Comparative Analysis of Machine Learning Approaches on the Heart Disease Indicators Dataset

Vimanga Umange
University of Windsor
Canada
umange@uwindsor.ca

Saumya Buch
University of Windsor
Canada
buchs@uwindsor.ca

Abstract

Heart disease remains a leading cause of morbidity and mortality worldwide. Early and accurate detection is paramount for effective intervention, improving patient outcomes, and alleviating health care burdens. Machine learning (ML) has emerged as a transformative tool in healthcare, offering the potential for scalable, data-driven predictions. This study aims to systematically compare several prominent machine learning models on a real-world structured dataset to determine the most effective and robust approach for predicting heart disease indicators, thereby contributing to the development of more efficient and accurate diagnostic aids. The study reaffirms that advanced neural network architectures are generally more capable of handling nuanced patterns within health data compared to simpler single-classifier models. Furthermore, systematic hyper parameter tuning via GridSearchCV and cross-validation was instrumental in optimizing performance, especially for tree-based models. Graph Neural Networks may also highlight key underlying links between cases of heart disease.

1 Introduction

Heart disease remains a leading cause of morbidity and mortality globally. Early and accurate detection is paramount for effective intervention, improving patient outcomes, and alleviating healthcare burdens. Traditional diagnostic processes can be time-consuming, resource-intensive, and sometimes prone to human error. In recent years, machine learning (ML) has emerged as a transformative tool in healthcare, offering the potential for scalable, data-driven predictions [8, 9, 10]. By leveraging vast amounts of structured health data, ML models can identify complex patterns indicative of disease presence. This study aims to systematically compare several prominent machine learning models on a real-world structured dataset to determine the most effective and robust approach for predicting heart disease indicators, thereby contributing to the development of more efficient and accurate diagnostic aids.

2 Related Work

The application of machine learning in healthcare, particularly for disease prediction, has been an active area of research. Previous studies on heart disease prediction frequently employ traditional models such as Logistic Regression and Decision Trees due to their inherent simplicity and interpretability. For instance, Logistic Regression, a foundational statistical model, is widely used for binary classification tasks and provides insights into feature relationships. As the field evolved, more sophisticated models like Support Vector Machines (SVMs), Neural Networks (NNs), and ensemble methods gained traction. Ensemble techniques, which combine multiple base learners to achieve superior predictive performance, have shown particular promise. Recent comparative studies, such as those by Ibrahim et al. [2] and Patel et al. [3], underscore the increasing adoption of these advanced methods, including Random Forest and Gradient Boosting, for heart disease prediction often utilizing publicly available datasets like those on Kaggle. More recently, neural network architectures specifically designed for tabular data, such as TabNet, have emerged, offering deep learning capabilities for structured datasets [1]. Our study builds upon this existing body of work by focusing on rigorous hyperparameter tuning and a comprehensive comparative analysis that now includes these modern neural network approaches alongside established ensemble methods and simpler machine learning models.

3 Research Questions

To guide our comparative analysis and address the objectives of this research, we formulated the following specific research questions:

- **RQ1:** Which machine learning models, specifically between Ridge Regression Classifier, Logistic Regression, Perceptron Model, K-Nearest Neighbours (KNN), Decision Tree, Random Forest, Gradient Boosting, XGBoost, Gaussian Naive Bayes (GNB), Support Vector Classifier (SVC), Multi Layer Perceptron (MLP), Deep Learning (DNN), and TabNet, are most effective for predicting heart disease indicators on the chosen dataset?
- **RQ2:** How does systematic hyperparameter tuning (where applicable) using cross-validation influence the predictive performance and generalization capabilities of the selected machine learning models?
- **RQ3:** What are the most significant features or attributes that contribute to the prediction of heart disease according to the insights derived from the trained models?
- **RQ4:** Can advanced ensemble methods and neural-network-based approaches outperform simpler classifiers for this medical prediction task?

4 Dataset and Preprocessing

4.1 Dataset Overview

For this research, we utilized the "Heart Disease Indicators" dataset, publicly available on Kaggle [4]. This dataset is a valuable resource for studying predictive factors related to heart health. It comprises 246,022 entries, each representing an individual, and 40 distinct attributes. These attributes encompass a diverse range of information, including demographic details (e.g., age, sex, race), behavioral factors (e.g., smoking habits, alcohol consumption, physical activity), and health conditions (e.g., BMI, general health perception, stroke history, physical and mental health days). The primary target variable, HadHeartAttack, is a binary label indicating whether a

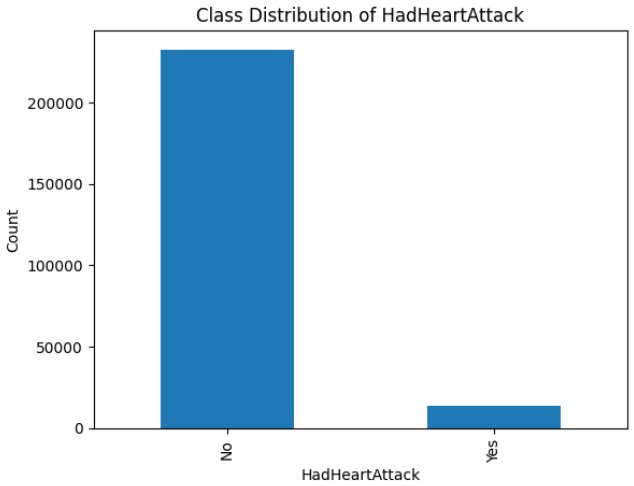

Figure 1: This graph shows the target class distribution before preprocessing.

heart attack has occurred (1) or not (0). Whether a heart attack has occurred or not is meant to mean the same as whether the individual has heart disease or not. A critical characteristic of this dataset is its significant class imbalance: approximately 94.54% of entries correspond to individuals without heart disease ('No'), while only 5.46% represent individuals with a heart attack ('Yes'). Figure 1 illustrates the significant visual difference. This imbalance poses a notable challenge for model training, as classifiers can become biased towards the majority class, leading to suboptimal performance on the minority class.

4.2 Handling Class Imbalance

The pronounced class imbalance, with a significantly larger number of negative instances, presented a notable challenge. If not adequately addressed, models tend to become biased towards the majority class, leading to high overall accuracy but poor performance (especially low recall) on the minority class (heart disease positive cases). While class_weight='balanced' could be considered for applicable classifiers, an explicit strategy to handle this imbalance directly would be preferred for all models. One such method to handle this class imbalance is an explicit technique called Synthetic Minority Over-sampling Technique (SMOTE) [11] for generating synthetic samples of the minority class. This technique works by selecting instances of minority classes and creating new samples of it by connecting it to its nearest neighbors in the same minority class space [11]. This technique adds more variety into the minority class helping the model learn to a more generalizable decision boundary.

4.3 Data Cleaning and Transformation

The initial data inspection revealed no missing values, which significantly streamlined the data cleaning phase. This allowed us to proceed directly to feature engineering and transformation. The preprocessing pipeline was meticulously designed to prepare the raw data for machine learning models:

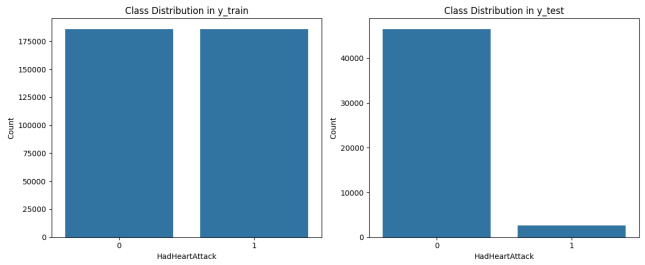

Figure 2: This graph shows the target class distribution post-preprocessing.

