Akshnoor Singh, 2232759
Ali Husain, 2277224
Anish Tottempudi, 1950018
Dinh Gam Phan, 2281486

Final Project

Introduction (Ali Husain, Anish Tottempudi):

Cardiovascular disease is a broad term which refers to a multitude of diseases that affect the heart. Cardiovascular diseases (CVDs), commonly referred to as "heart disease" are the leading cause in the United States. In 2022, over 700 thousand people were claimed by this affliction, accounting for roughly 1 in every 5 deaths in the nation. Every year in the United States, the total financial burden of cardiovascular diseases (cost of health care services, medicines, lost productivity due to death, etc.) is on the order of hundreds of billions of dollars (CDC.gov).

The question of interest in this project sets out to answer is: *How accurately can we predict the prevalence of heart disease?* To set out on this goal, this project utilizes a dataset provided from the website Kaggle. This dataset is a combination of 5 other data sets over 11 common features. These sets originate from Cleveland, Hungary, Switzerland, Long Beach VA, and another Heart dataset amounting to a total of 918 observations.

As stated, the data set has 918 observations, which in this case are individual patients tested for a variety of factors including whether they have heart disease. Each of these features could serve as possible components build into answers for the project's question. In the first column there is an age factor for each observation, the second column contains the gender of each observation, the third column contains different types of chest pain with 4 criteria Typical Angina, Atypical Angina, Non-Anginal Pain & Asymptomatic. Typical Angina is a type of pain derived from stress and physical activity, Atypical Angina is a type of chest pain that does not follow typical patterns of anginal pain, non-Anginal Pain has chest pain present but is not anginal and finally for Asymptomatic the patient has angina but no pain which can be particularly dangerous as the patient doesn't know of their condition. The fourth and fifth columns measure blood pressure and cholesterol respectively with the sixth column measuring the fasting blood sugar of the patient. The seventh column details the observations resting ECG results NV means heart activity is normal; ST means there is an abnormality with ST segment and t Wave and LVH means left ventricular hypertrophy which is indicates thickening of the hearts left ventricle. The eighth column displays the maximum heart rate achieved by the given observation, with the ninth being a column showing whether an observation as exercise-induced angina. The tenth observation Oldpeak is a measure of depression of ST with 0 or below no ST depression Mild 0.5 - 1 moderate to severe ST depression 1 and the eleventh observation indicates the slope, upsloping being ideal, flat less ideal and down sloping as bad. Lastly, the twelfth column provides whether the observation has heart disease or not.

Methodology (Anish Tottempudi, Dinh Gam Phan, Akshnoor Singh):

KNN:

We will be implementing two supervised machine learning models which aim to predict the prevalence of heart disease based on the 11 selected features. The two models we selected are K-nearest neighbors (KNN) and support vector machine (SVM).

We chose to use the KNN model mainly because it is conceptually straightforward and easy to implement while still being a highly accurate model for prediction. Some other advantages of the KNN model are that it makes no assumptions about the underlying data distribution and that it is flexible with a variety of distance metrics (Euclidean, Manhattan, Minkowski, etc.). Some disadvantages of KNN are that it is computationally expensive, it is limited in its ability to understand how the features are related to the output, and it is sensitive to outliers and scale.

For our KNN model, we used Euclidean distance as our distance metric. The formula is as follows:

$$d(x_{\text{test}}, x_i) = \sqrt{\sum_{j=1}^{p} \left(x_{test,j} - x_{i,j}\right)^2}.$$

SVM:

We will implement a support vector machine (SVM) model to predict the prevalence of heart disease based on the selected features. The SVM model was chosen because it is highly effective for both linear and non-linear classification problems. One major advantage of SVM is its ability to find the optimal decision boundary (or hyperplane) that maximizes the margin between different classes, which makes it a very strong classifier, especially for datasets with a clear margin of separation.

Another advantage of the SVM model is its flexibility, as it can be used with different kernel functions (linear, polynomial, radial basis function, etc.) to handle various types of data distributions. Some disadvantages of SVM include its sensitivity to the choice of hyperparameters and its computational cost for large datasets.

