Evolutionary Optimization of Artificial Neural Networks and Tree-Based Ensemble Models for Diagnosing Deep Vein Thrombosis

Ruslan Sorano[1], Kazi Shah Nawaz Ripon[2] and Lars Vidar Magnusson[1]

***Abstract*— Machine learning algorithms, particularly artificial neural networks, have shown promise in healthcare for disease classification, including diagnosing conditions like deep vein thrombosis. However, the performance of artificial neural networks in medical diagnosis heavily depends on their architecture and hyperparameter configuration, which presents virtually unlimited variations. This work employs evolutionary algorithms to optimize hyperparameters for three classic feed-forward artificial neural networks of pre-determined depths. The objective is to enhance the diagnostic accuracy of the classic neural networks in classifying deep vein thrombosis using electronic health records sourced from a Norwegian hospital. The work compares the predictive performance of conventional feed-forward artificial neural networks with standard tree-based ensemble methods previously successful in disease prediction on the same dataset. Results indicate that while classic neural networks perform comparably to tree-based methods, they do not surpass them in diagnosing thrombosis on this specific dataset. The efficacy of evolutionary algorithms in tuning hyperparameters is highlighted, emphasizing the importance of choosing the optimization technique to maximize machine learning models' diagnostic accuracy.**

I. Introduction

Deep Vein Thrombosis (DVT) [1] is a medical condition characterized by the formation of one or more blood clots, known as thrombi, in one of the body's large veins, commonly found in the lower limbs. These clots can partially or entirely block circulation in the vein, potentially leading to severe complications such as pulmonary embolism (PE). Nearly half of DVT cases may present with minimal or no symptoms, making early detection and diagnosis critical for effective intervention [2].

Artificial Neural Networks (ANNs) [3] are machine learning models widely used in various domains, including medical applications, such as disease diagnostics. Their computational algorithm is inspired by the biological neural networks of animal brains, designed to imitate how neurons in the brain process information. ANNs consist of interconnected nodes organized into layers: input layer, hidden layers, and output layer. Each connection between nodes is associated with a weight that determines the strength of the connection. During training, ANNs learn to adjust these weights through a process known as backpropagation, wherein the model iteratively updates its parameters to minimize the discrepancy between predicted and actual outputs.

[1]R. Sorano and L. V. Magnusson are with the Department of Computer Science and Communication, Østfold University College, Norway {ruslan.sorano, lars.v.magnusson}@hiof.no

[2]K. S. N. Ripon is with the Department of Computer Science, Oslo Metropolitan University, Norway kazi.ripon@oslomet.no

Machine learning (ML) models, including ANNs, have a range of hyperparameters (HPs) that play an important role in their performance. These parameters control the learning process of the algorithm and significantly influence its predictive capability. Fine-tuning the HPs [4] is essential in determining the efficacy of ML models. Various methods for HP tuning exist [5], ranging from manual grid search to automated techniques. In *manual tuning* [6], practitioners iteratively adjust HPs based on domain knowledge and intuition, which can be time-consuming and suboptimal, especially for complex models. *Grid search* [7] techniques systematically explore HP combinations within predefined ranges to identify the optimal configuration. While being effective, these methods may struggle with high-dimensional parameter spaces and computational expense. *Random search* techniques [8] explore HP combinations randomly within predefined ranges, offering an alternative to grid search. This approach may be more efficient for high-dimensional parameter spaces and can sometimes outperform grid search in finding optimal configurations. *Bayesian optimization* [9] is another approach for HP tuning that uses probabilistic models to select the next HP configuration based on the previous results. This method efficiently balances exploration and exploitation, often requiring fewer iterations to find optimal or near-optimal configurations than grid or random search, especially in high-dimensional spaces.

HPs in ANNs are parameters that govern the architecture and learning dynamics of the network, distinct from the weights learned during training. Key HPs include the number of layers, the number of neurons in each layer, activation functions and learning rates. The number of hidden layers in a neural network significantly influences its performance and efficiency. Adding hidden layers can enhance the network's ability to learn complex patterns and improve accuracy. While more hidden layers can increase accuracy, excessive complexity may lead to overfitting, where the model performs well on training data but poorly on new data. Typically, simpler models with one hidden layer may struggle with complex patterns but are computationally efficient. Increasing the number of layers will better balance complexity and computational cost, allowing for better representation of data features. Research suggests that implementing three hidden layers often provides a balance between time complexity and accuracy, offering optimal performance [10]. Limiting the configurations to three depths allows us to observe if the model complexity impacts performance without overwhelming computational resources. Training deeper networks can be computationally intensive;

hence, focusing on three depths allows for manageable experimentation.

In this context, Evolutionary Algorithms (EAs) [11] have emerged as a promising technique for efficiently searching the vast space of HPs to enhance the performance of ML models [12]. EAs draw inspiration from natural selection and genetic inheritance, iteratively evolving a population of candidate solutions to optimize a given objective function. By simulating the principles of survival of the fittest and genetic variation, EAs offer a robust and scalable framework for HP optimization in ML tasks. Existing studies on EAs for optimizing the HPs of ANNs have shown promising results across various domains [13], [14]. However, in the specific context of DVT prediction, this area remains underexplored.

Our earlier research [15] focused on optimizing ML models for predicting DVT using traditional techniques like grid search. Our findings showed that tree-based ML models outperformed other classifiers in diagnosing DVT. In our subsequent work [16], we employed an EA to fine-tune two tree-based ensemble ML models, namely Random Forest (RF) [17] and XGBoost (XGB) [18]. We analyzed the results of this evolutionary optimization approach from both single- and multi-objective perspectives and compared them with a conventional technique, random search. The outcomes confirmed that the EA approach is effective for optimizing the HPs of RF and XGB models and demonstrated comparable effectiveness or superiority over the more traditional random search optimization approach.

Building on these promising results, our current work focuses on utilizing an EA to enhance the predictive capabilities of ANNs and tree-based ensemble models for predicting DVT. We separately optimized three classic feed-forward ANNs with one, two, and three hidden layers, in addition to RF and XGB. Our research utilized the Ri-Schedule dataset, which was acquired during the study on the effectiveness of D-dimer testing as a stand-alone method for excluding DVT [19]. Leveraging this patient data, we compared the performance of optimized ANNs with that of tree-based ensemble models, RF and XGB, which have previously shown effectiveness in DVT diagnosis using the Ri-Schedule dataset [15], [16]. By benchmarking the predictive accuracy of ANNs against XGB and RF, we aim to assess the relative strengths and limitations of neural networks in diagnosing DVT. This comparative analysis will critically influence further research on predicting DVT with our dataset, guiding future endeavors toward more effective diagnostic approaches.

