Deep Learning Algorithm for Hypertension Prediction using Hyperparameter Optimization via Latin Hypercube Sampling in High-Performance Computing Environments

Bruno Victor Paiva da Silva
PPgEEC
UFRN
Natal, RN, Brazil
bruno.silva.704@ufrn.edu.br

Jhoan Fernandes de Oliveira
DETEC
UFERSA
Pau dos Ferros, RN, Brazil
jhoan.oliveira@alunos.ufersa.edu.br

Felipe Hidequel Santos da Silva
PPgEEC
UFRN
Natal, RN, Brazil
felipe.santos.087@ufrn.edu.br

Thiago Vinicius Paiva da Silva
DETEC
UFERSA
Pau dos Ferros, RN, Brazil
thiago.silva80312@alunos.ufersa.edu.br

Artur Santos Nogueira
DETEC
UFERSA
Pau dos Ferros, RN, Brazil
artur.nogueira32975@alunos.ufersa.edu.br

Walber Jose Adriano Silva
DETEC
UFERSA
Pau dos Ferros, RN, Brazil
walber.silva@ufersa.edu.br

***Abstract*—Hypertension is a chronic condition characterized by persistently elevated blood pressure. Due to its asymptomatic nature in the early stages, it is frequently underdiagnosed, leading to diagnosis only after the occurrence of cardiovascular complications. This silent progression highlights the need for early diagnostic tools capable of identifying at-risk individuals before irreversible damage occurs to target organs. To fill this clinical gap, this study proposes a Deep Learning Algorithm for Hypertension Prediction based on clinical variables. A Hyperparameter Optimization process was conducted to define the architecture and adjust the predictive capacity. The search space was systematically explored using Latin Hypercube Sampling (LHS) to define the neuron configuration, learning rate, dropout, and L2 regularization. Due to the computational complexity of evaluating multiple neural network architectures through 10-fold Stratified Cross-Validation, the optimization pipeline was implemented in High-Performance Computing (HPC) environments. The model achieved an accuracy of 90.21% and an F1-Score of 90.19%, demonstrating the balance achieved between precision and recall.**

***Index Terms*—deep learning, hypertension, high-performace computing, otimization, latin hypercube.**

I. Introduction

Hypertension is a chronic condition defined by elevated blood pressure [1]. It becomes more prevalent with aging and is considered a multifactorial disease, with oxidative stress being one of its main contributing factors [2]. According to Mills, Stefanescu, and He (2020) [3], hypertension is a leading cause of cardiovascular diseases and premature mortality worldwide. The progression of the disease is often asymptomatic, which leads to late diagnosis, frequently occurring only after complications in target organs have developed.

The fact that the disease begins before it is diagnosed highlights the importance of early detection. Identifying hypertension in its initial stages enables timely intervention, improving disease management and reducing mortality rates.

Artificial Intelligence (AI) has emerged as a powerful technology capable of analyzing large volumes of healthcare data to identify complex and predictive patterns [4].

Within this context, Machine Learning (ML), a subfield of AI, provides a set of techniques for building predictive models. Among these, Artificial Neural Networks (ANNs) represent a class of computational models inspired by the functioning of the human brain.

ANN architectures vary according to the type of data and tasks they address. Convolutional Neural Networks (CNNs) are particularly effective in processing spatial data, especially images, while Recurrent Neural Networks (RNNs) are designed for sequential data, such as text [5].

Among these architectures, the Multilayer Perceptron (MLP) stands out as a general-purpose model, commonly applied to classification and regression problems involving structured data. It is also recognized as one of the most versatile neural network architectures in terms of applicability [6].

MLPs have been widely used in diagnostic applications. However, the complexity of healthcare data and the need for more robust processing motivate the exploration of advanced approaches. In this context, the present study focuses on the development of a Multilayer Perceptron model optimized through structured hyperparameter tuning techniques, supported by high-performance computing resources, aiming to improve the accuracy and generalization of hypertension risk

prediction.

