.5 x IQ above or below Q3 and Q1, respectively. If this distance is 3 x IQ, it is considered an extreme outlier. The *Cleveland Heart Disease Database* dataset showed the presence of mild outliers in all real features. However, serum cholesterol presented an extreme outlier, which was removed before model training (the corresponding box-and-whiskers plot is available the appendix 3). [16]

Subsequently, numerical variables underwent scaling, a crucial step for various reasons: it helps machine learning algorithms that calculate distances between data to perform better, by bringing all features to the same scale, ensuring that no single feature has dominance over the others due to scaling. This is particularly notorious for algorithms like k-nearest neighbors, support vector machine and neural networks. Additionally, feature scaling can help optimization algorithms converge faster, prevent models from being biased towards certain features, and make the training process more stable. [22] Normalization and standardization are two common techniques used in data preprocessing to scale the features of a dataset. **Normalization** typically rescales the features to a range between 0 and 1 (or -1 and 1),and is useful when the distribution of the data does not follow a Gaussian distribution. What's more, normalization is sensitive to outliers, as it scales the data based on the minimum and maximum values of each feature. **Standardization** involves scaling the features so that they have a mean of 0 and a standard deviation of 1. This technique assumes that the data follows a Gaussian distribution and is less sensitive to the presence of outliers compared to normalization. [9]

Although there are several categorical features in the dataset, they were already label encoded, which involves replacing each set or category with a corresponding numerical identifier, maintaining the consistency of these assigned numbers across the entire feature set. [23] Nevertheless, label encoding might not be a good practice, since when directly converting categories into numerical values, relationships are introduced (such as distance and order), even if such relationships did not originally exist among the initial categories. This is specially an issue in models that perform distance measures such as K-nearest neighbors. So, for the non-ordinal categorical features (*cp*, *restecg*, *slope*, *thal*), **one-hot encoding** was performed. This method consists in generating an extra feature (*dummy*) for every distinct group within a categorical feature. Each observation is then labeled as either belonging (assigned a value of 1) or not belonging (assigned a value of 0) to that specific group, effectively converting a nominal feature to multiple binary ones. [23]

B. Feature Analysis

1) Heatmap: To assess the presumed linear relationships within the dataset features, a **heatmap** was generated using the *seaborn* library's *heatmap()* function, as illustrated in the appendix (figure 2). While heatmaps are traditionally applied to numerical features, the simplicity of categorical binary features makes them amenable to a numeric representation, thereby revealing potential relationships. In addition to all numerical features, three categorical features (*sex*, *fbs*, and *exang*) were included for analysis.

2) Principal Component Analysis: Interpreting the heatmap directly can be challenging, prompting a deeper analysis through **Principal Component Analysis** (PCA). PCA facilitates dimensionality reduction and addresses feature correlation, particularly beneficial for the Naive Bayes algorithm, which assumes feature independence. In a simplified manner, continuous data is standardized, the covariance matrix is computed, and eigenvectors and eigenvalues are determined to identify principal components. A feature vector guides the selection of principal components, and the database is recreated accordingly.[18] The *PCA* class from *sklearn.decomposition* was employed for the six numerical features. Subsequently, Cumulative Explained Variance (CEV) was plotted against the number of Principal Components (PC) to aid in choosing the optimal number. While PCA may compromise feature interpretability by combining them into PCs, it effectively mitigates overfitting and enhances data visualization, particularly when reducing dimensionality to three or fewer dimensions.

C. Data Models

1) Naive Bayes: The **Naïve Bayes** (NB) classifier is a highly simplified variant of the Bayesian probability model. It assumes that features are conditionally independent, meaning they have zero correlation to other features in the model. Additionally, the model assumes that all features contribute equally to the outcome. While these assumptions may not hold in real-world scenarios, it allows for simplified classification problems, requiring only a single probability for each variable. Despite the unrealistic assumption of independence, the Naïve Bayes classification algorithm performs well, especially when dealing with small sample sizes. [14] The mathematical expression for the Naive Bayesian probability model is as follows:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

$$P(y|X) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y) \cdot P(y)$$

$$y_{NB} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Gaussian Naive Bayes was implemented using *GaussianNB()* method from *scikit-learn*. It is specifically designed to work with continuous data that follows a Gaussian (normal) distribution. This classifier assumes that the likelihood of the features is Gaussian within each class. *CategoricalNB()* (from the same module) was also employed for only categorical features, with and without *fbs*.

