RESEARCH PROJECT REPORT — GROUP 6

CS534 Artificial Intelligence — Fall 2023

Predictive Analysis of Heart Disease Using AI Algorithms

 *Huibin Cui, Masters in Computer Science,*

 *Sonu Tejwani, Masters in Computer Science,*

 *Ben Tyler, BS in Computer Science & Data Science,*

 *Julie Vieira, BS/MS in Computer Science.*

ABSTRACT:

Heart disease is a condition that afflicts a significant portion of the population in the United States. Nearly twenty percent of deaths recorded each year can be linked to heart disease, with the toll weighing heavy on the family members whose loved one succumbed to the disease. In order to combat the problem, advancements in machine learning have been made which attempt to target the root cause of many medical issues. Multiple models have been proposed in the past, each with their own advantages and disadvantages. While all models were created in good faith, it is important to identify the most successful model in order to properly assess patients who are considered at risk. This paper explores heart disease predictive analysis through the use of five different state of the art methods: Naive Bayes, Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest. Each method is tested using a standard set of accuracy metrics to adequately assess which model is the best predictor. In order to test these models against one another, hyperparameter tuning and data preprocessing were included where necessary and ensured the experimentation was fair. The model which saw the most success from these trials, Random Forest, was then applied to a sample web application which is meant to encourage individuals to recognize the condition of their health and what actions they may take to improve it. Although Random Forest far exceeds the other SOTA methods in nearly every accuracy metric category, particular attention was given to the recall as this metric evaluates how many sick individuals were correctly classified as so. This metric holds far greater importance than others as it is important to know as early as possible the risk of heart disease.

Keywords: Heart disease, Disease classification, Predictive analysis, Hyperparameter tuning, Feature importance, Accuracy, Sklearn

SECTION I : INTRODUCTION

Heart disease is one of the leading causes of human mortality. In 2021, approximately 695,000 people died from heart disease in the United States, with an average of one person out of every five deaths succumbing to heart disease. On average, around 805,000 people experience heart attacks each year. Among them, roughly one-fifth of heart attacks are asymptomatic, causing damage without the patient being aware [2]. However, early intervention is crucial for managing heart disease, making regular heart disease detection indispensable. Detection of heart disease through early-stage symptoms is a great challenge in the current world scenario. Invasive methods of diagnosing the disease are expensive and painful. Therefore, there is a need for a technique that can diagnose heart disease in a non-invasive manner at less cost [1]. In addition, we propose an accessible visualization that can be easily understood throughout the general population. In order to accomplish this goal, a web application may be used to depict the predictions using the most accurate methods.

The proposed method employs the following algorithms to train the data.

A. Naïve Bayes
B. Logistic Regression
C. Support Vector Machine
D. Decision Trees
E. Random Forest

- Naïve Bayes is a probabilistic algorithm harnessed to estimate the likelihood of a patient's affliction with a specific ailment. It operates on the premise that observable symptoms or risk factors are independent of each other, making it computationally efficient and apt for diagnosing conditions characterized by well-defined symptoms or risk factors, such as certain infections. Nevertheless, Naïve Bayes encounters limitations in cases where the existence of substantial feature interdependencies impedes its capacity to accurately capture the intricate and multifaceted relationships inherent to the disease prediction process.
- Logistic Regression is a widely used predictive modeling tool that finds its application in forecasting the probability of an individual succumbing to a particular medical condition. This statistical method gauges the likelihood of disease occurrence by analyzing a patient's data and clinical features. Its interpretability renders it invaluable for risk assessment, particularly in contexts like cardiovascular disease prediction. Nonetheless, its effectiveness may be curtailed when confronting intricate interactions between numerous risk factors, which are characteristic of many multifaceted medical scenarios.
- Support Vector Machines (SVM) offer a robust framework for handling extensive health datasets, especially when intricate relationships exist among patient variables. SVMs excel at demarcating precise decision boundaries between individuals afflicted with diseases and those who are not, thus proving to be indispensable in critical diagnostic tasks like cancer diagnosis. It is worth noting that the computational cost of SVMs escalates when dealing with voluminous datasets, necessitating meticulous parameter tuning for optimal performance.
- Decision trees play a pivotal role in medical diagnosis by rendering straightforward and interpretable decision paths, thus aiding in the identification of key symptoms and risk factors. In the context of disease prediction, they are valuable for early detection models. Nonetheless, decision trees tend to be sensitive to variations in the data, and there is a risk of overfitting when applied to complex medical scenarios.
- Random Forests emerge as a highly effective choice, given their capability to accommodate multiple data features, uphold the significance of clinical variables, and mitigate the risk of overfitting. They find extensive application in diagnosing various diseases, including conditions like diabetes, characterized by intricate interactions among multiple risk factors. However, it is important to acknowledge that Random Forests demand more computational resources, making them best suited for applications where prediction accuracy takes precedence over computational efficiency.

In order to create a series of models that will predict heart disease in patients, the group plans to train and test the above methodologies using the libraries included in PyCharm. Sklearn is one library which includes pre-trained models and evaluations metrics that will be used to compare the methods and identify areas of weakness. In addition, using pre-trained models allows the group to evaluate the methods against themselves by testing different hyperparameter tuning methods and how each method interacts with different types of data, which are further detailed below. To evaluate the performance of the SOTA methods, a baseline for each evaluation metric has been determined. When looking at the accuracy, a score of 80% or higher is considered good for a model. Precision, recall, specificity, and F1-score should strive to have a score of at least 70%. Finally, Matthew's Correlation Coefficient is an indicator of a better model when it gets closer to +1, but values of 0.5 to 1 are generally acceptable.

To complete experimentation of these methods, we plan to run each of these methods on the data set using cross validation for hyperparameter tuning. This will further divide the data set into a predetermined number of folds. These folds will serve as replication in the experiment so that the models may be trained using the best possible attributes. Using these ideal parameters for each method, the models will produce quantifiable values for our evaluation metrics which include the following: accuracy, precision, recall, specificity, F1-score, Matthew's Correlation Coefficient (MCC), and confusion matrix.

In the subsequent sections, this report will review the methods in a general context and how each may be applied to the heart disease prediction problem outlined above. To accomplish this, a dataset with 70,000 entries has been acquired in order to train, test, and evaluate the models. This data set has 13 columns as follows: id, age, gender, height, weight, ao_hi, ap_lo, cholesterol, gluc, smoke, alco, active, cardio. The 'id' of each patient is used for the purpose of indexing and the column 'cardio' represents the actual value of whether or not the patient has heart disease, 0 being that they do not have heart disease and 1 being that they do have heart disease. This dataset was selected due to the fact that the number of entries was large enough that there would be plenty of data to use in both training and testing. For the purposes of this project, a ratio of 80:20 was used for training to testing data. In addition, a wide variety of attributes was included which showcased how different methods may be analyzed in

terms of whether the data was listed in a binary form, categorical form, or numerical range. Feature importance is highlighted for one of the experimental methods, Decision Trees, in order to exemplify and visualize the importance of some attributes compared to others in a general context. This conversation may therefore be expanded upon as certain attributes in all machine learning datasets hold more weight in the final prediction than others. Finally, we will review what has been learned through the implementation and testing of the models thus far and what steps can be taken moving forward.

id	age	gender	height	weight	ap_hi	ap_lo	cholester	gluc	smoke	alco	active	cardio
0	18393	2	168	62	110	80	1	1	0	0	1	0
1	20228	1	156	85	140	90	3	1	0	0	1	1
2	18857	1	165	64	130	70	3	1	0	0	0	1
3	17623	2	169	82	150	100	1	1	0	0	1	1
4	17474	1	156	56	100	60	1	1	0	0	0	0
8	21914	1	151	67	120	80	2	2	0	0	0	0
9	22113	1	157	93	130	80	3	1	0	0	1	0
12	22584	2	178	95	130	90	3	3	0	0	1	1
13	17668	1	158	71	110	70	1	1	0	0	1	0
14	19834	1	164	68	110	60	1	1	0	0	0	0

Table 1: Sample Data

SECTION II: SOTA LITERATURE REVIEW

Naïve Bayes:

Classification is the process of labeling each piece of data. A classification model consists of a training set and a test set. By studying the attributes and labels of the training set, the classifier can predict the labels of the test set based on their attributes [7]. We chose the Naïve Bayes classifier as one of the methods to predict whether patients have heart disease. The Naïve Bayes classifier applies the simple probability and assumptions of Bayes' theorem to predict labels. It is a supervised learning technique and a family of simple probabilistic classifiers [7]. Bayes' theorem is as follows:

$$P\left(C|x_1,\dots x_n\right) = \frac{P(c)P(x_1,\dots x_n|C)}{P(x_1,\dots x_n)} = P(C)P(x_1|C)\dots P(x_n|C) \quad (i)$$

In the Naïve Bayes classifier, we aim to maximize the probability values for each category, known as HMAP:

$$H_{MAP} = argmax(C|x_1,\dots x_n) = argmax(C)P(x_1|C)\dots P(x_n|C) \quad (ii)$$

Using HMAP, we can predict the data's categories. However, when its attributes are continuous, their probabilities can be very small, making it challenging to find the HMAP value. In such cases, Gaussian distribution is used, giving rise to Gaussian Naïve Bayes classifiers. Additionally, when the attributes are binary, the Bernoulli Naïve Bayes classifier is employed[12]. The Naïve Bayes classifier, compared to other classifiers, has strong independence assumptions, can be applied to large datasets, and despite its simplicity, often outperforms many complex classifiers [7] [13]. Therefore, we will attempt to use the Naïve classifier to detect whether patients have heart disease.