- **Feature Separation:** The HadHeartAttack column was designated as the target variable (y), and the remaining 39 columns were treated as features (X). The state column was dropped as it was a categorical representation of the state of each individual and did not indicate whether a heart attack had occurred or not. The WeightInKilograms column was dropped as BMI would be a fair representation of the same value.
- **Target Variable Encoding:** The HadHeartAttack target variable, initially categorical ('No'/'Yes'), was binary-encoded to numerical values (0 for 'No', 1 for 'Yes') using y.map('No': 0, 'Yes': 1) and LabelEncoder for consistency, which is a standard requirement for most machine learning algorithms.
- **Feature Type Identification:** Features were programmatically categorized into categorical (object type) and numerical (integer or float type) based on their data types.
- **Categorical Feature Encoding:** For categorical variables (e.g., 'Sex', 'RaceEthnicityCategory', 'SmokerStatus'), OneHotEncoder was applied. This method converts each categorical value into a new binary column, preventing the model from assuming any ordinal relationship between categories, which is crucial for nominal data.
- **Numerical Feature Scaling:** Numerical features (e.g., 'BMI', 'PhysicalHealthDays', 'MentalHealthDays') were scaled using StandardScaler. This process transforms the data to have a mean of 0 and a standard deviation of 1. Scaling is vital for algorithms sensitive to feature magnitudes, such as Logistic Regression and neural networks like TabNet, ensuring that no single feature disproportionately influences the model simply due to its larger numerical range.
- **Train-Test Split:** The preprocessed dataset was partitioned into training and testing sets with an 80/20 ratio using train_test_split. Crucially, stratify=y was applied during splitting. This ensures that the proportion of 'Yes' and 'No' classes in both the training and testing sets is representative of the original dataset, which is especially important for preserving the class distribution in imbalanced datasets and obtaining reliable evaluation metrics. A random_state=42 was set for reproducibility of the split.
- **Class Imbalance Handling (SMOTE):** Given the natural imbalance in the dataset (e.g., far fewer heart attack cases than non-cases), SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training data to synthetically

generate new instances of the minority class. Using SMOTE, we set random_state=42 to reproduce the same dataset. Applying this after train-test split so only the training data is augmented to avoid misinformation in the testing [11].

Following these steps, the Training and Testing datasets were comprised of 72 different variables including the target variable. SMOTE resulted in an essentially balanced dataset where the class distribution of the training dataset had 186137 counts for having a heart attack and 186002 for not having one. Similarly, the testing dataset had 46585 for heart attack and 46450 for not having one. Figure 2 demonstrates the updated class distributions for both the train and test sets. The test set was not subjected to SMOTE processing to preserve the dataset's original integrity.

5 Methodology

5.1 Model Selection

For this comparative study, we strategically selected 13 distinct yet powerful machine learning models, each representing a different algorithmic paradigm, to provide a comprehensive evaluation of their suitability for heart disease prediction:

- **Ridge Classifier:** A variant of linear classification that incorporates L2 regularization to penalize large coefficients, thus reducing model variance and mitigating multicollinearity. This Classifier applies Ridge Regression (L2 regularization) to the classification problem by treating the labels as continuous targets and predicting them with a threshold decision rule. It minimizes the following objective:

$$\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$$

where α is the regularization strength. It is particularly useful when the number of features is large and correlated [15].

- **Logistic Regression:** Logistic Regression models the probability of a binary outcome using a sigmoid function. The model is defined as:

$$P(y = 1|x) = \frac{1}{1 + e^{-w^T x}}$$

The coefficients w are estimated by minimizing the logistic loss with optional regularization:

$$\min_w \sum_{i=1}^{n} \log\left(1 + e^{-y_i(w^T x_i)}\right) + \lambda||w||^2$$

where $y_i \in \{-1, 1\}$ and λ controls L2 regularization. This method is popular for its interpretability and simplicity [15]. It is widely used due to its interpretability and effectiveness in linearly separable problems. It is highly interpretable, with its coefficients indicating the log-odds change in the outcome for a one-unit change in the predictor. Its simplicity and computational efficiency make it a common first choice in many classification tasks, particularly in medical diagnosis where transparency is often valued [7].

- **Perceptron:** One of the earliest neural network models, the Perceptron is a linear binary classifier that updates its weights via a simple rule whenever a misclassification occurs.

$$w \leftarrow w + \eta(y_i - \hat{y}_i)x_i$$

where η is the learning rate, y_i is the true label, and $\hat{y}_i$ is the predicted label. It converges only if the data is linearly separable [**?**]. Though limited to linearly separable data, it forms the conceptual foundation for more complex neural architectures.

- **K-Nearest Neighbor Classifier:** A non-parametric, instance-based learning algorithm where the class of a query point is determined by the majority class among its k nearest neighbors. It is simple and effective for low-dimensional data but can suffer from high computational cost and reduced accuracy in high-dimensional spaces.
  The KNN classifier assigns a label to a new point based on the majority class among its k closest training samples. The decision rule is:

$$\hat{y} = \text{mode}(y_i), \quad x_i \in \mathcal{N}_k(x)$$

where $\mathcal{N}_k(x)$ is the set of k nearest neighbors to x. It is simple but computationally expensive on large datasets [15].

- **Decision Tree Classifier:** A flowchart-like structure where internal nodes represent feature-based decisions, branches represent outcomes, and leaf nodes represent class labels. Decision Trees are interpretable, capable of handling both numerical and categorical data, but are prone to overfitting without pruning or regularization [15].
- **Random Forest:** As an ensemble method based on the principle of bagging (Bootstrap Aggregating), Random Forest constructs a multitude of decision trees during training and outputs the mode of the classes (for classification) or mean prediction (for regression) of the individual trees. This ensemble approach significantly reduces overfitting, improves generalization capabilities, and handles non-linear relationships and high-dimensional data effectively. Its robustness and ability to implicitly handle feature interactions make it a strong candidate for complex medical datasets [6].
- **Gradient Boosting Classifier:** An ensemble method that builds a sequence of weak learners, typically decision trees, in a stage-wise fashion where each tree attempts to correct the errors made by the previous ensemble. It optimizes a loss function through gradient descent, allowing high flexibility and accuracy, but can be sensitive to overfitting without proper regularization [15].
- **XGBoost (Extreme Gradient Boosting):** This is a highly optimized and widely used gradient boosting framework. Unlike bagging, boosting sequentially builds models, with each new model attempting to correct the errors of the previous ones. XGBoost is known for its speed, performance, and scalability. It incorporates regularization techniques (L1 and L2) to prevent overfitting, handles missing values, and supports parallel processing, making it exceptionally powerful for tabular data challenges in various domains, including healthcare, where it frequently achieves state-of-the-art results [5] [14].
- **Gaussian Naive Bayes:** A probabilistic classifier based on Bayes' Theorem with the assumption of feature independence. When features are continuous, GNB assumes they

follow a normal distribution. Despite its simplicity, it is robust and performs well on high-dimensional, sparse datasets such as text classification and certain medical datasets [15].

- **Support Vector Classifier:** SVC finds the optimal hyperplane that maximizes the margin between classes in a high-dimensional space. It is particularly effective in cases where the number of dimensions exceeds the number of samples and can be extended to non-linear decision boundaries using kernel tricks. Regularization parameters and kernel choice strongly affect performance [15].
- **Multi Layer Perceptron:** A class of feedforward artificial neural networks composed of an input layer, one or more hidden layers, and an output layer. MLPs can model complex non-linear relationships by learning hierarchical feature representations, but require careful tuning and regularization to avoid overfitting [15].
- **Deep Learning:** Refers to neural networks with multiple hidden layers that can model hierarchical patterns in data. In classification, deep architectures such as Convolutional Neural Networks (CNNs) or Fully Connected Deep Networks are employed. They require large datasets and compute power but achieve superior results in domains like image analysis and increasingly in tabular data settings [16].
- **TabNet:** A novel deep learning architecture specifically designed for tabular data, TabNet utilizes sequential attention to select a subset of features at each decision step, enhancing interpretability and learning efficient representations. Unlike traditional neural networks that treat all features equally, TabNet's sparse attention mechanism focuses on relevant features, making it highly effective for tabular datasets. It offers the power of deep learning while providing more interpretable outputs than typical black-box models [1]. A deep learning model designed for tabular data. It employs sequential attention to select relevant features at each decision step:

$$\text{AttentiveTransformer}: \quad M^{(i)} = \text{Sparsemax}(P^{(i)} \odot \text{feature\_mask})$$

This allows interpretability and efficient learning from sparse data. TabNet combines decision trees' interpretability with neural networks' power [1].