2) Logistic Regression: **Logistic Regression** is a supervised machine learning algorithm primarily employed for binary classification tasks. It utilizes a logistic function, commonly known as a sigmoid function. This function takes independent variables as input and generates a probability value within the range of 0 to 1. [11]

The logistic function is defined as:

$$P(Y = 1|X) = \frac{1}{1 + e^{-z}}, z = XW + b$$

The log-odds (logit) is another way to represent the logistic regression equation:

$$\log\left(\frac{P(Y = 1|X)}{1 - P(Y = 1|X)}\right) = z$$

Training the Logistic Regression model involves optimizing the weights and bias to maximize the likelihood of the observed outcomes. The *LogisticRegression()* function, from *scikit-learn*, is used to create and train logistic regression models in Python. Furthermore, the *LogisticRegression()* function provides various parameters to be customized, such as the *solver*, *regularization* (penalty), and the *C* parameter, providing margin for optimization of the model. *GridSearchCV* from *scikit-learn* is used to tune the hyperparameter *C*, which controls the regularization strength of the model.

3) Decision Trees: **Decision trees** are a popular machine learning model used for both classification and regression tasks. They work by recursively splitting the data into subsets based on the most significant feature at each node. This process continues until the data within a given subset is as homogenous as possible with respect to the target variable, or until a stopping criterion is met. The resulting tree structure consists of decision nodes and leaf nodes, where the decision nodes represent the features and the leaf nodes represent the target variable or the outcome.

The key steps in building a decision tree involve selecting the best feature to split the data, determining the splitting criterion (e.g., Gini impurity or entropy), and setting stopping criteria to control the tree's depth and prevent overfitting. Decision trees are valued for their interpretability, as they provide a clear visual representation of the decision-making process. However, they can become overly complex, so techniques like pruning are often used to control their size and prevent overfitting.

Some common hyperparameters that are tuned in decision trees include the maximum depth of the tree, the minimum samples required to split a node, the maximum number of leaf nodes, and the splitting criterion. Tuning these hyperparameters is essential for optimizing the model's performance, controlling its complexity, and avoiding overfitting. *GridSearchCV* was used to tune the maximum depth of the tree and the minimum samples required to split a node.

4) K-Nearest Neighbors: The **K-Nearest Neighbors** algorithm (KNN) is a supervised learning classifier that is based on distances and relies on proximity to classify data points. It uses a non-parametric method since it is not a particular finding of parameters for a functional form. It assumes that similar data points are typically found close to each other. It operates as a lazy classifier, memorizing the training data instead of learning fixed weights. As KNN is a distance-based algorithm, standardization should be performed. It is essential to ensure that all features are in a similar scale, preventing a feature with a larger scale from disproportionately influencing distance calculations. The next step is to choose the value of K, which is done by calculating the error for multiple K values (number of neighbors), ranging from 1 to 30. Choosing a small value for K may lead to overfitting, where the model performs well on the training data but poorly on unseen testing data. Additionally, it can result in sensitivity to noise, especially when dealing with large datasets. Using a distance measure such as the *Euclidean*, the distances between the new data point and its K nearest neighbors are calculated. The Euclidean distance function is defined as:

$$EuclideanDistance = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$$

Afterwards, the model identifies the class to which the majority of these neighbors belong. Selecting the class with the highest count among the K neighbors is assigning it to the new data point. [21]

D. Evaluation Metrics

1) Accuracy: Rate of correct classifications that a trained machine learning model achieves [15]

2) Precision: Number of true positives divided by the total number of positive predictions (true and false positives) [3]

3) Recall: The proportion of true positives among all actual positives. [4]

4) F1-Score: Harmonic mean of precision and recall.[5] Defined as:

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}}$$

5) ROC-AUC: The *Receiver Operating Characteristic Area Under the Curve* is a metric condensing a binary classification model's ability to distinguish positive and negative instances using the area under the ROC curve.