Logistic Regression:

The second method selected is logistic regression. Logistic regression is a method similar to linear regression, but is considered to be a better predictor of classifications with binary results. In linear regression, the line of best fit can extend into positive and negative infinity as this method is most often used for numerical outcomes. Logistic regression is a more suitable method for predicting heart disease because the values are restricted to the range of (0, 1) by a Sigmoid function, wherein 0 is representative of false and 1 is representative of true for the problem statement. Fig. 1 is a depiction of how the Sigmoid function restricts the range for the predictive probability.

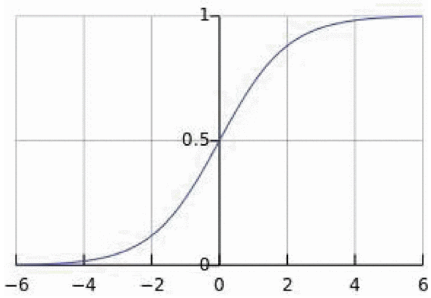

Fig. 1

In this case, the predictions will predict whether or not a patient has a form of heart disease [1]. The formula for logistic regression is as follows:

$$P(Y = 1|x) = \frac{1}{1 - e^{-value}} \ (iii)$$

where P(x) is the probability within the range of (0, 1) that the patient has heart disease, e is the base of natural logarithm, and $value$ takes on the form of $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$. β_0 is a constant and β_p is the j-th regression coefficient, j = 1, 2, …, p, to the independent variable x_i, i = 1, 2, …, n. The sample size is determined by n while the number of features utilized in logistic regression is represented by p [6]. For values that fall between 0 and 1, a threshold is set which determines what binary value can be assigned to them. Any predictions P(x) above the threshold are considered true and those that fall below the threshold are considered false. The value 0.5 is typically used as the threshold, but is determined on a case-by-case basis [1].

Support Vector Machines:

Support Vector Machines are a type of machine learning algorithm that is commonly used in classification tasks. They work by separating data points with a hyperplane that will maximize the margin (distance from closest data point), allowing for the classification of complex datasets with many variables[3]. In the context of heart disease detection, SVMs are valuable as they have the capacity to handle data attributes such as age, blood pressure, cholesterol, etc. while also having a high level of accuracy[8]. However, understanding the rationale behind how the SVM separates the data can be hard to interpret.

Decision Tree:

A decision tree is a machine learning algorithm used as both a regression technique and classification technique. It is a tree-structured classifier. As shown in Fig. 2, it constructs a tree of decision nodes where the internal nodes represent the attributes of data and edges denote the decision rules and each leaf node denotes the class label, i.e., the target class label if that path is chosen. In a decision tree, the prediction of the class will be as follows: The algorithm starts with the root node of the decision tree based on the decision rules; further nodes are selected, and it continues till the leaf node is reached; the class label at leaf node is the class label predicted [11].

Information gain for attribute selection measures:

1. Information Gain =Entropy(S) −[(Weighted Avg) * Entropy (each feature)]
2. Entropy(s) = − P(yes)log(P(yes)) − P(no) log(P(no))

$$Entropy = -\sum_{j=1}^{m} P_{ij} \ log_2 P_{ij}$$

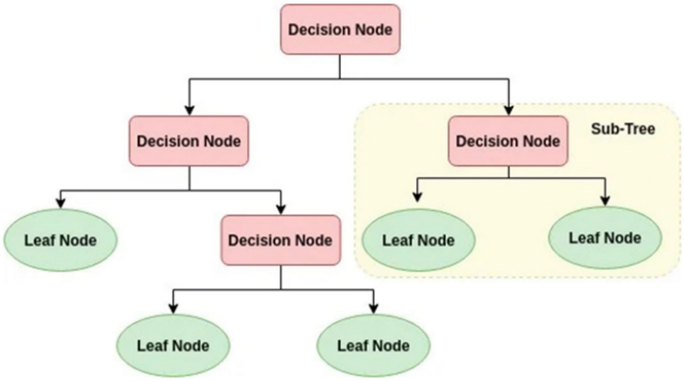

Fig. 2

Random Forest:

Random Forest is based on the concept of ensemble learning, which is a technique of uniting several classifiers to solve a large problem and to improve the performance of the model. As shown in Fig. 3, training data is divided into n subsets, and each subset is trained with a decision tree to design n models, prediction is done based on a voting system, i.e., the majority of classes are considered as the final predicted class [11]. The random forest algorithm addresses an issue posed by many other methods, overfitting of training data. When random forest is combined with the random search algorithm (RSA), studies have shown that this method will produce an accuracy comparable to the conventional approach using grid search algorithm. RSA determines the features that will be used to determine the appropriate hyperparameters at each level. As shown in Fig. 4, RSA will use masking at different levels, increasing from 1 to the number of features n, to determine which features will be utilized in the hyperparameter optimization. Due to the depth for which RSA is used, it still analyzes multiple different combinations of features without having to use computational power to review each combination as with grid search [4].

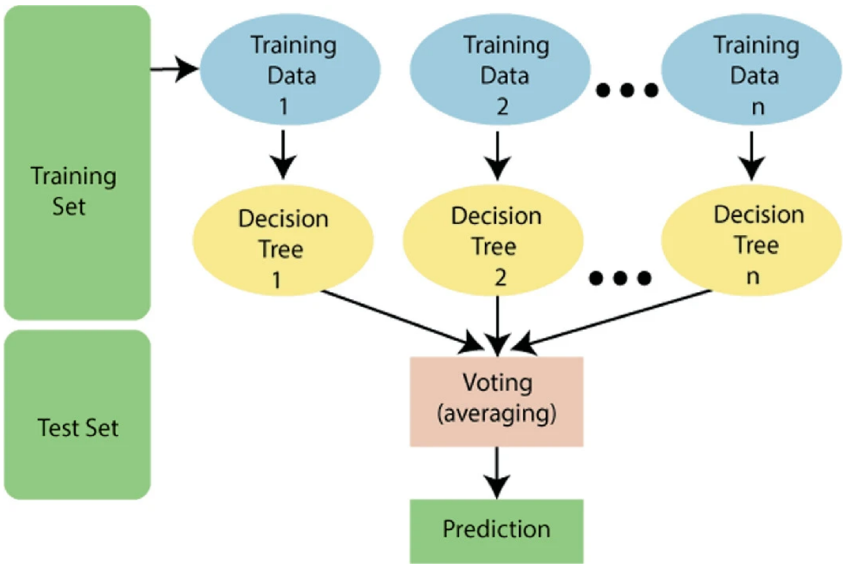

Fig. 3

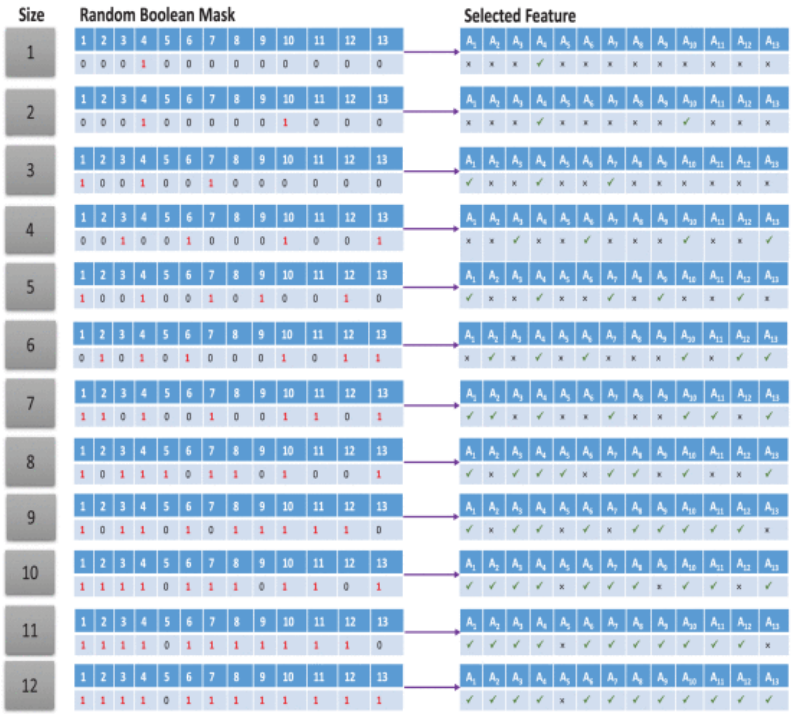

Fig. 4

SECTION III: PROPOSED METHODOLOGY

Data Analysis:

We initiated a comprehensive data analysis to extract valuable insights. Our first step involved scrutinizing the distribution of positive and negative risks across different age groups [Fig. 5]. We note that the highest risk of heart disease is observed among individuals aged between 50 and 64.