Linear Kernel Equation:

$$K(x_i, x_j) = \sum_{k=1}^{p} x_{ik} x_{jk}$$

Polynomial Kernel Equation:

$$K(x_i, x_j) = (1 + \langle x_i, x_j \rangle)^d = (1 + \sum_{k=1}^{p} x_{ik} x_{jk})^d$$

Radial Kernel Equation:

$$K(x_i, x_{i`}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i`j}\right)^2\right), \gamma > 0$$

Data Analysis (Ali Husain, Akshnoor Singh, Dinh Gam Phan)

SVM:

Before using Support Vector Machines (SVM), we performed preprocessing on the data. First, we encoded any categorical variables in our dataset. This step is crucial because SVM can only work with numerical data. We used one-hot encoding and label encoding, depending on the nature of the categorical variables. Just like we did when working with K-Nearest Neighbors (KNN), we handled outlier Cholesterol in our dataset by replacing empty values with the mean. Next, we scaled any numerical values in our dataset. This is important because SVM is sensitive to the scale of the data. If numerical variables are on different scales, it can affect the performance of the model. We typically use techniques like Min-Max scaling or Standardization (Z-score normalization) to ensure all features contribute equally. After scaling, we randomly split our dataset into training and testing sets in an 80/20 ratio.

```
# Split the data into training (80%) and test (20%) sets
set.seed(123)
split_index <- sample(1:nrow(heart), 0.8 * nrow(heart))
# Create training set
train_data <- heart[split_index, ]

# Create testing set
test_data <- heart[-split_index, ]
```

Now that both sets are scaled and ready, we must find the optimal cost parameter for our SVM model, we use a tuning function. To optimize the SVM model using different kernels, we utilized the tune function. We experimented with different kernel types—linear, polynomial, and radial basis function (RBF)—to see which one gives us the best results:

```
tune_out <- tune(svm,
                 train.x = train_x,
                 train.y = as.factor(train_y),
                 ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100)),
                 kernel = "linear")

tune_poly <- tune(svm,
                  train.x = train_x,
                  train.y = as.factor(train_y),
                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                                degree = c(2, 3, 4)),
                  kernel = "polynomial")

tune_radial <- tune(svm,
                    train.x = train_x,
                    train.y = as.factor(train_y),
                    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                                  gamma = c(0.001, 0.01, 0.1, 1)),
                    kernel = "radial")
```

Results from the SVM process:

Linear:

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
    1      3

- best performance: 0.1511847

- Detailed performance results:
    cost degree     error dispersion
1  1e-03      2 0.4442058 0.05335016
2  1e-02      2 0.4442058 0.05335016
3  1e-01      2 0.3378563 0.04222287
4  1e+00      2 0.2029804 0.06129906
5  5e+00      2 0.2099223 0.04233558
6  1e+01      2 0.2099408 0.04484014
7  1e+02      2 0.2003887 0.04280028
8  1e-03      3 0.4442058 0.05335016
9  1e-02      3 0.3720104 0.05466701
10 1e-01      3 0.1662532 0.03400535
11 1e+00      3 0.1511847 0.04929767
12 5e+00      3 0.1716031 0.04475486
13 1e+01      3 0.1879674 0.04465914
14 1e+02      3 0.2287486 0.04223843
15 1e-03      4 0.4442058 0.05335016
16 1e-02      4 0.4387264 0.05212778
17 1e-01      4 0.3188449 0.04630171
18 1e+00      4 0.1785265 0.05608074
19 5e+00      4 0.1934654 0.04211527
20 1e+01      4 0.2043873 0.03979079
21 1e+02      4 0.2452980 0.04001912
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1403924

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.1662718 0.03880820
2 1e-02 0.1417438 0.03616566
3 1e-01 0.1403924 0.04246939
4 1e+00 0.1417438 0.04249851
5 5e+00 0.1417253 0.04570279
6 1e+01 0.1417253 0.04570279
7 1e+02 0.1417253 0.04570279

