Predicting Cardiovascular Disease Using Machine Learning Algorithms

Alina Tariq
Department of Computer Science
Ryerson University
Toronto, Canada
alina.tariq@ryerson.ca

Anthony Greco
Department of Computer Science
Ryerson University
Toronto, Canada
anthony1.greco@ryerson.ca

Maryam Nada
Department of Computer Science
Ryerson University
Toronto, Canada
mnada@ryerson.ca

I. Introduction

A. Problem

According to the World Health Organization (WHO), cardiovascular diseases (CVDs), which refers to a group of heart and blood vessel-related disorders, are the leading cause of death worldwide, accounting for almost 18 million deaths in 2019 [15]. In Canada, CVDs are the second leading cause of death, behind cancer, and 1 out of 12 Canadians age 20 or above are diagnosed with heart disease and have a three times higher death rate compared to adults who do not have CVDs [2].

B. Significance and Challenge

With CVDs being so prevalent, this is a significant problem that affects a large subset of the world's population, their quality of life, and their life expectancy. Consequently, the sheer number of heart disease cases also place an undue burden on the public health system in Canada and other countries with free healthcare. In nations without free healthcare, seeking both a diagnosis and then treatment can in turn cause significant hardships for individuals who are unable to afford care.

Fortunately, heart diseases are considered to be both highly preventable (with estimates as high as 90%) and treatable provided you either follow a healthy lifestyle or receive an early diagnosis [3]. There's also several identifiable risk factors that can indicate an increased chance of developing cardiovascular disease including but not limited to high blood pressure, cholesterol levels, smoking, diabetes, inactivity, obesity, family history, gender and diet [4].

Although we cannot control people to take up healthier habits, we can still aim to contribute a solution to the problem by building models that can predict whether a person has cardiovascular disease using information about their health and risk factors. This can help mitigate some stress and financial cost placed upon the healthcare system and potential patients.

This task poses a few challenges, however. While there is information available about the different risk factors related to CVDs, not much is known about how these factors interact and compare to one another. Additionally, various tests need to be performed to be able to quantify a lot of these risk factors, which leads to a lack of data. Lastly, CVDs don't present homogeneously; rather, they affect individuals in a variety of ways with different risk factors and symptoms which can make it even more challenging to accurately predict using data [5].

C. Prior Work

Several studies have previously looked into this problem to attempt to build either single or ensemble (combined) models to help accurately predict heart disease. Dwivedi in [6] used a logistic regression algorithm and achieved an accuracy of 85%. Pouriyeh et al. [7], on the other hand, employed a stacking technique which combined Support Vector Machine and MLP Classifier models to obtain an accuracy of 84.15%.

A few novel studies have also used hybrid models to attempt to predict heart disease with the authors in [8] achieving an accuracy of 86.68% using a hybrid neural network consisting of a fuzzy neural network and artificial neural network (ANN) while the authors in [9] obtained an accuracy of 88.7% using a hybrid random forest with a linear model.

D. Overview

In our report, we will be further explaining the problem our project was focused upon along with details about our dataset and how it was preprocessed. Next, we will discuss the models we implemented, the techniques we employed to improve them, and the metrics we used to score them. Lastly, we will present, discuss, and interpret our results.

II. Problem Statement and Dataset

As mentioned previously, due to the prevalence of cardiovascular diseases worldwide and how it negatively impacts individuals' life expectancy, the goal of this project is to develop and train a machine learning model to accurately predict if a person has heart disease. In essence, we want to be able to collect basic demographic and health information about an individual and use it to correctly identify if a person has heart disease or not.

This type of problem is known as a binary classification problem in machine learning, wherein we want to use input date about different features (i.e. demographic/health

information) to be able to map it to one of two labels (i.e. “heart disease” or “no heart disease”).