The organization of this paper is as follows: Section II presents ANNs and their associated HPs. Additionally, it introduces EAs and their approach to optimizing HPs. Section III provides a comprehensive overview of our implementation methodology, including a detailed explanation of the employed optimization process. This section also contains an exploration of the evaluation metrics utilized to assess the effectiveness of our approach and a description of the dataset used in the experiments. Section IV presents and analyzes the outcomes of this study. In conclusion, Section V summarizes our work by recapitulating key findings, acknowledging its limitations, and proposing directions for future research.

II. Background

The healthcare sector's integration of ML techniques has witnessed a rapid surge in recent years, revolutionizing traditional medical practices. ML algorithms have emerged as indispensable tools, allowing clinicians to analyze vast and intricate datasets, facilitating disease diagnosis and enhancing patient outcomes. This paradigm shift has been pivotal in augmenting medical research endeavors and elevating the accuracy of medical predictions, consequently leading to improved patient outcomes. Among the numerous ML models, ANNs, RF and XGB stand out prominently for their effectiveness in healthcare applications [15], [16], [20]–[27].

ANNs [3] are computational models inspired by the biological neural networks of the human brain. ANNs excel at learning complex and nonlinear relationships from data, making them well-suited for tasks involving intricate patterns or high-dimensional feature spaces. ANNs have demonstrated notable success in various healthcare applications, including medical image analysis [28], clinical decision support [29], and disease risk prediction [30], [31]. Their ability to automatically extract relevant features from raw data and their capacity to model nonlinear relationships contribute to their effectiveness in capturing subtle cues and patterns indicative of disease states. The optimization of ANNs for medical diagnosis tasks relies heavily on fine-tuning the HPs, including the number of hidden layers, neurons per layer, learning rates, and activation functions. These HPs significantly impact the learning behavior and predictive capabilities of neural networks. However, traditional methods like grid search and random search for HP tuning can be time-consuming and computationally intensive [32].

EAs [11] have become powerful tools for solving optimization problems within ML, including HP tuning. These algorithms mimic natural evolutionary processes to iteratively explore the vast HP space and identify optimal configurations efficiently. Unlike traditional brute-force methods, which exhaustively search through all possible combinations of HPs, EAs employ a population-based approach, which enables them to navigate complex, high-dimensional spaces efficiently. Evolutionary operators, such as mutation and crossover, play critical roles within EAs by introducing genetic diversity and facilitating the exploration of the HP space. By evaluating, evolving and selecting candidate solutions over multiple generations, EAs can effectively focus on promising regions of the HP space, ultimately discovering configurations that yield optimal model performance. This evolutionary approach to HP optimization offers a robust and flexible framework for fine-tuning ML models capable of accommodating various optimization objectives and constraints while mitigating the computational burden of exhaustive search methods.

0.00224	1086	967	1061	Tanh	LeakyReLU	Sigmoid	Adam

Fig. 1: ANN-III chromosome representation.

0.04810	0.29005	147	7	1	0.9	0.6

Fig. 2: XGB chromosome representation.

In our previous work [16], the integration of EA for HP tuning of RF and XGB models exhibited promising results, surpassing the performance achieved through conventional methods. Building upon this success, we extend our approach to ANNs, utilizing EA to optimize neural network HPs. By applying EAs to ANNs, we aim to enhance model performance further, capitalizing on the evolutionary principles to achieve optimal configurations.

III. Evolutionary Optimization of Hyperparameters in Artificial Neural Networks for Classification of DVT

In our work, we employed an evolutionary algorithm to optimize HPs for ML models. In the context of this study, the chromosome serves as a genetic representation of the HPs of conventional feed-forward ANNs, XGB, and RF models.

A. Chromosome representation

ANN chromosome: For the ANN chromosome, three versions correspond to networks with 1, 2, and 3 hidden layers. The genes in the chromosome represent *learning rate (LR)*, that defines the step size for weight updates during the training process; *neurons per layer*, the number of neurons in each hidden layer; *activation functions*, the activation function for each hidden layer; and *optimizer*, the optimization algorithm during model training.

The number of genes for neurons and activations is determined by the number of layers in the ANN, ensuring a flexible and adaptable chromosome configuration. Fig. 1 shows a sample chromosome for ANN with three hidden layers. The value type and range of the genes are as follows:

- learning rate (Real): Initialized between 0.0001 and 0.1, with 10 points equally spaced in logspace.
- number of neurons (Integer): Ranging from 16 to 2048. Repetitive for each layer (1, 2, or 3 genes).
- activation function (Categorical): Options include *ReLU*, *Sigmoid*, *Tanh*, and *LeakyReLU*. Repetitive for each layer (1, 2, or 3 genes).
- optimizer (Categorical): Options include *Adagrad*, *Adam*, *RMSprop*, and *SGD*.

XGB chromosome: The XGB chromosome contains the following HPs controlling the behavior of the XGB model: *learning rate*, that dictates the step size shrinkage during each boosting iteration; *gamma*, represents the minimum loss reduction required to partition a leaf node further; *number of estimators*, determines the number of boosting rounds; *maximum depth*, the maximum depth of the decision tree; *minimum child weight*, the minimum sum of instance weight needed in a child; *subsampling ratio*, control the

146	sqrt	Entropy	24	10	3

Fig. 3: RF chromosome representation.

subsampling of training data; *column subsampling ratio*, control the subsampling of feature columns. An example of XGB chromosome is shown on Fig. 2. The value type and range of the genes are as follows:

- learning rate (Real): Initialized between 0.01 and 1.0, with 1000 points equally spaced in logspace.
- gamma (Real): Initialized between 0.01 and 10, with 1000 points equally spaced in logspace.
- number of estimators (Integer): Ranging from 100 to 500.
- maximum depth (Integer): Values ranging from 3 to 30.
- minimum child weight (Integer): Parameter values ranging from 1 to 10.
- subsampling ratio (Real): Values ranging from 0.1 to 1.0, with discrete values such as 0.1, 0.2, ..., 1.0.
- column subsampling ratio (Real): Values ranging from 0.1 to 1.0, with discrete values similar to subsampling ratio.