>
> best_model <- tune_out$best.model
>
> svm_best_pred <- predict(best_model, test_x)
>
> # Calculate test error
> test_error <- mean(svm_best_pred != test_y)
>
> cat("Test Error:", test_error, "\n")
Test Error: 0.1684783
>
> conf_matrix_tuned <- table(Predicted = svm_best_pred, Actual = test_y)
> print("Confusion Matrix:")
[1] "Confusion Matrix:"
> print(conf_matrix_tuned)
         Actual
Predicted  0  1
        0 69 16
        1 15 84
>
```

Polynomial:

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
    1      3

- best performance: 0.1511847

- Detailed performance results:
    cost degree     error dispersion
1  1e-03      2 0.4442058 0.05335016
2  1e-02      2 0.4442058 0.05335016
3  1e-01      2 0.3378563 0.04222287
4  1e+00      2 0.2029804 0.06129906
5  5e+00      2 0.2099223 0.04233558
6  1e+01      2 0.2099408 0.04484014
7  1e+02      2 0.2003887 0.04280028
8  1e-03      3 0.4442058 0.05335016
9  1e-02      3 0.3720104 0.05466701
10 1e-01      3 0.1662532 0.03400535
11 1e+00      3 0.1511847 0.04929767
12 5e+00      3 0.1716031 0.04475486
13 1e+01      3 0.1879674 0.04465914
14 1e+02      3 0.2287486 0.04223843
15 1e-03      4 0.4442058 0.05335016
16 1e-02      4 0.4387264 0.05212778
17 1e-01      4 0.3188449 0.04630171
18 1e+00      4 0.1785265 0.05608074
19 5e+00      4 0.1934654 0.04211527
20 1e+01      4 0.2043873 0.03979079
21 1e+02      4 0.2452980 0.04001912
```

```
>
> # Get the best model from tuning
> best_poly_model <- tune_poly$best.model
>
> # Make predictions on the test set
> svm_poly_pred <- predict(best_poly_model, test_x)
>
> # Calculate test error for polynomial kernel
> test_error_poly <- mean(svm_poly_pred != test_y)
>
> cat("Test Error (Polynomial Kernel):", test_error_poly, "\n")
Test Error (Polynomial Kernel): 0.1576087
>
> # Confusion matrix for polynomial kernel
> conf_matrix_poly <- table(Predicted = svm_poly_pred, Actual = test_y)
> print("Confusion Matrix (Polynomial Kernel):")
[1] "Confusion Matrix (Polynomial Kernel):"
> print(conf_matrix_poly)
         Actual
Predicted  0  1
        0 69 14
        1 15 86
>
> # Calculate accuracy for polynomial kernel
> accuracy_poly <- sum(diag(conf_matrix_poly)) / sum(conf_matrix_poly)
> cat("\nAccuracy (Polynomial Kernel):", round(accuracy_poly, 4), "\n")

Accuracy (Polynomial Kernel): 0.8424
```

Radial:

```
- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10 0.001

- best performance: 0.1376527

- Detailed performance results:
    cost gamma     error dispersion
1  1e-03 0.001 0.4442058 0.05335016
2  1e-02 0.001 0.4442058 0.05335016
3  1e-01 0.001 0.4442058 0.05335016
4  1e+00 0.001 0.1594224 0.04037031
5  5e+00 0.001 0.1417438 0.03616566
6  1e+01 0.001 0.1376527 0.04058332
7  1e+02 0.001 0.1389856 0.05019508
8  1e-03 0.010 0.4442058 0.05335016
9  1e-02 0.010 0.4442058 0.05335016
10 1e-01 0.010 0.1594224 0.04088352
11 1e+00 0.010 0.1417808 0.04494167
12 5e+00 0.010 0.1457979 0.05028530
13 1e+01 0.010 0.1485191 0.04383049
14 1e+02 0.010 0.1525176 0.04637067
15 1e-03 0.100 0.4442058 0.05335016
16 1e-02 0.100 0.4442058 0.05335016
17 1e-01 0.100 0.1417808 0.03956672
18 1e+00 0.100 0.1389485 0.03375188
19 5e+00 0.100 0.1553128 0.04230184
20 1e+01 0.100 0.1662162 0.04014916
21 1e+02 0.100 0.2085154 0.03675022
22 1e-03 1.000 0.4442058 0.05335016
23 1e-02 1.000 0.4442058 0.05335016
24 1e-01 1.000 0.4442058 0.05335016
25 1e+00 1.000 0.2166975 0.06013509
26 5e+00 1.000 0.2003517 0.05529870
27 1e+01 1.000 0.2003517 0.05529870
28 1e+02 1.000 0.2003517 0.05529870
```

```
> test_error_radial <- mean(svm_radial_pred != test_y)
>
> cat("Test Error (Radial Kernel):", test_error_radial, "\n")
Test Error (Radial Kernel): 0.173913
>
> # Confusion matrix for radial kernel
> conf_matrix_radial <- table(Predicted = svm_radial_pred, Actual = test_y)
> print("Confusion Matrix (Radial Kernel):")
[1] "Confusion Matrix (Radial Kernel):"
> print(conf_matrix_radial)
         Actual
Predicted  0  1
        0 69 17
        1 15 83
>
> # Calculate accuracy for radial kernel
> accuracy_radial <- sum(diag(conf_matrix_radial)) / sum(conf_matrix_radial)
> cat("\nAccuracy (Radial Kernel):", round(accuracy_radial, 4), "\n")

Accuracy (Radial Kernel): 0.8261
>
```

