Stage 3

Machine Learning Course

Major: Cloud Computing and Cybersecurity (Group CCC1)

Supervised By:

Nédra MELOULLI
Christophe RODRIGUES

Written By:

Alex QUILIS VILA
Carolina ROMERO DELGADO
Livia GAMERO HAMMERER

Date Last Edited: December 5, 2024

Contents

Chapter 1

Introduction and Business Scope

Heart disease remains one of the leading causes of death worldwide, highlighting the importance of early diagnosis to prevent fatalities and improve patient outcomes. This project aims to develop a machine learning model capable of predicting heart disease based on patient data. The goal is to assist healthcare professionals in identifying at-risk individuals and providing timely interventions. By achieving this, the solution seeks to reduce the strain on healthcare systems through early prevention and personalized treatment while enhancing overall patient care.

Chapter 2

Previous methods and limitations

2.1 Previous Methods

- Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), LightGBM, CatBoost, XGBoost and ensamble methods like Stacking Classifier and Voting Classifier. Those were implemented to predict heart disease.
- The Stacking Classifier was identified as the best-performing model with satisfactory metrics.

2.2 Limitations

However, many of these models struggle with class imbalance, leading to poor predictive performance for minority classes.

- Decision Tree risked overfitting due to its simplicity, even with limited depth.
- Random Forest provided competitive performance but required higher computational resources.
- Logistic Regression was insufficient for capturing non-linear relationships.
- KNN: High computational cost for large datasets.
- LightGBM, CatBoost and XGBoost: Require expertise in hyperparameter tuning for optimal performance.
- Ensemble approaches can increase complexity and runtime.

Chapter 3

Improvement Assumptions

To address these limitations, an advanced algorithms was considered, it was Neural Network with the following assumption: We developed a neural network model to enhance the performance of our classification task. The decision to use this model was based on its ability to capture complex non-linear relationships in the data, which simpler models might overlook. By addressing potential underfitting and leveraging techniques like dropout and SMOTE for class imbalance, we aimed to achieve better generalization and improved predictive accuracy.

Chapter 4

Problem formalisation and methods

4.1 New Algorithm description

A neural network is a computational model inspired by the way biological neural networks in the human brain work. It consists of layers of nodes, or "neurons," each of which performs a mathematical operation. These neurons are organized into layers: an input layer, one or more hidden layers, and an output layer. Each layer transforms the data it receives, with the goal of mapping input data to desired output, typically for tasks like classification, regression, or pattern recognition.

In a neural network, each connection between neurons has a weight that is adjusted during training. The training process involves using labeled data to adjust the weights through an algorithm called backpropagation, which helps minimize errors in the network's predictions. The network learns by updating its weights to reduce the difference between its predicted outputs and the true labels, typically through optimization algorithms like gradient descent.

For our heart disease prediction model, we use a neural network to classify whether a patient has heart disease or not. We input various health-related features into the network, such as age, cholesterol levels, blood pressure, and other relevant factors. The network processes these inputs to make a prediction on whether the patient is likely to have heart disease.

Our model consists of an input layer where each feature, such as age or cholesterol levels, is represented by a separate neuron. These inputs are then passed through one or more hidden layers, where the network learns the complex relationships between the different features. Finally, the output layer contains a single neuron that produces a value between 0 and 1, which represents the

probability of the patient having heart disease. A value closer to 0 indicates no heart disease, while a value closer to 1 indicates the presence of heart disease.

To capture complex relationships in the data, we use activation functions like ReLU (Rectified Linear Unit) in the hidden layers. This introduces non-linearity into the model, allowing it to learn more intricate patterns. In the output layer, we use the Sigmoid activation function to convert the network's output into a probability value, making it suitable for the binary classification task of predicting heart disease.

During the training process, we use backpropagation to adjust the weights of the network based on the error between the predicted output and the actual result. To minimize this error, we use an optimization algorithm like Adam, which updates the weights in a way that reduces the loss function, improving the accuracy of our model in predicting the presence or absence of heart disease.

4.2 Limitations

Although neural networks are powerful tools for making predictions, they come with certain limitations, especially when applied to tasks like heart disease prediction.

