COMPARISON OF ENSEMBLE TREE-BASED TECHNIQUES FOR PREDICTING HEART DISEASE

Isaiah Steinke[1], Carolyn Chesley[1], Jasmin Jean[2], and Raed Seetan[2]
[1]Department of Mathematics and Statistics, Slippery Rock University, Slippery Rock, PA, 16057
[2]Department of Computer Science, Slippery Rock University, Slippery Rock, PA, 16057
{ips1002, cac1036, jmj1024, raed.seetan}@sru.edu

ABSTRACT
As one of the leading causes of death worldwide, heart disease has received considerable attention from researchers. Hence, machine learning techniques have been studied to aid in the early detection of heart disease via fast and economical methods. In our study, we investigated the use of ensemble tree-based techniques to predict the presence or risk of heart disease for two datasets: an expanded version of the extensively studied Cleveland dataset and the Framingham dataset. Our results show that we are able to achieve an accuracy, a sensitivity, and a specificity of 100% for the expanded Cleveland dataset using random forests, bagging, and boosting. For the Framingham dataset, we can attain an accuracy of up to 85.7% using boosting; however, the sensitivity is only 6.6%. By tuning the boosting hyperparameters, we can increase the sensitivity to 14.1% but incur a minor decrease in accuracy to 82.1%. For both datasets, our methods tend to consistently select features related to a measure of cholesterol as the important variables in tree construction.

KEY WORDS
Coronary artery disease, decision trees, ensemble techniques, heart disease, machine learning, random forests

1. Introduction

Heart disease is the leading cause of death worldwide and is the cause of one in four deaths recorded in the United States, resulting in approximately 655,000 deaths annually [1], [2]. Heart disease is also known as cardiovascular disease (CVD) and refers to several types of heart conditions. The two main types of CVD are coronary artery disease (CAD) and myocardial infarctions or heart attacks [3]. CAD involves the blockage of coronary arteries caused by the build-up of cholesterol and fatty deposits called plaque inside arterial walls [4]. This plaque build-up blocks blood flow, which can cause chest pain (angina), weakness, pain in the arms or shoulder, and shortness of breath, eventually leading to heart failure. The main cause of myocardial infarctions is CAD, which occur when there is insufficient blood supply to the heart [3].

Owing to the widespread incidence and prevalence of heart disease and its negative impacts, the early detection of CAD is crucial to enable the implementation of effective treatments and decrease morbidity and mortality [5]. There are currently multiple medical screening and diagnostic techniques for heart disease, but some are expensive and invasive, potentially resulting in further health complications [4]. Thus, researchers have sought efficient and effective screening and diagnostic techniques that are both noninvasive and reliable. Therefore, the application of machine learning (ML) and data mining techniques for the detection and prediction of heart disease has been extensively studied in the literature. Data mining with large datasets has been shown to be useful in the discovery of patterns that aid in the early detection and prevention of heart disease [4].

In this paper, we examine the use of tree-based and ensemble ML and data mining techniques for the prediction of the presence or risk of heart disease. The performance of these techniques is assessed to determine the most useful and accurate techniques for two datasets. In addition, we compare our results with some results in the literature.

2. Related Work

There are numerous peer-reviewed studies discussing the use of various ML and data mining techniques to classify and predict heart disease [4]. The performance of various algorithms has been analyzed to determine the most useful and accurate algorithms for classification. Some widely used classification techniques for heart-disease prediction include decision tree classifiers, boosted decision trees, random forests, artificial neural networks (ANNs), support vector machines (SVMs), *k*-nearest neighbors (kNN), regression and naïve Bayes classifiers, logistic regression, and rule-based methods [4], [6]–[9].

The review by Alizadehsani *et al.* [4] found that 67 different datasets have been analyzed using ML techniques for the diagnosis of CAD. About 90% of the papers use supervised classification algorithms, with ANNs, decision trees, and SVMs as the top three. The performance of these ML techniques varies, with many techniques achieving values for the sensitivity, specificity, and accuracy that are >90%. In addition, they identified 21 studies that utilized ANNs, kNN, SVMs, convolutional neural networks (CNNs), decision trees, deep belief networks (DBNs), and fuzzy neural networks to achieve an accuracy of more than 98%.

