King Saud University
College of Computer and Information Sciences
Department of Information Technology

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

IT 461 Project Report

Heart Disease Prediction

Prepared by

Deema Alresheed, 442200938
Reem Almusharraf, 442200387
Bayan Albadri, 442201841
Sarah K Juwied, 442201381

Supervised by
Dr. Luluah Alhusain

First Semester 1445 H Fall 2023

Table Of Contents:

Introduction

Heart disease is one of the leading causes of death worldwide. According to the World Health Organization, heart disease causes 17.9 million deaths each year [1], which is about 31% of all deaths. Thus, predicting whether a person has heart disease is important because it can help doctors identify patients at risk for heart disease so they can be monitored and treated early.

Through machine learning algorithms, it is possible to predict whether a patient has heart disease based on their medical history and other factors. By utilizing this model, doctors and other healthcare professionals will be able to identify patients at risk for heart disease and intervene early in their treatment.

Background

Heart disease, which refers to various medical conditions that affect the heart and its blood vessels, is a major cause of death worldwide, making early detection and prevention of it essential. In contrast, machine learning techniques have emerged as powerful tools for analyzing complex medical data sets and predicting heart disease risk by developing algorithms that automatically learn patterns and relationships from data without being explicitly programmed. By taking advantage of these technologies, it is possible to develop accurate and reliable models that can help healthcare professionals make informed decisions and provide personalized care. One of the advantages of machine learning in this context is its ability to uncover hidden patterns and predictive insights within the data. By training models on large, diverse datasets, machine learning algorithms can identify complex relationships and risk factors that may not be readily apparent to human observers.

The performance of machine learning models for heart disease prediction is typically evaluated using various metrics, including accuracy, precision, recall, and f1-score. These metrics help assess the model's ability to correctly classify individuals into the presence or absence of heart disease categories.

Task

The task of the heart disease prediction model as shown in Figure 1 can be formulated as follows:

- Input: A patient's medical record which is a vector of features, such as BMI, Diabetes, age, sex and more.
- Output: A binary prediction of whether the patient has heart disease or not.

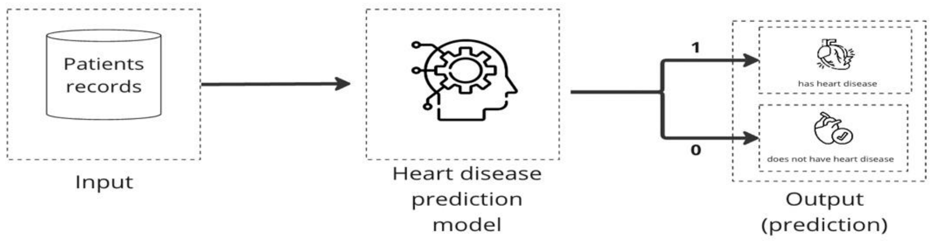


Figure 1 (illustrates a prediction model)

Related Works

Before we dive into our analysis, let's take a quick look at what others have done with our dataset. This "Related Works" section summarizes their key findings and methods, helping us build on their insights for our research.
A study conducted by MinThihaTun [2] and shared on Kaggle explored the "Heart Disease Health Indicators" [3] dataset. The data scientist employed binary logistic regression after performing data cleaning and preprocessing. The dataset was then split and converted to numpy arrays. The implementation of binary logistic regression involved calculating costs, computing gradients, optimizing with gradient descent, and evaluating metrics. Additionally, the data scientist suggested using an external logistic regression model from scikit-learn and demonstrated fitting and prediction steps with that model. The study achieved a final accuracy score of 15.4%.

Another study by Naman Srivastava [4] on the same dataset utilized a quantum kernel method for the Support Vector Machine (SVM) model. Following data cleaning and preprocessing, the dataset was divided into training and testing sets. A quantum kernel with 6 qubits was initialized using the PennyLane library. To facilitate SVM training, a kernel matrix was constructed by computing kernel values for pairwise combinations of training samples. The SVM model trained with the kernel matrix as the kernel parameter achieved an impressive accuracy score of 88%.

For the "*Personal Key Indicators of Heart Disease Dataset*" [5], Fatma Yousuf Mohamed conducted a study employing random forest as the chosen method [6]. After cleaning and preprocessing the dataset, it was split into training and testing sets. Various techniques were employed to address the issue of imbalanced data, with "Random oversampling" proving successful in achieving a balanced dataset. The study achieved a final accuracy score of 96.67%.

Another study conducted by Ziad Hamad on the same dataset [7]utilized the gradient boosting classifier (GBC) as the method of choice. After cleaning and verifying the values in each feature, categorical data was converted into numerical format. The data was then split and prepared for modelling. The GBC model achieved a final accuracy score of 98.5%.

In conclusion, comparing the accuracy results from the four studies revealed that the Gradient Boosting Classifier emerged as the most successful method, achieving the highest accuracy score of 98.5%.

Data

For our heart disease prediction project, we're utilizing two datasets: the "*Personal Key Indicators of Heart Disease Dataset*" [5] with 18 variables and 319,795 rows, and the "*Heart Disease Health Indicators Dataset*" [3] with 22 variables and 22,978 rows. Both datasets are considered clean, requiring no explicit cleaning procedures. We'll employ these datasets to train machine learning models, using "HeartDisease" and "HeartDiseaseorAttack" as explanatory variables, respectively. This approach allows us to analyze relationships between risk factors and heart health, developing predictive models for assessing an individual's susceptibility to heart disease. The clean nature of the datasets ensures a reliable exploration of these objectives within the machine learning framework.

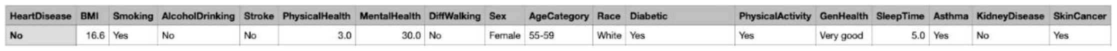
- "*Personal Key Indicators of Heart Disease Dataset*" [5]:
  Characteristics: 18 variables, 319,795 rows
  Explanatory Variable: "HeartDisease"

- "*Heart Disease Health Indicators Dataset*" [3]:
  Characteristics: 22 variables, 22,978 rows
  Explanatory Variable: "HeartDiseaseorAttack"

Table 1 (Dataset features)

