Random Forest Algorithm with Bayesian Optimisation for Heart Failure Prediction

Student ID: 23205123

COMP0172: AI for Biomedicine & Healthcare, University College London

Module Lead: Dr. Petru Manescu

27th November 2023

Word Count: 1999 words

Contents

Introduction

Heart failure affects over 64 million people [1] forming the leading cause of hospitalization worldwide [2]. Heart failure results in significant morbidity and mortality presenting a considerable social and economic burden, amounting to 2% of the total health costs in developed countries [2][1]. Alarmingly deaths from cardiovascular disease (CVD) have increased by 60% since 1990s, rising from 12.1 million to 20.5 million by 2021. Subsequently, this rise in heart disease has led to a growth in biomedical data, increasing the complexity of assessing heart failure by identifying an overwhelming number of potential contributing factors, such as gender [3], smoking [4], and physical activity [5]. The intricacy of the cardiovascular system and the complex causes of heart failure require the amalgamation of various data streams, including clinical test results, medical imaging, and patient demographic information. [6]. This has made it challenging to pinpoint the main contributing factors of heart failure reliably and manually. Therefore, building an effective strategy that can utilize large amounts of data for the preliminary detection of disease, the assessment of heart disease severity and the early estimation of adverse events is becoming progressively required.

How can ML help?

One tool that has been used to address this is machine learning (ML). ML has attempted to detect heart disease in previous cases [7][8][9][10] drawing significant attention in medical diagnostics. This is because ML can recognize patterns that are not immediate to human practitioners [11], by handling large volumes of data, and navigating the complex interactions of the numerous factors associated with heart failure. One such ML model implemented, is the Random Forest (RF) model. Within this study, we explore the reliability of RF for predicting heart failure considering data quality, feature engineering, model performance and hyperparameter tuning with Bayesian Optimisation.

Methodology

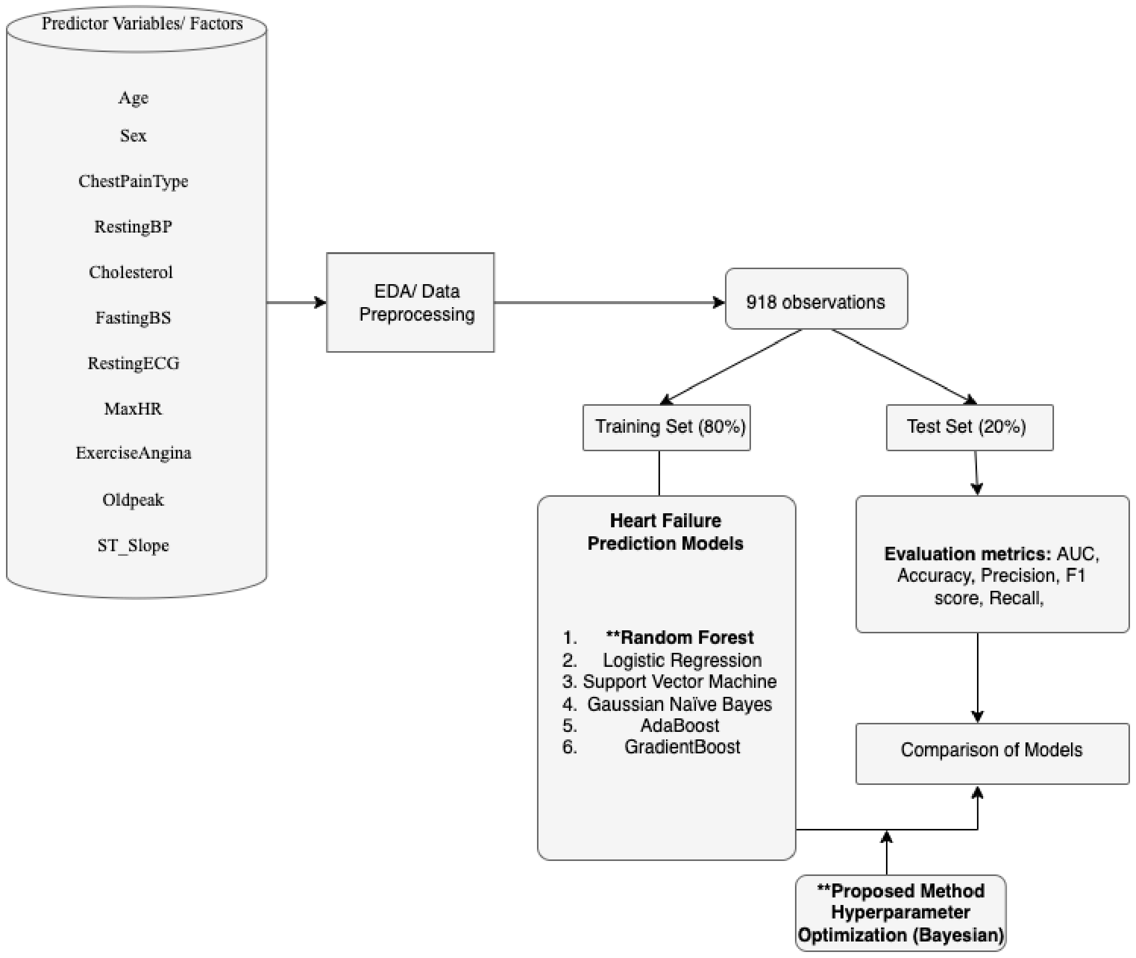


Figure 1: Machine Learning Pipeline for Heart Failure Prediction. This figure illustrates the methodology overview from data collection to model evaluation.