RF chromosome: The RF chromosome is composed of the genes representing the following HPs: *number of estimators*, that defines the number of decision trees in the forest; *maximum features*, the maximum number of features considered for splitting a node; *criterion*, defines the function to measure the quality of a split; *maximum depth*, the maximum depth of the tree; *minimum samples split*, the minimum number of samples required to split an internal node; *minimum samples leaf*, the minimum number of samples required to be a leaf node. An RF chromosome representation is shown on Fig. 3. The value type and range of the genes are as follows:

- number of estimators (Integer): Ranging from 100 to 1200.
- maximum features (Categorical): Options include $sqrt$, $log2$, and $None$.
- criterion (Categorical): Options include $gini$ and $entropy$.
- maximum depth (Integer): Values ranging from 5 to 30.
- minimum samples split (Integer): Parameter values ranging from 2 to 100.
- minimum samples leaf (Integer): Parameter values ranging from 1 to 10.

B. Evolutionary Operators

Evolutionary operators, such as crossover, simulated binary crossover (SBX), mutation, and polynomial mutation drive exploration and exploitation in EAs. They create new candidate solutions from existing ones, imitating natural selection. Understanding their interactions is essential for effective EA design and implementation in solving complex optimization problems, as they influence the algorithm's ability to navigate the solution space and achieve high-quality solutions.

Crossover, or recombination, emulates genetic recombination in biological organisms. It combines genetic material from two parent solutions to generate offspring, promoting exploration. Mechanisms like one-point, two-point, or uniform crossover influence offspring diversity and quality.

Simulated Binary Crossover (SBX) [33] is a specialized crossover operator used in real-valued optimization problems. Unlike conventional binary crossover, SBX performs operations involving real-valued parameters. Offspring are generated by randomly selecting a point between parents and using a probability distribution function based on a simulated binary distribution. This capability is valuable for ML models dealing with continuous HPs, allowing broader exploration beyond discrete choices. SBX enables EAs to explore the entire continuum of real values, enhancing HP space exploration, capturing subtle interactions, and improving ML algorithm performance and generalizability.

Mutation introduces randomness into the population by modifying individual solutions, helping maintain genetic diversity and preventing premature convergence. It can alter specific genes within predefined ranges. The mutation rate and extent of changes influence the algorithm's behavior.

Polynomial Mutation (PM) [34] is a mutation operator designed for real-valued optimization problems. It introduces small, controlled perturbations to gene values, emulating random mutations in biological evolution. A polynomial function controls the magnitude of changes, ensuring larger changes are less likely than minor ones, preventing excessive deviation from the current state. PM is characterized by mutation probability (likelihood of occurrence) and distribution index (degree of non-uniformity).

SBX and PM illegal values repair mechanism: While the SBX and PM operators are potent tools for exploring a problem's search space and generating diverse solutions, the offspring generated during the SBX or PM operation may have values outside the acceptable range, termed "illegal values." A repair mechanism is employed to address this, such as random re-initialization that replaces illegal values with valid ones. In this mechanism, a random value for x is drawn from a uniform distribution within the range defined by a valid parent gene and a boundary crossed by an offspring's gene value.

C. Optimization process

The primary objective of this study revolves around the application of EA to fine-tuning HPs to enhance the predictive accuracy of five ML models for diagnosing DVT. These models include three traditional feed-forward ANN models — configured with 1, 2, and 3 hidden layers and denoted as ANN-I, ANN-II, and ANN-III, respectively — as well as two tree-based ensemble models, XGB and RF. The three ANNs, XGB and RF, undergo HP optimization parallelly to ensure a fair comparison of their performance in accurately classifying DVT.

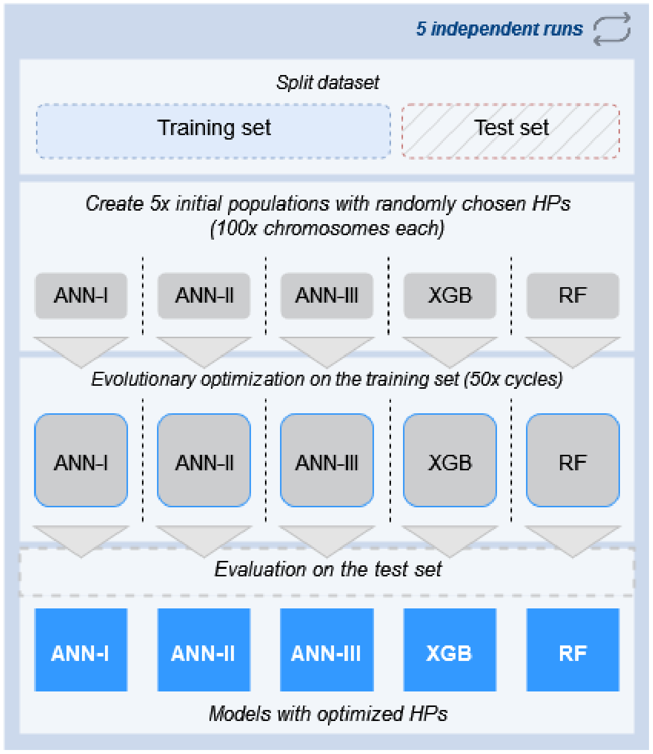

5 independent runs
Split dataset
Training set
Test set
Create 5x initial populations with randomly chosen HPs (100x chromosomes each)
ANN-I
ANN-II
ANN-III
XGB
RF
Evolutionary optimization on the training set (50x cycles)
ANN-I
ANN-II
ANN-III
XGB
RF
Evaluation on the test set
ANN-I
ANN-II
ANN-III
XGB
RF
Models with optimized HPs

Fig. 4: Implemented optimization process.

The EA is configured with specific parameters, including a population size of 100, 50 generations, and five runs with distinct random states. The parameters η_c (crossover distribution index) and η_m (mutation distribution index) are set to 20, while the crossover and mutation rates are established at 0.9 and 0.3, respectively. These parameters were chosen based on the previous experiments and research work [16], where similar configurations were found to be effective in achieving optimal results.

Fig. 4 illustrates the main stages of the implementation process. Five initial populations of ANN-I, ANN-II, ANN-III, XGB and RF models and the 60:40 training-test dataset split are generated at the beginning of each of the five runs using a new random state value. The evolutionary process unfolds in parallel, resulting in five unique sets of solutions during each cycle. A stratified 5-fold cross-validation approach is employed during training, ensuring an even data distribution across folds while maintaining consistent class distributions within each fold.

In each evolutionary cycle, the architectures of the ANN models are initialized from the information encoded in chromosomes. Subsequently, the ANNs undergo training, where the weights are adjusted through the learning process until a predefined stopping criteria is met. Similarly, the XGB and RF models are constructed with HPs derived from the chromosomes and trained using their respective algorithms. This iterative process ensures that the models evolve and improve performance over successive cycles. During the evolutionary optimization process, evolutionary operators such as crossover, mutation, SBX, and PM (depending on the type of gene data) are applied. Throughout the training phase, each solution is evaluated based on performance metrics derived from cross-validation, accurately representing the model's performance across the entire training set.