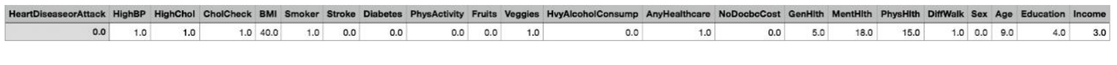
Dataset Name	Features
Personal Key Indicators of Heart Disease	*HeartDisease, BMI, Smoking, AlcoholDrinking, Stroke, PhysicalHealth, MentalHealth, DiffWalking, Sex, AgeCategory, Race, Diabetic, PhysicalActivity, GenHealth*(**General health**), *SleepTime, Asthma, KidneyDisease, SkinCancer.*
Heart Disease Health Indicators Dataset	*HeartDiseaseorAttack, HighBP*(**blood pressure(high)**), *HighChol*(**cholesterol (high)**), *CholCheck, BMI, Smoker, Stroke, Diabetes, PhysActivity*(**Physical Activity**), *Fruits, Veggies, HvyAlcoholConsump*(**alcohol consumption**), *AnyHealthcare, NoDocbcCost*(**quantity of out-of-pocket health care costs**), *GenHlth*(**General health**), *MentHlth*(**Mental Health**), *PhysHlth*(**Physical Health**), *DiffWalk*(**Difficulty walking**), *Sex, Age, Education, Income.*

- Examples:



HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	SleepTime	Asthma	KidneyDisease	SkinCancer
No	16.6	Yes	No	No	3.0	30.0	No	Female	55-59	White	Yes	Yes	Very good	5.0	Yes	No	Yes

Figure2 ("Personal Key Indicators of Heart Disease," Patient record)



HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	5.0	18.0	15.0	1.0	0.0	9.0	4.0	3.0

Figure3 ("Heart Disease Health Indicators Dataset," Patient record)

In terms of cleaning and preprocessing the data sets both of them are cleaned and no need for preprocessing in general.

Methods

This section covers the chosen machine learning algorithms and their rationale for our project. We'll also touch on any preprocessing, feature engineering, or dimensionality reduction techniques applied to optimize data quality and enhance overall machine learning performance.

Random Forest (RF)

For our heart disease prediction project, we selected the Random Forest algorithm for its adaptability in handling both classification and regression tasks, especially in complex datasets typical of health-related data. In the preprocessing phase, we focused on removing duplicate entries as a key technique. This step aimed to ensure the integrity of the dataset and improve the accuracy of our predictive model. We exploited Random Forest's intrinsic features, including its ability to analyze feature importance, to gain insights into the significant factors contributing to the prediction of heart disease.

Support Vector Machines (SVM)

Support Vector Machine (SVM) algorithm was strategically chosen for our heart disease prediction project. The decision is grounded in SVM's efficacy in handling complex datasets and its ability to delineate decision boundaries in high-dimensional spaces.

Throughout the preprocessing phase, categorical variables were encoded, ordinal categories were mapped, and one-hot encoding was applied to certain features. The 'AgeCategory' column underwent a mapping transformation for numerical representation. StandardScaler was employed to standardize the numerical values, ensuring consistent scaling across features. The SVM model was then fine-tuned using Grid Search, optimizing hyperparameters such as regularization (C) and kernel coefficient (gamma). Principal Component Analysis (PCA) was integrated into the pipeline for feature reduction, and balanced class weights were utilized to address potential imbalances in the dataset. This meticulous approach aims to enhance the SVM model's predictive capabilities and overall performance in our heart disease prediction task.

Neural Networks (NN)

Neural networks are known for their ability to learn complex patterns and relationships in data, making them suitable for tasks such as classification. The choice of a neural network for this problem is justified by its capability to handle non-linear relationships between input features and the target variable. Heart disease prediction often involves multiple factors and complex interactions, which can be effectively captured by neural networks. Throughout the preprocessing phase, categorical variables were encoded, ordinal categories were mapped, and one-hot encoding was applied to certain features. The choice of a neural network for this problem is justified by its capability to handle non-linear relationships between input features and the target variable. Heart disease prediction often involves multiple factors and complex interactions, which can be effectively captured by neural networks. Throughout the preprocessing phase, categorical variables were encoded, ordinal categories were mapped, and one-hot encoding was applied to certain features.

Experiment

This section summarizes the training process for each model, highlighting architecture, preprocessing, and hyperparameter tuning. We'll touch on dataset handling, feature engineering, and evaluation metrics, along with the libraries and computational resources used.

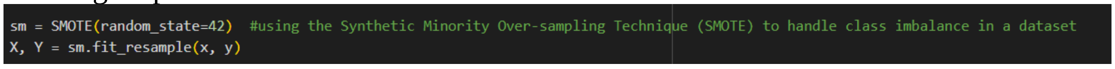
Random Forest (RF)

During the training phase, we identified an imbalance in both datasets. To address this issue, we employed different oversampling techniques for each dataset. Specifically, for the "*Personal Key Indicators of Heart Disease Dataset*" [6] we applied SMOTE oversampling, whereas for the "*Heart Disease Health Indicators Dataset*" [7]we utilized Random Oversampling.

To optimize the model performance, hyperparameter tuning was conducted using Grid Search. For the "Personal Key Indicators of Heart Disease,", we focused on tuning "max_depth" and "max_features" parameters, while for the "Heart Disease Health Indicators Dataset,", we explored a broader set of hyperparameters, including 'n_estimators,' 'max_depth,' 'min_samples_split,' and 'min_samples_leaf.'

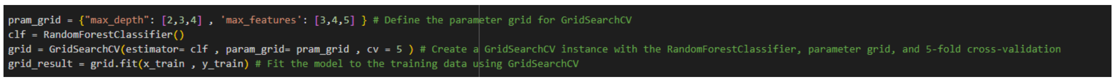
During this process, the computations were handled on the laptop's CPU, using Python libraries such as scikit-learn for machine learning algorithms, and imbalanced-learn for oversampling techniques.

The Evaluation of the models was assessed using a standard set of evaluation metrics, including accuracy, precision, recall, F1 score, and Confusion Matrix. These metrics provided a comprehensive understanding of the model's performance in addressing the challenges specific to each dataset.