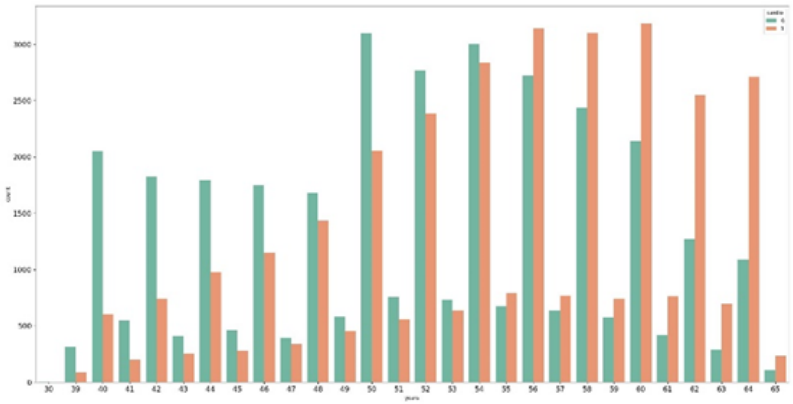

Fig. 5

Additionally, we delved into the gender distribution within the 70,000 records [Fig. 6]. Our findings revealed that more than 65% of the entries in our dataset are associated with female individuals.

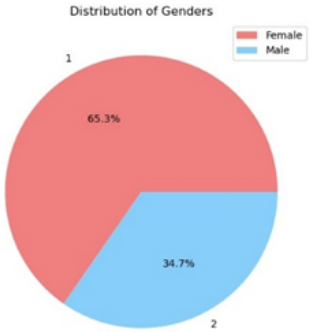

Fig. 6

We conducted a more in-depth analysis of the distribution of positive and negative risks within each sex [Fig. 7]. Furthermore, we investigated alcohol consumption patterns in both cohorts [Fig. 8].

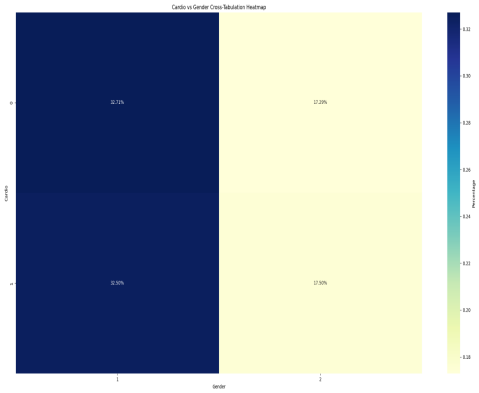

Fig. 7

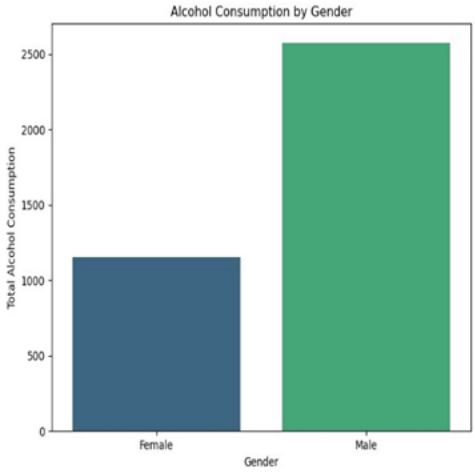

Fig. 8

Continuing with the analysis, a noteworthy observation emerged: within our dataset, the prevalence of smoking habits and alcohol consumption is approximately equal in both the at-risk and not-at-risk categories [Fig 9]. This implies that these factors may have minimal to no discernible impact on the risk status.

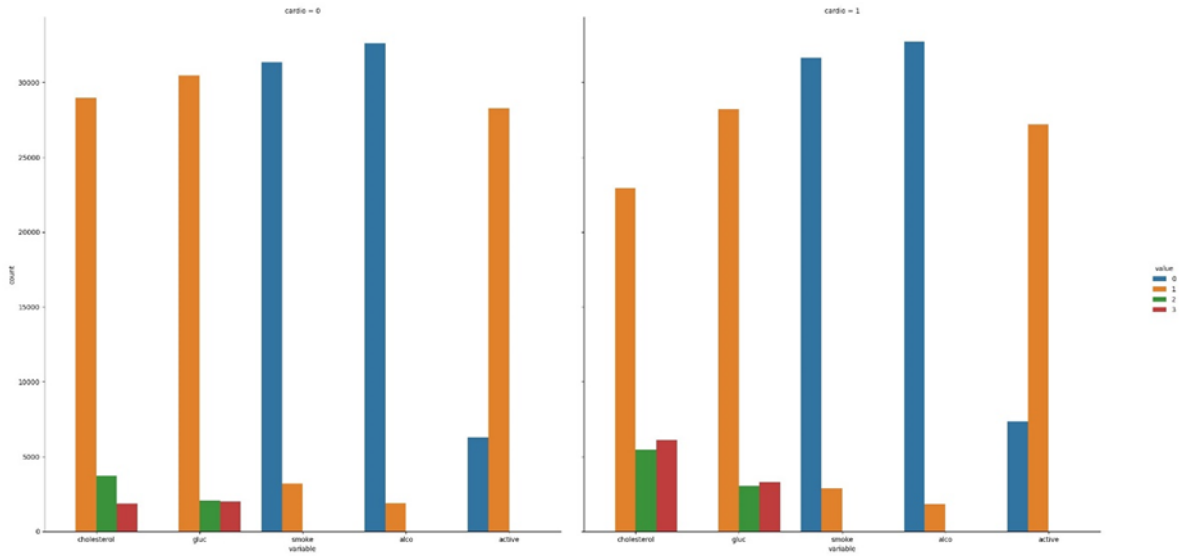

Fig. 9

Data Preprocessing:

We initiated the data preprocessing by eliminating extraneous attributes, such as ID and age in days. Subsequently, we conducted a thorough check for missing values. Addressing data quality, we strategically removed outliers in height and weight, employing a trim approach by eliminating 2.5% of values from both ends of the mean value to ensure a balanced distribution. Further refinement involved identifying entries where the systolic blood pressure exceeded the diastolic blood pressure—an implausible scenario [Fig. 10].

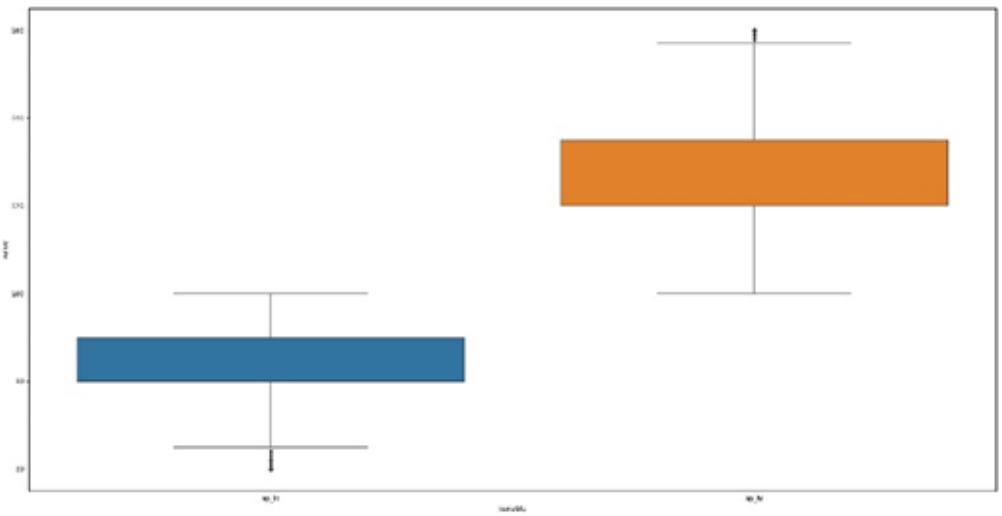

Fig. 10

Consequently, these entries were systematically removed. To address class imbalance, we executed an upsampling technique, bolstering the minority class to align with the size of the majority class [Fig. 11]. Finally, we standardized the data by employing label encoding, converting categorical fields into numerical values for enhanced analytical compatibility.