The primary objective of the model is to predict heart disease, and the paramount focus lies in minimizing false negatives. False negatives represent cases where the model predicts the absence of heart disease for patients who actually have it. While various metrics were considered during the hyperparameter tuning process, special emphasis was placed on Recall, F1-score, and ROC-AUC. These metrics played a decisive role in selecting the optimal hyperparameters, ensuring that the model is particularly adept at identifying individuals with heart disease and minimizing the risk of false negatives.

III. Results and Discussion

A. Data Preprocessing

At the outset, it's crucial to note that our dataset exhibited a clean structure with no missing or duplicate values. In the pursuit of optimizing model performance, we conducted thorough tests involving both normalization and standardization. Notably, across various models, standardization consistently outperformed normalization. The decision to favor standardization is substantiated by the visualizations presented in the appendix (figures 7, 4, 6, 5 and 8) illustrating the distributions of the features. These visual representations underscore the approximation of continuous feature distributions to Gaussian distributions. Consequently, standardization emerges as the preferred scaling technique, showcasing superior results. Furthermore, the dataset's characteristics include the presence of outliers. In such cases, standardization is inherently better equipped to handle outliers, aligning with our expectations. The robustness of standardization in the face of outliers contributes to its superior performance across the models evaluated. The bar plot of the target class (figure 9) allows the assumption of a balanced dataset.

B. Feature Analysis

Furthermore, by looking at the Density distributions of the features *chol* (figure 10) and *trestbps* (figure 11), and the normalized bar plot for the nominal feature *fbs* (figure 12), it's possible the deduct, individually, some of the correlation insignificance available in the heatmap (figure 2). From observing the Cumulative Explained Variance plot in figure 13, it's clear that 3 PC are enough to explain almost 100% of the variance, providing values of 71.532104% 17.005023% 9.418364%, sequentially.

C. Naive Bayes

In the initial attempt, Gaussian Naive Bayes was applied to all available features. However, recognizing that Gaussian Naive Bayes is specifically designed for continuous variables, only the continuous features were chosen for further analysis, aligning with the method's underlying assumptions. It's imperative to note that a Naive Bayes Classifier operates under the assumption of feature independence. Thus, when features exhibit high correlation, this can lead to a scenario where the model inadvertently magnifies the importance of these correlated features, resulting in an inflated influence on the overall model. Using the results from PCA, the 6 numerical features *age*, *trestbps*, *chol*, *thalach*, *oldpeak*, *ca*, can be successfully converted into 3 independent PCs. In the heatmap (figure 2) it is also possible to note this significant correlation between these features.

The subsequent evaluation of the Naive Bayes model produced the performance metrics showcased in the following table.

Features\Metric	**Recall**	**F1 Score**	**ROC AUC**
All (Train)	0.827	0.840	0.918
All (Test)	0.800	0.821	0.902
Numerical (Train)	0.710	0.738	0.842
Numerical (Test)	0.700	0.721	0.827
Numerical w/o ca (Train)	0.650	0.680	0.804
Numerical w/o ca (Test)	0.642	0.666	0.789
Numerical w/o thalach (Train)	0.640	0.696	0.818
Numerical w/o thalach (Test)	0.617	0.680	0.806
Numerical w/o trestbps and chol (Train)	0.694	0.725	0.835
Numerical w/o trestbps and chol (Test)	0.692	0.723	0.827
PCA Numerical (Train)	0.798	0.798	0.897
PCA Numerical (Test)	0.792	0.792	0.881
PCA Numerical and Categorical (Train)	0.556	0.620	0.760
PCA Numerical and Categorical (Test)	0.533	0.601	0.741
(CategoricalNB) Categorical (Train)	0.808	0.798	0.895
(CategoricalNB) Categorical (Test)	0.800	0.787	0.883
(CategoricalNB) Categorical w/o fbs (Train)	0.808	0.798	0.894
(CategoricalNB) Categorical w/o fbs (Test)	0.800	0.787	0.885

Table I
Performance Metrics for Different Models

According to the table, the combination of features that performed better was the one with all the features - numerical and categorical. Having in mind that Gaussian Naive Bayes assumes a normal distribution for each feature, this result is hard to explain since categorical features don't follow a normal distribution (even when label encoded as it is the case). Between the continuous features combination and the same without thalach, the metrics don't change too much suggesting that the thalach correlation with age and oldpeak isn't too significative. Probably a better approach would be to perform Gaussian Naive Bayes on continuous features and Categorical Naive Bayes (multinomial and/or bernoulli) and mix them but such task wasn't accomplished. It would also be expected to observe better results with the PCs. None of the removal of features proved useful for the success of Gaussian Naive Bayes. The Categorical Naive Bayes results with and without *fbs*, are more evidence to the possible irrelevance of this feature in the prediction of heart disease.