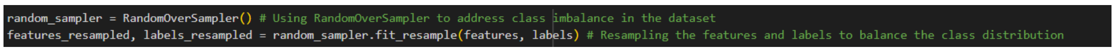
```
sm = SMOTE(random_state=42)  #using the Synthetic Minority Over-sampling Technique (SMOTE) to handle class imbalance in a dataset
X, Y = sm.fit_resample(x, y)
```

Figure4 ("Personal Key Indicators of Heart dataset", SMOTE oversampling)



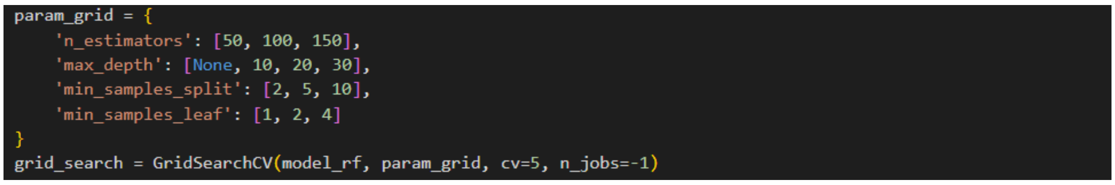
```
pram_grid = {"max_depth": [2,3,4] , 'max_features': [3,4,5] } # Define the parameter grid for GridSearchCV
clf = RandomForestClassifier()
grid = GridSearchCV(estimator= clf , param_grid= pram_grid , cv = 5 ) # Create a GridSearchCV instance with the RandomForestClassifier, parameter grid, and 5-fold cross-validation
grid_result = grid.fit(x_train , y_train) # Fit the model to the training data using GridSearchCV
```

Figure 5 ("Personal Key Indicators of Heart dataset", Grid Search)



```
random_sampler = RandomOverSampler() # Using RandomOverSampler to address class imbalance in the dataset
features_resampled, labels_resampled = random_sampler.fit_resample(features, labels) # Resampling the features and labels to balance the class distribution
```

Figure 6 ("Heart Disease Health Indicators Dataset," Random Oversampling)



```
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(model_rf, param_grid, cv=5, n_jobs=-1)
```

Figure 7 ("Heart Disease Health Indicators Dataset," Grid Search)

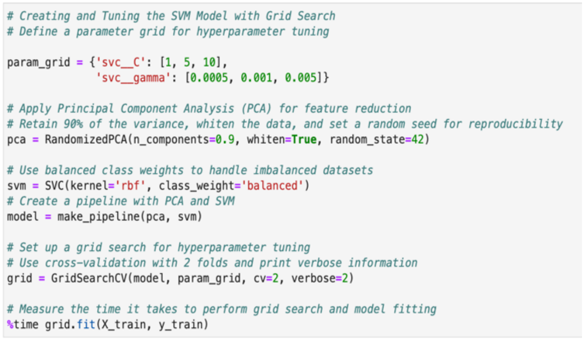
Support Vector Machines (SVM)

In crafting and training the Support Vector Machine (SVM) models for both datasets a meticulous approach was undertaken. The model architecture incorporated a Radial Basis Function (RBF) kernel, pivotal for its ability to handle non-linear relationships within the data. The SVM was structured within a pipeline that not only included the kernel but also incorporated Principal Component Analysis (PCA) for feature reduction. This PCA step was unique to each dataset, with the "*Heart Disease Health Indicators Dataset*", aiming to retain 95% variance and the "*Personal Key Indicators of Heart Disease Dataset*" [5] opting for 90%.

Before diving into the training phase, rigorous preprocessing tailored to each dataset was executed. For the "*Heart Disease Health Indicators Dataset*", numerical columns underwent scaling using the StandardScaler . Additionally, the "BMI" column was subject to feature engineering, categorizing it into distinct health status groups. On the contrary, the "*Personal Key Indicators of Heart Disease Dataset*" [5] underwent categorical encoding, mapping of ordinal features, and one-hot encoding for specific columns. For instance, the 'GenHealth' column, representing general health status, was mapped to numerical values, and the 'AgeCategory' column was transformed using a predefined dictionary. Moreover, categorical features like 'Race' and 'Diabetic' were subjected to one-hot encoding to create binary columns for each category. These preprocessing steps were crucial in preparing the datasets for subsequent training, ensuring compatibility with the Support Vector Machine (SVM) model and enhancing its predictive capabilities.

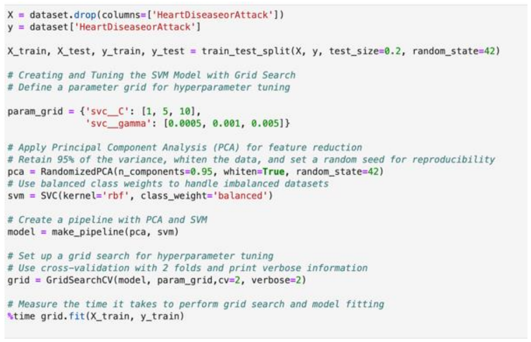
Before diving into the training phase, rigorous preprocessing tailored to each dataset was executed. For the "*Heart Disease Health Indicators Dataset*", numerical columns underwent scaling using the StandardScaler . Additionally, the "BMI" column was subject to feature engineering, categorizing it into distinct health status groups. On the contrary, the "*Personal Key Indicators of Heart Disease Dataset*" [5] underwent categorical encoding, mapping of ordinal features, and one-hot encoding for specific columns. For instance, the 'GenHealth' column, representing general health status, was mapped to numerical values, and the 'AgeCategory' column was transformed using a predefined dictionary. Moreover, categorical features like 'Race' and 'Diabetic' were subjected to one-hot encoding to create binary columns for each category. These preprocessing steps were crucial in preparing the datasets for subsequent training, ensuring compatibility with the Support Vector Machine (SVM) model and enhancing its predictive capabilities.

The subsequent training process as shown in Figures 8 and 9 involved a strategic split of the datasets into training and testing sets (80-20 split), ensuring diverse samples for model learning and evaluation. Hyperparameter tuning was executed through a grid search strategy, focusing on the regularization parameter (C) and the kernel coefficient (gamma). The hyperparameter grid was meticulously defined, encompassing values such as [1, 5, 10] for C and [0.0005, 0.001, 0.005] for gamma. The grid search was conducted using two-fold cross-validation for robust performance evaluation and reducing the probability of overfitting.



```
# Creating and Tuning the SVM Model with Grid Search
# Define a parameter grid for hyperparameter tuning

param_grid = {'svc__C': [1, 5, 10],
              'svc__gamma': [0.0005, 0.001, 0.005]}

# Apply Principal Component Analysis (PCA) for feature reduction
# Retain 90% of the variance, whiten the data, and set a random seed for reproducibility
pca = RandomizedPCA(n_components=0.9, whiten=True, random_state=42)

# Use balanced class weights to handle imbalanced datasets
svm = SVC(kernel='rbf', class_weight='balanced')
# Create a pipeline with PCA and SVM
model = make_pipeline(pca, svm)

# Set up a grid search for hyperparameter tuning
# Use cross-validation with 2 folds and print verbose information
grid = GridSearchCV(model, param_grid, cv=2, verbose=2)

# Measure the time it takes to perform grid search and model fitting
%time grid.fit(X_train, y_train)
```