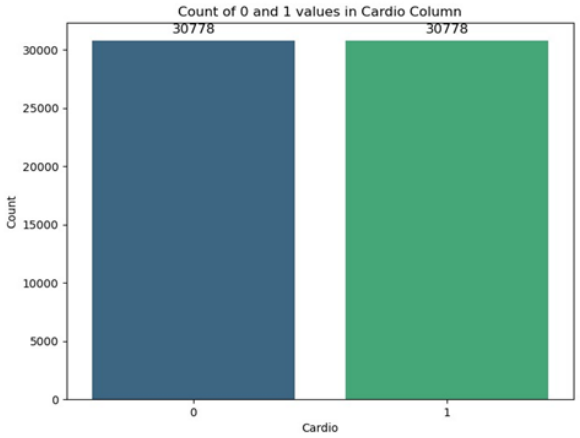

Fig. 11

Fig. 12 shows an overview of the entire process. It has been refined throughout training and testing to ensure all elements are taken into consideration including data cleaning, preprocessing, and balancing. Once the dataset was finalized, the outliers and nonexistent data points were removed. Random undersampling was also used to balance the minority and majority classes. Then, the data was split into training and testing sets, using a ratio of 80:20. The models were trained using this split data and then used to test the remaining samples. Evaluation metrics were produced in order to showcase the effectiveness of each solution.

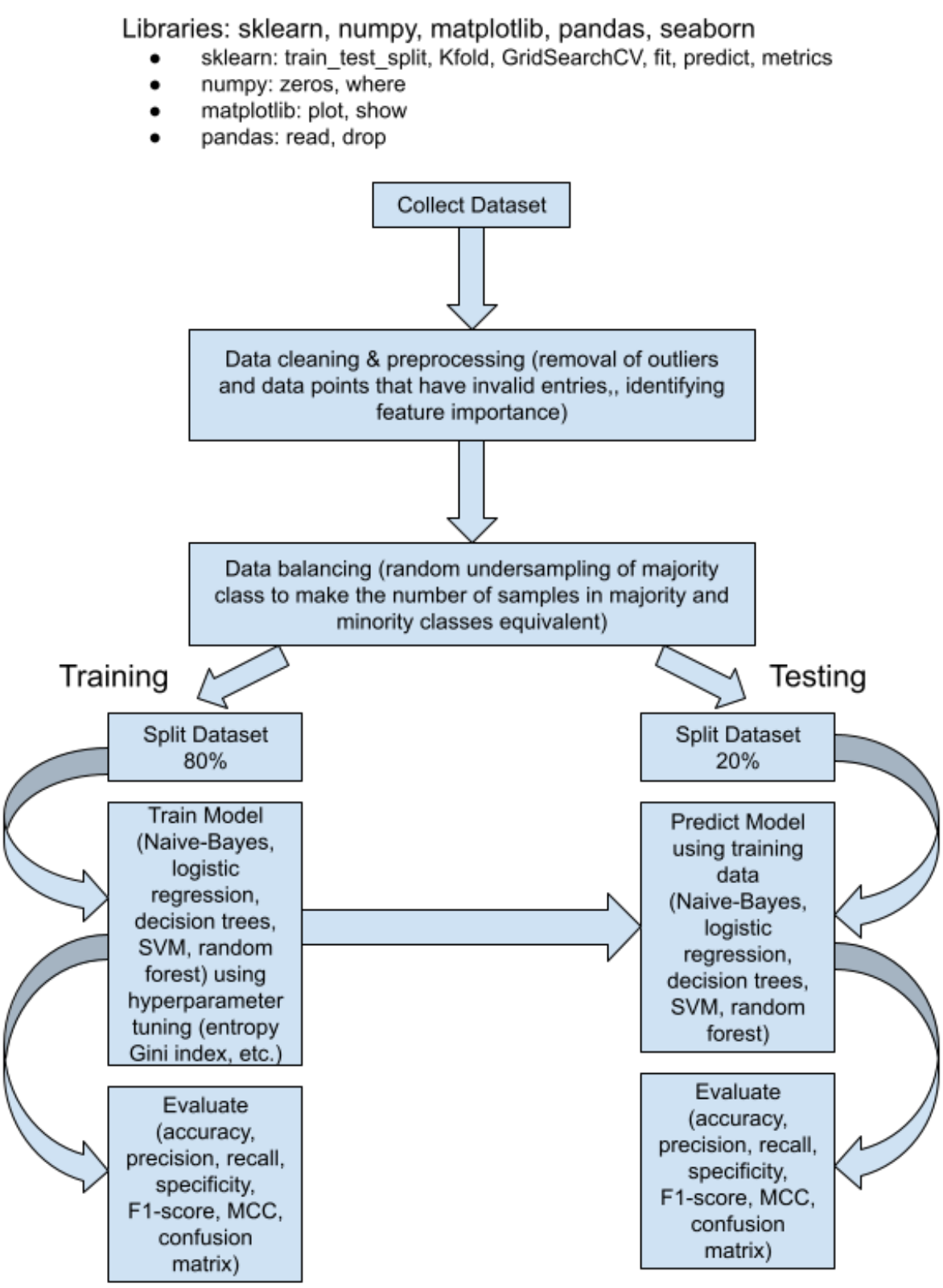
Libraries: sklearn, numpy, matplotlib, pandas, seaborn
sklearn: train_test_split, Kfold, GridSearchCV, fit, predict, metrics
numpy: zeros, where
matplotlib: plot, show
pandas: read, drop
Collect Dataset
Data cleaning & preprocessing (removal of outliers and data points that have invalid entries,, identifying feature importance)
Data balancing (random undersampling of majority class to make the number of samples in majority and minority classes equivalent)
Training
Testing
Split Dataset 80%
Split Dataset 20%
Train Model (Naive-Bayes, logistic regression, decision trees, SVM, random forest) using hyperparameter tuning (entropy Gini index, etc.)
Predict Model using training data (Naive-Bayes, logistic regression, decision trees, SVM, random forest)
Evaluate (accuracy, precision, recall, specificity, F1-score, MCC, confusion matrix)
Evaluate (accuracy, precision, recall, specificity, F1-score, MCC, confusion matrix)

Fig. 12

Hybrid Naive Bayes Classifier:

This dataset has 11 attributes, including continuous, binary, and categorical types. Therefore, I applied Gaussian Naive Bayes for continuous attributes, Bernoulli Naive Bayes for binary ones, and Categorical Naive Bayes for ordinal data. After training these classifiers with the respective attributes, I used them to predict on the test dataset. The results were combined to yield a (3, 2) 2D array. The label corresponding to the maximum values in this array was used as the prediction.

All these Naive Bayes classifiers are based on Bayes' Theorem:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

$P(y|x)$ is the posterior probability which is the probability of class y given predictor x.
$P(x|y)$ is the likelihood which is the probability of predictor x given class y.
$P(y)$ is the prior probability, which is the probability of class y without any information about x.
$P(x)$ is the marginal probability.
However, when predicting, $P(x)$ isn't directly computed, as it's constant across all classes, meaning it doesn't affect which class has the maximum posterior probability $P(y|x)$. So, when determining the class label with the highest $P(y|x)$, we can safely ignore the $P(x)$ term and focus on maximizing the numerator $P(x|y)P(y)$.
And we prefer to use $log(P(y|x)) + log(P(y))$ instead of $P(x|y)P(y)$. Because: 1) Multiplying many small probabilities in high-dimensional data can cause underflow issues, leading to inaccuracies or computational errors. 2) Addition is faster and more efficient than multiplication, especially with large datasets.
I also introduced smoothing in Bernoulli and Categorical Naive Bayes to prevent zero probabilities. This ensures that if a feature-label combination isn't in the training set, its probability doesn't default to zero, which could lead to inaccurate predictions.
And I implemented these classifiers using the numpy library from scratch. Next, I'll detail the Gaussian, Bernoulli, and Categorical Naïve Bayes classifiers.

Gaussian Naïve Bayes classifier:
1) Means and Variance Calculation (Training Phase)
For each class y and each feature x_i:

$$\mu_{yi} = \frac{1}{N_y} \sum_{x \in y} x_i$$

Where μ_{yi} is the mean of feature x_i for class y, and N_y is the number of instances of class y.

$$\sigma^2_{yi} = \frac{1}{N_y} \sum_{x \in y} (x_i - \mu_{yi})^2$$

Where σ^2_{yi} is the variance of feature x_i for class y.
2) Prior Probabilities (Training Phase)
For each class y:

$$P(y) = \frac{Number\ of\ instances\ of\ class\ y}{Total\ number\ of\ instances}$$

3) Log-Likelihood Calculation (Prediction Phase)
For Gaussian distribution, the natural log of probability density function is given by

$$ln\ P(x_i|y) = -\frac{1}{2} ln(2\pi\sigma^2_{yi}) - \frac{(x_i - \mu_{yi})^2}{2\sigma^2_{yi}}$$

4) Class Prediction (Prediction Phase)
For each instance and each class, calculate:

$$score(y) = ln\ P(y) + \sum_i ln\ P(x_i|y)$$

Predict the class with the highest score.