- Overfitting: Neural networks are highly flexible, which can lead them to learn not only the true patterns in the data but also the noise and small fluctuations. This makes the model perform well on the training data but struggle to generalize to unseen data. Regularization techniques like dropout and simplifying the network can help mitigate this issue.
- Requirement for Large Datasets: Neural networks often need large amounts of data to effectively learn patterns. If the dataset is too small, the model may not capture all the relevant patterns, leading to poor performance. In our case, the heart disease dataset may not provide enough examples to capture every possible relationship in the data.
- Interpretability: Neural networks are often referred to as "black boxes" because it's difficult to understand how the model makes predictions. In a sensitive domain like healthcare, it's important to know why a model makes a specific prediction.
- Computational Cost: Training neural networks, especially with large datasets and deep architectures, can be computationally expensive. It requires powerful hardware like GPUs or even distributed computing, which may not always be available, making it a limiting factor in resource-constrained environments.
- Sensitivity to Hyperparameters: Neural networks are sensitive to hyperparameters such as the learning rate, number of layers, and number of

neurons in each layer. The performance of the model can be heavily influenced by these choices, and finding the optimal combination requires experimentation and can be time-consuming. Incorrect hyperparameter choices can result in suboptimal performance.

Chapter 5

Methodology

5.1 Data preprocessing

- **Handling null and duplicate values:** Duplicates in the dataset were removed using drop_duplicates() to avoid any interference in the analysis.
- **Converting categorical variables to numerical values:** Several techniques were used to convert categorical variables into numeric values:
  - Binary columns (such as 'Yes' and 'No') were mapped to 1 and 0. The Sex column was mapped to 0 for "Female" and 1 for "Male".
  - The AgeCategory column was transformed into a numeric variable by calculating the average value of each age range.
  - The GenHealth column was assigned numeric values using an ordinal mapping.
  - The Diabetic column was mapped to numeric values based on the type of diabetes.
  - The Race column was encoded using LabelEncoder to convert categorical values into numeric ones.
- **Handling class imbalance with SMOTE:** SMOTE (Synthetic Minority Over-sampling Technique) was applied to balance the classes in the training data, generating synthetic examples for the minority class.
- **Scaling numerical features:** The StandardScaler was used to scale numerical features, ensuring the values are centered around 0 with a standard deviation of 1, which is useful for many machine learning algorithms, including neural networks.

5.2 Training and testing split

- 80% training, 20% testing.

- Applied cross-validation for robust evaluation.

5.3 New Algorithm Implementation and Hyperparameters

A sequential model is used, meaning the layers are stacked in a linear fashion (one on top of the other). This type of architecture is suitable for problems where the data flows sequentially through the network without branches or complex structures.

Model Layers

- **Input Layer:**
  - **Number of Neurons:** 64
  - **Activation Function:** ReLU (Rectified Linear Unit)
  - **Description:** The input layer receives the features of the data (in this case, scaled data). Each of the 64 neurons is connected to all the inputs of the model. The ReLU activation function is used to introduce non-linearity and allow the network to learn complex relationships between the inputs.
- **Hidden Layer:**
  - **Number of Neurons:** 32
  - **Activation Function:** ReLU
  - **Description:** This layer is responsible for learning more abstract representations of the data. Like the input layer, it uses the ReLU activation function to introduce non-linearity and improve the network's ability to learn complex patterns.
- **Dropout Layer:**
  - **Dropout Rate:** 50%
  - **Description:** The dropout layer is used to prevent overfitting. During training, 50% of the neurons are randomly turned off, forcing the network to learn more robust representations without over-relying on certain features of the data.
- **Output Layer:**
  - **Number of Neurons:** 1
  - **Activation Function:** Sigmoid

- **Description:** The output layer has only one neuron because the problem is a binary classification problem. The sigmoid activation function converts the output of the neuron into a value between 0 and 1, which can be interpreted as the probability that an instance belongs to a specific class.

Hyperparameters

- **Number of Epochs:**
  - **Value:** 20
  - **Description:** The number of epochs indicates how many times the entire training data is processed. Each epoch corresponds to one full pass through the dataset. In this case, the model will be trained for 20 epochs.
- **Batch Size:**
  - **Value:** 128
  - **Description:** The batch size defines how many samples are processed in each training step before updating the model weights. A batch size of 128 means that 128 examples will be processed before performing an update.
- **Optimizer:**
  - **Value:** Adam
  - **Description:** Adam is an adaptive optimizer widely used in neural networks. It combines the advantages of two other optimizers (Adagrad and RMSprop) to maintain a dynamically adjusting learning rate. This optimizer is effective for training neural networks on complex problems.
- **Loss Function:**
  - **Value:** Binary Crossentropy
  - **Description:** Binary crossentropy is a loss function used for binary classification problems. It compares the model's predictions with the true labels and computes a value that represents the difference between them. The goal is to minimize this difference during training.