A. Dataset

The “Heart Prediction Failure Dataset” used in this paper was obtained from Kaggle [10] and is a collection of five different independently available datasets [11, 12]: Cleveland, Hungarian, Switzerland, Long Beach VA, and Statlog (Heart). In total, the dataset consists of 918 observations over 11 common features — age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, max heart rate, exercise angina, old peak, and st slope. The features data, therefore, have a dimension of 918x11, while the labels data have a dimension of 918x1. The dataset is also not balanced with slightly more heart disease samples than no heart disease samples (Fig. 1).

III. Methods and Models

A. Preprocessing

The dataset was first examined for missing values including NaN and infinity values to determine if any samples needed to be removed. Next, the non-numerical features were examined and the binomial variables (i.e. sex and exercise angina) were converted to numerical values using an ordinal encoder while the multiclass variables (chest pain type, resting ECG, and st slope) were converted to a one-hot numerical array. A Spearman Correlation map (Fig. 2) was then generated using the preprocessed data to ensure there were no issues with multicollinearity which can affect the performance of classification models.

As the standard starter classification models, Logistic Regression and Naïve Bayes were implemented first, followed by k-Nearest Neighbour, Support Vector Machines, Decision Tree, Random Forest, and Artificial Neural Network, all of which are popular classification choices. The data was split into 7:3 train-test data, normalized or standardized depending on the model, and then used to train and evaluate baseline models of each algorithm using default parameters.

Instead of a manual training-validation split, k-fold cross validation with k=5 (4:1 training-validation split) was employed. K-fold cross validation works by splitting the training data into *k*-folds. It then uses $k - 1$ folds as the training set and the last fold as a validation test set, and then repeats the process until each fold has been used as the test set. This reduces bias in the model and also helps prevent overfitting [13].

B. Hyperparameter Optimization and Feature Selection

Once a baseline model had been evaluated, each algorithm

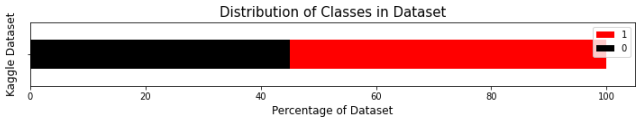

Fig. 1. *Distribution of classes in Kaggle dataset*

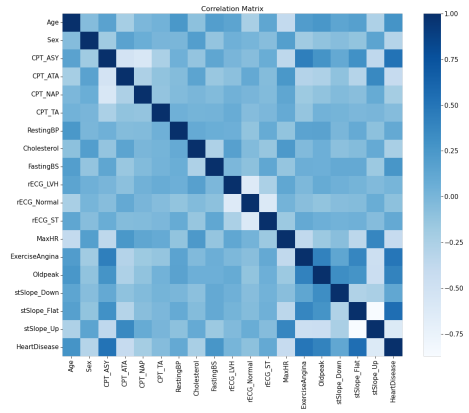

Fig. 2. *Spearman’s correlation heatmap of all the features in the dataset*

was tuned using hyperparameter optimization and feature selection. In hyperparameter optimization, multiple parameters for each model are tuned to find the combination of parameters that prevent overfitting and result in the best performance. In feature selection, different percentiles of features are used to train, cross-validate, and test the model to find the percentile of features that produce the highest score.

C. Bootstrap Aggregating (Bagging) and AdaBoost

Bagging is an ensemble meta-algorithm that is used to improve the accuracy and reduce the variance of machine learning classifiers. It is mainly used for unstable algorithms such as neural networks and decision trees. [14]

AdaBoost is a meta-algorithm that is used to improve the performance of other machine learning algorithms. It is mainly used to convert a weak algorithm into a strong one. AdaBoost adjusts the weights of the weak algorithms’ incorrect predictions to focus on difficult outliers in the data.

D. Logistic Regression and Naive Bayes

Logistic regression is a statistical algorithm that predicts the probability of the target variable given previous data. The target variable can only consist of binary values (two classes) which limits the number of datasets it could be applied on. Multinomial logistic regression expands logistic regression to multiclass problems (more than two classes). It is often used as a benchmark for other machine learning algorithms due to its simplicity and ease of implementation.