Bernoulli Naïve Bayes classifier:
1) Feature Presence Calculation (Training Phase)
For each class and feature x_i:

$$presence(y, x_i) = Number\ of\ instances\ where\ feature\ x_i\ is\ present\ and\ belongs\ to\ class\ y$$

2) Prior Probabilities (Training Phase)
For each class y:

$$P(y) = \frac{Number\ of\ instances\ of\ class\ y}{Total\ number\ of\ instances}$$

3) Feature Probability Calculation (Training Phase)
The probability of feature x_i being present in class y (with smoothing parameter α) is:

$$P(x_i = 1|y) = \frac{presence(y, x_i)+\alpha}{Number\ of\ instances\ of\ class\ y + 2\alpha}$$

And the probability of the feature being absent:

$$P(x_i = 0|y) = 1 - P(x_i = 1|y)$$

4) Log-Likelihood Calculation (Prediction Phase)
The log likelihood of an instance, given class y, is:

$$ln\ P(x_i|y) = x_i ln\ P(x_i = 1|y) + (1 - x_i) ln\ P(x_i = 0|y)$$

5) Class Prediction (Prediction Phase)
For each instance and each class, calculate:

$$score(y) = ln\ P(y) + \sum ln\ P(x_i|y)$$

Predict the class with the highest score.

Categorical Naïve Bayes classifier:
1) Feature Count Calculation (Training Phase)
For each class and category c_i of feature x_i:

$$count_{y,c_i} = \sum_{samples} \delta(x_i = c_i)$$

Where $count_{y, c_i}$ is the number of times category c_i of feature x_i appears in samples that belong to class y. δ is the indicator function.
2) Prior Probabilities (Training Phase)
For each class y:

$$P(y) = \frac{Number\ of\ instances\ of\ class\ y}{Total\ number\ of\ instances}$$

3) Feature Probability Calculation (Training Phase)
The probability of each category c_i of feature x_i given class y (with smoothing parameter α) is:

$$P(x_i = c_i|y) = \frac{count_{y,c_i}+\alpha}{\sum_{c_i}(count_{y,c_i}+\alpha)}$$

4) Log-Likelihood Calculation (Prediction Phase)
The log likelihood for a sample, given class y, is obtained by summing the log probabilities of its features' categories:

$$ln\ P(x_i|y) = \sum_i ln\left(\frac{count_{y,c_i}+\alpha}{\sum_{c_i}(count_{y,c_i}+\alpha)}\right)$$

5) Class Prediction (Prediction Phase)
For each instance and each class, calculate:

$$score(y) = ln\ P(y) + \sum ln\ P(x_i|y)$$

Predict the class with the highest score.

In response to insightful feedback received regarding our approach in implementing the Naïve Bayes Classifier, we have delved deeper into the architecture of our model. The understanding of the difference between "Hybrid" and "Ensemble" classifiers is an important distinction. “Hybrid” suggests a combination of methodologies while “Ensemble” suggests a result is obtained from the predictions made by running an algorithm multiple times on different subsets of a dataset. The proposed approach outlined above is indeed more accurately described as an "Ensemble Naïve Bayes Classifier." Precisely, this terminology better captures the parallel processing architecture of our model.

In our project, we effectively integrate Gaussian, Bernoulli, and Categorical Naïve Bayes classifiers to handle different types of attributes—continuous, binary, and categorical, respectively. The final prediction for each data instance is derived by combining the probabilistic outputs of these individual classifiers. Specifically, we assign different weights to the predictions from each classifier based on their relevance and performance characteristics. This weighting scheme allows us to leverage the strengths of each classifier type while compensating for their individual limitations. For instance, we tune each classifier separately using GridSearchCV, ensuring optimal performance for their respective attribute types. Then, we combine their probabilistic predictions. The weights (0.2 for GaussianNB, 0.3 for CategoricalNB, and 0.5 for BernoulliNB in our case) are empirically determined, but they can be adjusted according to the specific characteristics and requirements of the dataset. The final prediction is made by applying a threshold to the weighted sum of these probabilities. This ensemble approach not only enhances the model's overall accuracy but also ensures a more robust performance across varied data instances.

Logistic Regression:

Libraries and Modules Used:

- Data Manipulation: We utilized the ‘*pandas’* library to efficiently manage and preprocess the dataset.
- Data Visualization: The ‘*matplotlib*’ library enables us to create informative visualizations that aid in performance evaluation.
- Machine Learning: ‘*sklearn*’ is our primary library for machine learning tasks. It offers tools for model selection, hyperparameter tuning, and comprehensive model evaluation.

To complete the logistic regression model, the first step was to analyze the data being passed into the function we created. The parameters for the group’s logistic regression method were ‘X’ and ‘y’, which are representative of the features and attributed data points and the ground truth labels respectively. The binary values of the ground truth labels state whether the patient has heart disease, 1, or not, 0. Then, the data is split into training and testing sets using a ratio of 80:20 and 5 fold cross validation for hyperparameter tuning. Using the default penalty ‘l2’, the hyperparameters tested were ‘C’ and ‘solver’. ‘C’ represents the inverse of regularization strength, meaning that smaller values equate to stronger regularization and a lesser chance of overfitting. The values for ‘C’ for which we applied grid search on were [0.001, 0.01, 0.1, 1, 10, 100]. The ‘solver’ attribute determines optimization by performing different minimization techniques on the coordinates supplied by the dataset. Three solvers were used during testing, which were ['liblinear', 'newton-cg', 'newton-cholesky']. Using ‘GridSearchCV’ to complete an exhaustive search of parameters, the best parameters were determined to be {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}.

Once the ‘X’ and ‘y’ data sets are collected for training and testing, the logistic regression model is created using the ‘sklearn’ library ‘LogisticRegression’ class. This ‘fit’ function takes in both X_train and y_train. Using this newly created model, the ‘predict’ function in the same library is used to acquire the predictions for the X_test data set. Each prediction is a value between 0 and 1, with values closer to 1 indicating it is more likely a patient has heart disease and values closer to 0 indicating it is less likely that a patient has heart disease. These values are converted to binary values of 0 and 1, using a threshold of 0.5 to determine their classifications, in order to properly assess the evaluation metrics. Finally, the following metrics and visual representations are calculated or displayed: confusion matrix, accuracy, precision, recall, specificity, F1-score, and Matthew’s Correlation Coefficient, which are defined in the following section.

Support Vector Machine:

Libraries and Modules Used:

- ➢ Data Manipulation: We utilized the '*pandas'* library to efficiently manage and preprocess the dataset.
- ➢ Data Visualization: The '*matplotlib*' library enables us to create informative visualizations that aid in performance evaluation.
- ➢ Machine Learning: '*sklearn*' is our primary library for machine learning tasks. It offers tools for model selection, hyperparameter tuning, and comprehensive model evaluation.

SVM was completed using the 'sklearn' library's built-in 'SVC' function. The 'linear' kernel was applied to the model, which is used in the case where the data points are linearly separable. Then, the 'fit' method was used to train the model using the split dataset, which was categorized into the 'X_train' features and the 'y_train' ground truth labels. Finally, the 'predict' function was applied to the 'X_test' data in order to produce predictions. These predictions were compared against the corresponding ground truth labels 'y_test' in order to calculate the accuracy metrics outlined above.

Hyperparameter tuning using cross validation was not prioritized for SVM. It was determined to be too computationally expensive compared to the other SOTA methods. Although this is recognized to be an important step in the machine learning process, the devices from which the method was run face limitations and ultimately saw that the model's predictions did not converge when hyperparameter tuning was implemented. Despite this oversight, SVM still produced results that are considered up to the standards of the other evaluated methods as is detailed in the following section.

Decision Tree Classifier:

Libraries and Modules Used:

- ➢ Data Manipulation: We utilized the '*pandas'* library to efficiently manage and preprocess the dataset, while numpy supports essential numerical operations.
- ➢ Data Visualization: The '*seaborn*' and '*matplotlib*' libraries enable us to create informative visualizations that aid in performance evaluation.
- ➢ Machine Learning: '*sklearn*' is our primary library for machine learning tasks. It offers tools for model selection, hyperparameter tuning through grid search, cross-validation via the '*KFold*' module, and comprehensive model evaluation.