Tayefi *et al.* demonstrated that the decision tree algorithm is accurate, specific, and sensitive for investigating the risk factors of CAD [7]. They achieved an accuracy, a sensitivity, and a specificity of 94%, 96%, and 87%, respectively. Beunza *et al.* [10] investigated the application of decision trees, boosted decision trees, random forests, logistic regression, neural networks, and SVMs for the diagnosis and detection of heart disease. They found that the best algorithms were neural networks and SVMs when the area under the curve is used to assess performance. Xing *et al.* [8] found SVMs to be the best predictor of heart disease with an accuracy of 92.1%, followed by ANNs and decision trees with accuracies of 91% and 89.6%, respectively. Moreover, Amin *et al.* [9] showed that decision trees and kNN are able to obtain the highest precision with a lower accuracy.

In our study, we develop ML models for an expanded version [11] of the Cleveland dataset [12] that has been extensively studied in the literature [4]. To the best of our

knowledge, no other studies in the literature have investigated this expanded dataset. In addition, we build ML models for the Framingham dataset [13], which has been previously investigated [10]. However, Beunza *et al.* [10] did not employ bagging in the development of their models. Thus, we will compare the performance of our tree-based models with their results and assess the performance of bagging for this dataset.

3. Datasets and Data Preprocessing

3.1 Expanded Cleveland Dataset

Our first dataset [11] (hereinafter called *Cleve*) is an expanded version of the original Cleveland dataset [12] maintained by the Cleveland Clinic Foundation. The original dataset in [12] has been extensively studied in the literature [4]. The *Cleve* dataset has been augmented with data from databases maintained by the Hungarian Institute of Cardiology, the University Hospital of Zurich and Basel in Switzerland, and the V.A. Medical Center in Long Beach [11].

The *Cleve* dataset contains 1,025 instances of 14 attributes: age, sex, chest-pain type, resting blood pressure (BP), cholesterol, fasting blood sugar, resting electrocardiogram (ECG) results, maximum heart rate, exercise-induced angina, ST depression induced by exercise, the slope of the peak exercise ST segment, the number of major vessels colored by fluoroscopy, and thalassemia. These are a mixture of nominal, binary, and numeric attributes, and all categorical variables are encoded as integers. The response variable is binary-encoded, and 0 and 1 indicate that the person does not and does have heart disease, respectively.

The *Cleve* dataset does not contain any missing values and hence did not require extensive preprocessing. Exploratory data analyses detected the presence of some statistical outliers, but they were retained as-is since they appeared to be within typical measurement ranges and/or did not appear to be in error.

3.2 Framingham Dataset

The second dataset [13] (hereinafter called *Fram*) originates from a long-term, currently ongoing study of the cardiovascular health of the residents of Framingham, MA. It contains 4,238 instances of 16 attributes: age, sex, education, smoker status, cigarettes per day, BP medications, prevalent stroke, prevalent hypertension, diabetes status, cholesterol level, systolic BP, diastolic BP, body mass index (BMI), heart rate, and glucose level. These are a mixture of nominal, binary, and numeric attributes, and all categorical attributes are encoded as integers. The response variable is the 10-year risk of future coronary heart disease (CHD) and is a binary-encoded response variable—0: no risk and 1: risk of future CHD.

The *Fram* dataset contained missing values, and exploratory data analyses revealed the presence of statistical outliers for some attributes. All outliers were retained because we found that they were within acceptable measurement ranges and/or that they did not seem to in error. For the missing values, we initially attempted to build regression models for the numeric attributes for imputation. For the categorial/nominal attributes, we built models based on kNN, logistic regression, and decision trees in R for imputation. In all cases, we found that the resulting models did not perform well for imputation or did not provide better performance than simpler models that imputed a constant value.