After each of the five independent runs, the final generation, which encapsulates HPs of ML models, is evaluated on a holdout test set. The results are then sorted in descending order based on the primary metric, accuracy, followed by the secondary metric, recall. The best-performing model is selected and its HPs and performance metrics are documented and presented in section IV.

D. Evaluation metrics

Our evaluation metrics primarily focus on *accuracy*, a standard measure of classification performance, and additionally include *recall* for a comprehensive comparison [35]. Accuracy measures the overall correctness of the model's predictions, representing the ratio of correctly predicted instances to the total number of instances. Accuracy provides a general overview of the model's performance but may not be suitable for imbalanced datasets, where one class dominates the other. Recall (also called sensitivity) measures the proportion of actual positive instances that are correctly identified by the model. Recall emphasizes the model's ability to capture positive instances accurately, without missing them.

For a detailed account of our findings during training sessions, we collected these metrics for each model: accuracy (mean, min, max, std) and recall (mean, min, max, std). The best HPs and classification metrics for each model, derived from each of the five independent runs, are reported, along with average values for the entire experiment. We also utilize Receiver Operating Characteristic *(ROC)* and Precision-Recall *(PRC)* plots to visually represent the training results, showcasing our models' discriminative capabilities [35]. ROC plots illustrate the trade-off between true positive rate and false positive rate, offering a comprehensive view of model's sensitivity across different decision thresholds. Conversely, PRC plots emphasize precision and recall, providing a more nuanced perspective on model performance, especially in scenarios with imbalanced class distributions [36].

A detailed breakdown of the model's predictive performance on the test set is provided through metrics such as *True Negatives (TN)*, *False Positives (FP)*, *False Negatives (FN)*, and *True Positives (TP)*. TN represents instances correctly identified as the negative class, while FP signifies instances incorrectly classified as positive. Conversely, FN represents instances erroneously classified as negative, and TP denotes instances correctly classified as positive. We calculate other classification metrics, such as *specificity*, *precision* and *False Positive Rate*, presented in tables and used for graphic plots. Additionally, *Area Under the Curve (AUC)* values for both ROC and PRC quantitatively measure the models' discriminatory capabilities. AUC ROC evaluates the trade-off between recall and specificity, while AUC PRC emphasizes precision and recall [35].

We employed McNemar's test [37] to systematically compare and evaluate the performance of the five ML models, utilizing a 95% confidence interval. McNemar's test is a statistical method suitable for comparing predictive models, and it is particularly useful for detecting differences in performance within paired datasets [38]. The outcomes of these comparative analyses are collected in contingency tables, offering a structured depiction of the models' classifications and highlighting areas of agreement and disagreement. These contingency tables serve as crucial elements in calculating McNemar's test statistic. McNemar's test assesses the significance of differences in predictive accuracy between paired models by focusing on the discordant cells. The test statistic (χ^2) is calculated as follows:

$$\chi^2 = \frac{(|b - c| - 1)^2}{b + c} \qquad (1)$$

where b represents the number of instances where one model predicts positive while the other predicts negative, and c represents the number of instances where one model predicts negative while the other predicts positive in the contingency table. This formula quantifies the discrepancy between the two models in their misclassifications, providing a statistical measure of the significance of the differences observed.

E. Data Source

Our research utilized Ri-Schedule data [19] - an Electronic Health Record (EHR) [39] dataset focusing on patients suspected of having DVT. This dataset was gathered at the Emergency Department of Østfold Hospital Trust in Sarpsborg, Norway. The original Ri-Schedule dataset consisted of 1800 patient records and 195 variables containing numerical and categorical data. These variables included personal details such as age, gender, weight, height, clinical symptoms, risk factors, vital signs, laboratory results, knee and ankle measurements, prescription and follow-up data. The binary target variable represented a positive or negative DVT diagnosis. The diagnosis was decided through D-dimer [40] values and confirmed with compression ultrasonography examinations.

Several steps were taken to pre-process the dataset for machine learning analysis. Duplicate entries were identified and removed, ensuring retention of the most complete or latest information for each patient ID. Subsequently, irrelevant attributes for ML analysis, those with highly sparse data or conflicting information, were eliminated. Additionally, two variables containing circumference measurements of left and right knees and ankles were combined into a new variable representing the absolute difference in these measurements. We employed univariate imputation to deal with missing fields, replacing them with mean, median, and mode values depending on the attributes' meaning and valid ranges. Following these pre-processing steps, the dataset comprised 1392 samples and 44 independent variables. The target attribute contained 1116 negative and 276 positive values. At the beginning of each experiment, the dataset was randomly split into training (60%) and test (40%) sets, maintaining class distribution through labels for stratified sampling. The training and test sets contained 835 and 557 samples, respectively, with a consistent negative-to-positive ratio (4:1). Finally, to ensure uniformity, the magnitudes of values across different independent variables were standardized to a range of [0, 1].

IV. Experimental Results and Analysis

With the goal of maximizing the predictive capabilities of machine learning models for diagnosing DVT on the Ri-Schedule dataset, this study utilized EA to fine-tune HPs of ANN-I, ANN-II, ANN-III, XGB and RF models. Our evaluation metrics primarily focus on *accuracy*, a standard measure of classification performance, and additionally include *recall* and other metrics for a comprehensive comparison.

A. Training results

The evolutionary approach resulted in ANN-I's mean accuracy of 88.22% and a mean recall of 51.80%. Similarly, ANN-II and ANN-III achieved mean accuracies of 88.38% and 88.22%, respectively, with slightly higher mean recalls of 54.71% and 54.96%, respectively. Despite architectural variations, the classic feed-forward ANN models exhibited comparable accuracies, suggesting robustness in their predictive capabilities and implying that the ANNs may have reached their peak performance in terms of accuracy. XGB and RF performed better than ANN models during the training phase. XGB, in particular, demonstrated performance with a mean accuracy of 89.05% and a mean recall of 64.82%. RF exhibited a mean accuracy of 89.15% and a mean recall of 56.74%. The detailed results are shown in Tables I-V. The tables also contain the tuned HPs for the models that achieved the highest accuracy in each run. In Tables I to III, the abbreviations LR, Neur, Act, Opt, Sig and LReLU stand for Learning Rate, Number of Neurons, Activation Function, Optimizer, Sigmoid and LeakyReLU respectively. Similarly, for Table IV, the abbreviations NE, MD, MCW, Sub, and CS represent the Number of Estimators, Maximum Depth, Minimum Child Weight, Subsampling Ratio, and Column Subsampling Ratio, respectively. Lastly, in Table V, the abbreviations NE, MF, C, MD, MSS, and MSL denote the Number of Estimators, Maximum Features, Criterion, Maximum Depth, Minimum Samples Split, and Minimum Samples Leaf, respectively.