We start with the Decision Tree Classifier. After initializing the model, we define a search space for hyperparameters, such as the criterion for splitting nodes and the maximum depth of the tree. To identify the best hyperparameters, we use cross-validation with a predefined number of folds, ensuring reliable and unbiased model selection. Following hyperparameter tuning, we proceed to train the Decision Tree model. Its performance is evaluated using a comprehensive set of classification metrics, including accuracy, precision, recall, F1-score, specificity, and the Matthews Correlation Coefficient (MCC). Visualizations, such as a confusion matrix and feature importance plots, aid in understanding the model's behavior and the significance of individual features.

1) Entropy:
   a) Entropy is a measure of impurity or disorder in a set of data. In the context of a decision tree, it's used as a criterion to determine the best attribute for splitting a node.
   b) The entropy of a node is given by:

$$H(S) = -p_1 log_2 p_1 - p_2 log_2 p_2 - \ldots\ldots\ldots - p_c log_2 p_c$$

   Where S is is the set of samples, c is the number of classes, and p_i is the proportion of samples belonging to class i.

2) Gini Impurity:
   a) Gini Impurity measures the probability of misclassifying a randomly chosen element if it was labeled according to the class distribution in the node.

b) Gini Impurity is calculated as:

$$Gini(S) = 1 - \sum (p_i)^2$$

Where $Gini(S)$ is the Gini Impurity, and p_i is the proportion of samples belonging to class i.

- Data Preparation and Splitting:
  - Load the dataset (e.g., "cardio_train.csv").
  - Split the dataset into features (X) and the target variable (y) for cardiovascular disease prediction. Further split the data into training and validation sets using the train_test_split function.
- Decision Tree Model:
  - Create an instance of the DecisionTreeClassifier, initializing it with a random seed for reproducibility.
  - Define hyperparameters for the Decision Tree model, including the criterion (Gini impurity or Entropy) and maximum depth.
  - Set up k-fold cross-validation using KFold with the specified number of folds (e.g., 10) to assess model performance.
- Model Tuning:
  - Use GridSearchCV to perform hyperparameter tuning by searching for the best combination of criterion and maximum depth. This process is carried out over the training data using k-fold cross-validation.
- Model Training and Evaluation:
  - Train the Decision Tree model on the training data and assess its performance on both the training and validation sets.
  - Calculate evaluation metrics, including accuracy, precision, recall, F1-score, specificity, and Matthews Correlation Coefficient (MCC).
    Generate a classification report for both training and validation sets to obtain insights into the model's performance.
  - Calculate the confusion matrix to visualize the true positives, true negatives, false positives, and false negatives.
- Feature Importance:
  - Determine the feature importance of the Decision Tree model to identify the most relevant features for classification.
  - Visualize important features using a bar plot.

Random Forest:
Random Forest is an ensemble learning method that combines multiple Decision Trees to improve predictive performance and reduce overfitting. Each Decision Tree is trained on a bootstrapped sample of the data, and feature selection is also randomized. Random Forest aggregates the predictions of individual trees to make a final prediction.
The mathematical expressions for Random Forest are more complex due to the ensemble nature of the model. The primary idea is that each tree's output is considered, and the final prediction is made based on majority voting or averaging, depending on the problem (classification or regression). The individual trees are weighted equally.
While there isn't a single formula for Random Forest, it can be thought of as an ensemble of Decision Trees, and the concept of Information Gain is still relevant when building individual trees.
In summary, Decision Trees are based on entropy and information gain to partition data, while Random Forest is an ensemble of Decision Trees. The mathematical representations of Random Forest are not as straightforward as Decision Trees but involve the combination of multiple trees to make predictions.

- Data Preparation and Splitting:
  - For the Random Forest model, you'll again use the dataset split into features (X) and the target variable (y) for cardiovascular disease prediction.
  - Split the data into training and validation sets.
- Random Forest Model:

    - Create an instance of the RandomForestClassifier with parallel processing (n_jobs=-1) and set a random seed for reproducibility.
- Hyperparameter Tuning:
    - Define a set of hyperparameters for the Random Forest model, such as the number of trees (n_estimators) and maximum depth.
    - Employ GridSearchCV for hyperparameter tuning over the training data with k-fold cross-validation.
- Model Training and Evaluation:
    - Train the Random Forest model using both the default settings and the best hyperparameters found during tuning.
    - Evaluate the model's performance, calculate evaluation metrics (accuracy, precision, recall, F1-score, specificity, MCC), and generate a classification report for both training and validation sets.
    - Visualize the confusion matrix.

SECTION IV: EXPERIMENTAL RESULTS AND DISCUSSIONS

Performance evaluation of the proposed work is done based on the following measures:

Confusion Matrix is a matrix that is used to evaluate the performance of a model. The four terms associated with the confusion matrix which is used to determine the performance matrices are:

- True Positive (TP): An outcome when the positive class is correctly predicted by the model.
- True Negative (TN): An outcome when the negative class is correctly predicted by the model.
- False Positive (FP): An outcome when the positive class is incorrectly predicted by the model.
- False Negative (FN): An outcome when the negative class is incorrectly predicted by the model.

1. Accuracy: the ratio of the number of correct predictions given by the model to the total number of instances [12].

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)}$$

2. Precision: measures the proportion of individuals predicted to be at risk of developing heart disease and had a risk of developing heart disease[12].

$$Precision = \frac{TP}{(TP+FP)}$$

3. Recall/Sensitivity: measures the proportion of individuals who were at risk of developing heart disease and were predicted by the algorithm to be at risk of developing heart disease [4].

$$Recall = \frac{TP}{(TP+FN)}$$

4. Specificity: measures the proportion of individuals who were not at risk of developing heart disease and were predicted by the algorithm to not be at risk of developing heart disease [4].

$$Specificity = \frac{TN}{(TN+FP)}$$

5. F1 Score: the harmonic mean of precision and recall [12].

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

6. Matthew's Correlation Coefficient: statistical analysis of binary classification [4].

$$MCC = \frac{(TP * TN - FP * FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

To execute the experiment using the above performance measures, we have used 10-fold, 5-fold, and 3-fold cross-validation which will divide the samples into n sections and apply the algorithm to n sections for training using a different combination of hyperparameters in each step. Once the optimal set of hyperparameters is determined, they will be applied to the testing set. This will provide uniformity in comparing the results for each experiment and test whether or not the ratio of training to testing data has an influence on the results. In addition, the data set which we plan to use will have a variety of features that can be manipulated. Examples of features included in past data sets for heart disease are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, and the number of major vessels colored by fluoroscopy [4]. Experimentation may also include training using the features that cannot be changed, such as age and family history, compared to those that can be changed, like lifestyle.

Classifier	Accuracy	Precision	Recall	Specificity	F1 Score	MCC
Naïve Bayes	70.87%	75.39%	61.17%	80.39%	67.54%	0.42
Logistic Regression	71.43%	75.41%	64.24%	78.73%	69.38%	0.44
SVM	70.56%	78.41%	55.80%	84.99%	65.20%	0.43
Decision Tree	77.16%	77.49%	77.20%	82.54%	77.11%	0.55
Random Forest	82.22%	82.34%	82.24%	85.26%	82.21%	0.65

Table 2: Results from testing for each evaluation method and model

Hybrid Naïve Bayes Classifier:
Accuracy: 70.87% - The model predicts correctly in 70.87% of cases; considered decent.
Precision: 75.39% - When predicting positive, the model is correct 75.39% of the time; generally good with few false positives.
Recall: 61.17% - The model correctly identifies 61.17% of actual positive cases; higher recall is preferable for medical diagnoses.
Specificity: 80.39% - The model accurately identifies 80.39% of actual negative cases; effective at spotting true negatives.
F1 score: 67.54% - Represents the balance between precision and recall.
MCC: 0.42 - Ranges from -1 to 1, with 1 being perfect. Higher values are better, indicating a positive correlation between predictions and actuals.

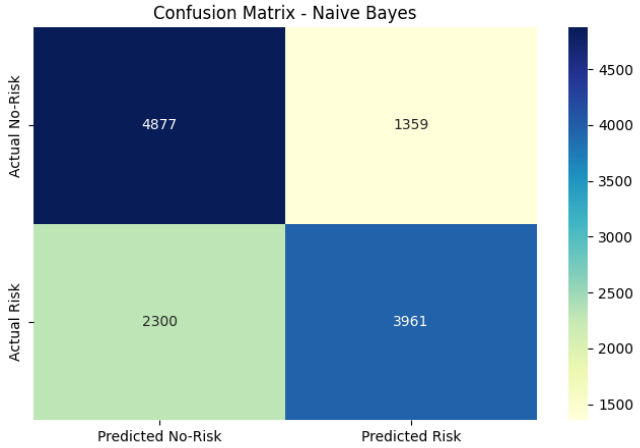

Fig. 13

Logistic Regression:
Accuracy: 71.43% - The model predicts correctly in 71.43% of cases; considered decent.
Precision: 75.41% - When predicting positive, the model is correct 75.41% of the time; generally good with few false positives.
Recall: 64.24% - The model correctly identifies 64.24% of actual positive cases; higher recall is preferable for medical diagnoses.
Specificity: 78.73% - The model accurately identifies 78.73% of actual negative cases; generally good with true negatives.
F1 score: 69.38% - Represents the balance between precision and recall.
MCC: 0.44 - Ranges from -1 to 1, with 1 being perfect. Higher values are better, indicating a positive correlation between predictions and actuals.