The Naive Bayes classifier uses Bayes's Theoreom to classify data. It assumes that each of the features in the dataset are independent. This may negatively affect the classifier's accuracy if some features are dependent on each other.

E. K-Nearest Neighbour (KNN)

KNN classifies data by comparing the distance from it to it's K closest neighbors. A "majority voting" system is often used where K neighbours class majority is predicted for the inputted data.

F. Support Vector Machine (SVM)

The SVM classifier finds a hyperplane that divides a dataset into its corresponding classes. It does this by finding

support vectors which are unique points in the dataset that influence how the hyperplane is generated.

G. Decision Tree and Random Forest

A decision tree classifier is a tree-like structure that consists of many nodes. Each internal node acts as some form of test on the combined features where the branch is the result of each test. The leaf nodes represent the predicted label for the features given and are only reached after every subsequent test. Decision trees are prone to overfitting which can be fixed by limiting the size of branches or by using a random forest classifier.

A random forest classifier is an ensemble learning method that combines the results of several distinct decision trees to minimize overfitting. It often outperforms decision tree classifiers due to the aggregation of predictions rather than depending on a single model.

H. Artificial Neural Network (ANN)

ANN classifiers mimic biological neural networks by using artificial neurons that have weights and transmit signals based on input signals (features). Learning is done by adjusting the weights of every artificial neuron. Signals travel through each layer of neurons where they end at an output layer where a prediction is made. ANN's are capable of modelling extremely complex data but are slower at learning compared to other algorithms.

I. Evaluation Metrics

Multiple evaluation metrics — precision, recall, F1, and accuracy — were used to score the performance of each model. These were all chosen, because they're all excellent for binary classification problems. However, accuracy was used as the primary evaluator for comparing performance between different models.

IV. Results and Discussions

We were able to train, cross-validate, and test our data using each of the seven aforementioned models with relatively successful results (Fig. 3). Using accuracy as our primary indicator of performance, we were able to achieve the highest results with a logistic regression model and our lowest with a decision tree classifier.

Bias is an error a classifier makes by making incorrect assumptions of relationships within the data. We want to minimize bias to increase the accuracy of predicting the labels from the features. In the learning curve plots, bias is visualized by the y-axis. In other words, the higher a point on the plot reaches, the less bias there is.

Variance is a sensitivity error to fluctuations in the data. We want to minimize variance to avoid modelling the random noise in the dataset and avoid overfitting. Variance is visualized by the light green and red shading in the learning curve plots. The greater their width, the greater the variance.

A. Logistic Regression

The logistic regression model had the best performance of all our models after extensive parameter tuning with an accuracy of 86.74% (Table I). This would point to the possibility that the data we have is linearly separable, but due to our implementation of feature selection, it is also possible that the set of features that produced the optimal results consisted of input data that was linearly separable.

If the linear separable property does ring true for which there is some prior evidence [6], however, then our results are in line with what is expected. From a fundamental failure analysis perspective, logistic regression isn't likely to overfit, has flexibility with fitting due to the regularization parameter, and still works well when there's a lack of data [16].

TABLE I. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Logistic Regression	87.35	87.11	86.74

To see if the model could be further improved, Bagging and AdaBoost classifiers were applied to the optimized models. The learning curves for the default, bagging, and AdaBoost models can be seen in Figure 3.

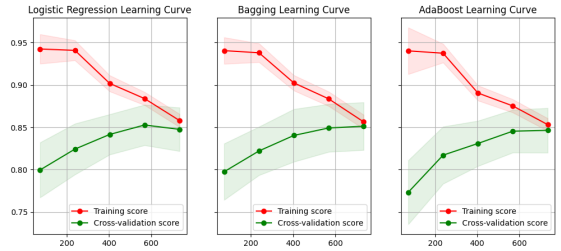

Fig. 3. *Logistic Regression Learning Curves*

We tried to reduce the variance and bias using bagging and AdaBoost. Bagging reduced the bias at a cost of higher variance. AdaBoost reduced the variance at a cost of higher bias. Therefore all classifiers are optimal as they each have their unique bias-variance trade offs.