Here is a breakdown of the findings for each kernel type. For the linear kernel, we conducted a grid search to find the optimal cost parameter. The results indicated that the best cost value was C = 0.1, which yielded an accuracy of 83.15%. This suggests that a moderate penalty for misclassification is effective for this dataset. Next, we applied the polynomial kernel. During tuning, we varied both the cost and degree of the polynomial. The optimal parameters turned out to be C = 1 and degree = 3, resulting in an accuracy of 84.24%. This improvement over the linear kernel indicates that the polynomial kernel can capture more complex relationships in the data. Finally, we explored the radial (RBF) kernel. The tuning process revealed that the best parameters were C = 10 and gamma = 0.001, achieving an accuracy of 82.61%. This kernel performed the worst among all tested. in conclusion, while all kernels provided reasonable accuracy, the polynomial kernel outperformed both linear and radial kernels in this case.

KNN:

When beginning our K-Nearest Neighbors (KNN) analysis, we carefully examined and considered all the variables in the dataset. The dataset includes six categorical variables (Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope, FastingBS) and five quantitative variables (Age, RestingBP, Cholesterol, MaxHR, Oldpeak). We noticed that some of the observations had a Cholesterol value of 0 which is scientifically impossible in a living human being. We considered removing this variable, but we realized that cholesterol is an essential

attribute for predicting heart disease. To solve this dilemma, we decided to assign the empty with the mean value of Cholesterol.

We took various steps to fine-tune our KNN model. To prepare the data for analysis, we converted all categorical variables into binary values using one-hot encoding to prevent assumptions of ordinality. Additionally, we scaled the variables to avoid larger values such as Cholesterol or MaxHR, from dominating the distance-based calculations in KNN. This preprocessing ensured that all of the variables contributed to the analysis accurately. Finally, we implemented K-means clustering into our model,

Before proceeding we randomly split the dataset, allocating 80% of the data for training and reserved the remaining 20% as a test set. To find the best way to split our data, we initially tried used K-fold cross-validation:

```
train_control <- trainControl(
  method = "cv",
  number = 5,      # k = 5 folds (80/20 split)
  savePredictions = "final"  # Save predictions for later use
)

# Run k-fold CV with KNN
heart_scaled$HeartDisease <- as.factor(heart_scaled$HeartDisease)
knn_fit <- train(
  HeartDisease ~ .,
  data = heart_scaled,
  method = "knn",
  trControl = train_control,
  tuneGrid = expand.grid(k = c(1, 3, 10, 13, 15)) # k values to test
)
```

However, as it turns out, the model was able to predict more accurately when we randomly sampled as shown below:

```
train_idx <- sample(1:nrow(heart_scaled), size = 0.8 * nrow(heart_scaled))
train_data <- heart_scaled[train_idx, ]
test_data <- heart_scaled[-train_idx, ]
```

After we determined the split, we ran KNN on the training and testing data and obtained the following results:

```
> # Run KNN
> k <- 17
> predicted <- knn(train = train_X,
+                  test = test_X,
+                  cl = train_y, k = k)
>
> # Evaluate Model Performance
> accuracy <- mean(predicted == test_y)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.869565217391304"
>
> # Confusion Matrix
> print(table(Predicted = predicted, Actual = test_y))
         Actual
Predicted  0  1
        0 69  7
        1 17 91
```

