Stage 4

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 19, 2024

Contents

1 Introduction and Business Scope **3**

2 Models used **4**

3 Problem formalisation and methods **5**
3.1 Algorithms description 5
3.2 Limitations 6

4 Methodology **7**
4.1 Data preprocessing 7
4.2 Training and testing split 7
4.3 Algorithms Implementation and Hyperparameters 8

5 Results **10**
5.1 Metrics 10
5.2 Overfitting 10
5.3 Evaluation Comparison with Previous Solutions 10

6 Discussion and Conclusion **12**
6.1 Discussion 12
6.2 Conclusion 12

List of Tables

5.1 Models Results 11

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

Models used

Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), LightGBM, CatBoost, XGBoost, ensamble methods like Stacking Classifier and Voting Classifier and neural networks. Those were implemented to predict heart disease.

Chapter 3

Problem formalisation and methods

3.1 Algorithms description

- **Logistic Regression**: A linear model used for probabilistic predictions.
- **Decision Tree**: A non-linear model that splits data into branches based on feature thresholds.
- **Random Forest**: An ensemble method that builds multiple trees and aggregates their results for better accuracy and robustness.
- **K-Nearest Neighbors (KNN)**: A non-parametric algorithm that classifies a data point based on the majority class of its nearest neighbors.
- **LightGBM**: A gradient-boosting framework optimized for speed and performance with categorical data support.
- **CatBoost**: A gradient-boosting algorithm designed specifically to handle categorical features efficiently.
- **XGBoost**: An optimized gradient-boosting framework that uses parallel processing, tree pruning, and regularization techniques to enhance performance.
- **Ensemble Model**: Techniques that combine multiple base models to improve predictive performance compared to using a single model.
  - Voting Classifier: Ensemble method that combines multiple base models and makes a final prediction based on their individual predictions. There are two types of voting:

  - Stacking Classifier: Ensemble technique where multiple base models are trained and their predictions are used as input to a final model (called the meta-model). The meta-model learns how to optimally combine the predictions of the base models.

- **Neural network**: 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.

3.2 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.
- Computational Cost: Training neural networks, especially with large datasets and deep architectures, can be computationally expensive.

Chapter 4

Methodology

4.1 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.

4.2 Training and testing split

- 80% training, 20% testing.

- Applied cross-validation for robust evaluation and to help reduce overfitting.

4.3 Algorithms Implementation and Hyperparameters

The following models were implemented with their respective hyperparameters:

- **Logistic Regression**:
  - **Hyperparameters**: Maximum iterations (max_iter = 1000) and regularization parameter ($C = 0.1$).
- **Random Forest**:
  - **Hyperparameters**: Number of estimators ($n\_estimators = 100$), maximum depth (max_depth = 10), and minimum samples required to split (min_samples_split = 10).
- **Decision Tree**:
  - **Hyperparameters**: Maximum depth (max_depth = 5).
- **KNN**:
  - **Key Hyperparameter**: `n_neighbors` set to 5, representing the number of neighbors considered.
  - **Preprocessing**: Features normalized using `StandardScaler` to ensure consistent distance calculations.
- **LightGBM**:
  - **Key Hyperparameter**: `num_leaves` set to 31 (for optimal trade-off between complexity and speed).
  - **Learning Rate**: Set to 0.05 for better performance with fewer iterations.
  - **Preprocessing**: Categorical features were handled automatically by LightGBM, no manual encoding required.
- **CatBoost**:
  - **Key Hyperparameter**: `iterations` set to 1000, `learning_rate` set to 0.1 for optimal trade-off between speed and performance.
  - **Preprocessing**: Categorical features automatically detected and processed without manual encoding.
- **XGBoost**:

- **Key Hyperparameter**: `max_depth` set to 6, `learning_rate` set to 0.1 for fast convergence.
- **Regularization Parameters**: `alpha` set to 0.5 to prevent overfitting, `lambda` set to 0.5 to control model complexity.

- **Neural networks**:
  - **Input Layer:**
    - Number of Neurons: 64
    - Activation Function: ReLU (Rectified Linear Unit)
  - **Hidden Layer:**
    - Number of Neurons: 32
    - Activation Function: ReLU
  - **Dropout Layer:**
    - Dropout Rate: 50%
  - **Output Layer:**
    - Number of Neurons: 1
    - Activation Function: Sigmoid
  - **Hyperparameters**:
    - Number of Epochs: 20
    - Batch Size: 128
    - Optimizer: Adam
    - Loss Function: Binary Crossentropy

Chapter 5

Results

5.1 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.

5.2 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.

5.3 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 5.1: Models Results

Chapter 6

Discussion and Conclusion

6.1 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.

6.2 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.