B. Naïve Bayes

Naïve Bayes surprisingly performed only slightly worse compared to our Logistic Regression classifier with an accuracy of 85.47% (Table II). This is somewhat surprising since Naïve Bayes tends to assume that features are unrelated to each other. However, it is a model that is resilient to noise, which could potentially be the reason it performed so well [17]. The learning curve of the Naives Bayes algorithm can be seen in Figure 4. For the meta algorithms, Bagging increased variance and AdaBoost did not work on the classifier so these plots were not included.

TABLE II. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Naïve Bayes	86.80	86.71	85.47

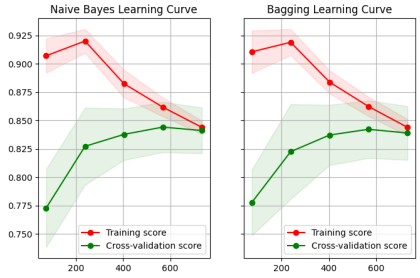

Fig. 4. *Naïve Bayes Learning Curve*

C. *k-Nearest Neighbour*

K-Nearest Neighbour (KNN) was also a model that performed well with the optimized model resulting in a test accuracy score of 86.67% (Table III). This is somewhat surprising, given that KNN is ideal for non-linear situations and the only model that did better was one that works well for linear situations. It also does not make assumptions about the dataset, tends to give equal importance to each feature, and can be susceptible to noise which can cause it to overfit [18].

However, it is possible that due to feature selection both models performed well and were able to locate their respective linear and non-linear feature combinations. The optimized model and Bagging classifier learning curves can be seen in Figure 5. Bagging reduced both the bias and variance making it the optimal meta-algorithm to use with KNN. AdaBoost did not work correctly with the KNN classifier so this plot wasn't included.

TABLE III. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
K-Nearest Neighbours	96.34	87.61	86.67

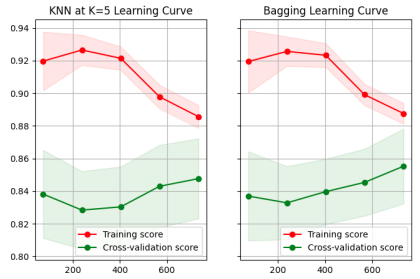

Fig. 5. *k-Nearest Neighbour Curve*

D. *Support Vector Machine*

The SVM model we trained and optimized also had a fairly good score compared to the other models with an accuracy score of 85.89% (Table IV). This result is great since SVM's parameters can drastically change the performance and can be tricky to optimize. While it is also well suited to smaller datasets and binary classification, SVM is also at the risk of overfitting to the training set, and based on our training score, it seems that to an extent it did. However, it is possible that it outperformed expectations since SVMs look at interactions between different features and are good at capturing relationships [19].

TABLE IV. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Support Vector Machines	90.14	87.73	85.89

The learning curve of the SVM classifier can be seen in Figure 6. The classifier has the lowest bias of any algorithm tested so far with similar variance as the others. Bagging had a negative impact and AdaBoost did not work on the classifier so these plots were not included.

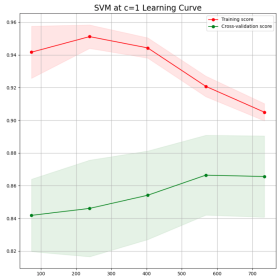

Fig. 6. *Support Vector Machine Learning Curve*

E. *Decision Tree*

The decision tree model even after being optimized was still our worst performing model with an accuracy score of 82.10% on the test set. However, this was to be expected given that it is not a classifier known for generally doing well. It is quite sensitive to deviations in the data, susceptible to a class imbalance bias, and also has a tendency to overfit to training data. Although in our case, it was able to recover from overfitting, it seems as if the noise in our data set and the classifier possibly assigning higher value to the wrong features caused it to have the lowest score [20].