Dataset Description & Exploratory Data Analysis

With the Heart Failure prediction dataset from Kaggle [12], we performed EDA to investigate the characteristics of 918 observations which contain 11 predictor variables and 1 binary target feature. From Table 1, the dataset's predictor variables included continuous numerical variables such as age, resting blood pressure, cholesterol and maximum heart rate achieved (MaxHR). It contains categorical variables such as sex and 'chestpaintype'. The binary target feature 'HeartDisease' indicates whether the individual has heart disease (1) or not (0). Preliminary checks include establishing the coherence of dataset by looking at missing values, duplicated rows and significant outliers, shown in Figure 3. However, upon removing outliers, dataset reduced to 518 observations which might decrease dataset integrity so these observations are retained. No missing values or duplicated rows were identified. In Appendix 1.5, with Figures 10 and 11 show additional dataset characteristics.

Table 1: The heart disease dataset attributes detailed information

Attribute	Dataset Description	Values/Range
Age	Numerical and Continuous	28–77
Sex	Categorical with cardinality 2	M or F
ChestPainType	Categorical with cardinality 4	ATA, NAP, ASY, TA
RestingBP	Numerical and Continuous	0–200 [mm Hg]
Cholesterol	Numerical and Continuous	0–603 [mm/dl]
FastingBS	Numerical and Binary	0 or 1
RestingECG	Categorical with cardinality 3	Normal, ST, LVH
MaxHR	Numerical and Continuous	60–202
ExerciseAngina	Categorical and Binary	Yes or No
Oldpeak	Numerical and Continuous	−2.6 to 6.2
ST_Slope	Categorical with cardinality 3	Up, Flat or Down
Heart Disease	Numerical and Binary	0 or 1

With Pearson's Correlation Coefficient metric, we used seaborn heatmap function to visualize the relationships between the predictor and target variables. It can be seen in Figure 2, that all predictors are moderately correlated with 'HeartDisease' except for RestingECG (0.057). Despite this, we have maintained this variable in our dataset as numerous literature have demonstrated that this is a significant feature in heart failure prediction [13][14][15].

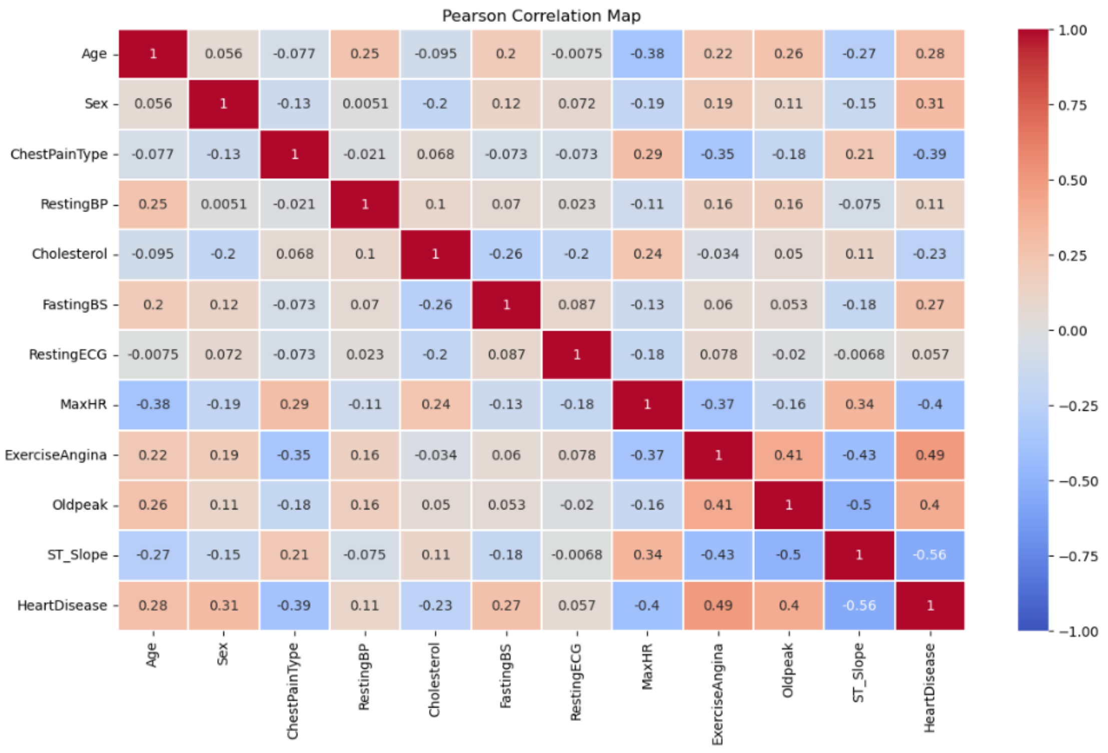


Figure 2: Pearons's Correlation Coefficient Heatmap of 'heart.csv' illustrating relationship between individual predictor variables and target variable against each other