II. Theoretical review

A. Hypertension

Hypertension, also known as high blood pressure, is a condition in which the blood pressure in the arteries remains persistently elevated. According to Williams et al. (2018) [7], hypertension can be classified into three stages based on blood pressure levels, the presence of cardiovascular risk factors, hypertension-mediated organ damage, or comorbidities. The first stage, termed uncomplicated, corresponds to hypertensive patients without target-organ damage or associated diseases. The second stage, termed asymptomatic disease, is characterized by the presence of target-organ damage, stage 3 chronic kidney disease, or diabetes mellitus without organ damage. The third stage, considered the most advanced, is called established disease and is associated with the presence of established cardiovascular disease, chronic kidney disease of stage 4 or higher, or diabetes mellitus with target-organ damage.

B. Neural Network

In risk classification, blood pressure levels are associated with various additional factors, such as advanced age, excessive salt consumption, sedentary lifestyle, among others. To perform an aggregated analysis of these factors, computational methods such as Artificial Neural Networks (ANNs) can be applied.

The first artificial neural network architecture was proposed in 1943 by McCulloch and Pitts, designed as a simplified computational model intended to represent the cooperative functioning of biological neurons in animal brains. This model aimed to demonstrate the capacity of artificial neurons to perform complex calculations based on propositional logic [8].

The Multilayer Perceptron (MLP) is one of the simplest ANN architectures (Géron, 2021). An MLP is composed of an input layer, one or more hidden layers, and a final layer called the output layer. The layers closest to the input layer are referred to as lower layers, while the layers closer to the output layer are called upper layers. Except for the output layer, all other layers are fully connected.

MLPs are generalizations of linear models that perform multiple processing steps to reach a decision [9], in which the prediction made by a linear regression is presented in Equation 1.

$$\hat{y} = w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_p \cdot x_p + b \qquad (1)$$

Where $\hat{y}$ is a weighted sum of the input features x_0 to x_p, weighted by the learned coefficients w_0 to w_p. In an MLP, the process of calculating weighted sums is repeated multiple times: first, computing hidden units that represent an intermediate processing stage, which are then recombined using weighted sums to generate the final result.

III. Related works

This section presents scientific publications related to the topic of this research. The work of Zhao et al. (2021) [10], is presented, in which the authors predict hypertension risks using machine learning models and compare their performance . Next, we present the work of Marin and Goga [11], in which hypertension prediction is performed using a dataset composed of different population groups and four machine learning algorithms. Moving on, we have the work of Dond (2025) [12] and Gong, Chen and Ding (2023) [13] in which they explore machine learning techniques related to blood pressure measurements and lifestyle habits.

Detailing the work, the study by Zhao et al. (2021) investigates the prediction of hypertension risk, a widely prevalent chronic condition, using machine learning models capable of operating with simple and easily collected risk factors. Using a dataset composed of 29,700 samples obtained from physical examinations, the authors initially selected the main risk factors through univariate logistic regression.

Based on these selected variables, four algorithms Random Forest, CatBoost, a multilayer perceptron (MLP) neural network, and logistic regression—were optimized using 10-fold cross-validation for hyperparameter tuning. The final performance was evaluated using metrics such as the area under the curve (AUC), which measures the model's overall ability to discriminate between classes, as well as accuracy, sensitivity, and specificity, which measures the model's ability to correctly identify negative samples in the test set.

Among the evaluated models, Random Forest achieved the best performance, as presented in Table I.

TABLE I
Performance metrics reported by Zhao et al. [10] for a classical machine learning model evaluated on a dataset of 29,700 samples.

Metric	Value
Accuracy	82%
Sensitivity	83%
Specificity	81%
AUC	92%

The study by Marin and Goga (2019) aimed to apply machine learning techniques to identify hypertension across different population groups, including the general population, hypertensive patients, and pregnant women with or without preeclampsia. This work utilized public datasets from the University of Massachusetts Amherst. The study by Dond (2025) explores the use of machine learning techniques to detect hypertension, leveraging the capability of algorithms to analyze large datasets, identify patterns, and make predictions based on various clinical and lifestyle factors. The patient data include blood pressure measurements, demographic information, and lifestyle habits.