For the education (105 missing values) and heart rate (1 missing value) attributes, we deleted the records containing missing values. For education, there were four different categories, and the distribution of these categories was heavily biased to values of 1 and 2. Thus, it was difficult to determine a suitable imputation value. For the BP medications (53 missing values) and cigarettes per day (29 missing values) attributes, we replaced the missing values with 0. For BP medications, this categorical value was mostly 0 in the dataset. For cigarettes per day, this numeric attribute was highly skewed with about half of the values being 0 or 1. Given the relatively low number of missing values for these attributes, the imputation of 0 seems to be of relatively low risk.

For the rest of the attributes containing missing values, we imputed their median values. These include cholesterol level (53 missing values, median: 234), glucose level (388 missing values, median: 78), and BMI (19 missing values, median: 25.39). After imputation, the cleaned *Fram* dataset contained 4,132 instances.

3.3 Preparation of the Training and Test Sets

Training and test sets were created in R (v. 4.0.2) by random sampling at an 80/20 ratio for the training/test sets and saved as *.csv files for both the *Cleve* and *Fram* datasets. These files were then processed into *.arff files for analysis in Weka (v. 3.8.4). Since the response variable for *Fram* is skewed towards 0 (~85.4% of the records are 0 for the response variable), we ensured that both the training and test sets retained a roughly 85%/15% ratio of 0/1 for the response variable.

Typical choices for the training/test set split are 60/40, 70/30, and 80/20 [14], [15]. We chose an 80/20 split for two reasons. First, the 80/20 split allows for more data in the training set, thereby improving the parameter estimates for our classifiers, resulting in better model performance [14], [16]. This is more important for the *Cleve* dataset since it is relatively small. Second, Beunza *et al.* [10] use an 80/20 split in their analyses; hence, we used the same split to ensure a fair comparison.

4. Analysis Methods

4.1 Tree-Based Algorithms

As discussed in Section 2, tree-based and ensemble methods have been extensively used in the prediction of heart disease. Hence, we chose the decision tree as a baseline method. A decision tree is a nonparametric method that creates a tree-based classification model that can extract hidden knowledge from large databases. The advantages of decision trees are that they are easy to implement and the resulting tree is easy to interpret. Each node in the tree represents a test on an attribute value, each branch represents an outcome of a test, and the leaves represent classes or class distributions [17].

To improve classification accuracy, we will additionally use bagging, boosting, and random forests. Bagging allows the variance of individual classifiers to be reduced and helps avoid overfitting the data [17]. In boosting, weights are randomly assigned to each training tuple to boost the accuracy of a learning method [17]. Random forests are made up of many decision trees, and individual decision trees are generated with a randomly selected subset of attributes at each node. The accuracy of random forests is dependent on the strength of the individual classifiers and the level of dependence between them [17].

We investigated the use of these methods in both R and Weka. In R, the decision tree was implemented with the "tree" (v. 1.0-40) library, where a tree is grown with binary recursive partitioning according to the algorithm by Breiman *et al.* [18]. We set the algorithm to use the Gini index as the splitting criterion. The random forests and bagging ensemble methods were implemented with the "randomForest" (v. 4.6-14) library, where the bagging method is essentially achieved by allowing the random forests algorithm to choose among all predictors at each split. This library uses the algorithm by Breiman [19], and the number of trees and the number of predictors were the only hyperparameters tuned to improve performance. Finally, the "gbm" (v. 2.1.8) library was used as the boosting ensemble method. This library uses the AdaBoost algorithm, and the number of trees, the interaction depth, and the shrinkage (i.e., the learning rate) were tuned to improve performance.

In Weka, we employed the "J48" package to build a decision tree, which utilizes the C4.5 algorithm. The "Bagging" package was used for the bagging ensemble method with "J48" as the base learner. All other parameters were left at their default values. For the random forests method, we used the "RandomForest" package; this also uses the algorithm by Breiman [19]. Here again, the number of trees and the number of predictors were tuned to improve performance. Finally, the "AdaBoostM1" package was used as the boosting ensemble method, which employs the AdaBoost M1 algorithm. "J48" was chosen as the base learner, and only the number of iterations was tuned to improve performance.