D. Logistic Regression

The following table shows the average performance metric values on the training and test sets for the C hyperparameter tuning using GridSearchCV with a 5-fold cross-validation.

Considering that the optimal hyperparameter selection is crucial for achieving the best model performance, we observe variations in Recall, F1-score, and ROC AUC as the C

param_C	Recall	F1 Score	ROC AUC
0.001 (Train)	0.200	0.333	0.904
0.001 (Test)	0.200	0.324	0.895
0.01 (Train)	0.719	0.804	0.918
0.01 (Test)	0.708	0.794	0.910
0.1 (Train)	0.825	0.865	0.935
0.1 (Test)	0.775	0.824	0.915
1 (Train)	0.827	0.858	0.942
1 (Test)	0.767	0.817	0.912
10 (Train)	0.838	0.865	0.943
10 (Test)	0.775	0.816	0.904
100 (Train)	0.840	0.866	0.943
100 (Test)	0.792	0.826	0.903
1000 (Train)	0.840	0.866	0.943
1000 (Test)	0.783	0.822	0.902

Table II
PERFORMANCE METRICS FOR DIFFERENT VALUES OF C

parameter changes. Given the objective of prioritizing high recall without compromising the F1-score, the model with C=100 stands out as the top performer. This model achieves a Recall of 0.792, F1-score of 0.826, and ROC AUC of 0.902. These results indicate that the model effectively identifies a substantial proportion of true positive cases while maintaining a balanced trade-off between precision and recall.

E. Decision Trees

The following table presents the performance metrics for different combinations of hyperparameters (Max Depth and Min Samples Split) during the tuning process of a Decision Tree Classifier. The metrics include Recall, F1 Score, and ROC AUC for both the training and test sets. The goal is to identify the combination that optimally balances model complexity and generalization performance.

Max Depth	Min samples split	Recall	F1 Score	ROC AUC
3 (Train)	2	0.810	0.838	0.912
3 (Test)	2	0.708	0.733	0.811
3 (Train)	4	0.810	0.838	0.912
3 (Test)	4	0.708	0.733	0.811
3 (Train)	8	0.810	0.838	0.912
3 (Test)	8	0.708	0.733	0.811
5 (Train)	2	0.902	0.939	0.982
5 (Test)	2	0.708	0.721	0.752
5 (Train)	4	0.898	0.934	0.981
5 (Test)	4	0.683	0.710	0.753
5 (Train)	8	0.885	0.914	0.974
5 (Test)	8	0.692	0.714	0.790
7 (Train)	2	0.967	0.983	0.999
7 (Test)	2	0.708	0.718	0.748
7 (Train)	4	0.940	0.964	0.997
7 (Test)	4	0.700	0.729	0.770
7 (Train)	8	0.910	0.929	0.988
7 (Test)	8	0.692	0.708	0.792

Table III
PERFORMANCE METRICS FOR DIFFERENT VALUES OF max_depth AND $min\_samples\_split$

Observing the performance metrics, it becomes apparent that the model exhibits a notable drop in performance on the test set, indicating a tendency to overfit the training data. This overfitting tendency tends to escalate with an increase in Max Depth. The hyperparameter combination that demonstrates the most favorable performance is associated with Max Depth = 3. Interestingly, this holds true for any value of Min Samples Split, as the performance metrics remain consistent across various split configurations (Recall = 0.708, F1-Score = 0.733, ROC-AUC = 0.811).

It's essential to note that the observed challenges in decision tree performance could be attributed, in part, to the relatively small dataset. Decision trees, being highly flexible, can intricately fit the training data, potentially leading to overfitting in scenarios with limited instances.