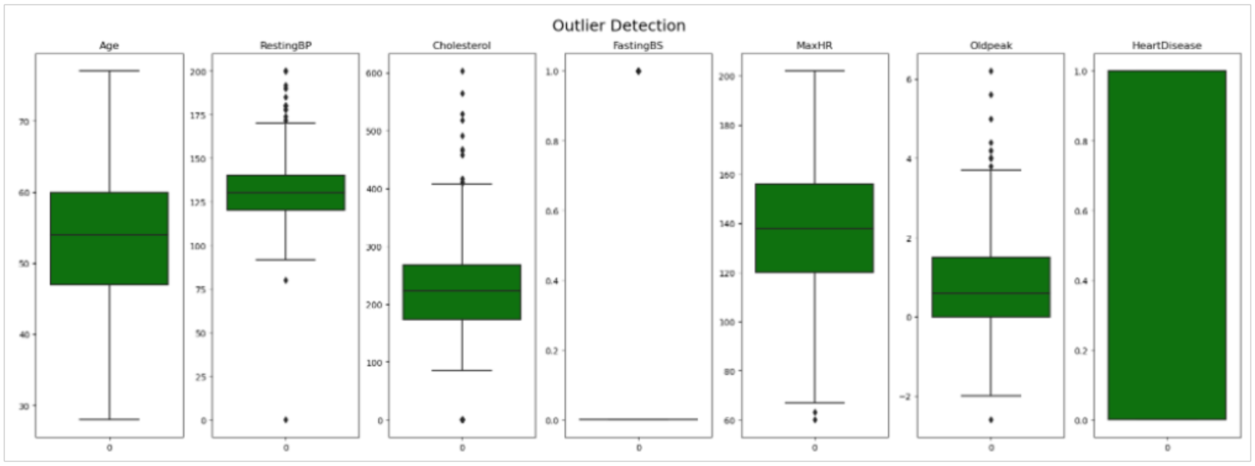
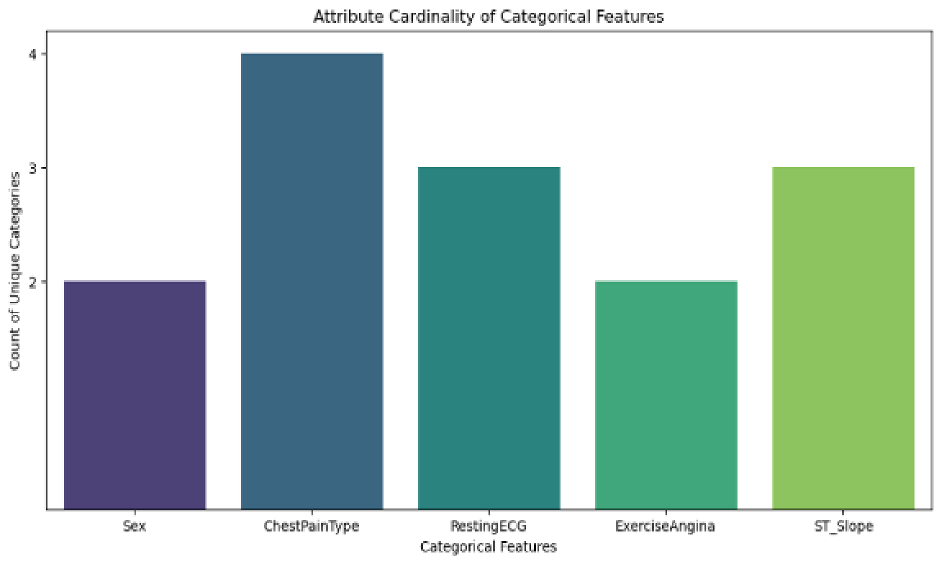


Figure 3: Outliers detected in 'heart.csv' dataset where RestingBP and Cholesterol show the highest number of outliers

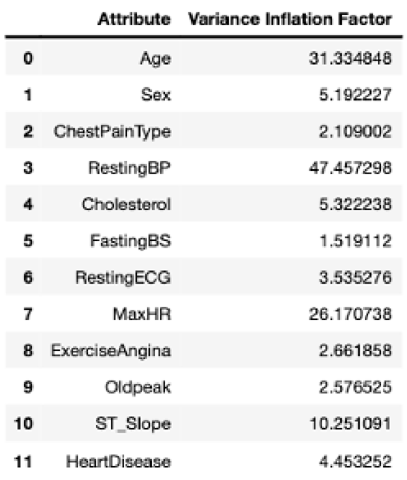
Data Preprocessing: Feature Selection & Engineering