Chapter 6

Results

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Neural network	0.7400	0.2000	0.6500	0.3100	0.7000

Table 6.1: Model Results

6.1 Metrics

- Accuracy: Provided the overall correctness of predictions.
- Precision: Measured the relevance of positive predictions.
- Recall: Evaluated the model's ability to capture true positives.
- F1-Score: Balanced metric for precision and recall.
- ROC-AUC: Assessed the performance of models by comparing true positive rates against false positive rates.

6.2 Overfitting

The models demonstrated minimal signs of overfitting, with the ensemble models showing more balanced performance across the evaluation metrics compared to the individual models. While the Stacking classifier performed the best overall in terms of accuracy, the Voting classifier achieved the highest ROC AUC score, indicating strong generalization across various metrics.

6.3 Evaluation Comparison with Previous Solutions

Models	Accuracy	Precision	Recall	F1-Score	ROC AUC
Logistic Regression	0.6995	0.1820	0.6661	0.2859	0.6844
Random Forest	0.6968	0.1952	0.7547	0.3101	0.7229
Decision Tree	0.6526	0.1797	0.7987	0.2935	0.7184
KNN	0.7407	0.1778	0.5160	0.2644	0.6395
LightGBM	0.7230	0.1984	0.6796	0.3071	0.7035
CatBoost	0.7443	0.2025	0.6231	0.3056	0.6897
XGBoost	0.7497	0.2025	0.6031	0.3032	0.6837
Voting classifier	0.7454	0.2028	0.6204	0.3057	0.7729
Stacking classifier	0.8178	0.1984	0.3345	0.2491	0.7518
Neural network	0.7400	0.2000	0.6500	0.3100	0.7000

Table 6.2: Models Results

Chapter 7

Discussion and Conclusion

7.1 Discussion

- The Neural Network model achieved competitive performance with an accuracy of 74.00%, a precision of 20.00%, a recall of 65.00%, and an F1-score of 31.00%, indicating its capability to capture complex patterns in the data. Its ROC AUC score of 0.7000 shows it performed well at distinguishing between the classes but did not outperform the ensemble models.
- Among the individual models, XGBoost and LightGBM continued to demonstrate their robustness with balanced accuracy and F1-scores. However, their performance was slightly eclipsed by ensemble models in terms of both accuracy and ROC AUC.
- The Voting Classifier stood out with the highest ROC AUC (0.7729), emphasizing that combining classifiers improves the ability to discriminate between classes effectively. Its overall metrics suggest it is a strong candidate for tasks where reliable class separation is crucial.
- The Stacking Classifier, despite having a lower F1-score (0.2491), achieved the highest accuracy (81.78%) and a strong ROC AUC (0.7518), making it suitable for scenarios prioritizing accuracy over precision or recall.
- The ensemble approaches (Voting and Stacking) outperformed most individual models, including the Neural Network, further confirming the benefits of leveraging diverse classifiers to enhance prediction robustness.

7.2 Conclusion

From the results, we observe the following key insights:

- **Ensemble Models Outperform Individual Models:** The Voting and Stacking Classifiers remain the top-performing approaches, with the Voting Classifier achieving the highest ROC AUC (0.7729) and the Stacking Classifier the highest accuracy (81.78%). These models effectively leverage the strengths of multiple classifiers.
- **Neural Network as a Competitive Candidate:** The Neural Network demonstrated competitive results, with an accuracy comparable to the best-performing individual models (e.g., XGBoost and LightGBM). However, it fell short of ensemble models in key metrics like ROC AUC and accuracy, suggesting the need for optimization to unlock its full potential.
- **Recommendation for Ensemble Methods:** Given the overall performance, ensemble approaches (Voting and Stacking Classifiers) remain the best choice for this task, balancing accuracy, class separation, and robustness. Future work could explore integrating Neural Networks into ensemble frameworks to potentially improve results further.