4.2 Performance Metrics

Since we have binary classification problems for both datasets, we built models using the training set and obtained the confusion matrices after prediction with these models on the test set. After identifying the numbers of true positives *TP*, true negatives *TN*, false positives *FP*, and false negatives *FN* from the confusion matrices, we then calculated the accuracy *AC*, misclassification rate *MR*, sensitivity *SE*, specificity *SP*, and F_1 score (denoted F_1 for short hereinafter). These quantities are expressed as follows:

$$AC = \frac{TP+TN}{TP+TN+FP+FN}, \tag{1}$$

$$MR = \frac{FP+FN}{TP+TN+FP+FN}, \tag{2}$$

$$SE = \frac{TP}{TP+FN}, \tag{3}$$

$$SP = \frac{TN}{TN+FP}, \tag{4}$$

$$F_1 = \frac{2\times PR\times SE}{PR\times SE}, \tag{5}$$

where the precision $PR = TP/(TP + FP)$. In general, we tuned hyperparameters where possible to maximize *AC*. However, we also paid close attention to *SE* because the rate of false negatives is critical for medical diagnostics.

In addition to the performance metrics in Eqs. (1)–(5), we also tabulated the runtimes of the various algorithms in both R and Weka. All runtimes were obtained on a computer equipped with an AMD Ryzen 7 3800X processor (running at 3.9/4.5 GHz base/boost clocks) and 32 GB of RAM.

5. Results

5.1 Expanded Cleveland Dataset

Table 1 summarizes the runtimes and performance metrics of all algorithms in both Weka and R for the *Cleve* dataset. Surprisingly, we are able to achieve perfect classification performance (i.e., an accuracy, a sensitivity, and a specificity of 100%) on our test set for many algorithms: random forests, bagging, and boosting in R and random forests and boosting in Weka. We attribute these results to the simplicity and size of the dataset, the lack of missing values, and relatively low amount of noise. It is possible that overfitting may play a small role. However, the hyperparameters for the random forests and boosting algorithms to not appear to be sufficiently extreme to

Table 1. Results for the *Cleve* dataset.

	Method	Runtime [s]	*AC*	*MR*	*SE*	*SP*	F_1
R	Decision Tree	0.03	90.2%	9.8%	89.6%	90.9%	90.5%
	Random Forests	0.09	100%	0%	100%	100%	100%
	Bagging	0.11	100%	0%	100%	100%	100%
	Boosting	0.21	100%	0%	100%	100%	100%
Weka	Decision Tree	0.01	96.1%	3.9%	92.5%	100%	96.1%
	Random Forests	0.16	100%	0%	100%	100%	100%
	Bagging	0.03	98.1%	1.9%	96.2%	100%	98.1%
	Boosting	0.11	100%	0%	100%	100%	100%

Table 2. Results for the *FRAM* dataset.

	Method	Runtime [s]	*AC*	*MR*	*SE*	*SP*	F_1
R	Decision Tree	0.08	80.3%	19.7%	9.9%	92.3%	12.8%
	Random Forests	1.39	85.0%	15.0%	4.1%	98.9%	7.5%
	Bagging	1.54	84.5%	15.5%	4.1%	98.3%	7.3%
	Boosting	2.61	85.7%	14.3%	6.6%	99.3%	11.9%
Weka	Decision Tree	0.03	81.7%	18.3%	9.1%	94.2%	12.7%
	Random Forests	4.8	85.1%	14.9%	2.5%	99.3%	4.7%
	Bagging	0.34	83.3%	16.7%	6.6%	96.5%	10.4%
	Boosting	1.04	83.4%	16.6%	9.1%	96.2%	13.8%

suggest overfitting: the numbers of trees and predictors are 100 and 4, respectively, for the random forests algorithm, and the tree depth and the number of trees are 2 and 500, respectively, for boosting. We also checked the performance of these algorithms at a different training/test set split of 70/30 (results not shown) to test for possible overfitting. Here again, we were able to obtain perfect classification performance on the test set using the random forests algorithm in R for modest hyperparameter values (number of trees = 50, number of predictors = 4).