Figure 8 (shows the Hyperparameter tuning and training pipeline for SVM model on Personal Key Indicators of Heart Disease Dataset)



```
X = dataset.drop(columns=['HeartDiseaseorAttack'])
y = dataset['HeartDiseaseorAttack']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating and Tuning the SVM Model with Grid Search
# Define a parameter grid for hyperparameter tuning

param_grid = {'svc__C': [1, 5, 10],
              'svc__gamma': [0.0005, 0.001, 0.005]}

# Apply Principal Component Analysis (PCA) for feature reduction
# Retain 95% of the variance, whiten the data, and set a random seed for reproducibility
pca = RandomizedPCA(n_components=0.95, whiten=True, random_state=42)
# Use balanced class weights to handle imbalanced datasets
svm = SVC(kernel='rbf', class_weight='balanced')

# Create a pipeline with PCA and SVM
model = make_pipeline(pca, svm)

# Set up a grid search for hyperparameter tuning
# Use cross-validation with 2 folds and print verbose information
grid = GridSearchCV(model, param_grid,cv=2, verbose=2)

# Measure the time it takes to perform grid search and model fitting
%time grid.fit(X_train, y_train)
```

Figure 9 (shows the Hyperparameter tuning and training pipeline for SVM model on Heart Disease Health Indicators Dataset)

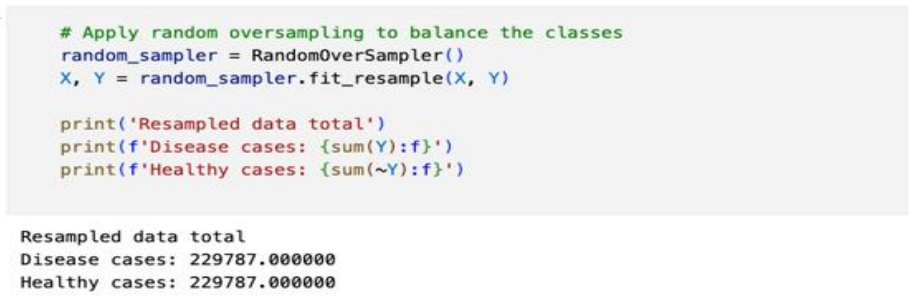
Following the training, evaluation metrics centred around accuracy, with the confusion matrix offering a visual depiction of the model's classification performance. The models' efficiency was showcased on standard CPU resources, and crucial code snippets illustrated key training steps. The overall approach not only provided a high-level understanding of the SVM architecture and its training but also delved into the specifics of dataset preprocessing, hyperparameter tuning, and model evaluation. The classification report, including metrics such as precision, recall, and F1-score, offered a detailed breakdown of the model's performance in different classes, providing insights into its strengths and areas for improvement.

Neural Networks (NN)

During the training phase, we identified an imbalance in both datasets. To address this issue, we employed Random Oversampling for both.

The model architecture is a simple feedforward neural network with the following components: Input layer: the input features are flattened using tf.keras.layers.Flatten. This layer is essential when the input data is not already flattened, as it transforms the input into a one-dimensional array. also have Hidden Layer: A dense (fully connected) hidden layer with 32 units and a ReLU activation function ('relu'). This layer is responsible for learning the complex patterns in the data. Moreover, the dropout layer: A dropout layer with a dropout rate of 0.2. Dropout is a regularization technique that randomly sets a fraction of input units to zero during training. It helps prevent overfitting by introducing noise and reducing reliance on specific neurons. finally Output Layer: The output layer has a single unit and a sigmoid activation function ('sigmoid'). This architecture suggests that you are working on a binary classification task, as the sigmoid activation outputs values between 0 and 1, suitable for binary decisions. and in terms of compilation: The model is compiled using the Adam optimizer with a specified learning rate (my_learning_rate). The loss function is set to 'binary_crossentropy', which is common for binary classification problems. The metric to monitor during training is accuracy.

The steps for training the model are: for "*Personal Key Indicators of Heart Disease Dataset*" First, we loaded and explored the dataset, the columns with missing values were dropped and then we explored the data type for every column to make sure it has only numeric values so we can apply the neural network in it. After converting the binary values to numeric applying one-hot encoding for the categorical values and normalizing the 'BMI' column, we apply random oversampling to balance the classes. Moreover in "*Heart Disease Health Indicators Dataset*": The first step is to load and explore the data to check if there is any missing or null value and information about the data and correlation between the features; after that, we have an overview of it; now the second step is the preprocessing part, which involves dropping the features we don't want, like "Income" and "Education", since there is a strong relationship between them, which may cause confusion for training. then we apply one-hot encoding for categorical features: "Age", "GenHlth", "Diabetes", and normalized numeric features with values other than 0-1 as MentHlth, PhysHlth, and BMI. Finally, save the preprocessed data set, and now for data splitting, this part separates the dataset into features (X) and the target variable (Y) that you want to predict. After checking the class distribution and figuring out the imbalanced data, apply random oversampling as shown in Figure 11 to balance the classes by generating synthetic samples for the minority class.



```
# Apply random oversampling to balance the classes
random_sampler = RandomOverSampler()
X, Y = random_sampler.fit_resample(X, Y)

print('Resampled data total')
print(f'Disease cases: {sum(Y):f}')
print(f'Healthy cases: {sum(~Y):f}')
```

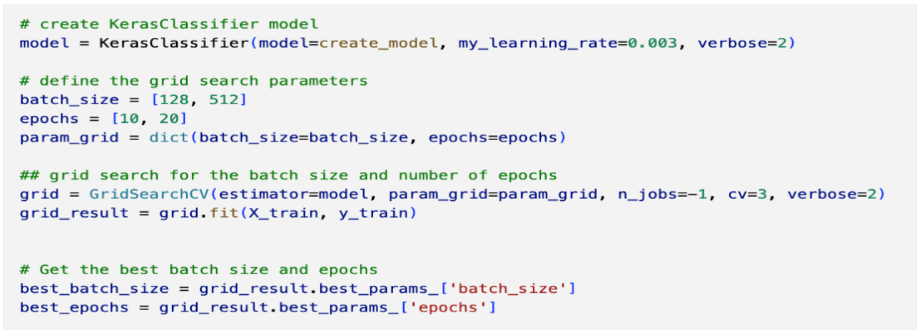
```
Resampled data total
Disease cases: 229787.000000
Healthy cases: 229787.000000
```

Figure 10 ("Heart Disease Health Indicators Dataset," Random Oversampling)

The next step is to split the dataset into 80% for the training set, 10% for the test set, and 10% for the validation set then check if the classes are still balanced. Then define all the functions that will be used for training and hyperparameter tuning. The hyperparameter Tuning will be done using a grid search to tune the batch size, number of epochs, number of neurons, and activation function in the hidden layers.