In Figure 5, ROC and PRC plots are presented to visually compare the performance of each of the five ML models across five independent runs. XGB and RF consistently exhibit higher curves for both ROC and PRC, indicating greater discriminatory power in the DVT classification task on the Ri-Schedule dataset compared to ANN models. The clear delineation between ensemble tree-based and ANN models emphasize the effectiveness of tree-based ensemble learning techniques in handling the complexities of DVT prediction tasks.

B. Test results

The classification metrics for each model on the test set are presented in Table VI. The abbreviations used in these tables are as follows: True Negatives (TN), False Positives (FP), False Negatives (FN), True Positives (TP), Area Under the Receiver Operating Characteristic Curve (AUC ROC), and Area Under the Precision-Recall Curve (AUC PRC). The primary metrics considered were accuracy and recall, with supplementary metrics providing additional context. RF emerged as the best-performing model on the holdout test set, with a mean accuracy of 89.01% and a mean recall of 57.45%. RF consistently demonstrated a high accuracy and recall across different runs, showcasing its robustness in handling the DVT classification task. XGB closely followed RF, with a mean accuracy of 88.01% and a mean recall of 64.18%. XGB demonstrated high accuracy and particularly exceled in recall, indicating its effectiveness in correctly identifying positive instances.

The three ANNs exhibited competitive performance, with mean accuracies ranging from 86.82% to 87.86% and mean recalls ranging from 47.82% to 49.09%. While ANNs performed reasonably well, they generally lagged behind XGB and RF in terms of both accuracy and recall. The tree-based ensemble models showcased superior performance, particularly in the recall, indicating their efficacy in correctly identifying positive instances.

To conduct a McNemar's test, we created contingency tables that showed the number of cases where each model correctly or incorrectly predicted the outcome. Based on these tables, the test calculates a χ^2 statistic and p-values, which can provide insights into the significance of differences in predictive performance between model pairs. Table VII presents the mean values of the contingency tables, χ^2 values, and p-values based on the predictions of different model combinations across five different runs.

Initially, our analysis focused on the p-values obtained from McNemar's test. When comparing ANN models with XGB and RF, in most cases, the p-values were higher than 0.05, indicating no significant difference in performance between ANN models and XGB or RF. Expanding our analysis, we examined instances where ANN models had more incorrect predictions than XGB and RF, providing additional insights into relative performance. The pairwise comparison of the number of wrong predictions for each ANN model against the tree-based models revealed that in each combination of ANNs with XGB or RF models, both XGB and RF had fewer incorrect predictions. Overall, there was no consistent evidence across multiple runs indicating a significant difference in performance between ANN models and tree-based models based on McNemar's test statistical method. However, the analysis of instances of misclassification showed that ANN models exhibited a higher frequency of incorrect predictions than XGB and RF in all the comparisons.

TABLE I: Hyperparameters and classification metrics for ANN-I on training set.

Run	LR	Neur	Act	Opt	Accuracy				Recall			
					mean	min	max	std	mean	min	max	std
1	0.0242	1300	Tanh	Adam	0.8778	0.8323	0.9042	0.0259	0.4635	0.3636	0.5455	0.0821
2	0.0536	1670	LReLU	Adam	0.8790	0.8503	0.9222	0.0247	0.4824	0.3939	0.7273	0.1240
3	0.0267	1888	LReLU	Adam	0.8898	0.8683	0.9281	0.0206	0.5667	0.5000	0.7273	0.0822
4	0.0234	1438	Tanh	Adam	0.8743	0.8503	0.8982	0.0182	0.5121	0.4242	0.5758	0.0511
5	0.0272	819	Tanh	Adam	0.8898	0.8623	0.9281	0.0223	0.5656	0.4546	0.6765	0.0894
Mean					0.8822	0.8527	0.9162	0.0223	0.5180	0.4273	0.6504	0.0858

TABLE II: Hyperparameters and classification metrics for ANN-II on training set.

Run	LR	Neur		Act		Opt	Accuracy				Recall			
							mean	min	max	std	mean	min	max	std
1	0.0126	169	1896	Tanh	ReLU	Adam	0.8778	0.8264	0.8982	0.0264	0.5722	0.5152	0.6364	0.0409
2	0.0121	443	1607	Tanh	LReLU	Adam	0.8790	0.8443	0.9162	0.0252	0.4945	0.3030	0.6667	0.1468
3	0.0098	248	1603	Tanh	ReLU	Adam	0.8874	0.8623	0.9222	0.0198	0.6203	0.5455	0.6667	0.0502
4	0.0104	236	1775	Tanh	ReLU	Adam	0.8826	0.8623	0.9102	0.0184	0.5182	0.3939	0.6061	0.0732
5	0.0070	1876	152	Tanh	LReLU	Adam	0.8922	0.8683	0.9281	0.0227	0.5305	0.4242	0.6667	0.0845
Mean							0.8838	0.8527	0.9150	0.0225	0.5471	0.4364	0.6485	0.0791

TABLE III: Hyperparameters and classification metrics for ANN-III on training set.

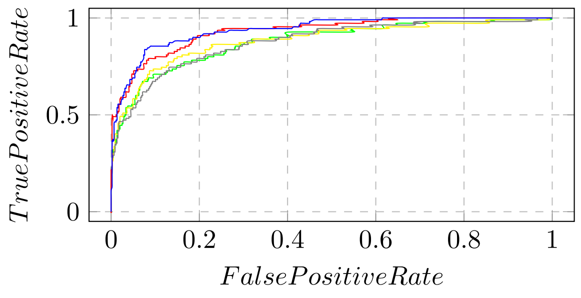
Run	LR	Neur			Act			Opt	Accuracy				Recall			
									mean	min	max	std	mean	min	max	std
1	0.0015	719	1910	632	Tanh	Tanh	Sig	Adam	0.8767	0.8503	0.8982	0.0154	0.5423	0.4546	0.6061	0.0511
2	0.0012	592	588	1936	Tanh	LReLU	Tanh	Adam	0.8790	0.8563	0.9162	0.0222	0.5184	0.3939	0.6667	0.1015
3	0.0019	1845	582	1449	Tanh	LReLU	Tanh	Adam	0.8862	0.8683	0.9162	0.0169	0.5845	0.5455	0.6364	0.0329
4	0.0023	1511	1736	464	Tanh	LReLU	Tanh	Adam	0.8767	0.8623	0.8862	0.0097	0.5549	0.4412	0.6667	0.0747
5	0.0039	1429	458	331	Tanh	LReLU	Sig	Adam	0.8922	0.8623	0.9281	0.0211	0.5480	0.4242	0.7273	0.1041
Mean									0.8822	0.8599	0.9090	0.0171	0.5496	0.4519	0.6606	0.0729

TABLE IV: Hyperparameters and classification metrics for XGB on training set.