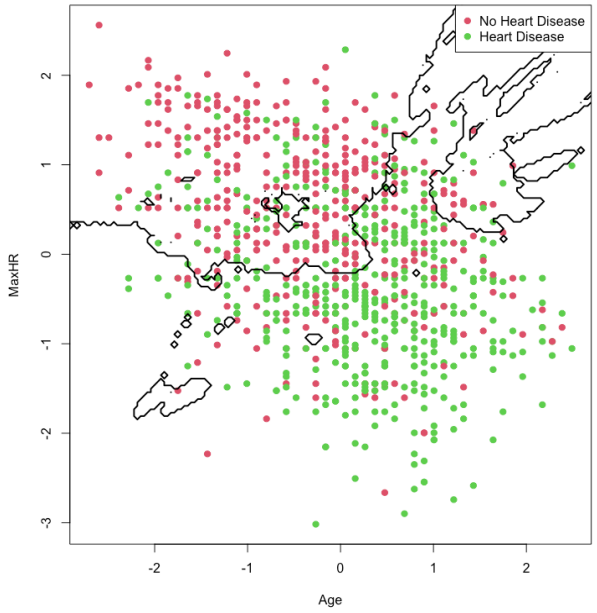
Comparative Analysis:

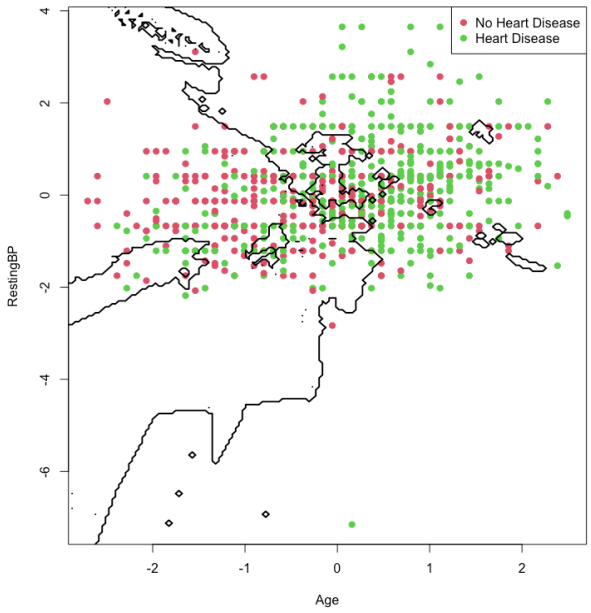
In our project, we compared the performance of K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) for predicting the prevalence of heart disease. After thorough processing and analysis, we found that KNN outperformed SVM in terms of both accuracy and ease of implementation. KNN achieved higher accuracy with a test error of with simpler preprocessing steps and less computational cost compared to the more complex SVM models.

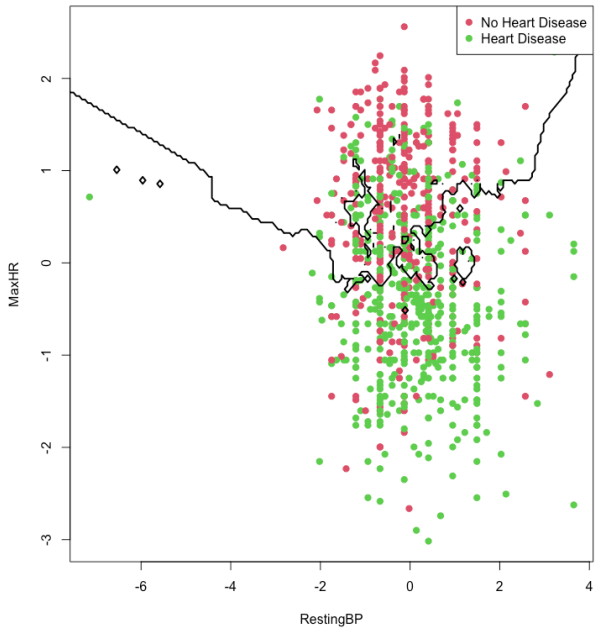
The output below showcases the KNN function ran on the entire data set. As shown in the output, the model showcased an accuracy of 0.8649 very similar to the original one.