F. K-Nearest Neighbors

Initially, an analysis was conducted on the F1 rate for K values ranging from 1 to 30 to determine the most suitable values to be utilized, as illustrated in the graph shown in the figure 1.

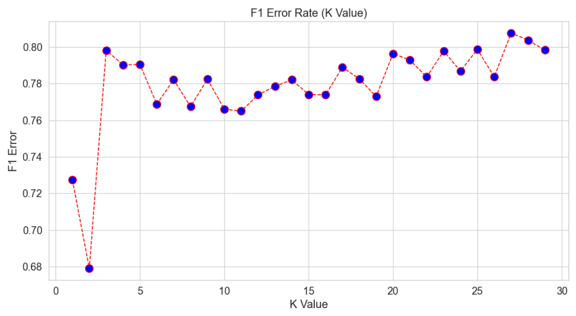

Figure 1. Error rate of K

In order to assess the KNN algorithm, the evaluation metrics were calculated for the local minima of the graph shown in Figure 1, resulting in the following table:

K	Recall	F1	ROC AUC
2	0.738	0.848	0.973
6	0.810	0.859	0.945
8	0.769	0.829	0.938
11	0.775	0.817	0.929

Table IV
PERFORMANCE METRICS FOR DIFFERENT VALUES OF K APPLIED ON DATA TRAIN

K	Recall	F1	ROC AUC
2	0.575	0.679	0.839
6	0.717	0.769	0.888
8	0.708	0.767	0.893
11	0.733	0.765	0.898

Table V
PERFORMANCE METRICS FOR DIFFERENT VALUES OF K APPLIED ON DATA TEST

Based on the performance metrics obtained and illustrated in the tables IV and V, when evaluating the training data, K=6 stands out, demonstrating significantly superior performance, only surpassed by K=2 in the ROC AUC metric. However, K=2 is considered too low and should be excluded early in the analysis due to the high risk of overfitting. Regarding the training set, it is evident that K=11 exhibits the best performance (Recall = 0.733333, F1-Score = 0.764934, ROC-AUC = 0.897639) with a slightly lower but not significantly different F1 score compared to K=8. Furthermore, K=11 is a value that presents a low probability of overfitting occurring.

IV. Conclusion

In conclusion, this work undertook a comprehensive analysis of machine learning models for the prediction of heart disease, employing a dataset from the Cleveland Heart Disease Database. The methodology encompassed rigorous data preprocessing, feature analysis, and evaluation of various machine learning algorithms, including Naive Bayes, Logistic Regression, Decision Trees, and K-Nearest Neighbors.

The data preprocessing phase involved addressing missing values, outliers, and scaling numerical features. Standardization emerged as the preferred scaling technique, showcasing consistent superiority across models. The analysis of features, using techniques like heatmap, Principal Component Analysis (PCA), and univariate distributions, provided valuable insights into their relationships and importance.

Machine learning models were trained and evaluated using performance metrics such as Recall, F1 Score, and ROC AUC. Logistic Regression, after hyperparameter tuning using GridSearchCV, demonstrated the most balanced and promising performance, achieving a Recall of 0.792, F1-Score of 0.826, and ROC AUC of 0.902. Decision Trees, while exhibiting overfitting tendencies, highlighted the importance of careful tuning to balance model complexity and generalization.

References

[1] Statlog (Heart). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C57303.

[2] ascot cardiology group. Diagnostic resting ecg. https://ascotcardiologygroup.co.nz/services/diagnostic-resting-ecg/, 2014. Accessed January 4, 2024.

[3] C3.ai. Precision. https://c3.ai/glossary/machine-learning/precision/, 2014. Accessed January 15, 2024.

[4] C3.ai. Recall. https://c3.ai/glossary/data-science/recall/, 2014. Accessed January 15, 2024.

[5] C3.ai. What is the f1 score? https://c3.ai/glossary/data-science/f1-score/, 2014. Accessed January 15, 2024.

[6] Mayo Clinic. Heart disease. https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118, 2022. Accessed January 2, 2024.

[7] Serviço Nacional de Saúde. Doenças cardiovasculares. https://www.sns.gov.pt/noticias/2017/10/04/doencas-cardiovasculares/, 2017. Accessed January 2, 2024.