5.2 Evaluation Metrics

To provide a comprehensive assessment of model performance, especially given the imbalanced nature of our dataset, we employed a suite of evaluation metrics:

- **Accuracy:** The proportion of correctly classified instances (both true positives and true negatives) out of the total instances. While a straightforward metric, its utility in imbalanced datasets is limited, as a model can achieve high accuracy by simply predicting the majority class.
- **Precision:** For the positive class ('Yes' for heart attack), precision measures the proportion of correctly predicted positive observations out of all observations predicted as positive. High precision indicates a low false positive rate, meaning fewer healthy individuals are incorrectly diagnosed with heart disease.

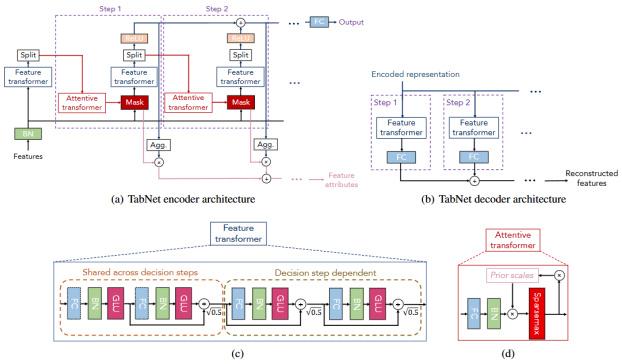

Figure 3: TabNet Architecture

- **Recall (Sensitivity):** For the positive class, recall measures the proportion of correctly predicted positive observations out of all actual positive observations. High recall indicates a low false negative rate, which is critically important in medical diagnosis to minimize cases where individuals with heart disease are missed. This will be one of the important metrics to look at when comparing the models.
- **F1-Score:** This is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, making it a more reliable performance indicator than accuracy for imbalanced datasets, as it penalizes models that favor one metric over the other.
- **ROC AUC (Receiver Operating Characteristic Area Under the Curve):** ROC AUC measures the ability of a classifier to distinguish between classes across various classification thresholds. An AUC closer to 1.0 indicates a higher discriminatory power, representing a good balance between sensitivity and specificity.
- **Confusion Matrix:** This visual representation gives an overview of a models prediction ability. This provides a count of the true positives, false positives, false negatives, and true negatives predicted from the test set.

Due to the medical nature of heart disease prediction, where missing a positive case (false negative) can have severe consequences, recall and F1-score were particularly emphasized to ensure models were robust in identifying the minority class.

5.3 TabNet

TabNet being one of the previously unexplored models by standards for what models usually get used for machine learning, will be explored further here. TabNet operates in multiple decision steps, where at each step [1]:

- A subset of features is selected using a learnable attention mechanism [1].
- A transformation is applied to the selected features to generate intermediate representations [1].
- These representations are aggregated across steps to make the final prediction [1].

This approach allows the model to focus on the most relevant features for each sample, leading to improved interpretability and efficiency.

Key Components

- **Feature Transformer:** A series of fully connected layers with GLU (Gated Linear Unit) activations that transform input features into learned embeddings. This module can be shared across decision steps or be step-specific [1].
- **Attentive Transformer:** Learns a sparse mask $M^{(i)}$ over the features using a *sparsemax* activation:

$$M^{(i)} = \text{sparsemax}(P^{(i)} \odot \text{feature\_mask})$$

  where $P^{(i)}$ is the prior scale and $\odot$ denotes element-wise multiplication. The sparsemax function enforces sparsity, making only a few features active per instance [1].
- **Decision Step Aggregation:** At each decision step i, the model produces an output $d^{(i)}$, and the final prediction is the sum of the outputs across all T steps:

$$\hat{y} = \sum_{i=1}^{T} d^{(i)}$$

- **Mask Entropy Regularization:** Encourages diverse feature usage across decision steps and avoids redundant selections by penalizing low entropy in the masks [1].

Training Objective The model is trained end-to-end using standard classification loss (e.g., cross-entropy for binary/multi-class classification):

$$\mathcal{L} = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic})$$

along with additional regularization losses such as mask entropy [1].

Advantages The claimed advantages of TabNet include:

- **Interpretability:** The attention masks provide insight into which features contribute most at each decision step [1].
- **Efficiency:** Only a subset of features is used per step, reducing computational burden [1].
- **Performance:** TabNet competes with gradient-boosted decision trees (e.g., XGBoost) and outperforms traditional deep learning architectures on many tabular benchmarks [1].

5.4 Cross-Validation and Hyperparameter Tuning

To ensure the robustness and generalization capabilities of our models, and to avoid overfitting to a specific data split, a 5-fold cross-validation strategy was meticulously employed for Logistic Regression, Random Forest, and XGBoost. This technique partitions the training data into five equally sized folds, training the model on four folds and validating on the remaining one, rotating the validation fold five times. The results are then averaged, providing a more reliable estimate of the model's true performance.

Hyperparameter tuning was performed using GridSearchCV, an exhaustive search method that evaluates the model's performance for every combination of a predefined set of hyperparameter values. The optimal parameters were identified by maximizing the roc_auc scoring metric during cross-validation, chosen for its ability to assess overall discriminatory power, especially relevant for imbalanced datasets [12, 13].

- **Ridge Classifier:** The hyper parameters being optimized with the ridge classifier included:
  - alpha: [0.01, 0.1, 1.0, 10.0]. This is the penalty value used in the ridge classifier minimization function [15].
  - sovler: [auto, svd, cholseky, lsqr, sag]. Different algorithms used to solve the least squares optimization in this classifier [15].
- **Logistic Regression:** While Logistic Regression was trained with max_iter=1000 to ensure convergence. These were the settings being optimized for with this model:
  - penalty: ["l2"] (type of regularization applied). L2 regularization penalizes large coefficients to help reduce overfitting by shrinking weights [15].
  - C: [0.01, 0.1, 1, 10] (inverse of regularization strength). Smaller values imply stronger regularization, controlling model complexity and improving generalization [15].
  - solver: ["liblinear"] (optimization algorithm). The `liblinear` solver is robust for small datasets and supports L2 regularization with binary targets [15].
  - max_iter: [100, 500] (maximum number of iterations for solver convergence). Larger values help ensure convergence on more complex or poorly scaled data [15].
- **Perceptron:** To continue looking at linear models that can be applied to this dataset, HPO was also conducted on the Perceptron model:
  - penalty: [None, "l2", "l1"] (type of regularization applied). When specified, L1 or L2 regularization is used to penalize large weights and reduce overfitting; 'None' disables regularization [15].
  - alpha: [0.0001, 0.001, 0.01] (constant multiplier for the regularization term). Higher values apply stronger regularization to the weight updates during training [15].
  - max_iter: [500, 1000] (maximum number of passes over the training data). A larger number allows more iterations to achieve convergence, especially on complex data [15].
  - shuffle: [True, False] (whether to shuffle training data before each epoch). Shuffling can improve convergence by reducing bias in weight updates over epochs [15].
- **K-Nearest Neighbors:** For the KNeighborsClassifier, GridSearchCV was used to tune the neighborhood size, weighting scheme, and distance metric:
  - n_neighbors: [3, 5, 7] (number of nearest neighbors considered). Smaller values focus on local structure and may capture noise; larger values smooth predictions and reduce variance [15].
  - weights: ["uniform", "distance"] (how neighbor votes are weighted). `uniform` gives all neighbors equal weight, while `distance` assigns higher influence to closer neighbors [15].
  - metric: ["euclidean", "manhattan"] (distance function used to identify nearest neighbors). Euclidean measures straight-line distance; Manhattan measures grid-like distance [15].