The categorical features underwent cardinality checks to determine the suitability of encoding techniques,shown in Figure 4A. Low cardinality in features 'Sex', 'ChestPainType', 'RestingECG', and 'ST_slope' indicated one-hot encoding could be applied without concerns of dimensionality explosion [16]. We encoded categorical variables using one-hot-encoding. To diagnose multicollinearity, complementary to Pearson's heatmap, we used variance inflation factor (VIF) to prevent inaccurate prediction results in the ML models. Multicollinearity is present when VIF is higher than 10 [17]. In Figure 4B, age, MaxHR and RestingBP show significantly high values (31.33, 26.17, and 47.45 respectively). However, when these factors were compared to the heatmap and literature [18][19][20], they demonstrate a relationship between these factors and heart failure. Therefore, we retained the 3 factors. From the preprocessing steps, the new dataset is shown below in Figure 5.

A)

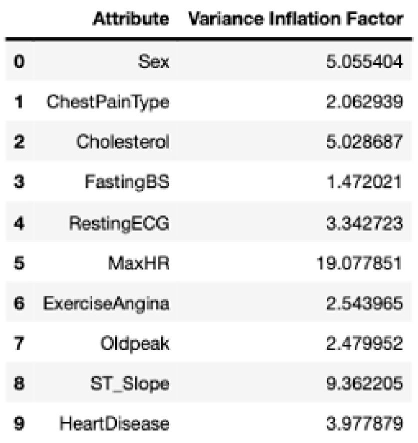


B)



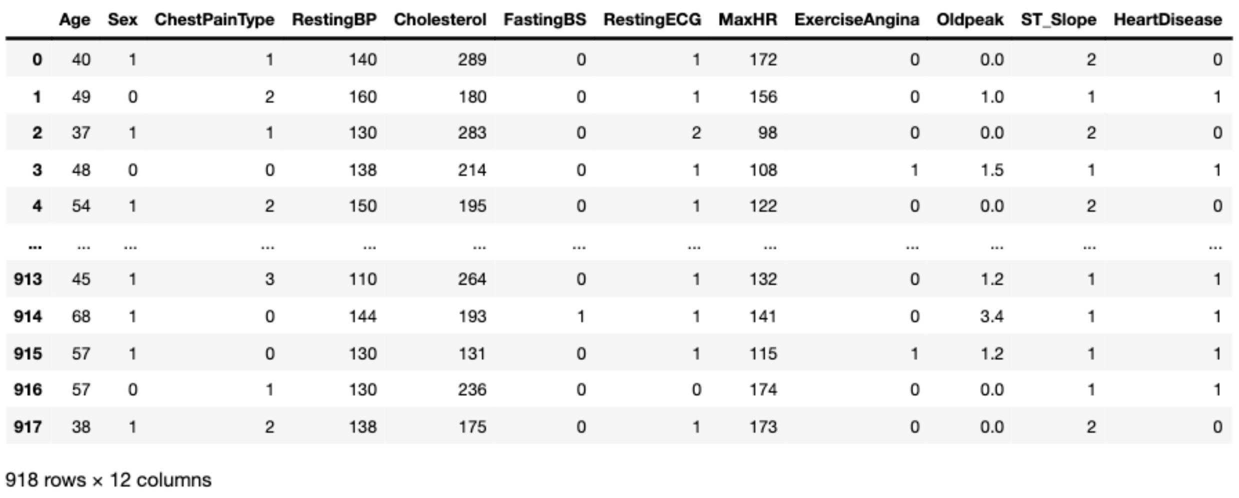
	Attribute	Variance Inflation Factor
0	Age	31.334848
1	Sex	5.192227
2	ChestPainType	2.109002
3	RestingBP	47.457298
4	Cholesterol	5.322238
5	FastingBS	1.519112
6	RestingECG	3.535276
7	MaxHR	26.170738
8	ExerciseAngina	2.661858
9	Oldpeak	2.576525
10	ST_Slope	10.251091
11	HeartDisease	4.453252

C)



	Attribute	Variance Inflation Factor
0	Sex	5.055404
1	ChestPainType	2.062939
2	Cholesterol	5.028687
3	FastingBS	1.472021
4	RestingECG	3.342723
5	MaxHR	19.077851
6	ExerciseAngina	2.543965
7	Oldpeak	2.479952
8	ST_Slope	9.362205
9	HeartDisease	3.977879

Figure 4: A) Bar chart illustrating cardinality of 5 categorical features- how many unique variables present; B) Variance Inflation factor (VIF) of features in 'heart.csv' dataset; C) VIF of features in 'heart.csv' when the high VIF attributes: Age and RestingBP are removed