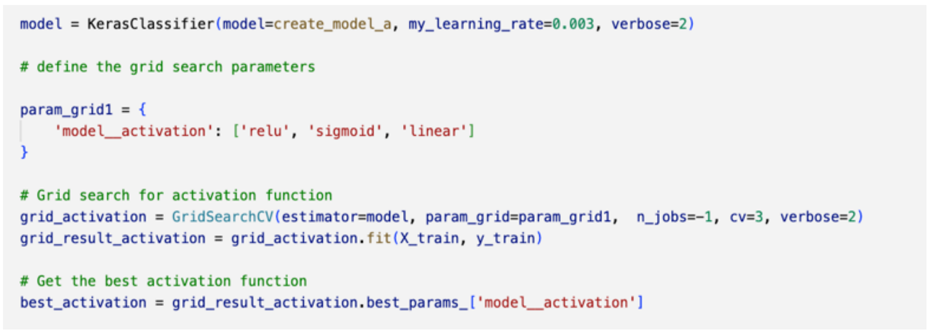
The range of values of batch size was: 128, 512 epochs number:
128, 512 as shown in Figure 11 and Figure 14; the number of neurons: 32, 64, 128 shown in Figure 12 and Figure 15; activation function: relu, sigmoid, linear shown in Figure 13 and Figure 16. Finally, after tuning this value using the grid search by training the model with these values, we found that the optimal value that gave the best results for heart disease Health indicators dataset was: batch size: 512; epochs: 20; number of neurons: 128; and activation function: relu and for Personal Key Indicators of Heart Diseasewas : batch size: 128; epochs: 20; number of neurons: 32; and activation function: relu.

In addition to the evaluation step, evaluate the performance of the trained model in multiple ways using evaluation metrics and evaluation using loss and accuracy for the model on the test set. Moreover, do a more detailed evaluation using a classification report, which includes precision, recall, F1-score, and support for each class in the test set. Lastly, it visualizes the confusion matrix, which helps in understanding the model's performance in different classes.



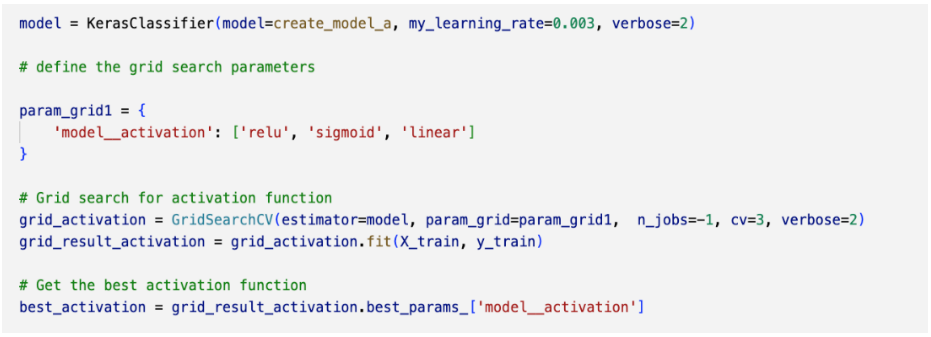
```
# create KerasClassifier model
model = KerasClassifier(model=create_model, my_learning_rate=0.003, verbose=2)

# define the grid search parameters
batch_size = [128, 512]
epochs = [10, 20]
param_grid = dict(batch_size=batch_size, epochs=epochs)

## grid search for the batch size and number of epochs
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3, verbose=2)
grid_result = grid.fit(X_train, y_train)


# Get the best batch size and epochs
best_batch_size = grid_result.best_params_['batch_size']
best_epochs = grid_result.best_params_['epochs']
```

Figure 11 ("Heart Disease Health Indicators Dataset", Tuning batch size and number of epochs using grid search)



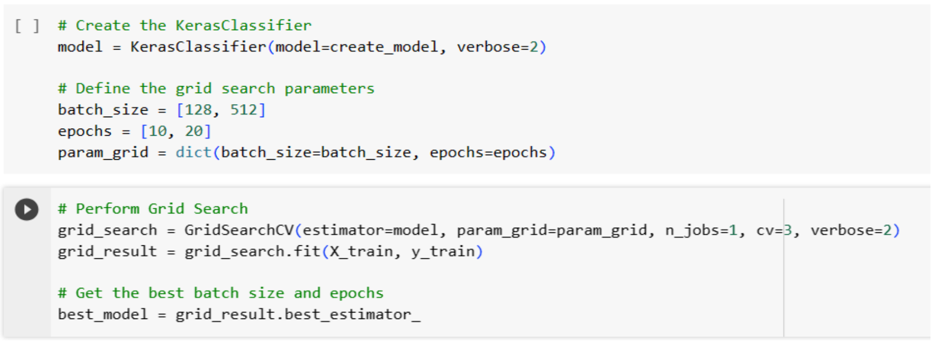
```python
model = KerasClassifier(model=create_model_a, my_learning_rate=0.003, verbose=2)

# define the grid search parameters

param_grid1 = {
    'model__activation': ['relu', 'sigmoid', 'linear']
}

# Grid search for activation function
grid_activation = GridSearchCV(estimator=model, param_grid=param_grid1,  n_jobs=-1, cv=3, verbose=2)
grid_result_activation = grid_activation.fit(X_train, y_train)

# Get the best activation function
best_activation = grid_result_activation.best_params_['model__activation']
```

Figure 12 ("Heart Disease Health Indicators Dataset", Tuning the number of neurons using grid search)



```python
model = KerasClassifier(model=create_model_a, my_learning_rate=0.003, verbose=2)

# define the grid search parameters

param_grid1 = {
    'model__activation': ['relu', 'sigmoid', 'linear']
}

# Grid search for activation function
grid_activation = GridSearchCV(estimator=model, param_grid=param_grid1,  n_jobs=-1, cv=3, verbose=2)
grid_result_activation = grid_activation.fit(X_train, y_train)

# Get the best activation function
best_activation = grid_result_activation.best_params_['model__activation']
```