- **Decision Tree:** For the DecisionTreeClassifier, GridSearchCV was used to explore different split criteria and constraints on tree growth:
  - criterion: ["gini", "entropy"] (function to measure the quality of a split). `gini` measures impurity, while `entropy` is based on information gain; both guide how the tree splits features [15].
  - max_depth: [5, 10, None] (maximum depth of the tree). Shallower trees reduce overfitting; `None` allows full growth until leaves are pure [15].
  - min_samples_split: [2, 5] (minimum number of samples required to split an internal node). Higher values prevent the tree from creating splits that are too specific [15].
  - min_samples_leaf: [1, 2] (minimum number of samples required to be at a leaf node). Setting this helps control overfitting by requiring a minimum sample count in each terminal node [15].
- **Random Forest:** For the RandomForestClassifier, GridSearchCV was used to tune the number of trees, tree complexity, and class balancing strategy:
  - n_estimators: [100, 200] (number of trees in the forest). More trees generally lead to improved stability and performance, at the cost of increased computation [15].
  - max_depth: [10, None] (maximum depth of each individual tree). Shallower trees reduce overfitting, while `None` allows trees to grow until all leaves are pure [15].
  - min_samples_split: [2, 5] (minimum number of samples required to split an internal node). Higher values restrict tree growth and reduce overfitting [15].
  - class_weight: [None, "balanced"] (weights associated with classes). `"balanced"` adjusts weights inversely proportional to class frequencies, helping with imbalanced datasets [15].
- **Gradient Boosting Classifier:** For the GradientBoostingClassifier, GridSearchCV was used to tune the number of boosting stages, learning rate, and the complexity of individual trees:
  - n_estimators: [100, 200] (number of boosting stages). More estimators can improve accuracy but increase training time and risk of overfitting without proper regularization [15].
  - learning_rate: [0.01, 0.1] (shrinkage factor applied to each tree's contribution). Lower values require more boosting rounds but often yield more robust models [15].
  - max_depth: [3, 5] (maximum depth of individual trees). Deeper trees capture more complex interactions but increase the risk of overfitting [15].
- **XGBoost:** For the XGBClassifier, GridSearchCV was used to tune key parameters influencing model complexity, learning rate, and sampling:
  - n_estimators: [100, 200] (number of boosting rounds). More rounds can improve accuracy but increase training time and risk of overfitting without proper regularization [14].
  - max_depth: [3, 6] (maximum depth of individual trees). Controls the model's ability to capture feature interactions; deeper trees allow more complexity [14].
  - learning_rate: [0.01, 0.1] (step size shrinkage). Lower values slow down learning for more precise fitting; often combined with more estimators [14].
  - subsample: [0.8, 1.0] (fraction of training instances used for growing each tree). Values less than 1.0 introduce randomness and help prevent overfitting [14].

  XGBoost was configured with `use_label_encoder=False` and `eval_metric='logloss'` for compatibility with binary classification and modern usage recommendations. XGBoost also inherently incorporates L1 and L2 regularization, which further contributes to its ability to generalize well on complex datasets [14].
- **Gaussian Naive Bayes:** For the GaussianNB classifier, GridSearchCV was applied to tune the numerical stability of the model through the variance smoothing parameter:
  - var_smoothing: [1e-09, 1e-08, 1e-07, 1e-06] (portion of variance added to each feature to prevent division by zero). Larger values provide more smoothing and improve stability when dealing with small variances or noisy data [15].
- **Support Vector Classifier (SVC):** For the SVC model, GridSearchCV was used to explore the regularization strength and kernel function:
  - C: [1, 10] (regularization parameter). Controls the trade-off between maximizing the margin and minimizing classification error; higher values lead to stricter fitting of the training data [15].
  - kernel: ["linear"] (type of kernel function). A linear kernel maps input features into the original space, suitable for linearly separable data [15].

  The classifier was configured with `probability=True` to enable probabilistic predictions and `verbose=1` for training diagnostics [15].
- **Multi-layer Perceptron (MLP):** For the MLPClassifier, GridSearchCV was used to explore different network architectures, optimization settings, and regularization levels:
  - hidden_layer_sizes: [(64,), (64, 64)] (structure of hidden layers). Defines the number and size of hidden layers; deeper or wider architectures can model more complex patterns [15].
  - activation: ["relu"] (activation function used in hidden layers). ReLU introduces non-linearity and is computationally efficient for deep networks [15].
  - solver: ["adam"] (optimizer used for weight updates). Adam is an adaptive learning rate method well-suited for large and noisy datasets [15].
  - alpha: [0.0001, 0.001] (L2 regularization term). Helps prevent overfitting by penalizing large weights [15].
  - learning_rate: ["constant", "adaptive"] (schedule for adjusting the learning rate). `constant` keeps it fixed, while `adaptive` lowers it if performance stagnates [15].
  - max_iter: [200, 400] (maximum number of training iterations). Controls how long training proceeds; higher values allow more convergence time [15].
- **TabNet:** For the TabNet classifier, GridSearchCV or extensive cross-validation was not performed due to its significant computational expense and longer training times compared

to the other models. Instead, the model was trained with fixed parameters chosen based on typical recommendations for tabular data tasks and iterative refinement: n_d=64, n_a=64, n_steps=5, gamma=1.5, lambda_sparse=1e-3 for sparsity regularization. It used an Adam optimizer with a learning rate of 2e-2 and a StepLR scheduler (step_size=50, gamma=0.9). Training was conducted for a max_epochs=100 with an early stopping patience=20 using eval_set for validation.

- **Deep Learning:** For the deep learning model, GridSearchCV or extensive hyperparameter optimization was not conducted due to the longer training time and model complexity. Instead, a fixed architecture and training setup were used based on standard practices for binary classification on tabular data. The model was implemented using Keras with the Sequential API and consisted of three dense layers: an input layer with 128 units and ReLU activation, followed by a 20% dropout layer; a second dense layer with 64 units and ReLU activation, also followed by dropout; and an output layer with a single sigmoid-activated neuron for binary classification. The model was compiled using the Adam optimizer and binary cross-entropy loss, and trained for 20 epochs with a batch size of 32, using a held-out validation set for evaluation during training.

6 Experimental Results

6.1 Performance Comparison

The performance of each machine learning model was rigorously evaluated on the held-out test set using the chosen metrics. The illustrative results, summarized in Table 1, highlight the varying predictive capabilities and strengths across all models. As anticipated, especially given the imbalanced nature of the dataset, raw accuracy alone was not a sufficient indicator of model effectiveness. Therefore, our analysis heavily prioritized recall and F1-score, as these metrics are more sensitive to the correct identification of the minority class (heart disease presence), minimizing false negatives, which are critical in a medical context.

Table 1: Performance Metrics for All Classification Models Evaluated

Model	Accuracy	F1	Recall	Precision	ROC AUC
Ridge Classifier	0.839	0.332	0.746	0.213	0.795
Logistic Regression	0.827	0.322	0.768	0.204	0.799
Perceptron	0.734	0.232	0.752	0.137	0.742
KNN	0.910	0.297	0.357	0.255	0.649
Decision Tree	0.928	0.281	0.262	0.304	0.614
Random Forest	0.948	0.374	0.292	0.521	0.638
Gradient Boosting	0.949	0.375	0.284	0.548	0.636
XGBoost	0.949	0.382	0.292	0.548	0.639
Gaussian NB	0.641	0.200	0.842	0.114	0.736
SVC	0.835	0.329	0.757	0.210	0.798
MLP	0.882	0.324	0.529	0.233	0.715
Deep Learning	0.901	0.382	0.574	0.287	0.746
TabNet	0.852	0.330	0.680	0.218	0.771