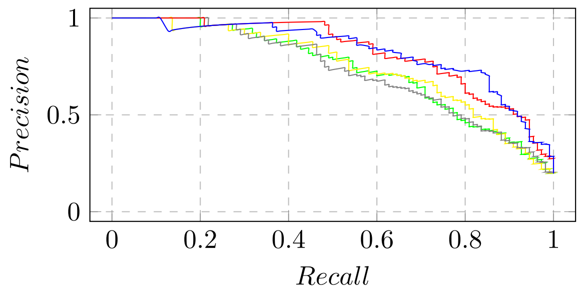
Run	LR	Gamma	NE	MD	MCW	Sub	CS	Accuracy				Recall			
								mean	min	max	std	mean	min	max	std
1	0.0165	0.1604	219	25	1	0.7	0.7	0.8778	0.8323	0.9042	0.0253	0.5902	0.4849	0.6667	0.0661
2	0.0206	0.4333	268	26	1	0.7	0.5	0.8862	0.8563	0.9102	0.0211	0.6264	0.4242	0.8182	0.1254
3	0.0198	2.9385	233	29	1	0.9	0.6	0.8946	0.8623	0.9162	0.0210	0.7168	0.6667	0.7576	0.0317
4	0.0623	0.3002	450	19	1	0.9	0.7	0.8922	0.8743	0.9102	0.0137	0.6390	0.5588	0.7273	0.0567
5	0.0373	0.2901	147	7	1	0.9	0.6	0.9018	0.8563	0.9401	0.0297	0.6688	0.5455	0.7576	0.0735
Mean								0.8905	0.8563	0.9162	0.0221	0.6482	0.5360	0.7455	0.0707

TABLE V: Hyperparameters and classification metrics for RF on training set.

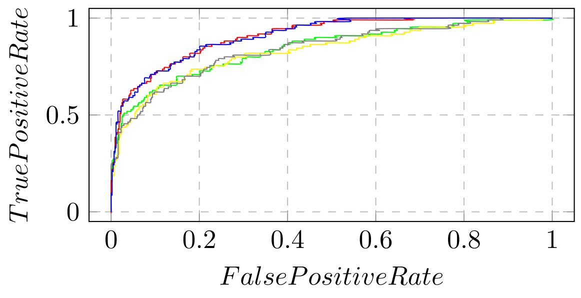
Run	NE	MF	C	MD	MSS	MSL	Accuracy				Recall			
							mean	min	max	std	mean	min	max	std
1	718	None	entropy	18	2	1	0.8814	0.8443	0.9042	0.0229	0.5358	0.4242	0.6364	0.0819
2	696	sqrt	entropy	18	4	2	0.8910	0.8683	0.9222	0.0179	0.5300	0.3939	0.6667	0.0898
3	486	sqrt	entropy	16	8	1	0.8970	0.8743	0.9162	0.0167	0.5840	0.4546	0.6471	0.0693
4	328	sqrt	entropy	14	9	2	0.8934	0.8743	0.9102	0.0116	0.5850	0.4706	0.6667	0.0644
5	259	sqrt	entropy	29	10	3	0.8946	0.8743	0.9162	0.0154	0.6023	0.5455	0.6667	0.0490
Mean							0.8915	0.8671	0.9138	0.0169	0.5674	0.4578	0.6567	0.0709



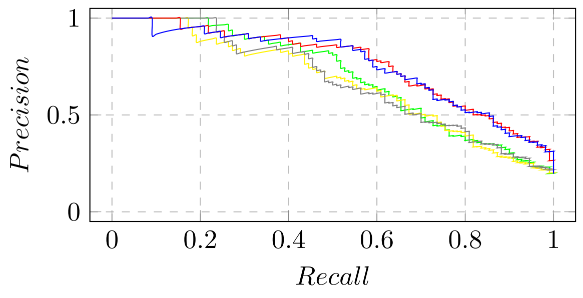
(a) Run 1 on training set.



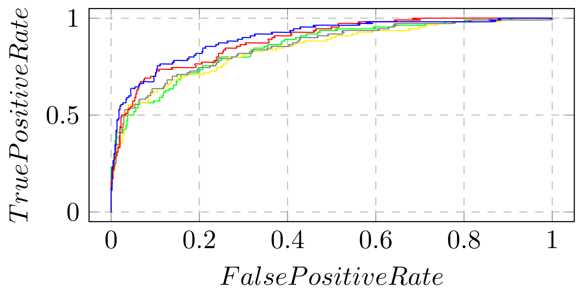
(b) Run 1 on training set.



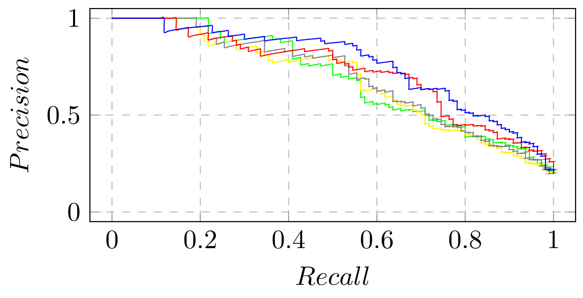
(c) Run 2 on training set.



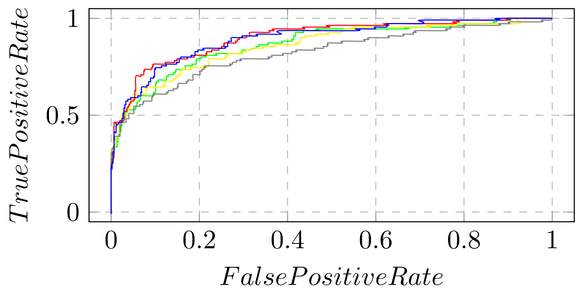
(d) Run 2 on training set.