	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	1	1	140	289	0	1	172	0	0.0	2	0
1	49	0	2	160	180	0	1	156	0	1.0	1	1
2	37	1	1	130	283	0	2	98	0	0.0	2	0
3	48	0	0	138	214	0	1	108	1	1.5	1	1
4	54	1	2	150	195	0	1	122	0	0.0	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	1	3	110	264	0	1	132	0	1.2	1	1
914	68	1	0	144	193	1	1	141	0	3.4	1	1
915	57	1	0	130	131	0	1	115	1	1.2	1	1
916	57	0	1	130	236	0	0	174	0	0.0	1	1
917	38	1	2	138	175	0	1	173	0	0.0	2	0

918 rows × 12 columns

Figure 5: Preprocessed 'heart_num_filtered.csv' dataset all with numeric columns for ML implementation

Machine Learning Models Implementation

This study is designed to evaluate the performance of Random Forest (RF) algorithm against 5 other established binary classification ML models. We hypothesize that RF algorithm would be the best suited for this binary classification task based on numerous literature which utilize analogous heart datasets [21][22][23]. RF is an effective algorithm for binary classification tasks because it builds multiple decision trees and merges them together for a stable prediction. RF has the capacity to handle large datasets with numerous variables making it particularly suitable for complex medical datasets where multiple factors influence the outcome [22].

The dataset was randomized and partitioned into training and test set using train_test_split() function, 80% and 20% respectively, where the test set is utilized for evaluation purposes. The StandardScaler() function was used for feature scaling (normalization), to ensure that every feature varies within the same range. For ML implementation, this ensures that features with wider numerical ranges do not dominate, which may bias analysis towards less informative attributes. Feature scaling will be able to mitigate the phenomenon and consequently improve ML classification performance [24][25]. Each model was trained using the same training and test set to standardize the performance evaluation.

The 5 other supervised learning algorithms, extracted from relevant literature on specific analogous heart failure datasets, were Logistic Regression [26], Gaussian Naïve Bayes[27], AdaBoost [28], GradientBoost [29], and Support Vector Machines [30]. In Appendix 1.2, it explains how these 5 algorithms perform this binary classification task.

Appropriate Evaluation Metrics

The appropriate evaluation metrics hinges on the data characteristics and the model's binary classification task. For this study, we employed 5 metrics on the test set to assess the RF model's performance against 5 other algorithms. From Figure 6, our imbalanced dataset with differing counts of presence of heart disease, indicates that we needed more than just accuracy as it can be misleading alone, since it does not distinguish between numbers of correctly classified examples of different classes. [31]. The area under receiver operating (AUC) curve provides a robust measure, evaluating models at various classification thresholds, which is more informative than just the F1 score [32]. Precision and recall will offer additional insights into model's ability to correctly identify positive cases and trade-off between the two. These metrics, detailed in Appendix 1.3, are required for evaluation of the model's capabilities where balance between sensitivity (recall) and precision is important [33].

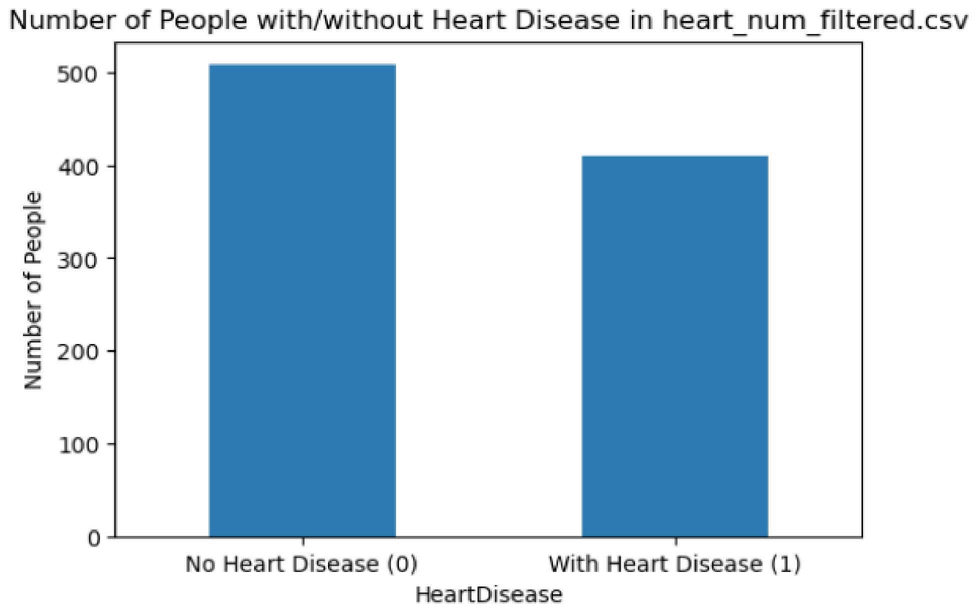


Figure 6: Barplot of Counts of 0's and 1's in the preprocessed 'heart_num_filtered.csv'