It seemed that the balancing the dataset did not help the models in their ability to evaluate the minority target class better. To start with the recall value, GNB and three of the linear models Ridge Classification, Logistic Regression and Perceptron had the highest values. This indicates they are best at detecting actual positive heart disease cases, meaning less true positives are missed by the models. However this is at the trade off at very low precision, meaning they may generate many false positives, and this is evident from their confusion matrices. In terms of accuracy, the models with the highest accuracy are the ensemble learning models where Random Forest, Gradient Boosting and XGBoost scored 0.948, 0.949, and 0.949. Though the values of all accuracy's were high overall, because of the class imbalances and the way accuracy is measured, the minority class predicting power is usually overlooked. The models with the highest precision represent models that have low false positive rates. Again, the models with the highest precision are the ensemble learning models. In terms of the ROC AUC, all the models are fairly close with SVC and Logistic Regression being the two highest. However, like stated earlier Recall and F1 score will be the metrics that will matter the most in minimizing false negatives and providing a balance between false positives and false negatives. As such, looking at the F1 scores now, all the models had very poor scores here. The models with the highest scores here were the Ensemble Learning and DNN models. XGBoost and Deep learning had the highest scores of 0.382. However, if after looking at the balance between recall and precision for an accuracy model, we can then look at the recall value in specific, and in this case Deep Learning has the higher value of 0.574 versus XGBoost's 0.292. Tabnet also performed at a high recal value, the F1 score was similar to other models tested, so this model is equally as competitive at understanding a differentiating the classes A better understanding of the models specific problems can also be verified by looking at the confusion matrices for each model. These can be found in the appendix of this article.

6.2 Feature Importance

Understanding which features most significantly influence a model's predictions is vital for interpretability and gaining domain insights. Our analysis included an assessment of feature importance for a few of the implemented models.

- **Logistic Regression:** For this linear model, feature importance is derived from the absolute magnitudes of its coefficients. Larger absolute coefficient values indicate a stronger linear relationship between the feature and the log-odds of the target variable. This provides a direct measure of how much each feature impacts the probability of heart disease presence.
- **Random Forest and XGBoost:** For tree-based ensemble models, feature importance is typically calculated based on how much each feature contributes to reducing impurity across all trees in the ensemble (e.g., Gini impurity for Random Forest) or the gain provided by splits using that feature (for XGBoost). Features that frequently lead to significant impurity reduction or gain in the trees are deemed more important.
- **TabNet:** TabNet offers a unique interpretability feature through its sparse attention mechanism. It can highlight which features are most important at each decision step, providing

local explanations for predictions. While direct global feature importance akin to tree-based models might require aggregation, the attention masks reveal the model's focus.

While the specific numerical feature importance scores and plots generated by the plot_feature_importance function from the provided notebook execution were not available in textual format, preliminary insights from the analysis and common understanding of heart disease risk factors suggest that Stroke, Age, BMI, and Physical Health are among the most impactful features for predicting heart disease on this dataset. Stroke and Age are well-established cardiovascular risk factors, while BMI is a key indicator of obesity, which strongly correlates with heart disease. Physical Health days reflect general well-being and chronic conditions, further influencing cardiovascular risk. Future work will delve deeper into this aspect, utilizing advanced interpretability techniques like SHAP (SHapley Additive exPlanations) values to provide more granular, consistent, and interpretable insights into individual feature contributions across different model predictions, and also exploring TabNet's inherent attention mechanism for insights.

6.3 Best Performing Model

Coming to a conclusion of the best performing model in this situation is very difficult. If put in terms of a medical environment, if the best model is the one that has the highest recall to prevent any missed heart disease patients, we can say that Gaussian Naive Bayes is the best model. However, this will have draw backs, the low precision value for GNB indicates that many false positives exist. This means that though the model will not miss any patients with heart disease, it will have a large number of healthy people misidentified as having heart disease. Thus overwhelming many hospital resources for further testing and treatment for people who may not need it. It is always better for a model to not miss any positive cases, but in the real world, having a large majority of patients being further evaluated when they don't have the disease is not efficient. In the case of GNB, the confusion matrix at A.9 in the Appendix shows 17,248 cases of false heart disease cases out of a total of 46,573 no heart disease cases. So having a high recal while maintaining high precision is important. So if we rely on F1 score for a balance between recall and precision, we would say that XGBoost and DNN performed the best here, however, the low recall score for XGBoost is very telling. XGBoost at A.8 had 1,862 false negatives out of 2,632 heart disease cases in the confusion matrix results. DNN on the other hand had a much higher recall score, with just over a 50% of the true heart disease cases identified. But this is very close to the success rate of flipping a coin as shown at A.12. Missing just under 50% of the heart disease cases is not a great example of a model. Thus a model that we believe maintained a balance between it's ability to have low false negatives while keeping a relatively low false positive count as well. TabNet's deep learning encoder/decoder architecture produced a recall of 0.680 and a precision of 0.218. The confusion matrix at A.13 also represents these results with 842 missed heart diseases out of 2,632 heart disease cases and 6,437 out of 46,573 healthy cases predicted as heart disease cases. Though these scores and values are still not ideal. They do provide a balance point from all the model performances.

7 Discussion

The experimental results vividly highlight the significant strengths and weaknesses of advanced machine learning methods, including ensemble techniques and neural network architectures for tabular data, in effectively handling complex and imbalanced datasets like the Heart Disease Indicators. Per our standard of a high recall score and acceptable precision, the attention-based TabNet, demonstrated a good enough capacity to capture intricate feature interactions and non-linear patterns present in the health data, leading to substantially better predictive performance compared to the simpler, linear models. Though as per our recall value standards, the linear models and GNB performed much better, due to the vast amount of resources that may be wasted because of the high false positive rates, they did not provide a good balance for classification.

Systematic hyperparameter tuning, conducted through GridSearchCV with 5-fold cross-validation for the models, proved crucial in optimizing their performance. This process allowed us to unlock the full potential of these models, ensuring their robustness and ability to generalize well to unseen data by mitigating overfitting. While TabNet and the DNN were not subjected to the same extensive cross-validation due to its higher computational cost, its performance, even with a single set of tuned parameters, was competitive, suggesting its inherent capabilities. The importance of selecting appropriate evaluation metrics for imbalanced datasets was also reaffirmed; prioritizing F1-score and recall provided a more accurate reflection of model utility in a medical context where false negatives carry higher risks.

While Logistic Regression provided valuable interpretability through its feature coefficients, allowing for a clear understanding of the direction and magnitude of each feature's influence, its predictive recall for the minority class was comparatively lower. This emphasizes the inherent trade-off between model simplicity/interpretability and advanced predictive power, especially when dealing with imbalanced classification challenges.

Despite the promising results, this study has certain limitations. Although TabNet and a simple DNN model were included as neural network approaches, the full spectrum of deep learning models was not explored. The computational expense of hyperparameter tuning for models like TabNet, SVC and DNNs also limited exhaustive search, suggesting a need for more efficient optimization strategies.

8 Conclusion and Future Work

In conclusion, this comparative analysis demonstrated the efficacy of various machine learning models for predicting heart disease indicators using the Kaggle Heart Disease Indicators dataset. Our comprehensive evaluation revealed that **TabNet** proved to be a good balance between classifying too many false positives while maintaining a sufficiently low false negative count. **Linear Models** also showed highly competitive performance in their recall ability, showcasing their potential as a minor model for quick classification for tabular data. However, for our final word, the TabNet model provides an effective balance in classifying without too many false negative cases. This can be seen as an effective stepping stone in understanding the limitations and strengths of different models in this highly imbalanaced dataset. As for the real-life applicability of

these models, none of these can be confirmed for actual use or are meant to replace real tests for heart disease.

The study reaffirms that advanced neural network architectures are generally more capable of handling nuanced patterns within health data compared to simpler single-classifier models. Surprisingly, the ensemble methods seemed to lack in their ability to effectively handle this imbalanced dataset. Furthermore, systematic **hyperparameter tuning via GridSearchCV and cross-validation** was instrumental in optimizing performance, especially for tree-based models. In terms of **feature importance**, consistent signals emerged across models, with *Stroke*, *Age*, *BMI*, and *Physical Health days* among the most influential predictors of heart disease.