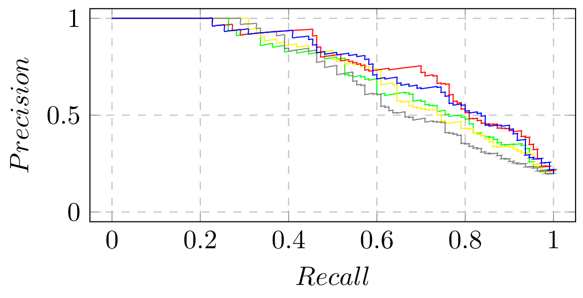
(e) Run 3 on training set.



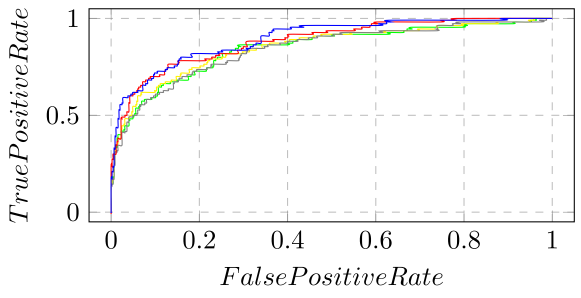
(f) Run 3 on training set.



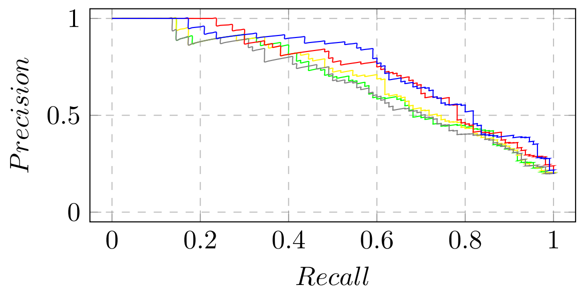
(g) Run 4 on training set.



(h) Run 4 on training set.



(i) Run 5 on training set.



(j) Run 5 on training set.

Fig. 5: ROC and PRC curves for ANN-I (green), ANN-II (yellow), ANN-III (gray), XGB (red) and RF (blue) for runs 1-5

TABLE VI: Mean values of classification metrics for ML models on test set.

Model	**TN**	**FP**	**FN**	**TP**	**Accuracy**	**Recall**	**Specificity**	**Precision**	**AUC ROC**	**AUC PRC**
ANN-I	431	16	57	53	0.8686	0.4782	0.9647	0.7728	0.8584	0.7057
ANN-II	433	14	58	52	0.8715	0.4764	0.9687	0.7922	0.8547	0.7030
ANN-III	430	17	56	54	0.8682	0.4891	0.9615	0.7619	0.8478	0.6914
XGB	420	27	39	71	**0.8801**	**0.6418**	0.9387	0.7255	0.9024	0.7663
RF	433	14	47	63	**0.8901**	**0.5745**	0.9678	0.8167	0.9080	0.7754

TABLE VII: Contingency table for McNemar's test (average values across 5 runs).

Model 1	**Model 2**	**Both correct**	**Model 1 wrong**	**Model 2 wrong**	**Both wrong**	χ^2	**p-value**
ANN-I	ANN-II	476.0	9.4	7.8	63.8	0.62	0.62
ANN-I	ANN-III	474.0	9.6	9.8	63.6	1.14	0.44
ANN-I	XGB	464.0	26.2	19.8	47.0	1.07	0.5
ANN-I	RF	470.4	25.4	13.4	47.8	3.81	0.21
ANN-II	ANN-III	475.6	8.0	9.8	63.6	0.39	0.67
ANN-II	XGB	463.4	26.8	22.0	44.8	0.89	0.43
ANN-II	RF	471.8	24.0	13.6	47.6	2.72	0.2
ANN-III	XGB	462.0	28.2	21.6	45.2	1.49	0.29
ANN-III	RF	470.4	25.4	13.2	48.0	3.87	0.13
XGB	RF	481.0	14.8	9.2	52.0	1.23	0.45

ANNs are generally less efficient than tree-based ensemble models like RF and XGB for tabular datasets with a relatively small number of samples and a mix of feature types (integer, real, categorical, Boolean). Tree-based models are more robust to the presence of uninformative or redundant features in the dataset. ANNs, on the other hand, struggle with such features, and their performance degrades significantly when uninformative features are present [41]. Tabular datasets often contain a mix of numerical (integer, real) and categorical (Boolean, ordinal, nominal) features. Tree-based models can naturally handle heterogeneous data types without the need for extensive feature engineering or encoding schemes, while ANNs require special architectures and techniques to handle such data effectively [42], [43]. Moreover, tree-based models are better suited for learning irregular and complex patterns in the data, which are common in tabular datasets. ANNs, particularly standard architectures, tend to learn overly smooth solutions and struggle with capturing such irregularities [41]. While deep learning approaches may achieve competitive performance on very large tabular datasets [42], tree-based ensembles like XGB remain the state-of-the-art for most small to medium-sized heterogeneous tabular datasets, which are common in many applications [43]. In summary, the robustness to uninformative features, ability to handle heterogeneous data types, capacity to learn irregular patterns, and efficiency with limited data and training time make tree-based ensemble models more effective than ANNs for tabular data such as the Ri-Schedule dataset.

V. Conclusion

The analysis of the models' performance on both the training and holdout test sets reveals that the tree-based ensemble models, XGB and RF, outperform conventional feed-forward ANNs for classifying DVT on Ri-Schedule data. Although ANNs exhibited comparable performance, their accuracy and recall were slightly lower than those of the tree-based models. XGB and RF consistently demonstrated superior performance across various evaluation metrics, showcasing their effectiveness in handling the complexities of the disease classification task. These tree-based ensemble models leverage the collective intelligence of decision trees, effectively capturing intricate patterns within the dataset and yielding higher predictive accuracy. However, the ANN models demonstrated stability and consistency across different HP configurations, suggesting their reliability in predictive tasks. The results of this study suggest that further research in enhancing DVT diagnostics on the Ri-Schedule dataset should explore tree-based ensemble methods, such as XGB and RF, rather than classic feed-forward ANNs. In conclusion, the comparative analysis provided insights into the relative performance of ANNs and tree-based ensemble methods for DVT diagnosis, highlighting the importance of algorithm selection in clinical decision-support systems.

References

[1] J. A. López, C. Kearon, and A. Y. Lee, "Deep venous thrombosis," *ASH Education Program Book*, vol. 2004, no. 1, pp. 439–456, 2004.

[2] J. Hirsh and A. Y. Lee, "How we diagnose and treat deep vein thrombosis," *Blood, The Journal of the American Society of Hematology*, vol. 99, no. 9, pp. 3102–3110, 2002.