Bayesian Optimization Hyper-parameter Tuning of Random Forest

For our proposed method RF, we wanted to verify whether hyperparameter optimization would increase the model's performance significantly, as suggested in literature [34][35][36]. For RF, the hyperparameter space was carefully defined into 4 key parameters of the RandomForestClassifier which are n_estimators, max_depth, min_samples_split and max_features [34]. These were set up with the code in Appendix 1.4, with specified bounds to ensure computational efficiency. The objective function for Bayesian optimization

(BO) was established using the negative mean squared error (MSE), derived from a 3-fold cross-validation of the training set. Less negative MSE indicates higher accuracy so the aim is to maximise the objective function. Best parameters were retrieved after 5 initial random runs and 15 BO steps. Model with tuned hyperparameters were assessed on the test set, with same metrics as described above.

Results: How does Random Forest perform?

Table 2 compares RF to other ML models. LR balances well, with 84.78% accuracy and 86.27% F1 score, yet recall issues affect its F1 score. RF outshines LR, boasting the top accuracy (88.59%) and F1 score (90.32%), reflecting better precision-recall harmony. GaussianNB is consistent in labeling positives despite lower accuracy (84.24%) and precision. SVM shows robustness with high accuracy (86.41%) and the best AUC (94.91%). AdaBoost, while precise, trades off with lower recall, impacting its F1 score (86.83%). GradientBoost's performance is solid, with slight room for improvement in recall and F1 score.

Table 2: **Model evaluation metrics**

Model	**Accuracy**	**AUC**	**Precision**	**Recall**	**F1 Score**
Logistic Regression (LR)	0.847826	0.900837	0.907216	0.822430	0.862745
Random Forest (RF)	0.885870	0.943848	0.924528	0.882883	0.903226
Gaussian Naïve Bayes (GaussianNB)	0.842391	0.908970	0.882353	0.841121	0.861244
Support Vector Machine (SVM)	0.864130	0.949144	0.894231	0.869159	0.881517
AdaBoost classifier	0.853261	0.914735	0.908163	0.831776	0.868293
GradientBoost Classifier	0.875000	0.936521	0.920000	0.859813	0.888889

Initially, the RF model achieved an accuracy of 88.59%, the best among the ML models implemented. In Table 2, the model's AUC of 0.9438 reflects its strong discriminative ability to differentiate between classes. Precision, at 92.45%, indicates a high rate of true positive predictions relative to all positive predictions made, while a recall of 88.29% shows that the model correctly identified a large proportion of actual positives. The F1 Score of 90.32% suggests a well-balanced model considering both precision and recall.

Table 3: **Comparison of Random Forest model performance with and without Bayesian Optimization**

Model	**Accuracy**	**AUC**	**Precision**	**Recall**	**F1 Score**
Random Forest	0.885870	0.943848	0.924528	0.882883	0.903226
Random Forest + Bayesian Optimization (BO)	0.902174	0.946131	0.926606	0.909910	0.918182

Table 3 compares the performance of the RF classifier before and after BO hyperparameter tuning. After applying BO for hyperparameter tuning, the model's performance improved across all metrics. Accuracy increased to 90.22%, showing that the model makes more correct predictions post-tuning. The AUC saw a slight improvement to 0.9461, which suggests a marginal increase in the model's ability to distinguish between classes. Notably, precision improved to 92.66%, and recall saw a significant rise to 90.99%, indicating that the model is now more effective at identifying true positives. The F1 Score increased to 91.82%, confirming that the tuned model provides a better balance between precision and recall.

Discussion & Evaluation

Improvements to Random Forest Model

Overall, standard RF outperforms the other models, and RF + BO outperforms all. The standard RF algorithm has demonstrated performance over other standard models in the comparative analysis for several reasons. Aside from the evaluation metrics, RF is more computationally efficient as it is an embedded method where feature selection is integrated into the classifier algorithm, where they rank importance based on metrics on Gini's Importance score [37]. In this study, we can attempt to understand how each feature contributes to the RF predictions to further improve the feature engineering/selection component of model implementation. As there is a built-in feature importance with sci-kit learn for RF, we used this to analyze the results of the feature importance as shown in Figure 7 below.