Gong, Chen, and Ding (2023) propose a more reliable method for hypertension prediction based on non-invasive signals, such as electrocardiogram (ECG) and photoplethysmo-

gram (PPG). The authors employ causal inference techniques to select variables that are truly related to hypertension, rather than relying solely on correlation. The study is based on data from the National Health and Nutrition Examination Survey. To obtain the results, four machine learning algorithms were employed. The first, Gaussian Naive Bayes, is a probabilistic classifier. The second and third, logistic regression and Random Forest, are based on statistical modeling and ensemble learning, respectively. Finally, Support Vector Machines (SVM) operates by finding an optimal decision boundary between data classes. The results showed that the models were effective in identifying individuals without hypertension but correctly classified only approximately half of the cases in which hypertension was present, indicating a limitation in sensitivity for positive detections. Nevertheless, the authors emphasize that the automatic detection of abnormal health parameters using machine learning techniques is an important tool for monitoring and preventing critical health issues across different population groups.

Based on the reviewed studies, it can be observed that there is extensive literature employing machine learning techniques to classify the risk of hypertension using selected clinical and lifestyle-related parameters, with the support of high-performance computing tools to identify optimal hyperparameters. These studies generally report performance results above 80%, often focusing on the application of deep learning algorithms. The primary objective across these works is the early detection of hypertension risk, enabling preventive interventions at early stages.

IV. Development

The methodology of this work followed the practical steps of a complete machine learning project, as described in Chapter 2 of [14].

A. Problem overview

In the problem overview phase, the aim was to understand and establish the scientific and practical motivation for the study. The main objective of the method is to develop a deep learning model to detect the risk of hypertension in its early stages.

B. Data acquisition

From the second stage onwards, we use a dataset from a public database, which addresses clinical data of patients. The database used was provided by the Kaggle platform and is called: Hypertension Risk Prediction Dataset. It contains, among the data provided, 1985 records, 2 binary classes, and 11 influence features generated from clinical insights and public health data patterns. The variables in the database are:

1) *Age*: Patient's age (in years);
2) *Salt_Intake*: Daily salt intake (in grams);
3) *Stress_Score*: Scale from 0 to 10 measuring psychological stress level;
4) *BP_History*: Previous blood pressure history: normal, prehypertension, hypertension;
5) *Sleep_Duration*: Average number of hours of sleep per day;
6) *BMI*: Body Mass Index (BMI), a measure of obesity based on weight/height;
7) *Medication*: Type of medication used (None, Beta-blocker, Diuretic, ACE inhibitor, Other);
8) *Family_History*: Family history of hypertension;
9) *Exercise_Level*: Level of physical activity (Low, Moderate, High);
10) *Smoking_Status*: Smoker or Non-smoker;
11) *Has_Hypertension*: Presence of hypertension;

C. Data exploration and visualization

In stage 3, data exploration, the objective was to understand the distribution of features and identify possible class imbalances. The dataset was loaded using the pandas library, chosen because, according to Wes (2013), it is widely recognized as one of the main data manipulation tools in Python [15]. Pandas enables vectorized operations, the generation of descriptive statistics, and the creation of visualizations directly from the data.

In the context of this work, pandas was used to load the hypertension prediction dataset, analyze the class distribution, and identify imbalances. Class balance was evaluated using a distribution graph, as illustrated in the Figure 1.

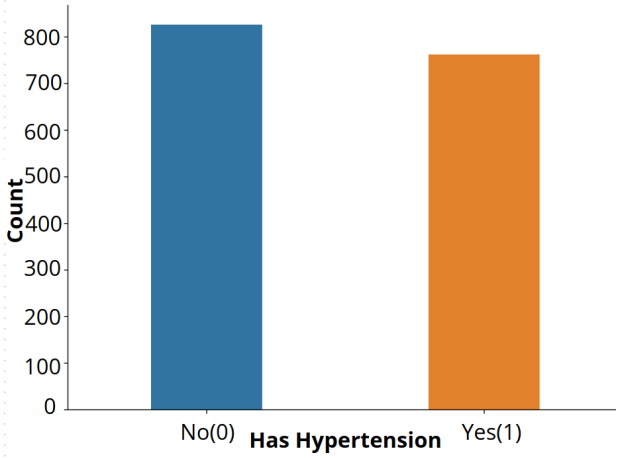


Fig. 1. Distribution of classes before the application of the SMOTE technique, illustrating the imbalance between the binary classes (No and Yes) for hypertension prediction.