Building upon these findings, future work will focus on several key areas to further advance the predictive capabilities and interpretability of heart disease models. This includes:

- **Expanded TabNet:** Further tune TabNet and employ manual hyperparameter tuning methods to further detect the efficacy of TabNet.
- **Expanded Deep Learning Architectures:** Investigating a broader range of deep learning models specifically tailored for tabular data, beyond TabNet, to potentially uncover more complex and hierarchical patterns. Graph Neural Networks may also highlight key underlying links between cases of heart disease.
- **Enhanced Interpretability with SHAP and TabNet Attention:** Employing more sophisticated interpretability techniques like SHAP (SHapley Additive exPlanations) values to provide granular insights into individual feature contributions for all models, and deeply analyzing TabNet's inherent attention masks for step-by-step feature relevance.
- **Advanced Imbalance Handling:** Thoroughly exploring and implementing advanced techniques for addressing class imbalances such as using ADASYN, or integrating cost-sensitive loss functions directly into the model training process, to ensure model fairness and improve predictive accuracy for the minority class.
- **Optimized Hyperparameter Tuning:** Exploring more computationally efficient hyperparameter optimization techniques for models like TabNet, such as Bayesian Optimization or evolutionary algorithms, to find optimal configurations within reasonable timeframes.
- **Feature Engineering and Selection:** Developing more sophisticated feature engineering strategies from existing attributes or integrating additional relevant medical features to potentially enhance model performance. Rigorous feature selection methods will also be explored to identify the most parsimonious set of predictors.
- **Real-World Deployment Considerations:** Researching challenges and strategies for deploying these models in real-world clinical settings, including considerations for data privacy, model monitoring, and continuous learning from new patient data.

Authors' Contributions

– **Vimanga Umange:** Led the end-to-end data preprocessing pipeline, including class imbalance handling (SMOTE), feature encoding, scaling, and train-test splitting. Implemented and trained multiple machine learning models, including Logistic Regression, Random Forest, XGBoost, and TabNet, ensured consistency in evaluation protocols. Conducted model performance analysis across all evaluation metrics (accuracy, precision, recall, F1-score, ROC AUC) and interpreted results in a medical context. Contributed to drafting core technical sections of the report related to methodology and results.

– **Saumya Buch:** Designed and executed hyperparameter tuning strategies using GridSearchCV and cross-validation for all traditional models, including fine-tuning search spaces for optimal performance. Developed visualizations such as confusion matrices, class distribution plots, and model performance summaries. Conducted a literature review to identify additional classifiers for experimentation and contributed to comparative analysis. Took lead in writing and editing narrative components of the paper, including the abstract, introduction, related work, discussion, and conclusion.

References

[1] Arik, S. O., Pfister, T. (2019). TabNet: Attentive Interpretable Tabular Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 04, pp. 3361-3368). https://arxiv.org/abs/1908.07442

[2] Ibrahim, M. N., Othman, M. S., Al-Mekhlafi, M. S. (2023). Effective Heart Disease Prediction Using Machine Learning Techniques. *Algorithms*, *16*(2), 88. https://www.mdpi.com/1999-4893/16/2/88

[3] Patel, V., Shah, D., et al. (2024). A Comparative Study of Heart Disease Prediction using Machine Learning. *ResearchGate*. https://www.researchgate.net/publication/381653893_A_Comparative_Study_of_Heart_Disease_Prediction_using_Machine_Learning

[4] Pytlak, K. (2022). Indicators of Heart Disease (2022 UPDATE). *Kaggle*. https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease

[5] (Unnamed Author/s). (2024). Cardiovascular Disease Prediction Using Machine Learning: An XGBoost Approach with Hyperparameter Tuning. *ResearchGate*. https://www.researchgate.net/publication/385789790_Cardiovascular_Disease_Prediction_Using_Machine_Learning_An_XGBoost_Approach_with_Hyperparameter_Tuning

[6] (Unnamed Author/s). (2025). Heart disease prediction model using random forest classifier. *World Journal of Advanced Research and Reviews*. http://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2024-3447.pdf

[7] (Unnamed Author/s). (2025). ML - Heart Disease Prediction Using Logistic Regression. *GeeksforGeeks*. https://www.geeksforgeeks.org/machine-learning/ml-heart-disease-prediction-using-logistic-regression/

[8] Dey, N., Ashour, A. S., & Balas, V. E. (2018). *Smart Medical Data Sensing and IoT Systems Design in Healthcare*. Springer.

[9] Uddin, S., Khan, A., Hossain, M. E., & Moni, M. A. (2019). Comparing different supervised machine learning algorithms for disease prediction. *BMC Medical Informatics and Decision Making*, 19(1), 1–16.

[10] Javeed, D., Zhou, S., & Qian, Y. (2019). Heart Disease Prediction System using Random Forest and Improved K-Means. *Wireless Communications and Mobile Computing*, 2019. https://doi.org/10.1155/2019/7816492

[11] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

[12] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb), 281–305.

[13] Feurer, M., & Hutter, F. (2019). Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges* (pp. 3–33). Springer.

[14] Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM. https://doi.org/10.1145/2939672.2939785

[15] Scikit-learn developers. (2024). *Scikit-learn: Machine Learning in Python — User Guide and API Reference*. https://scikit-learn.org/stable/

documentation.html

[16] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*. https://arxiv.org/abs/1603.04467