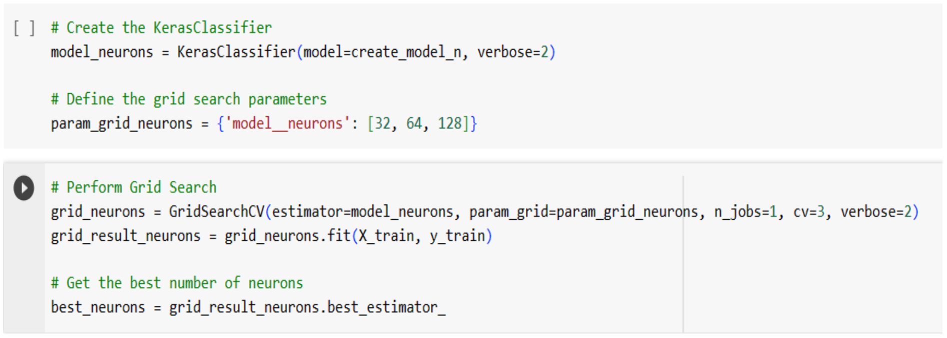
Figure 13 ("Heart Disease Health Indicators Dataset", Tuning activation function using grid search)



```python
# Create the KerasClassifier
model = KerasClassifier(model=create_model, verbose=2)

# Define the grid search parameters
batch_size = [128, 512]
epochs = [10, 20]
param_grid = dict(batch_size=batch_size, epochs=epochs)
```

```python
# Perform Grid Search
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=1, cv=3, verbose=2)
grid_result = grid_search.fit(X_train, y_train)

# Get the best batch size and epochs
best_model = grid_result.best_estimator_
```

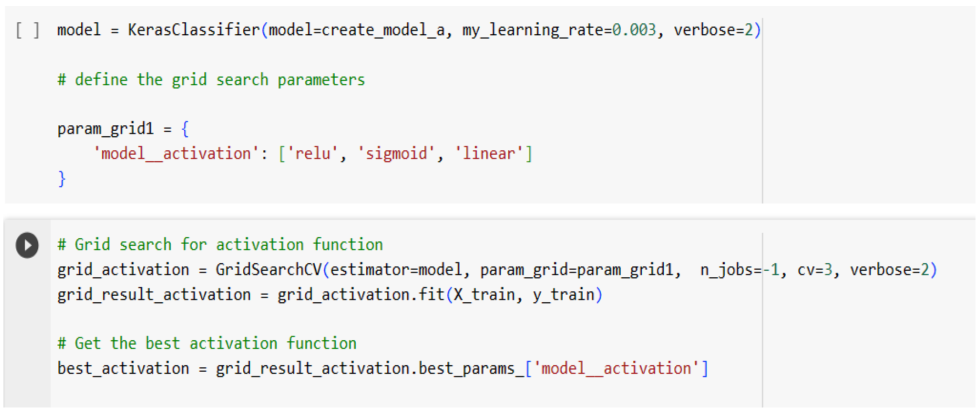
Figure 14 ("Personal Key Indicators of Heart Disease", Tuning batch size and number of epochs using grid search)



```python
# Create the KerasClassifier
model_neurons = KerasClassifier(model=create_model_n, verbose=2)

# Define the grid search parameters
param_grid_neurons = {'model__neurons': [32, 64, 128]}
```

```python
# Perform Grid Search
grid_neurons = GridSearchCV(estimator=model_neurons, param_grid=param_grid_neurons, n_jobs=1, cv=3, verbose=2)
grid_result_neurons = grid_neurons.fit(X_train, y_train)

# Get the best number of neurons
best_neurons = grid_result_neurons.best_estimator_
```

Figure 15 ("Personal Key Indicators of Heart Disease", Tuning the number of neurons using grid search)



```
[ ] model = KerasClassifier(model=create_model_a, my_learning_rate=0.003, verbose=2)

    # define the grid search parameters

    param_grid1 = {
        'model__activation': ['relu', 'sigmoid', 'linear']
    }
```

```
# Grid search for activation function
grid_activation = GridSearchCV(estimator=model, param_grid=param_grid1,  n_jobs=-1, cv=3, verbose=2)
grid_result_activation = grid_activation.fit(X_train, y_train)

# Get the best activation function
best_activation = grid_result_activation.best_params_['model__activation']
```

Figure 16 ("Personal Key Indicators of Heart Disease", Tuning activation function using grid search)

Following the training, evaluation metrics centred around accuracy, with the confusion matrix offering a visual depiction of the model's classification performance. The models' efficiency was showcased on standard CPU resources, and crucial code snippets illustrated key training steps. The overall approach not only provided a high-level understanding of the NN architecture and its training but also delved into the specifics of dataset preprocessing, hyperparameter tuning, and model evaluation. The classification report, including metrics such as precision, recall, and F1-score, offered a detailed breakdown of the model's performance in different classes, providing insights into its strengths and areas for improvement.

Results and Discussion

This section summarizes performance metrics of machine learning models, using tables for comparison and visualizations like confusion matrices. It analyzes training and validation performance across epochs, highlighting strengths and weaknesses.

Results and performance:

Table 2 (Personal Key Indicators of Heart Disease Dataset)

	Random Forest	SVM	Neural Network
Accuracy	81.79%	75.22%	77.11%
Recall	85.61%	75%	77 %
Precision	79.59%	90%	77 %
F1-score	82.49%	80%	77 %

Table 3 (Heart Disease Health Indicators Dataset)

	Random Forest	SVM	Neural Network
Accuracy	97.07%	70.61%	77.32%
Recall	99.80%	71%	77%
Precision	94.63%	90%	78%
F1-score	97.15%	77%	77%

Table 2 reveals the performance of Random Forest, SVM, and Neural Network models on the Personal Key Indicators of Heart Disease Dataset. Random Forest stands out with an 81.79% accuracy, excelling in recall at 85.61%. While SVM shows a high precision of 90%, its overall accuracy and recall trail behind Random Forest. The Neural Network, consistent but not exceptional, underscores Random Forest's effectiveness. For this dataset, Random Forest proves robust, offering high accuracy and crucially high recall for identifying potential heart disease cases.

In Table 3, Random Forest impresses with a 97.07% accuracy and exceptional 99.80% recall on the Heart Disease Health Indicators Dataset. Conversely, SVM struggles with a 70.61% accuracy, emphasizing its challenges in this context despite a notable precision of 90%. The Neural Network, falling between Random Forest and SVM, suggests a balanced performance. Consistent F1-scores across models highlight the need to carefully weigh precision and recall trade-offs based on specific goals. Here, Random Forest emerges as the top performer, showcasing its potential for accurately identifying heart disease cases from health indicators.