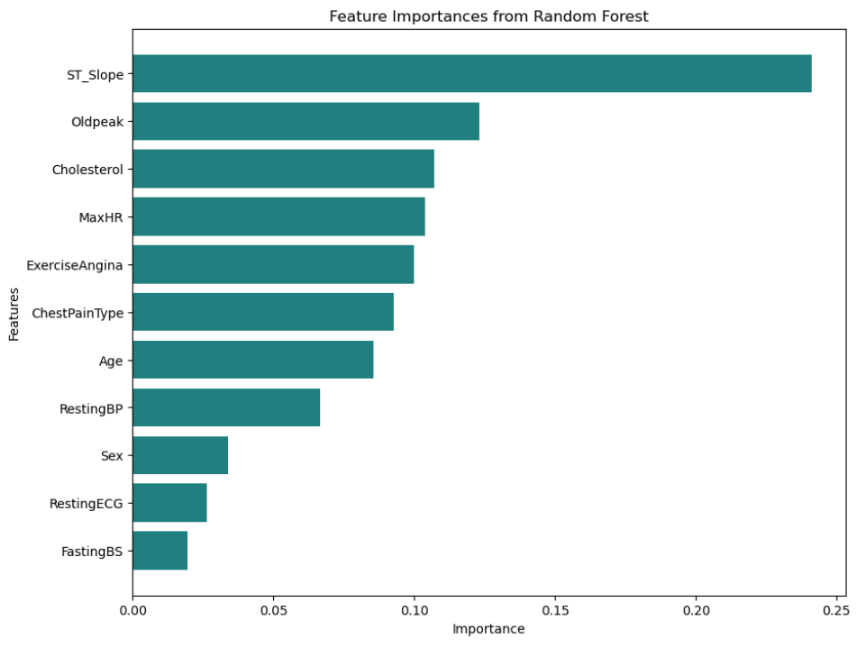


Figure 7: Ranking of 'heart_num_filtered.csv' features with Gini's Impurity Score, an embedded function in Random Forest

Consequently, the RF model can be improved by removing less important features to reduce model complexity (potentially Fasting Blood Sugar from Figure 7), which can help prevent overfitting. Additionally, we can investigate and integrate analogous datasets which contain more important features that are related to the heart failure such as systolic patient's systolic and diastolic blood pressure [38], and patient's history on smoking and alcohol intake. Another problem we can investigate into to attempt improving model's performance is data leakage. This problem introduces information from the test set or target variable into training process, artificially inflating model performance on training data but diminishing its effectiveness on the new data [39]. Such leakage can cause RF algorithm to overestimate the importance of features that correlate with the target, despite lack of true predictive power. This misleads confidence in both model's accuracy and significance of these features [40]. From Figure 7 above, we can attempt to remove most important features to validate whether they are important or it is a data leakage phenomenon.

From Table 3, RF with Bayesian Optimization (RF+BO) outperforms the standard RF model, based on the evaluation metrics. With BO, this takes into account previous iterations results when choosing the next hyperparameters which makes it relatively more suitable than random/ grid search [41]. However, it is worth exploring other alternative optimisation algorithms or feature selection methods such as recursive feature elimination (RFE) [29] as it is shown by Burcu et. Al (2018) [42] to improve the RF's performance when combined with RFE with a high dimensional biomedical dataset.

Improvements to Study

Additionally, for the improvement of experimental design, we could have explored different data preprocessing techniques such as other normalization techniques such as Variable Stability Scaling [24], which might lead to different insights, potentially better. More advanced ensemble techniques like XGBoost or LightBGM could have also been investigated in this study to accurately compare a model for this heart failure prediction task. Moreover, hyperparameter tuning of the 5 other ML classifiers should have also been evaluated instead of just the best standard model proposed in this study. Including other detailed evaluation metrics such as ROC curve plots would offer a more nuanced view of model performance.

Conclusion

Challenges of deploying Random Forest + Bayesian Optimisation model in clinical settings

In deploying RF+BO models in clinical settings, arise primarily from the varied and complex nature of clinical data. These range from converting unstructured health records into formats which ML algorithms can interpret, to ensuring that the models remain accurate as healthcare practices evolve and patient data changes. This latter point, known as data shift[43], can significantly degrade model performance if not properly managed, leading to unreliable predictions. Furthermore, constructing efficient data pipelines to handle the diversity and velocity of clinical data is critical, requiring sophisticated systems to collect, process, and feed data into ML models without significant delays. Addressing these challenges is essential for the successful integration of ML into clinical workflows, ensuring that predictions are timely, accurate, and free from bias, ultimately supporting better patient outcomes.

References

1. Savarese G, Becher PM, Lund LH, Seferovic P, Rosano GMC, and Coats AJS. Global burden of heart failure: a comprehensive and updated review of epidemiology. Cardiovascular Research 2022 Feb; 118. DOI: 10.1093/cvr/cvac013

2. Tripoliti EE, Papadopoulos TG, Karanasiou GS, Naka KK, and Fotiadis DI. Heart Failure: Diagnosis, Severity Estimation and Prediction of Adverse Events Through Machine Learning Techniques. Computational and Structural Biotechnology Journal 2017; 15:26–47. DOI: 10.1016/j.csbj.2016.11.001

3. Sciomer S, Moscucci F, Salvioni E, Marchese G, Bussotti M, Corrà U, and Piepoli MF. Role of gender, age and BMI in prognosis of heart failure. European Journal of Preventive Cardiology 2020 Dec; 27:46–51. DOI: 10.1177/2047487320961980. [Accessed on: 2021 Oct 1]