The learning curves can be seen in Figure 7. The original classifier has both low bias and variance but we decided to test bagging and AdaBoost anyways. Bagging greatly reduced bias while also slightly reducing variance. AdaBoost seems to overfit the training data but has reduced bias while slightly reducing variance. Therefore the bagging meta-algorithm is optimal as it has the lowest bias and variance.

F. *Random Forest*

The Random Forest model also performed quite well on our test set with an accuracy of 86.02% (Table VI). Based on previous research done into this problem, Random Forest was a classifier that had done well for this dataset [9] so this result was expected even if the classifier seemed to be overfitted to

TABLE V. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Decision Tree	88.41	84.61	82.10

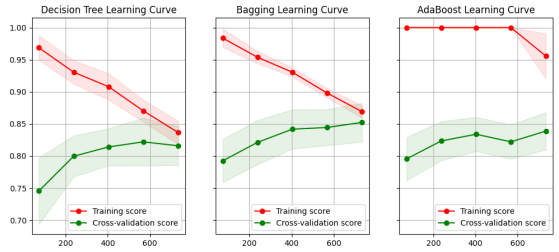

Fig. 7. *Decision Tree Learning Curve*

training data. Despite that, this score makes inherent sense since it is a model that is robust to outliers, has a low risk of overfitting, and works well with a mixture of numerical and categorical data [21].

Another point to note however is that the Random Forest model does not expect linear features, which is in line with the results for KNN, our second best model, but not in line with the results for Logistic Regression, our best model. To see if this model could be further improved, however, we applied to it the Bagging and AdaBoost classifiers and the resulting learning curves can be seen in Figure 8.

TABLE VI. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Random Forest	95.75	87.84	86.02

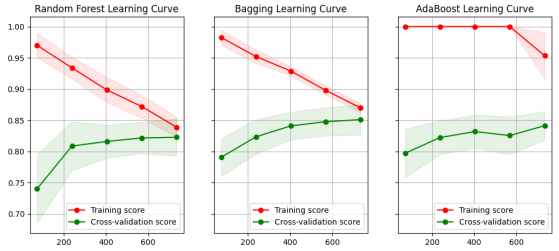

Fig. 8. *Random Forest Learning Curve*

Bagging reduced both the bias and variance. AdaBoost overfitted the training data but reduced the variance and slightly reduced bias. Therefore both the bagging and AdaBoost classifiers are viable as they perform better than the original classifier.

G. Artificial Neural Network

The Artificial Neural Network model that we optimized has a test accuracy score of 84.71% (Table VII). Although the score is satisfactory, it is also our second worst performing model even if it did markedly improve its performance compared to the baseline model of ANN. Although ANNs are very good at figuring out complex relationships and are good with handling non-linear data which seemed to be in a way the case for our data set, they also require a good chunk of training data in order to perform well. Since our data set was on the smaller side, it is entirely possible that it did not have sufficient data required to make good predictions. The learning curve of the optimized model can be seen in Figure 8.

TABLE VII. OPTIMIZED MODEL ACCURACY FOR LOGISTIC REGRESSION

Classifier	Train (%)	Dev (%)	Test (%)
Artificial Neural Network	87.18	86.78	84.71

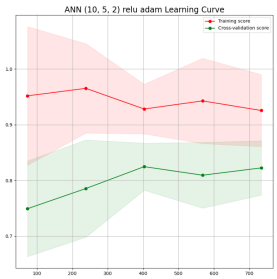

Fig. 9. *Artificial Neural Network Learning Curve*

H. Summary of Results

Overall, while none of our models were excellent at predicting heart disease in the training set, performing under 87%, it should be noted that even the worst of our models were not entirely terrible, achieving an accuracy of 82% and above once optimized. Although hyperparameter optimization is an important part of training and testing your machine learning model, in our case, most of our classifiers did not significantly improve their score when compared to the baseline version (Fig. 10) with the exception of three models — Decision Tree, K-Nearest Neighbour, and Artificial Neural Network.