D. Data preparation for Machine Learning algorithms

After data visualization, transformations were applied to the available dataset in order to standardize and preprocess the data for model training. Initially, categorical variables were converted into numerical format using one-hot encoding, and the target variable values "Yes" and "No" were mapped to 1 and 0, respectively. This conversion is necessary because machine learning models operate on numerical values. Next, the data were split into training and testing sets using the

train_test_split function from the scikit-learn library [16], which was divided into 80% training data and 20% test (validation) data, in order to evaluate the model's generalization capability. The training set was used to learn patterns, while the test set was kept separate for validation. This separation helps mitigate issues such as overfitting.

During the preparation of the training set, a class imbalance was observed in the target variable, with a predominance of samples from patients without hypertension. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) [17] was applied, which generates synthetic samples of the minority class based on interpolation between nearest neighbors. This approach balances the classes and improves the model's ability to correctly identify positive hypertension cases.

Finally, the data were scaled using the StandardScaler method, also from the scikit-learn library. This procedure transforms each feature to have zero mean and unit variance, preserving the original data distribution while ensuring that all features contribute proportionally during the training process.

E. Model selection and training

In step 5, the architecture of the machine learning model responsible for making predictions was defined. A multilayer perceptron (MLP) network was selected due to its ability to model non-linear relationships between variables and adapt to different patterns present in the data. The model was implemented using the TensorFlow and Keras libraries [18], which are widely used in deep learning applications. This choice provided adequate control over network layers, activation functions, and training parameters, in addition to offering support for acceleration via Graphics Processing Units (GPUs), thereby reducing processing time.

1) Neural network architecture construction: To define the optimal deep learning model architecture, a structured hyperparameter optimization process was implemented using the Optuna framework [19]. The search space was explored using the Latin Hypercube Sampling (LHS) technique [20], which ensures a statistically uniform exploration of the parameter space.

A total of 30 independent trials were conducted, in which the algorithm simultaneously optimized the number of neurons (32 to 128), the learning rate (5×10^{-4} to 5×10^{-3}), the dropout rate (0.2 to 0.5), and the L2 regularization coefficient (10^{-4} to 10^{-2}).

To perform the model validation, a 10-fold Stratified Cross-Validation approach was adopted. This stratified method ensures that the proportional representation of the 'Yes' and 'No' classes is maintained across all 10 folds, avoiding biased training splits. Furthermore, instead of using a traditional exhaustive Grid Search, which is computationally prohibitive for continuous hyperparameter ranges, the search space was explored using the Latin Hypercube Sampling (LHS) technique [20]. The objective of the search was to maximize the model's F1-score across the folds, ensuring robust generalization. The model was structured with dense layers using the ReLU (Rectified Linear Unit) activation function in the hidden layers, aiming to mitigate the vanishing gradient problem and accelerate convergence. For the output layer, the sigmoid activation function was used, which is suitable for binary classification tasks. To prevent overfitting and optimize training time, an early stopping mechanism was adopted, monitoring the validation loss. Training was interrupted if no improvement was observed after a defined patience period, ensuring that the model retained the weights corresponding to its best generalization performance.

F. Fine-tuning

In stage 6, fine-tuning, techniques established in the literature were employed to mitigate overfitting. In particular, the dropout technique was used, which consists of randomly deactivating neurons during training, forcing the network to learn more robust and independent representations. This approach was combined with L2 regularization (weight decay), which penalizes large weights, thereby smoothing the model's decision function.

G. Solution presentation

In step 7, after training and fine-tuning the model, the obtained results were organized and presented for interpretation. In this stage, the objective was to validate the model's performance and demonstrate the effectiveness of the applied techniques. Evaluation metrics such as accuracy, precision, recall, and F1-score were used, along with the confusion matrix. These metrics enable a detailed analysis of the model's behavior, considering both correct and incorrect predictions, while the confusion matrix provides a class-wise breakdown of predictions, highlighting true positives, false positives, true negatives, and false negatives.

H. System deployment, monitoring, and maintenance

The final deployment phase consisted of structuring the computational environment to ensure reproducibility on HPC infrastructures. The execution of multiple neural network architectures requires significant computational resources, making parallel programming paradigms essential to distribute the workload efficiently, a foundational concept detailed by Pacheco (2011) [21]. The model was encapsulated using automation scripts managed by SLURM, which handle resource allocation, execution environment initialization, and job monitoring. The system was configured to generate detailed performance logs, enabling the tracking of metric convergence during training. Finally, maintenance is supported by storing the optimized hyperparameters in configuration files, allowing the model to perform inference on new data without requiring the re-execution of the optimization pipeline.