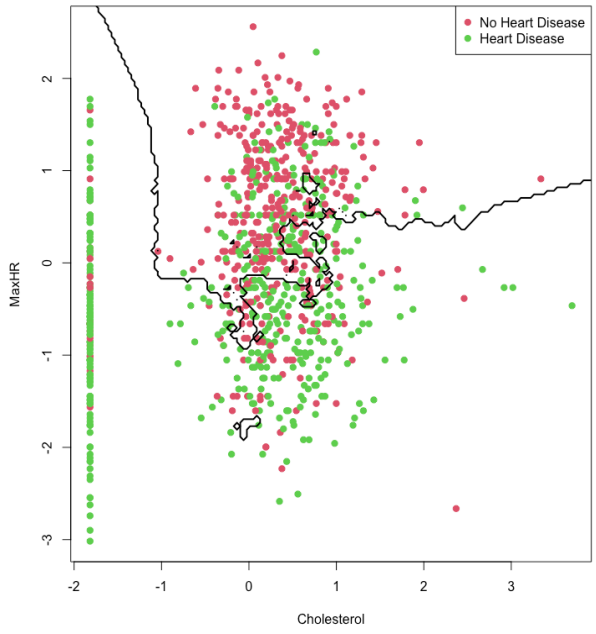
```
> # Run KNN on the entire dataset with the best k value
> k <- 17
> predicted <- knn(
+   train = predictors,
+   test = predictors,
+   cl = response,
+   k = k
+ )
>
> # Evaluate Model Performance
> accuracy <- mean(predicted == response)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.864923747276688"
>
> # Confusion Matrix
> print(table(Predicted = predicted, Actual = response))
         Actual
Predicted   0   1
        0 350  64
        1  60 444
```

The graphs below showcase KNN on two features that we decided were the most impactful in predicting heart disease. These pairs are: Max heart rate & age, Resting blood pressure & age, Max heart rate & Resting Blood Pressure and, Max heart rate & cholesterol.


No Heart Disease
Heart Disease
MaxHR
Age


No Heart Disease
Heart Disease
RestingBP
Age


No Heart Disease
Heart Disease
MaxHR
RestingBP


No Heart Disease
Heart Disease
MaxHR
Cholesterol

Conclusion (Ali Husain, Anish Tottempudi, Dinh Gam Phan, Akshnoor Singh):

How accurately can we predict the prevalence of heart disease? Our findings show that machine learning models such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) can effectively predict heart disease with reasonable accuracy among these two strategies KNN proved to be the better-performing for this dataset, achieving a high level of accuracy while being computationally efficient and straightforward to implement.

The data analysis posed several challenges. One significant difficulty was dealing with missing or incorrect data points, such as cholesterol values of zero, which are biologically impossible. To address these issues, we replaced missing or invalid values with the mean, but this approach might have introduced bias. Another challenge was ensuring that categorical features like chest pain type and resting ECG results were encoded properly without losing critical information about their relationships.

Scaling the data for distance-based calculations in KNN was also crucial but introduced its own challenges in ensuring consistency between training and testing sets. For SVM tuning hyperparameters like cost, degree, and kernel type required substantial computational time and expertise, which highlighted its complexity compared to KNN. To improve the data analysis, we could consider studies with more data, less missing data, and trying different results in which we remove all sections with invalid entries like the zeroes for cholesterol values. In the end, we decided to move forward with KNN thanks to its heightened accuracy in predicting this disease. We hope to use a model like this to predict more conditions to those who may not know or even aware of their case.

References

"Heart Disease Facts." *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, 24 Oct. 2024, http://www.cdc.gov/heart-disease/data-research/facts-stats/.

"Cardiovascular Diseases (Cvds)." *World Health Organization*, World Health Organization, 11 June 2021, www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-%28cvds%29.

"2024 Heart Disease and Stroke Statistics Update Fact Sheet." *American Heart Association*, www.heart.org/-/media/PHD-Files-2/Science-News/2/2024-Heart-and-Stroke-Stat-Update/2024-Statistics-At-A-Glance-final_2024.pdf. Accessed 22 Nov. 2024.

Fedesoriano. "Heart Failure Prediction Dataset." *Kaggle*, 10 Sept. 2021, www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data.