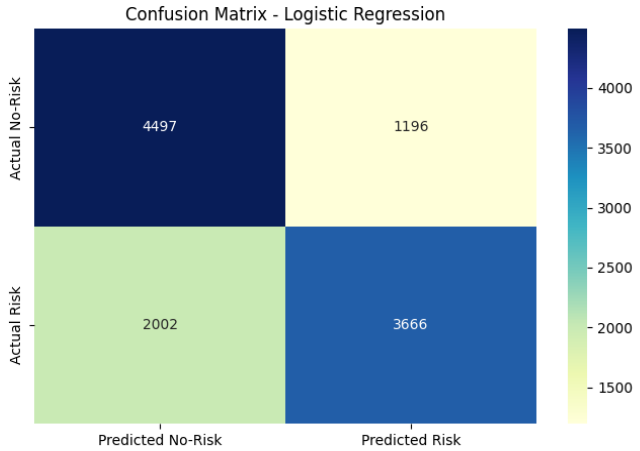

Fig. 14

Support Vector Machines:
Accuracy: 70.56% - The model predicts correctly in 70.56% of cases; considered decent.
Precision: 78.41% - When predicting positive, the model is correct 78.41% of the time; generally good with few false positives.

Recall: 55.80% - The model correctly identifies 55.80% of actual positive cases; higher recall is preferable for medical diagnoses.
Specificity: 84.99% - The model accurately identifies 84.99% of actual negative cases; effective at spotting true negatives.
F1 score: 65.20% - Represents the balance between precision and recall.
MCC: 0.43 - Ranges from -1 to 1, with 1 being perfect. Higher values are better, indicating a positive correlation between predictions and actuals.

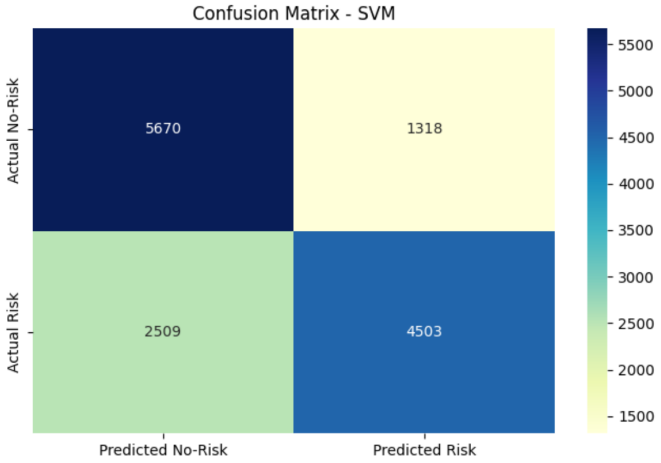

Fig. 15

Decision Tree:
Accuracy: 77.16% - The model predicts correctly in 77.16% of cases; relatively low.
Precision: 77.49% - When predicting positive, the model is correct 77.49% of the time; generally good with few false positives.
Recall: 77.20% - The model correctly identifies 77.20% of actual positive cases; higher recall is preferable for medical diagnoses.
Specificity: 82.54% - The model accurately identifies 82.54% of actual negative cases; generally good with true negatives.
F1 score: 77.11% - Represents the balance between precision and recall.
MCC: 0.55 - Ranges from -1 to 1, with 1 being perfect. Higher values are better, indicating a positive correlation between predictions and actuals.
The feature importance graph of the Decision Tree [Fig. 17] aligns with our observation from the data analysis, indicating that smoking and alcohol are among the least significant features in influencing the outcome.

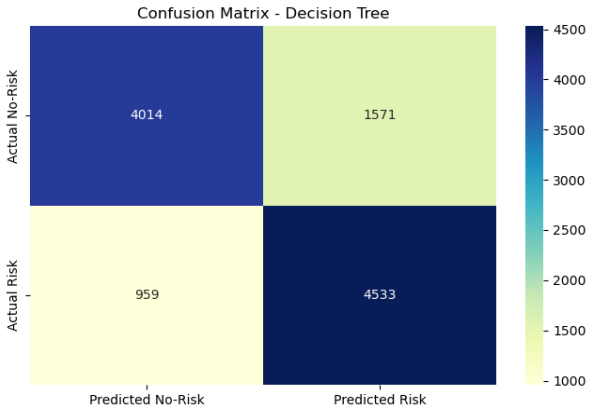

Fig. 16

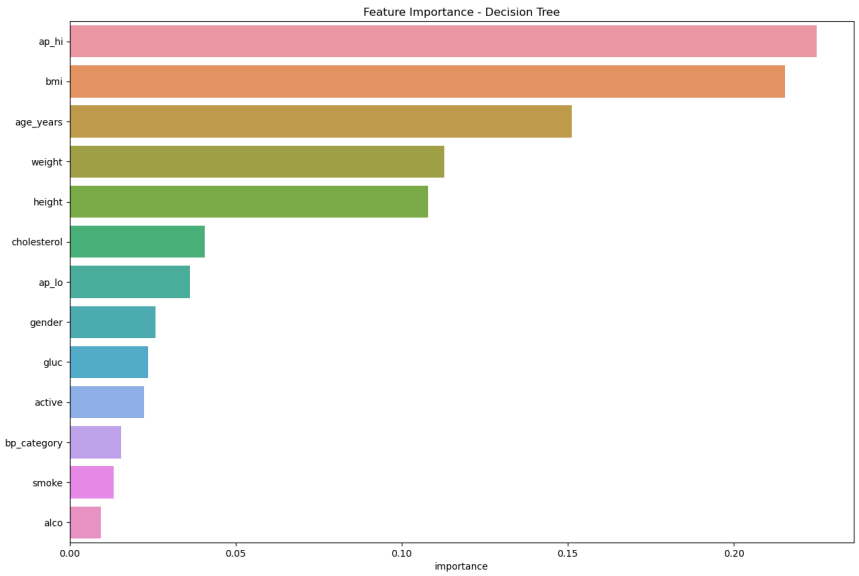

Fig. 17

Random Forest:
Accuracy: 82.22% - The model predicts correctly in 82.22% of cases; considered decent.
Precision: 82.34% - When predicting positive, the model is correct 82.34% of the time; generally good with few false positives.
Recall: 82.24% - The model correctly identifies 82.24% of actual positive cases; higher recall is preferable for medical diagnoses.
Specificity: 85.26% - The model accurately identifies 85.26% of actual negative cases; generally good with true negatives.
F1 score: 82.21% - Represents the balance between precision and recall.
MCC: 0.65 - Ranges from -1 to 1, with 1 being perfect. Higher values are better, indicating a positive correlation between predictions and actuals.

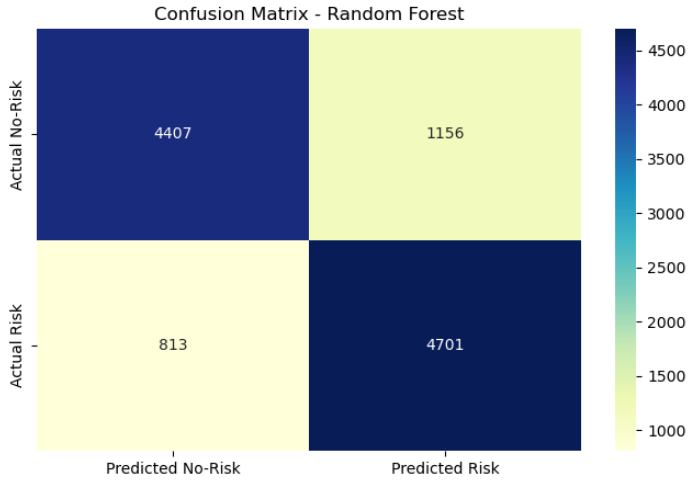

Fig. 18

SECTION V: LESSONS LEARNED

Ben:
This project has been invaluable in understanding the processes that go into predictive analysis AI. I focused primarily on Support Vector Machines which focuses on making a hyperplane to separate data points on a graph. Coding this program taught me the sklearn library. I also have learned how to tweak an algorithm to give you more accurate results. Going into this project I had no idea how one would go about making a project like this and it is amazing how many methods are possible within just this project and how heavily it would be optimized.