V. Results

To obtain the results, experiments were conducted on the supercomputer of the Federal University of Rio Grande do Norte, at NPAD (Nucleo de Processamento de Alto Desempenho), using a computational node from the AMD 512

partition with 256 GB of RAM. Based on the model training and the Figure 2, it was possible to obtain the loss function graph, which represents how much the model is wrong in its predictions.

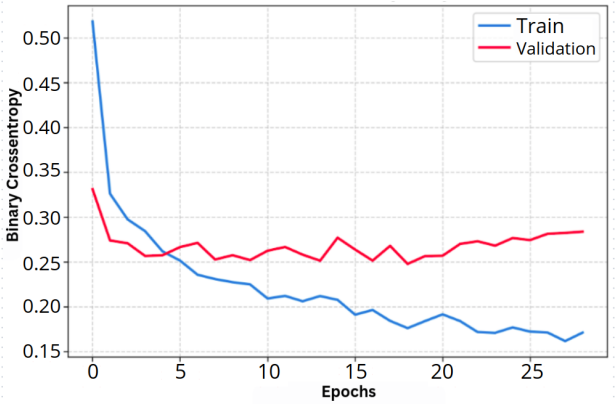


Fig. 2. Graph of the loss function, showing the lines during the periods relating to training and validation.

Additionally, the evolution of accuracy over the training epochs was recorded, as illustrated in the Figure 3, showing the accuracy values for both the training and validation sets.

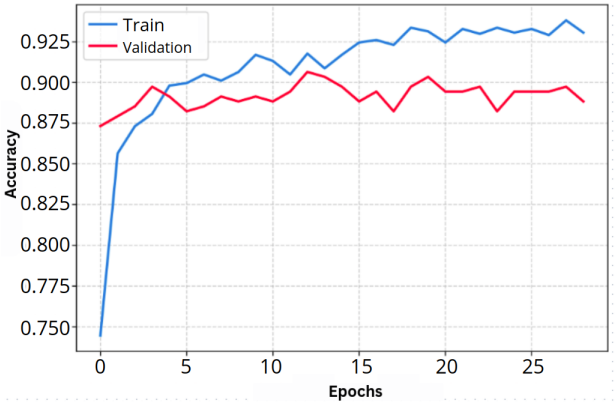


Fig. 3. Graph of the acurracy, showing the lines during the periods relating to training and validation.

It is observed that the loss decreases consistently, converging to a stable level for both the training and validation sets, indicating that the model has learned representative patterns from the data without exhibiting underfitting. Simultaneously, the accuracy shows an increasing trend, stabilizing at values above 90%. The close alignment between the training and validation curves demonstrates the effectiveness of the regularization techniques (dropout and L2) as well as the early stopping mechanism, ensuring that the model achieves robust generalization without showing signs of overfitting or divergence, even after extended training.

Through cross-validation, more reliable estimates of the model's performance were obtained for the accuracy, precision, recall, and F1-score metrics, reducing the dependence on a single data split. Table II presents the values of these metrics.

TABLE II
MODEL PERFORMANCE METRICS OBTAINED USING STRATIFIED 10-FOLD CROSS-VALIDATION.

Metric	Value
Accuracy	0.9021
Precision	0.9030
Recall	0.9008
F1-Score	0.9019

From Table II, it can be observed that the evaluation metrics are well balanced, indicating consistent model performance. The model achieved accuracy, precision, recall, and F1-score values of approximately 90%. In addition to the metrics obtained through cross-validation, the model's confusion matrix was also computed. As shown in Figure 4, it provides a detailed view of correct and incorrect predictions, including true positives, false positives, true negatives, and false negatives.