In addition, we can achieve quite good classification performance with a simple decision tree in both R and Weka, with Weka having a slightly lower runtime. At worst, the decision tree in R is able to achieve an accuracy of at least 90% and a sensitivity of just under 90%.

Comparing the runtimes of the algorithms in R versus Weka, we find that Weka is slightly faster in general, except for the random forests algorithm. For the decision tree and random forests algorithms, this difference is likely due to differences in the implementations of these algorithms in R and Weka, as the decision tree algorithm has no hyperparameters to tune and the random forest algorithm used the same values for the number of trees (100) and number of predictors (4) in both R and Weka. For bagging, the Weka implementation does not have many hyperparameters to tune, whereas the number of trees can be tuned for the implementation in R. Finally, the difference in runtimes for boosting is likely due to differences in the hyperparameters, e.g., the number of trees. An increase in the number of trees considerably increases the runtime, and we suspect that the number of trees is higher for our R results (from the Weka documentation, it does not appear that the number of trees may be varied).

An advantage of the R implementations is that we are able to assess the importance of the attributes for the various algorithms. For the decision tree, thalassemia, the number of major vessels colored by fluoroscopy, ST depression induced by exercise, resting ECG results, cholesterol, chest-pain type, exercise-induced angina, age, sex, and maximum heart rate are used in its construction. This results in a tree that is seven levels deep. For random forests/bagging, the five most important variables (in decreasing order) are thalassemia, the number of major vessels colored by fluoroscopy, chest-pain type, maximum heart rate, and ST depression induced by exercise before there is a significant drop-off in the mean decrease in the Gini index. Finally, the four most important variables for boosting (in decreasing order) are thalassemia, the number of major vessels colored by fluoroscopy, chest-pain type, and ST depression induced by exercise before there is a significant decrease in importance. We see that thalassemia, chest-pain type, the number of major vessels colored by fluoroscopy, maximum heart rate, and ST depression induced by exercise consistently appear as the top variables of importance for all algorithms.

5.2 Framingham Dataset

Table 2 summarizes the runtimes and performance metrics for all algorithms in Weka and R for the *Fram* dataset. Overall, we achieve fairly consistent performance for all algorithms, with the accuracy ranging from a low of 80.3% (decision tree, R) to a high of 85.7% (boosting, R). Notably, the sensitivity is less than 10% for all algorithms, with the highest sensitivity obtained for the decision tree in R. This is important because it indicates a high false negative rate, which should be avoided in diagnostic medical tests. For boosting in R, we can increase the sensitivity with further hyperparameter tuning; a sensitivity

of 14.1% can be achieved at the cost of a modest decrease in accuracy (to 82.1%).

The reason for the high false negative rate is likely due to the nature of the *Fram* dataset. In our cleaned full dataset, only 15.2% of the instances have 1 for the response variable. Our training and test sets have also been randomly sampled to retain roughly an 85%/15% ratio of 0/1. Therefore, a simple classifier that always classifies as 0 will achieve an accuracy of 85.4% for the *Fram* test set but a sensitivity of 0%. Because of the relatively low proportion of response variables with values of 1, the algorithms have fewer instances to learn positive outcomes and more for learning negative outcomes. Many methods have been discussed in the literature to overcome this issue, many of which involve sampling the dataset to balance classes in the training set by either oversampling the minority class or undersampling the majority class.

Comparing the runtimes of the algorithms, we again see that Weka achieves lower runtimes for the decision tree, bagging, and boosting methods and a higher runtime for random forests. The reasoning for this difference is the same as it is for the *Cleve* dataset discussed in Section 5.1.