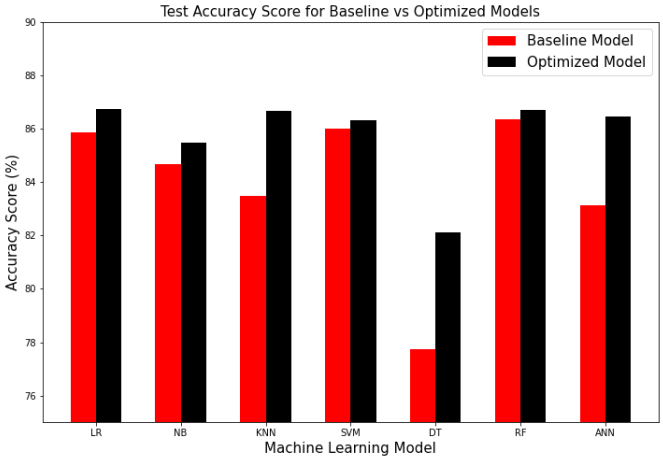

Fig. 10. *Test Set Accuracy Score for Baseline vs Optimized Models*

I. Bias-Variance-Noise Trade-Off

We made use of the bias-variance-noise trade-off primarily during the hyperparameter optimization phase while selecting the right parameters to tune and the correct set of values to tune them over. Simpler, linear models such as Logistic

Regression and Naïve Bayes tend to be low-variance and high bias models. For these models, it was imperative that we select parameters which would reduce bias albeit increase variance. For Logistic Regression this meant tuning the regularization strength as well as adding a regularization parameter, whereas for Naïve Bayes where there were no parameters to tune, the k in k-fold cross validation was used since lower values of k tend to be more biased.

Other more complex models, however, tend to have higher variance and lower bias, and in our case this was a majority of our models: KNN, SVM, and Decision Tree. For KNN, we tuned the number of neighbours (along with weights and distance metrics) since a higher number of neighbours increases bias and reduces variance. For SVM, we focused mainly on the regularization parameter and kernel related parameters, since we know that smaller regularization values cause a decrease in variance. For Decision Trees, due to its poor baseline performance, there were quite a few appealing parameters to tune, and we chose to focus on the complexity parameter for tree pruning along with maximum tree depth since they're both known to affect variance.

Random Forest models are unique in the sense that they take multiple high variance decision trees and employ a number of techniques to reduce variance, resulting in a low variance and low bias model. However, it should be noted that we still carefully considered which parameters and the values over which we could tune these realistically, settling on the number of estimators, the maximum depth of a tree, and the minimum number of samples required at a leaf node.

Overall, we used our understanding of the bias-variance-noise trade-off to select parameters and their ideal ranges when tuning out models.

J. Future Direction

Although we were able to implement and test a total of seven different machine learning models for this project and even attempted classifiers meant to boost and improve optimized models, we were unable to take our implementation a step further and explore hybrid and other ensemble models which combine two or more algorithms to make predictions. Despite it not being quite as well explored, there is emerging evidence that combined models are better at predicting heart disease using his dataset [7, 8, 9]. If given a chance to continue working on this project, therefore, we would like to explore these other types of machine learning models to see how they stack up against the relatively standard ones we used for our project.

V. Implementation and Code

We started by preprocessing the dataset to make the data usable for the classifiers. We used the pandas library to read in the raw data, check for missing/invalid values, and encode the categorical variables. This is done in preprocessing.py and helper.py.