[8] Erika Edwards. Heart disease patients with exercise-induced chest pain may not need stents. https://www.nbcnews.com/health/heart-health/heart-disease-patients-exercise-induced-chest-pain-may-not-need-n1082441, 2019. Accessed January 4, 2024.

[9] GeeksforGeeks. Normalization vs standardization. https://www.geeksforgeeks.org/normalization-vs-standardization/, 2021. Accessed January 14, 2024.

[10] GeeksforGeeks. Understanding data attribute types — qualitative and quantitative. https://www.geeksforgeeks.org/understanding-data-attribute-types-qualitative-and-quantitative/, 2022. Accessed January 3, 2024.

[11] GeeksforGeeks. Logistic regression in machine learning. https://www.geeksforgeeks.org/understanding-logistic-regression/, 2023. Accessed January 14, 2024.

[12] BHF Health Intelligence Team. Global Heart Circulatory Diseases Factsheet. *British Heart Foundation*, (June):1–12, 2023.

[13] Jennifer Huizen. What is serum cholesterol? https://www.medicalnewstoday.com/articles/321519, 2021. Accessed January 4, 2024.

[14] IBM. Naïve bayes classifiers. https://www.ibm.com/topics/naive-bayes. Accessed January 14, 2024.

[15] Iguazio. What is model accuracy in machine learning? https://www.iguazio.com/glossary/model-accuracy-in-ml/, 2014. Accessed January 15, 2024.

[16] Zakaria Jaadi. A step-by-step explanation of principal component analysis (pca). https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm. Accessed January 15, 2024.

[17] Jennifer Nelson. Thallium stress test. https://www.healthline.com/health/thallium-stress-test, 2023. Accessed January 4, 2024.

[18] National Institute of Standards and Technology. What are outliers in the data? https://builtin.com/data-science/step-step-explanation-principal-component-analysis. Accessed January 15, 2024.

[19] Adam Rowden. What does st depression on an ecg result mean? https://www.medicalnewstoday.com/articles/st-depression-on-ecg, 2022. Accessed January 4, 2024.

[20] RxList. Definition of fasting blood glucose. https://www.rxlist.com/fasting$_blood_glucose/definition.htm, 2020. AccessedJanu$

[21] Tavish Srivastava. A complete guide to k-nearest neighbors. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/, 2024. Accessed January 14, 2024.

[22] Raghav Vashisht. When to perform a feature scaling? https://www.atoti.io/articles/when-to-perform-a-feature-scaling/, 2021. Accessed January 14, 2024.

[23] Eugenio Zuccarelli. Handling categorical data, the right way. https://towardsdatascience.com/handling-categorical-data-the-right-way-9d1279956fc6, 2020. Accessed January 14, 2024.