We again noted the important variables used in the construction of our trees using the R algorithms. For the decision tree, 13 of the 15 predictors were used to construct a very deep and complex tree with 275 terminal nodes (we did not attempt to prune this tree; hence, it is possible that there is some overfitting). For the random forests algorithm, the systolic BP, BMI, cholesterol level, age, glucose level, diastolic BP, and heart rate were the most important variables in decreasing order. For boosting, age, systolic BP, diastolic BP, glucose level, cholesterol level, cigarettes per day, and BMI were the most important in decreasing order. Notably, the systolic BP, diastolic BP, and cholesterol level, which are measures typically used to diagnose heart disease, appear to be important in all algorithms. Further, a measure of cholesterol—cholesterol level for the *Fram* dataset and cholesterol for the *Cleve* dataset—is an important variable common to both datasets.

5.3 Comparison with Literature Results

Latha *et al.* [6] used boosting and bagging to improve classification accuracy with the original Cleveland dataset [12]. With bagging, they realized accuracies of 79.87% and 80.53% when using the C4.5 decision tree and random forest algorithms, respectively, as base learners. These results are worse than ours in Table 1, where we are able to achieve an accuracy of 100% using C4.5 as a base learner for bagging in Weka and for bagging in R. For boosting, Latha *et al.* report accuracies of 75.9% and 78.88% when using the C4.5 decision tree and random forest algorithms as base learners. Again, we are able to achieve better performance according to Table 1—an accuracy of 100%. The reason why our results may be better is because our *Cleve* dataset is much larger than the original Cleveland dataset [12].

Table 3. Comparison of our results with those of Beunza *et al.* [10] for the *Fram* dataset.

	Method	*AC*	*SE*	*SP*
This Study	Decision Tree	80.3%	9.9%	92.3%
	Random Forests	85.0%	4.1%	98.9%
	Bagging	84.5%	4.1%	98.3%
	Boosting	85.7%	6.6%	99.3%
Beunza *et al.* [10]	Decision Tree	84%	8%	98%
	Random Forests	78%	30%	87%
	Boosting	84%	5%	99%
	SVM	68%	69%	68%
	Neural Network	71%	64%	72%
	Logistic Regression	66%	69%	66%

Beunza *et al.* [10] investigated the classification performance for the *Fram* dataset using R. They considered three variations of this dataset; for a fair comparison, we have summarized their results in Table 3 for the variation where they imputed the mean values for missing values (the closest variation to ours) in addition to our results. The results that they obtained for the decision tree and boosting methods are similar to ours, with the largest difference being within ±6% for the specificity. Beunza *et al.* used the C5.0 decision tree and corresponding boosting function in the "C50" R library for their decision tree and boosting algorithms, respectively. Hence, it is likely that the newer version of the decision tree algorithm is the cause of the rather large increases in accuracy (80.3% vs. 84%) and specificity (92.3% vs. 98%) compared to ours.

Notably, they are able to attain much higher sensitivities using the random forests, SVM, neural network, and logistic regression techniques, although this results in greatly decreased accuracy compared to their decision tree baseline (up to 18% when comparing the decision tree and logistic regression methods). In particular, the SVM and logistic regression techniques are able to achieve a sensitivity of 69% but have less than 70% accuracy. To the best of our knowledge, we are using the same R library and function as Beunza *et al.* for the random forest algorithm. Hence, we assume that the differences from our results are due to differences in hyperparameter settings and the minor differences in our dataset due to the slightly different imputation methods and the differences in sampling the training and test sets.

6. Conclusion

We have studied the use of ensemble tree-based algorithms for the prediction of heart disease. For the *Cleve* dataset, we are able to achieve perfect classification performance using random forests, bagging, and boosting. For the *Fram* dataset, we can achieve an accuracy that is comparable to a simple classifier that always classifies as 0 with careful

hyperparameter tuning while realizing an improved but still low sensitivity. For a modest tradeoff in accuracy (a decrease of ~3%), we can increase the sensitivity by 7.5%, although this still results in many false negatives.

Owing to the unbalanced nature of the *Fram* dataset, we obtained a large number of false negatives, which means that our classifiers are unlikely to be of use in medical diagnostics. Therefore, it may be worthwhile to investigate techniques that balance the training set to further improve classification performance. In addition, as shown by Beunza *et al.* [10], it appears to be possible to greatly increase the sensitivity using other techniques such as SVMs, logistic regression, or neural networks. These techniques should also be investigated.