We used the scikit-learn library to split the data, implement all the machine learning and meta-learning models, train, cross-validate, and test the models, generate score metrics, and perform hyperparameter optimization. In addition to scikit-learn, pandas and numpy both assisted in supporting roles to accomplish the above tasks. These can be found in helper.py and optimization.py.

The seaborn matplotlib library was also used to generate all relevant plots, graphs, and figures. The functions for these can be found in optimization.py, plots.py, and helper.py.

In addition to these libraries, the documentation from scikit-learn which demonstrated implementation of each of the functions, learning curves, score metrics and more was useful in completing this project.

References

[1] https://github.com/anthfgreco/heart-disease-machine-learning

[2] Government of Canada, "Heart Disease in Canada," Available: https://www.canada.ca/en/public-health/services/publications/diseases-conditions/heart-disease-canada.html. [Accessed Oct 1 2021].

[3] H. C. McGill, C. A. McMahan and S. S. Gidding, "Preventing Heart Disease in the 21st Century: Implications of the Pathobiological Determinants of Atherosclerosis in Youth (PDAY) Study," *Circulation*, vol. 117, *(9)*, pp. 1216-1227, 2008. doi:10.1161/CIRCULATIONAHA.107.717033. [Accessed Nov 30 2021]

[4] National Health Service, "Cardiovascular Disease," Available: https://www.nhs.uk/conditions/cardiovascular-disease/ [Accessed Oct 1 2021].

[5] A. R. Folsom, "Heterogeneity of cardiovascular diseases among populations-recognition and seminal explanations," *American Journal of Epidemiology*, vol. 185, *(11)*, pp. 1000-1001, 2017. doi:10.1093/aje/kwx073. [Accessed Nov 30 2021].

[6] A. K. Dwivedi, "Performance evaluation of different machine learning techniques for prediction of heart disease," *Neural Computing & Applications*, vol. 29, *(10)*, pp. 685-693, 2016. doi:10.1007/s00521-016-2604-1 [Accessed Oct 1, 2021].

[7] S. Pouriyeh et al, "A comprehensive investigation and comparison of machine learning techniques in the domain of heart disease," in 2017. doi:10.1109/ISCC.2017.8024530.

[8] H. Kahramanli and N. Allahverdi, "Design of a hybrid system for the diabetes and heart diseases," *Expert Systems with Applications*, vol. 35, *(1)*, pp. 82-89, 2008. doi: 10.1016/j.eswa.2007.06.004.

[9] S. Mohan, C. Thirumalai and G. Srivastava, "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques," *IEEE Access*, vol. 7, pp. 81542-81554, 2019. doi:10.1109/ACCESS.2019.2923707.

[10] fedesoriano, "Heart Failure Prediction Dataset," Available: https://www.kaggle.com/fedesoriano/heart-failure-prediction.

[11] UCI Machine Learning Repository, "Heart Disease Data Set," Available: https://archive.ics.uci.edu/ml/datasets/heart+disease.

[12] UCI Machine Learning Repository, "Heart Disease Data Set," Available: https://archive.ics.uci.edu/ml/datasets/statlog+(heart).

[13] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," Available: https://machinelearningmastery.com/k-fold-cross-validation.

[14] Leo Breiman, "Bagging Predictors", Available: https://www.stat.berkeley.edu/~breiman/bagging.pdf

[15] World Health Organization, "Cardiovascular diseases (CVDs)," Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[16] Holy Python, "Logistic Regression," Available: https://holypython.com/log-reg/.

[17] Holy Python, "Naive Bayes," Available: https://holypython.com/nbc/.

[18] Holy Python, "K-nearest Neighbors," Available: https://holypython.com/knn/.

[19] Holy Python, "Support Vector Machine," Available: https://holypython.com/svm/.

[20] Holy Python, "Decision Trees," Available: https://holypython.com/dt/.

[21] Holy Python, "Random Forest," Available:https://holypython.com/rf/.