V. Appendix

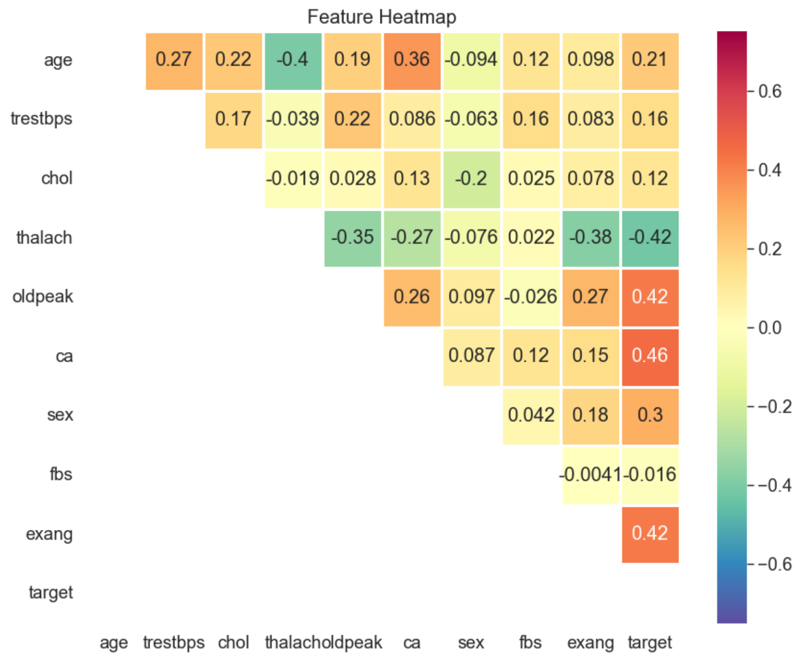

Figure 2. Feature Heatmap

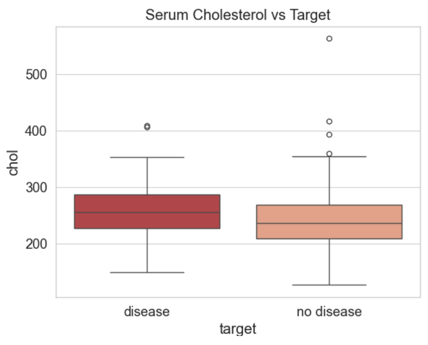

Figure 3. Serum Cholesterol VS Target Box Plot

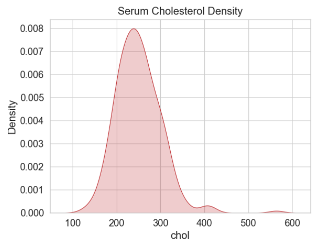

Figure 4. Serum Cholesterol Density Plot

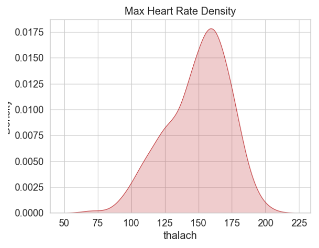

Figure 5. Max Heart Rate Density Plot

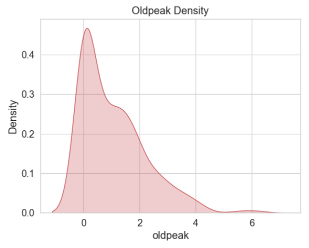

Figure 6. Oldpeak Density Plot

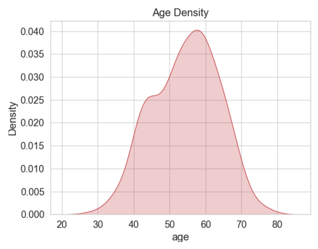

Figure 7. Age Density Plot

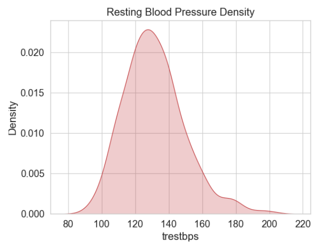

Figure 8. Resting Blood Pressure Density Plot

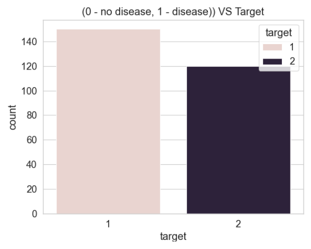

Figure 9. Target Bar Plot

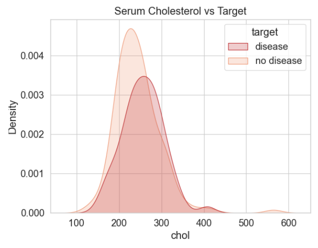

Figure 10. Serum Cholesterol VS Target Density Plot

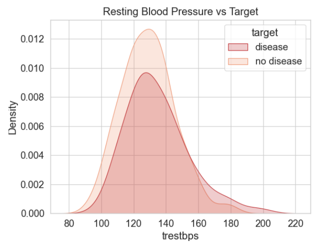

Figure 11. Resting Blood Pressure VS Target Density Plot

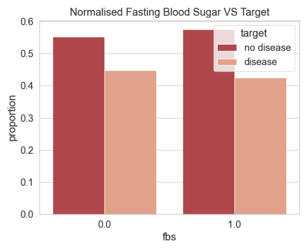

Figure 12. Fasting Blood Sugar VS Target Normalized Bar Plot

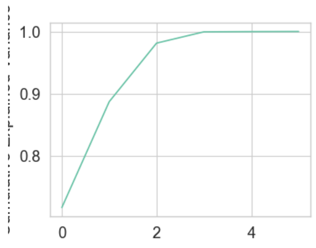

Figure 13. Principal Component VS Cumulative Explained Variance Plot