Visualizations:

- Classification Report:

1. Random Forest (RF):

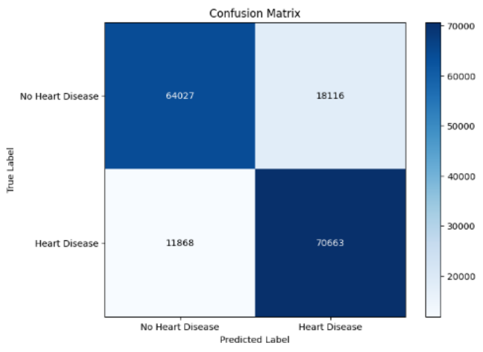


Figure 17 (Confusion Matrix for Personal Key Indicators of Heart Disease Dataset)

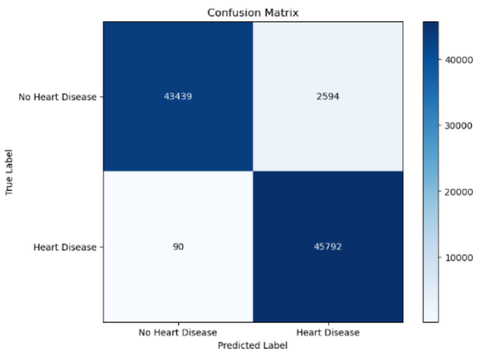


Figure 18 (Confusion Matrix for Heart Disease Health Indicators Dataset)

2. Neural Network (NN):

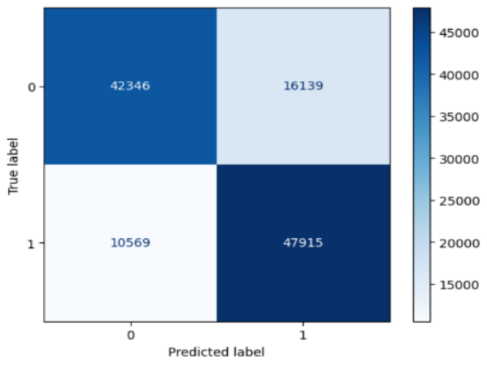


Figure 19 (Confusion Matrix for Personal Key Indicators of Heart Disease Dataset)

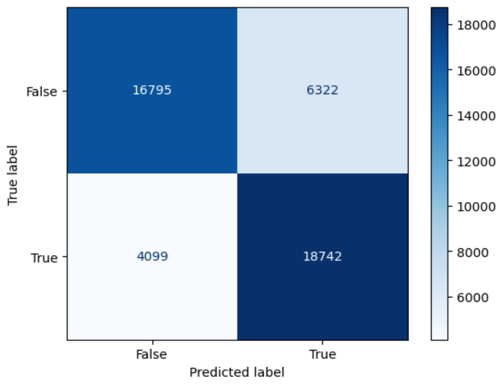


Figure 20 (Confusion Matrix for Heart Disease Health Indicators Dataset)

3. Support Vector Machines (SVM):

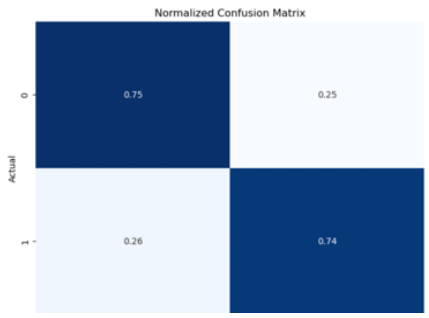


Figure 21 (Confusion Matrix for Personal Key Indicators of Heart Disease Dataset)

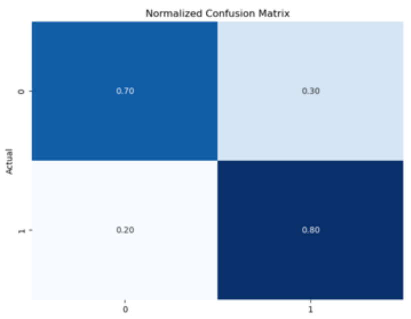


Figure 22 (Confusion Matrix for Heart Disease Health Indicators Dataset)

- ROC Curve:

1- Random Forest (RF):

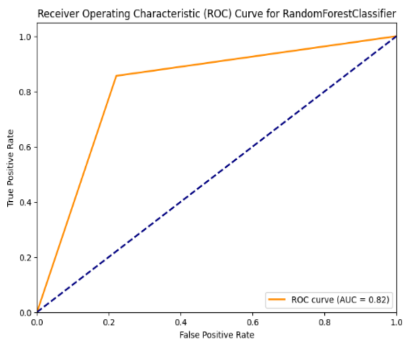


Figure 23 (ROC Curve for Personal Key Indicators of Heart Disease Dataset)

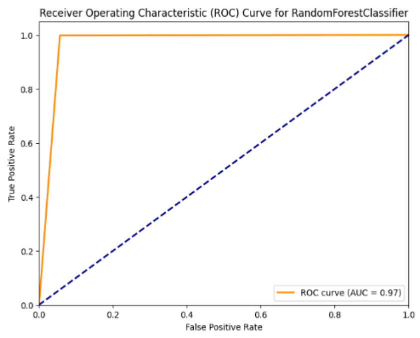


Figure 24 (ROC Curve for Heart Disease Health Indicators Dataset)

2- Neural Network (NN):

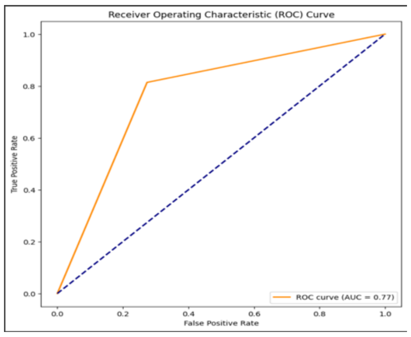


Figure 25 (ROC Curve for Personal Key Indicators of Heart Disease Dataset)

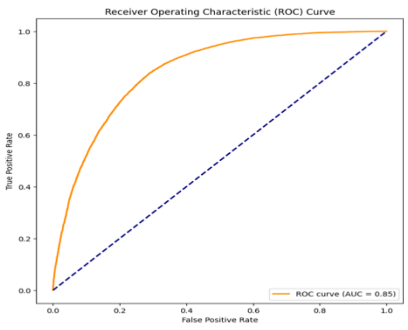


Figure 26 (ROC Curve for Heart Disease Health Indicators Dataset)

3- Support Vector Machines (SVM):