[3] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[4] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[5] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33, 2019.

[6] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on Machine Learning*. PMLR, 2013, pp. 115–123.

[7] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for NAS," *arXiv preprint arXiv:1912.06059*, 2019.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.

[9] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[10] M. Uzair and N. Jamil, "Effects of hidden layers on the efficiency of neural networks," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*. IEEE, 2020, pp. 1–6.

[11] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[12] L. Tani, D. Rand, C. Veelken, and M. Kadastik, "Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics," *The European Physical Journal C*, vol. 81, pp. 1–9, 2021.

[13] E. Cantú-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 915–927, 2005.

[14] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-performance Computing Environments*, 2015, pp. 1–5.

[15] R. Sorano, L. V. Magnusson, and K. Abbas, "Comparing effectiveness of machine learning methods for diagnosis of deep vein thrombosis," in *International Conference on Computational Science and Its Applications*. Springer, 2022, pp. 279–293.

[16] R. Sorano, K. S. N. Ripon, and L. V. Magnusson, "Evolutionary multi-objective optimization of hyperparameters for decision support in healthcare," in *Handbook of Formal Optimization*. Springer, 2023, pp. 1–26.

[17] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[19] S. G. Fronas, A. E. A. Dahm, H. S. Wik, C. T. Jørgensen, J. Gleditsch, N. Raouf, R. Holst, F. A. Klok, and W. Ghanima, "Safety and feasibility of Rivaroxaban in deferred workup of patients with suspected deep vein thrombosis," *Blood Advances*, vol. 4, no. 11, pp. 2468–2476, Jun. 2020.

[20] X. Chen, M. Hou, and D. Wang, "Machine learning-based model for prediction of deep vein thrombosis after gynecological laparoscopy: A retrospective cohort study," *Medicine*, vol. 103, no. 1, p. e36717, 2024.

[21] E. E. Contreras-Luján, E. E. García-Guerrero, O. R. López-Bonilla, E. Tlelo-Cuautle, D. López-Mancilla, and E. Inzunza-González, "Evaluation of machine learning algorithms for early diagnosis of deep venous thrombosis," *Mathematical and Computational Applications*, vol. 27, no. 2, p. 24, 2022.

[22] Y. Fei, J. Hu, W.-Q. Li, W. Wang, and G.-Q. Zong, "Artificial neural networks predict the incidence of portosplenomesenteric venous thrombosis in patients with acute pancreatitis," *Journal of Thrombosis and Haemostasis*, vol. 15, no. 3, pp. 439–445, 2017.

[23] B. Kainz, M. P. Heinrich, A. Makropoulos, J. Oppenheimer, R. Mandegaran, S. Sankar, C. Deane, S. Mischkewitz, F. Al-Noor, A. C. Rawdin *et al.*, "Non-invasive diagnosis of deep vein thrombosis from ultrasound imaging with machine learning," *NPJ Digital Medicine*, vol. 4, no. 1, p. 137, 2021.

[24] T. D. Martins, J. M. Annichino-Bizzacchi, A. V. C. Romano, and R. Maciel Filho, "Artificial neural networks for prediction of recurrent venous thromboembolism," *International Journal of Medical Informatics*, vol. 141, p. 104221, 2020.

[25] L. Ryan, S. Mataraso, A. Siefkas, E. Pellegrini, G. Barnes, A. Green-Saxena, J. Hoffman, J. Calvert, and R. Das, "A machine learning approach to predict deep venous thrombosis among hospitalized patients," *Clinical and Applied Thrombosis/Hemostasis*, vol. 27, p. 1076029621991185, 2021.

[26] W. Sheng, X. Wang, W. Xu, Z. Hao, H. Ma, and S. Zhang, "Development and validation of machine learning models for venous thromboembolism risk assessment at admission: a retrospective study," *Frontiers in Cardiovascular Medicine*, vol. 10, 2023.

[27] J. Willan, H. Katz, and D. Keeling, "The use of artificial neural network analysis can improve the risk-stratification of patients presenting with suspected deep vein thrombosis," *British Journal of Haematology*, vol. 185, no. 2, pp. 289–296, 2019.

[28] K. Dembrower, Y. Liu, H. Azizpour, M. Eklund, K. Smith, P. Lindholm, and F. Strand, "Comparison of a deep learning risk score and standard mammographic density score for breast cancer risk prediction," *Radiology*, vol. 294, no. 2, pp. 265–272, 2020.

[29] V. E. Staartjes, M. P. de Wispelaere, W. P. Vandertop, and M. L. Schröder, "Deep learning-based preoperative predictive analytics for patient-reported outcomes following lumbar discectomy: feasibility of center-specific modeling," *The Spine Journal*, vol. 19, no. 5, pp. 853–861, 2019.

[30] T. M. Dumont, A. I. Rughani, and B. I. Tranmer, "Prediction of symptomatic cerebral vasospasm after aneurysmal subarachnoid hemorrhage with an artificial neural network: feasibility and comparison with logistic regression models," *World Neurosurgery*, vol. 75, no. 1, pp. 57–63, 2011.

[31] K. Sekaran, P. Chandana, N. M. Krishna, and S. Kadry, "Deep learning convolutional neural network (CNN) with Gaussian mixture model for predicting pancreatic cancer," *Multimedia Tools and Applications*, vol. 79, no. 15-16, pp. 10 233–10 247, 2020.

[32] A. F. Cooper, Y. Lu, J. Forde, and C. M. De Sa, "Hyperparameter optimization is deceiving us, and how to stop it," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3081–3095, 2021.

[33] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[34] K. Deb, M. Goyal *et al.*, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[35] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[36] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PloS One*, vol. 10, no. 3, p. e0118432, 2015.

[37] B. S. Everitt, *The analysis of contingency tables*. CRC Press, 1992.

[38] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[39] Health IT Playbook, "Electronic health records," https://www.healthit.gov/playbook/electronic-health-records/, accessed: April 4, 2024.

[40] E. D. Johnson, J. C. Schell, and G. M. Rodgers, "The D-dimer assay," *American Journal of Hematology*, vol. 94, no. 7, pp. 833–839, 2019.

[41] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" *Advances in Neural Information Processing Systems*, vol. 35, pp. 507–520, 2022.

[42] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[43] S. Marton, S. Lüdtke, C. Bartelt, and H. Stuckenschmidt, "GRANDE: Gradient-based decision tree ensembles," *arXiv preprint arXiv:2309.17130*, 2023.