References

[1] Centers for Disease Control and Prevention, About heart disease, https://www.cdc.gov/heartdisease/about.htm (accessed Sept. 16, 2020).

[2] World Health Organization, Cardiovascular diseases (CVDs), https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) (accessed Sept. 20, 2020).

[3] V.C. Scanlon & T. Sanders, *Essentials of anatomy and physiology* (Philadelphia, PA: F.A. Davis Company, 2011), p. 265.

[4] R. Alizadehsani, M. Abdar, M. Roshanzamir, A. Khosravi, P.M. Kebria, F. Khozeimeh, S. Nahavandi, N. Sarrafzadegan, & U.R. Acharya, Machine learning-based coronary artery disease diagnosis: A comprehensive review, *Computers in Biology and Medicine*, *111*, 2019, 103346.

[5] A.C. Skelly, R. Hashimoto, D.I. Buckley, E.D. Brodt, N. Noelck, A.M. Totten, J.R. Lindner, R. Fu, & M. McDonagh, Noninvasive testing for coronary artery disease, Agency for Healthcare Research and Quality, Rockville, MD, USA, Comparative Effectiveness Review No. 171, AHRQ Publication No. 16-EHC011-EF, March 2016. [Online]. Available: https://effectivehealthcare.ahrq.gov/products/coronary-artery-disease-testing/research.

[6] C.B.C. Latha & S.C. Jeeva, Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques, *Informatics in Medicine Unlocked*, *16*, 2019, 100203.

[7] M. Tayefi, M. Tajfard, S. Saffar, P. Hanachi, A.R. Amirabadizadeh, H. Esmaeily, A. Taghipour, G.A. Ferns, M. Moohebati, & M. Ghayour-Mobarhan, hs-CRP is strongly associated with coronary heart disease (CHD): A data mining approach using decision tree algorithm, *Computer Methods and Programs in Biomedicine*, *141*, 2017, 105–109.

[8] Y. Xing, J. Wang, Z. Zhihong, & G. Yonghong, Combination data mining methods with new medical data to predicting outcome of coronary heart disease, *2007 International Conf. on Convergence Information Technology (ICCIT 2007)*, Gyeongju, South Korea, 2007, 868–872.

[9] M.S. Amin, Y.K. Chiam, & K.D. Varathan, Identification of significant features and data mining techniques in predicting heart disease, *Telematics and Informatics*, *36*, 2019, 82–93.

[10] J.-J. Beunza, E. Puertas, E. García-Ovejero, G. Villalba, E. Condes, G. Koleva, C. Hurtado, & M.F. Landecho, Comparison of machine learning algorithms for clinical event prediction (risk of coronary heart disease), *Journal of Biomedical Informatics*, *97*, 2019, 103257.

[11] D. Lapp, 2019, Heart disease dataset, Kaggle. [Online]. Available: https://www.kaggle.com/johnsmith88/heart-disease-dataset.

[12] UCI Machine Learning Repository, Jul. 1988, Heart disease data set, UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+disease.

[13] Dileep, Jun. 2019, Logistic regression to predict heart disease, Kaggle. [Online]. Available: https://www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression.

[14] B. Boehmke & B.M. Greenwell, *Hands-on machine learning with R* (Boca Raton, FL: CRC Press, 2020).

[15] K.P. Murphy, *Machine learning: A probabilistic perspective* (Cambridge, MA: MIT Press, 2012).

[16] I.H. Witten, E. Frank, & M.A. Hall, *Data mining: Practical machine learning tools and techniques* (Burlington, MA: Morgan Kaufmann, 2011), 3rd ed.

[17] J. Han, M., Kamber, & J. Pei, *Data mining: Concepts and techniques* (Waltham, MA: Elsevier/Morgan Kaufmann, 2012), 3rd ed.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, & C. J. Stone, *Classification and regression trees* (Belmont, CA: Wadsworth, 1984).

[19] L. Breiman, Random forests, *Machine Learning*, *45*(1), 2001, 5–32.