Huibin:
In the course of this project, I embarked on the challenging yet rewarding task of developing a Naïve Bayes classifier that integrates Gaussian, Bernoulli, and Categorical models. This intricate process not only broadened my comprehension of each classifier's unique characteristics but also deepened my appreciation of the subtle deviations and complexities involved in applying Bayes' Theorem in real-world scenarios. I paid particular attention to maintaining a delicate balance between precision and recall, a critical aspect in fields requiring high accuracy, such as medical diagnostics. Although I possessed a basic understanding of Naïve Bayes principles, the practical application of this project honed my skills further, particularly in the realm of advanced mathematical concepts and the utilization of sophisticated tools like numpy. I tackled challenging aspects such as mitigating data underflow and implementing smoothing techniques, which were pivotal in ensuring the robustness and reliability of our predictions. In essence, this project significantly enhanced my technical acumen and provided valuable insights into the complexities of classifier design and optimization.

Julie:
This project has been pivotal in solidifying my understanding of the methods used in heart disease detection listed above. I focused primarily on logistic regression and SVM, but have also learned quite a bit from conversations with my team members about the other methods. The main learning objective I will be able to take away from working on logistic regression and SVM is how the different accuracy measures work. This has given me a better perspective on how to properly evaluate methods and determine where improvements can be made. Before this project, I had never heard of the methods we are using so learning about how each one works and viewing the diagrams of each process was very beneficial. This project has also taught me what tools are available in Python libraries, such as sklearn, that help to evaluate and understand how the models work. Finally, the importance and confusion matrix graphs are important in understanding the effectiveness of each model as well as which attributes contribute the most to machine learning.

Sonu:
Working on this project has been a valuable learning experience. I have gained several insights and skills throughout the process. Firstly, I've deepened my understanding of hyperparameter tuning and cross-validation. By exploring various hyperparameters and using techniques like grid search and K-fold cross-validation, I've learned the importance of optimizing machine learning models for improved performance. In this project, I've also leveraged powerful libraries like sklearn and data visualization tools like Matplotlib and Seaborn, which have expanded my toolkit for working with machine learning and data analysis.
The experience with the RandomForestClassifier and hyperparameter tuning through GridSearchCV underscored the significance of optimizing model parameters for disease prediction. By systematically exploring various combinations of n_estimators and max_depth, we learned that fine-tuning these parameters can substantially enhance the model's performance. The 10-fold cross-validation and the ability to parallelize computations with n_jobs greatly expedited the optimization process. Additionally, the comparison between the best parameters and default settings demonstrated the substantial impact of parameter tuning on predictive accuracy.

SECTION VI: CONCLUSION AND FUTURE WORK

This project embarked on an in-depth analysis of various machine learning methodologies, specifically delving into Naive Bayes, Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest algorithms, to address the challenge of heart disease prediction. Each model was tested and evaluated using a set of rigorous accuracy metrics to discern the most effective approach. It was observed that the Random Forest algorithm, with its intricate ensemble of decision trees and capability to handle complex datasets, emerged as the superior model. The distinct advantage of the Random Forest model lies in its sophisticated mechanism for processing multifaceted data relationships and its intrinsic defense against overfitting, which is particularly beneficial in a domain as complex as medical diagnostics. The high degree of accuracy achieved by the Random Forest model underscores its significance and potential as a reliable tool for the prediction of heart disease, positioning it as a pivotal element in the arsenal of predictive analytics.

Future efforts will be directed towards enhancing the models' performance through advanced feature engineering and more sophisticated preprocessing techniques. This advancement will be pursued through the implementation of advanced feature engineering—crafting and selecting predictive variables that are more indicative of heart disease—and the application of more nuanced preprocessing techniques to enhance the quality of the data fed into the models. In the realm of model architecture, we will investigate the integration of hybrid and ensemble methods, aiming to harness the collective strengths of various algorithms to build a composite model of even greater predictive power. In addition to these technical enhancements, we plan to create a platform between our research and its practical deployment. This application will function as a platform for end-users, allowing them to enter specific health-related parameters and promptly obtain an evaluation of their potential risk for heart disease. The objective is to elevate public consciousness about heart disease risks and to catalyze a shift towards proactive health management among individuals.

Timeline Schedule:

This is the schedule we created according to our timeline in Phase 1 and Phase 2. All tasks below were completed over the course of the project. One improvement we did not have time to execute was finding more efficient ways to execute the website visualization.

Week 1	Find appropriate data set with enough entries to undergo preprocessing and cleaning and have enough entries remaining to effectively conduct testing
Week 2	Construct the methods for each SOTA algorithm: Naive Bayes, Logistic Regression, Support Vector Machine, Decision Trees, Random Forest
Week 3	Analyze the attributes for which have the greatest impact on our experimental results

Week 4	Combine two datasets to create more features to work with and augment the data
Week 5	Balance and reprocess the dataset by evaluating any missing or invalid data entries
Week 6	Analyze the accuracy metrics for each of the SOTA methods in order to determine which methods are more successful than others and in which areas we should look to improve the models
Week 7	Complete the implementation of improvements for each SOTA method
Week 8	Using the most successful method, construct the visualization: a web application to help users test their personal info to see whether or not they are at risk of heart disease
Week 9	Complete presentation materials including slideshow and finalize all necessary demonstrations
Week 10	Complete final writeup of work and compile all visuals to help convey topics

REFERENCES:

1. Ambesange, S., Vijayalaxmi, A., Sridevi, S., Venkateswaran, & Yashoda, B. (2020, July 28). Multiple Heart Diseases Prediction using Logistic Regression with Ensemble and Hyper Parameter tuning Techniques [Digital Library]. IEEE Xplore. https://doi.org/10.1109/WorldS450073.2020.9210404
2. CDC. (2023, May 15). Heart Disease Facts | cdc.gov [Government Organization]. Centers for Disease Control and Prevention. https://www.cdc.gov/heartdisease/facts.htm
3. Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019, November 21). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification [Digital Library]. IEEE Xplore. https://ieeexplore.ieee.org/document/8908018
4. Javeed, A., Noor, A., Nour, R., Qasim, I., Yongjian, L., & Zhou, S. (2019, November 7). An Intelligent Learning System Based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection [Digital Library]. IEEE Xplore. https://doi.org/10.1109/ACCESS.2019.2952107
5. Jun, Z. (2021). The Development and Application of Support Vector Machine. Journal of Physics: Conference Series, 1748(5), 052006. https://doi.org/10.1088/1742-6596/1748/5/052006
6. Latifah, F. A., Slamet, I., & Sugiyanto. (2020). Comparison of heart disease classification with logistic regression algorithm and random forest algorithm. AIP Conference Proceedings, 2296(1), 020021. https://doi.org/10.1063/5.0030579
7. Marathe, N., Gawade, S., & Kanekar, A. (2021). Prediction of heart disease and diabetes using naive Bayes algorithm. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 447–453. https://doi.org/10.32628/cseit217399
8. Pisner, D. A., & Schnyer, D. M. (2020). Chapter 6—Support vector machine. In A. Mechelli & S. Vieira (Eds.), Machine Learning (pp. 101–121). Academic Press. https://doi.org/10.1016/B978-0-12-815739-8.00006-7
9. Ramalingam, V. V., Dandapath, A., & Raja, M. (2018). Heart disease prediction using machine learning techniques: A survey. International Journal of Engineering & Technology, 7, 684. https://doi.org/10.14419/ijet.v7i2.8.10557
10. Rameshar, V., Hasan, A., Shongwe, T., & Williams, R. (n.d.). Heart Disease Prediction using Machine Learning Techniques. Retrieved September 23, 2023, from https://ieeexplore.ieee.org/document/9655783/
11. Redrowthu Ph.D, V., Gajavelly, K., Nikhath, A., Vasavi, R., & Anumasula, R. R. (2022). Heart Disease Prediction Using Decision Tree and SVM (pp. 69–78). https://doi.org/10.1007/978-981-16-7389-4_7
12. S A, S. (2021). Comparative Study of Naive Bayes, Gaussian Naive Bayes Classifier and Decision Tree Algorithms for Prediction of Heart Diseases. International Journal for Research in Applied Science and Engineering Technology, 9(3), 475–486. https://doi.org/10.22214/ijraset.2021.33228
13. Salmi, N., & Rustam, Z. (2019). Naïve Bayes Classifier Models for Predicting the Colon Cancer. IOP Conference Series: Materials Science and Engineering, 546(5), 052068. https://doi.org/10.1088/1757-899X/546/5/052068

14. Zou, X., Hu, Y., Tian, Z., & Shen, K. (2019). Logistic Regression Model Optimization and Case Analysis. 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 135–139. https://doi.org/10.1109/ICCSNT47585.2019.8962457