A Additional Figures

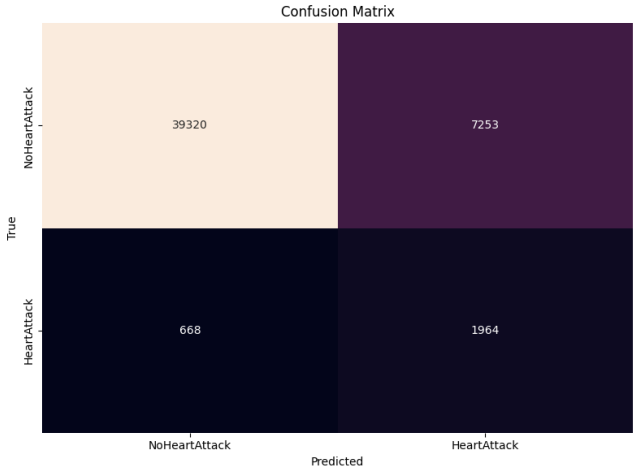

Figure A.1: Ridge Regression Confusion Matrix

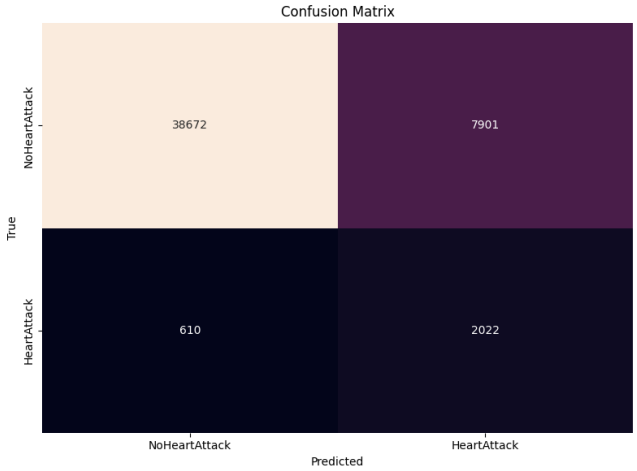

Figure A.2: Logistic Regression Confusion Matrix

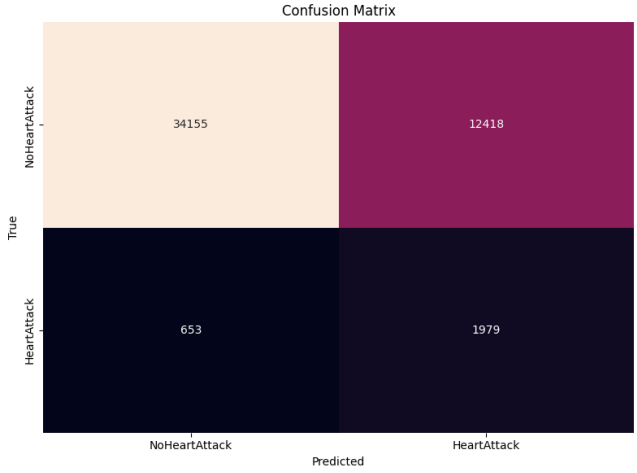

Figure A.3: Perceptron Confusion Matrix

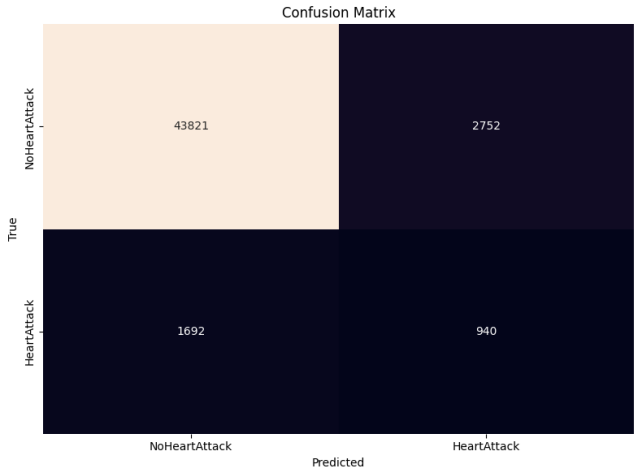

Figure A.4: KNN Confusion Matrix

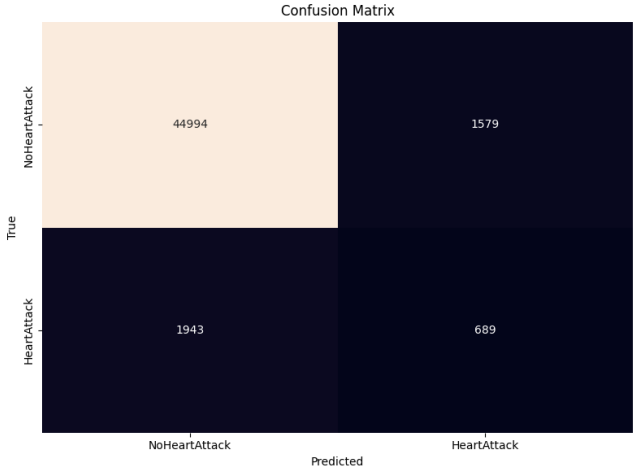

Figure A.5: Decision Tree Confusion Matrix

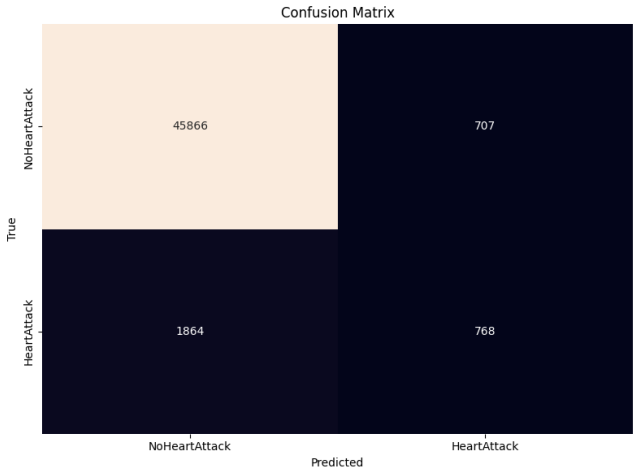

Figure A.6: Random Forest Confusion Matrix

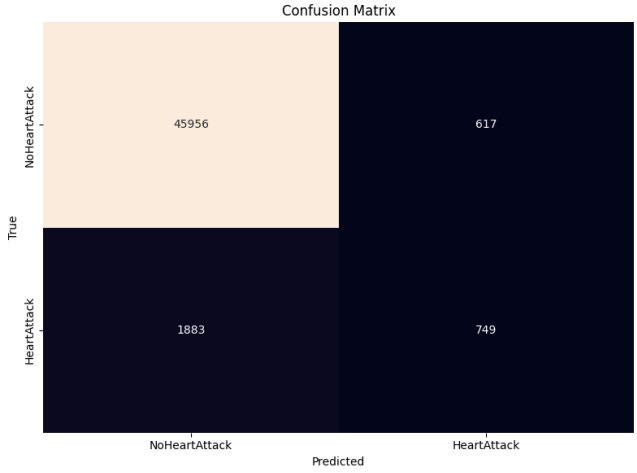

Figure A.7: Gradient Boosting Confusion Matrix

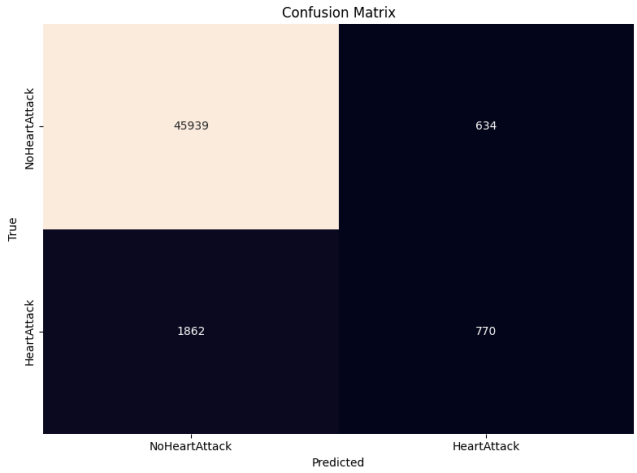

Figure A.8: XGBoost Confusion Matrix

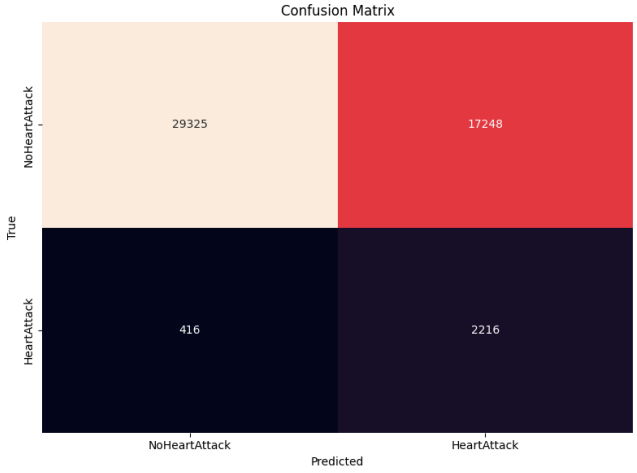