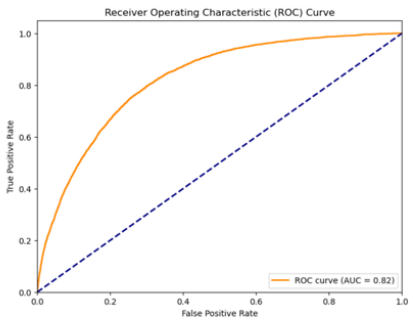
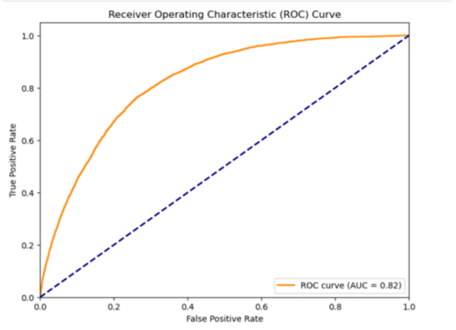


Figure 27 (ROC Curve for Personal Key Indicators of Heart Disease Dataset)



28 (Confusion Matrix for Heart Disease Health Indicators Dataset)

- <u>Loss plots:</u>

Neural Network (NN)

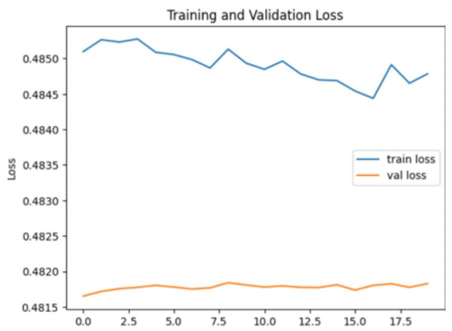


Figure 29 (Loss for Personal Key Indicators of Heart Disease Dataset)

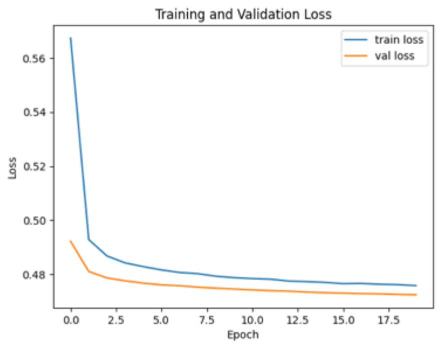


Figure 30 (Loss plot for Heart Disease Health Indicators Dataset)

- <u>Accuracy plots</u>

Neural Network (NN)

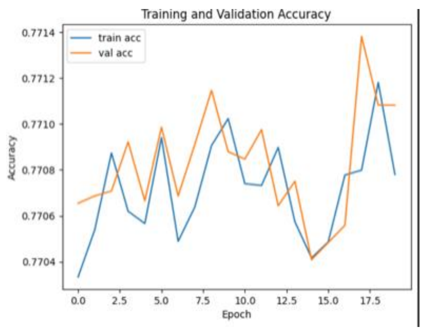


Figure 31 (Accuracy plot for Personal Key Indicators of Heart Disease Dataset)

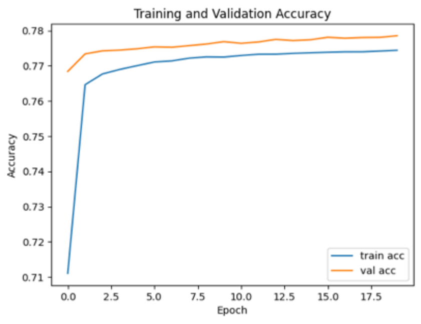


Figure 32 (Accuracy plot for Heart Disease Health Indicators Dataset)

As shown is Figure 29, Figure 30, Figure 31 ,and Figure 32, depicts the training and validation loss and accuracy over the number of epochs for a machine-learning model. Training loss steadily decreases and accuracy increases, indicating better fitting to training data. Validation loss also decreases accuracy increases, showing generalization to new data. The generalization gap, the difference between training and validation loss, is relatively small, suggesting good generalization capacity. This implies the model's readiness for real-world applications.

Conclusion

In summarizing the key findings and outcomes of the project, we observed notable variations in the performance of three different models: Random Forest, SVM, and Neural Network, across two distinct datasets. The Random Forest model consistently outperformed others, achieving the highest accuracy, recall, precision, and F1-score on both datasets. This suggests its robustness in handling the classification task related to heart disease indicators.

However, it's crucial to acknowledge certain challenges and limitations encountered during the project. All models faced difficulties in dealing with imbalanced classes, particularly regarding the minority class, leading to varied performance across metrics. SVM, despite exhibiting high precision, struggled with overall accuracy compared to Random Forest. Neural Network, while demonstrating competitive accuracy, encountered challenges in precision and recall.

Possible improvements and future directions for the project involve addressing class imbalance more effectively. Techniques such as undersampling, or employing advanced algorithms designed for imbalanced datasets could be explored. Additionally, the project could benefit from experimenting with deep learning architectures, such as Convolutional Neural Networks (CNNs), especially if there are spatial or hierarchical patterns within the data.

Contributions

Table 4 (Contributions)

Name	Role
Reem Almusharraf	SVM for Both Datasets
Deema Alresheed	NN for Heart Disease Health Indicators dataset
Bayan Albadri	NN for Personal Key Indicators dataset
Sarah K Juwied	RF for Both Datasets

References

[1] "Cardiovascular diseases," World Health Orgnization , [Online]. Available: https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1. [Accessed 15 9 2023].

[2] M. T. TUN, "binary logistic regression," 7 2025. [Online]. Available: https://www.kaggle.com/code/minthihatun/binary-logistic-regression/notebook. [Accessed 17 9 2023].

[3] A. TEBOUL, "Heart Disease Health Indicators Dataset," 2021. [Online]. Available: https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset. [Accessed 9 9 2023].

[4] N. SRIVASTAVA, "Heart Disease prediction using QML," [Online]. Available: https://www.kaggle.com/code/srivnaman/heart-disease-prediction-using-qml/notebook. [Accessed 17 9 2023].

[5] K. PYTLAK, "Indicators of Heart Disease (2022 UPDATE)," [Online]. Available: https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease?resource=download. [Accessed 9 9 2023].

[6] F. Y. MOHAMED, "Random Forest Handling Imbalanced with Acc=97%," 8 2023. [Online]. Available: https://www.kaggle.com/code/fatmayousufmohamed/random-forest-handling-imbalanced-with-acc-97. [Accessed 17 9 2023].

[7] Z. HAMADA, "Visualizing and predict heart disease (acc 98.5%)," 2 2023. [Online]. Available: https://www.kaggle.com/code/ziadhamadafathy/visualizing-and-predict-heart-disease-acc-98-5. [Accessed 17 9 2023].