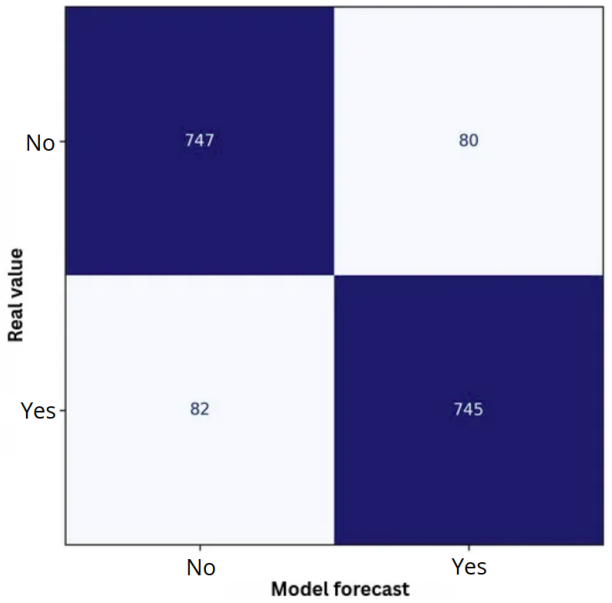


Fig. 4. Matrix Confusion (10 Folds).

From Figure 4, it can be observed that most predictions are concentrated along the main diagonal, corresponding to true positives and true negatives. In contrast, the off-diagonal elements indicate a smaller number of false positives and false negatives. Overall, the model produced 1,492 correct predictions and 162 incorrect predictions, demonstrating strong classification performance.

VI. CONCLUSION

This work aimed to develop a Multilayer Perceptron neural network model using the Optuna hyperparameter optimizer

and the Latin Hypercube Sampling technique to find the best combination of hyperparameters. The model's efficiency was evaluated using classification metrics, specifically assessing the number of correct and incorrect predictions, precision, recall, and F1-score. Results above 90% indicate that the model achieved a high rate of correct predictions compared to false negatives and positives. Based on the results obtained, future work proposes testing the model with larger datasets, as well as using other optimization strategies, such as Bayesian Optimization. Furthermore, to address the limitations of generating synthetic medical data via SMOTE, future studies will implement class weighting directly into the loss function to handle data imbalance dynamically. Another contribution would be to utilize traditional algorithms known in the literature, such as Random Forest, Decision Tree, and Gradient Boosting Machine, in an ensemble analysis.

VII. Acknowledgments

I would like to thank the LAPPS (Laboratory of Parallel Architectures for Signal Processing) coordination for granting NPAD access to conduct scientific research.

References

[1] J. Krzemińska, M. Wronka, E. Młynarska, B. Franczyk, and J. Rysz, "Arterial hypertension—oxidative stress and inflammation," *Antioxidants*, vol. 11, no. 1, p. 172, 2022.

[2] A. J. Donato, D. R. Machin, and L. A. Lesniewski, "Mechanisms of dysfunction in the aging vasculature and role in age-related disease," *Circulation research*, vol. 123, no. 7, pp. 825–848, 2018.

[3] K. T. Mills, A. Stefanescu, and J. He, "The global epidemiology of hypertension," *Nature reviews nephrology*, vol. 16, no. 4, pp. 223–237, 2020.

[4] E. J. Topol, "High-performance medicine: the convergence of human and artificial intelligence," *Nature medicine*, vol. 25, no. 1, pp. 44–56, 2019.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] F. A. Gomide, "Redes neurais artificiais para engenharia e ciências aplicadas: curso prático," 2012.

[7] B. Williams *et al.*, "2018 ESC/ESH Guidelines for the management of arterial hypertension: The Task Force for the management of arterial hypertension of the European Society of Cardiology (ESC) and the European Society of Hypertension (ESH)," *European Heart Journal*, vol. 39, no. 33, pp. 3021–3104, 2018. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/30165516/

[8] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[9] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[10] H. Zhao *et al.*, "Predicting the risk of hypertension based on several easy-to-collect risk factors: A machine learning method," *Frontiers in Public Health*, vol. 9, p. 619429, 2021.

[11] I. Marin and N. Goga, "Hypertension detection based on machine learning," in *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*, 2019, pp. 1–4.

[12] V. T. Dond, "Using predictive analytics to detect hypertension with machine learning," *International Journal of Scientific Research in Engineering and Management*, 2025, unverified reference.

[13] K. Gong, Y. Chen, and X. Ding, "Causal inference for hypertension prediction," in *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2023, pp. 1–4.

[14] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly Media, 2019.

[15] W. McKinney, *Python for Data Analysis*. O'Reilly Media, 2012.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[20] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[21] P. Pacheco and M. Malensek, *An introduction to parallel programming*. Morgan Kaufmann, 2021.