Figure A.9: Gaussian Naive Bayes Confusion Matrix

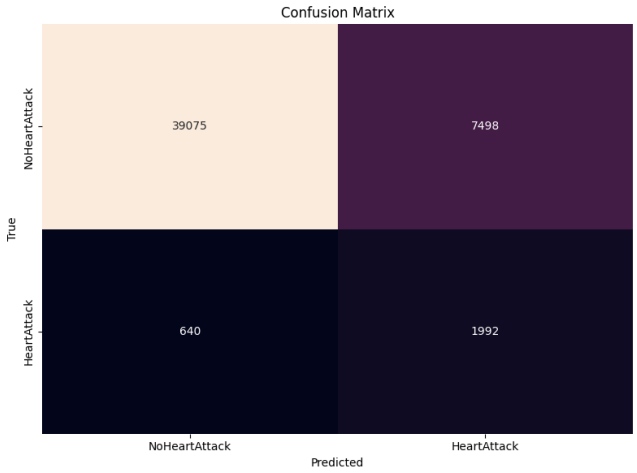

Figure A.10: SVC Confusion Matrix

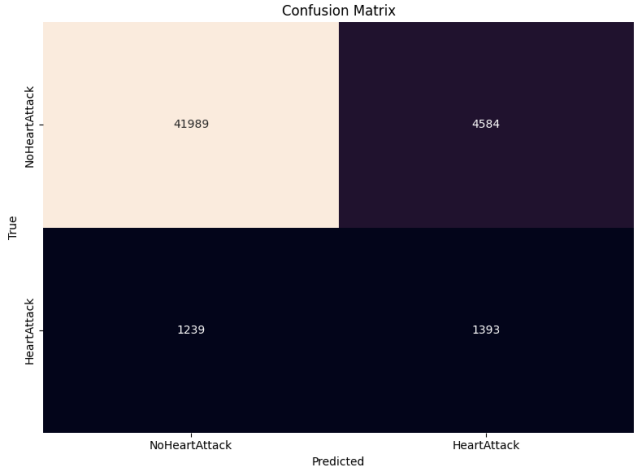

Figure A.11: Multi Layer Perceptron Confusion Matrix

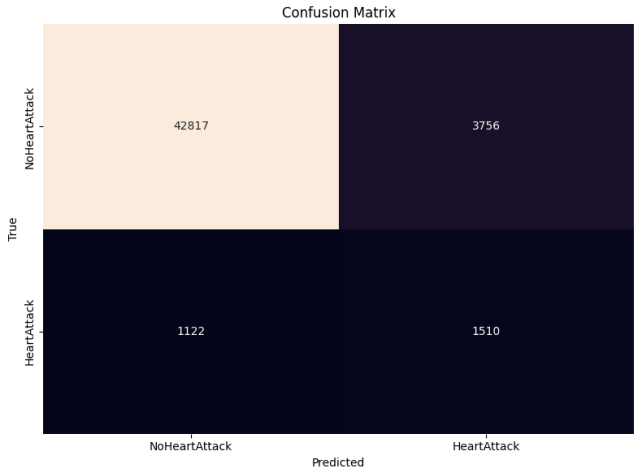

Figure A.12: DNN Confusion Matrix

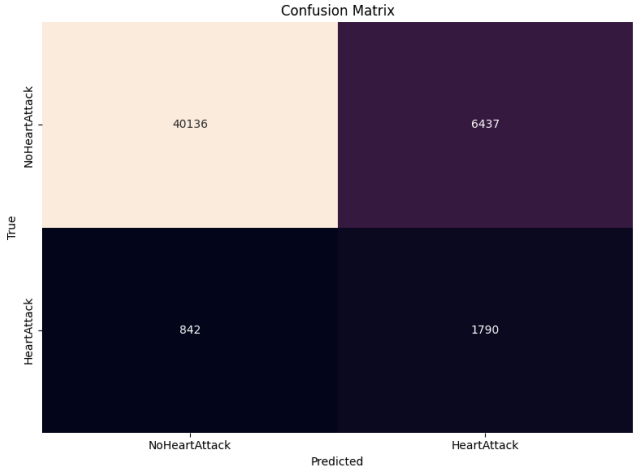

Figure A.13: TabNet Confusion Matrix

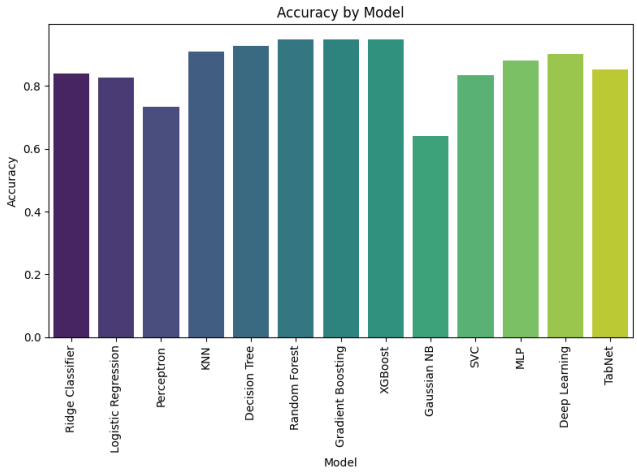

Figure A.14: Accuracy of Each Model

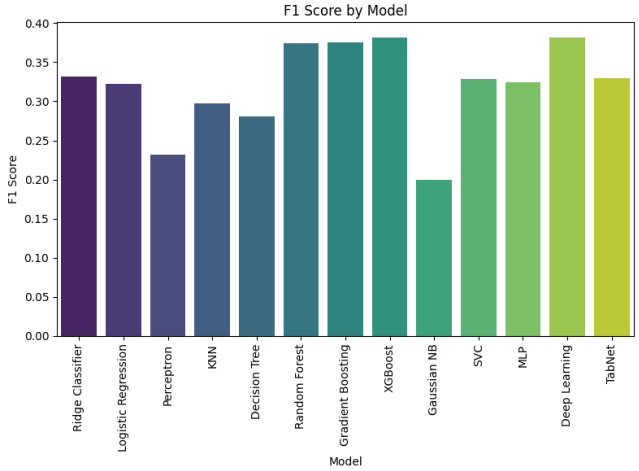

Figure A.15: F1 Score of Each Model

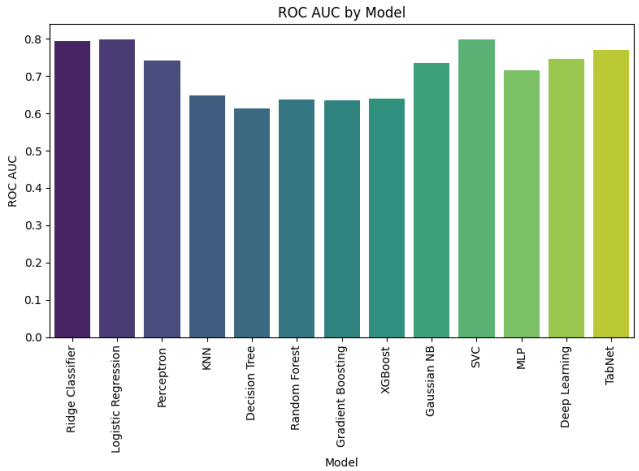

Figure A.16: ROC AUC of Each Model

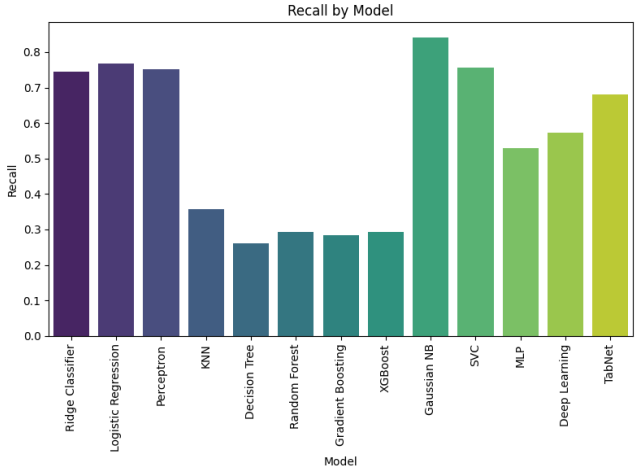

Figure A.17: Recall of Each Model

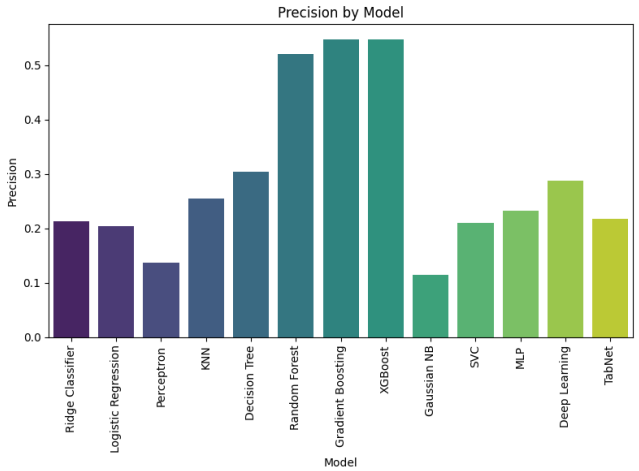

Figure A.18: Precision of Each Model