4. Son YJ and Lee HJ. Association between persistent smoking after a diagnosis of heart failure and adverse health outcomes: A systematic review and meta-analysis. Tobacco Induced Diseases 2020 Jan; 18. DOI: 10.18332/tid/116411

5. Cattadori G, Segurini C, Picozzi A, Padeletti L, and Anzà C. Exercise and heart failure: an update. ESC Heart Failure 2017 Dec; 5:222–32. DOI: 10.1002/ehf2.12225. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5880674/

6. Mahmud I, Kabir MM, Mridha MF, Alfarhood S, Safran M, and Che D. Cardiac Failure Forecasting Based on Clinical Data Using a Lightweight Machine Learning Metamodel. Diagnostics 2023 Jan; 13:2540. DOI: 10.3390/diagnostics13152540. Available from: https://www.mdpi.com/2075-4418/13/15/2540 [Accessed on: 2023 Oct 3]

7. Rajdhan A and Agarwal A. HEART DISEASE PREDICTION USING MACHINE LEARNING. International Research Journal of Modernization in Engineering Technology and Science 2023 Mar. DOI: 10.56726/irjmets34179. [Accessed on: 2023 Apr 17]

8. Jindal H, Agrawal S, Khera R, Jain R, and Nagrath P. [Confluence 2020 Front Matter]. 2020 Jan. DOI: 10.1109/confluence47617.2020.9058231. [Accessed on: 2023 Sep 3]

9. Sahoo PK and Jeripothula P. Heart Failure Prediction Using Machine Learning Techniques. SSRN Electronic Journal 2020. DOI: 10.2139/ssrn.3759562. [Accessed on: 2021 Mar 21]

10. Uyar K and İlhan A. Diagnosis of heart disease using genetic algorithm based trained recurrent fuzzy neural networks. Procedia Computer Science 2017; 120:588–93. DOI: 10.1016/j.procs.2017.11.283. [Accessed on: 2020 Mar 12]

11. Awan SE, Sohel F, Sanfilippo FM, Bennamoun M, and Dwivedi G. Machine learning in heart failure. Current Opinion in Cardiology 2018 Mar; 33:190–5. DOI: 10.1097/hco.0000000000000491. [Accessed on: 2022 May 16]

12. . Heart Failure Prediction Dataset. www.kaggle.com, 2021. Available from: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

13. MADIAS JE. The Resting Electrocardiogram in the Management of Patients with Congestive Heart Failure: Established Applications and New Insights. Pacing and Clinical Electrophysiology 2007 Jan; 30. DOI: 10.1111/j.1540-8159.2007.00586.x. [Accessed on: 2022 Aug 9]

14. Al Hinai G, Jammoul S, Vajihi Z, and Afilalo J. Deep learning analysis of resting electrocardiograms for the detection of myocardial dysfunction, hypertrophy, and ischaemia: a systematic review. European Heart Journal - Digital Health 2021 Aug; 2:416–23. DOI: 10.1093/ehjdh/ztab048. [Accessed on: 2023 Jan 9]

15. Bauer DC. Association of Major and Minor ECG Abnormalities With Coronary Heart Disease Events. JAMA 2012 Apr; 307:1497. DOI: 10.1001/jama.2012.434. [Accessed on: 2019 Apr 19]

16. Brownlee J. Ordinal and One-Hot Encodings for Categorical Data. Machine Learning Mastery, 2020 Jun. Available from: https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

17. Kim JH. Multicollinearity and misleading statistical results. Korean Journal of Anesthesiology 2019 Jul; 72:558–69. DOI: 10.4097/kja.19087

18. Rodgers JL, Jones J, Bolleddu SI, Vanthenapalli S, Rodgers LE, Shah K, Karia K, and Panguluri SK. Cardiovascular Risks Associated with Gender and Aging. Journal of Cardiovascular Development and Disease 2019 Apr; 6:19. DOI: 10.3390/jcdd6020019. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6616540/

19. KETEYIAN SJ, KITZMAN D, ZANNAD F, LANDZBERG J, ARNOLD JM, BRUBAKER P, BRAWNER CA, BENSIMHON D, HELLKAMP AS, and EWALD G. Predicting Maximal HR in Heart Failure Patients on -Blockade Therapy. Medicine Science in Sports Exercise 2012 Mar; 44:371–6. DOI: 10.1249/mss.0b013e318234316f. [Accessed on: 2023 Apr 10]

20. Zhao M, Chen Y, Wang M, Wang C, Yao S, Li Y, Zhang S, Yun C, Wu S, and Xue H. Relationship between resting heart rate and incident heart failure in patients with hypertension: The Kailuan Cohort Study in China. The Journal of Clinical Hypertension 2020 Oct; 22:2325–31. DOI: 10.1111/jch.14062. [Accessed on: 2021 Mar 21]

21. Sumwiza K, Twizere C, Rushingabigwi G, Bakunzibake P, and Bamurigire P. Enhanced cardiovascular disease prediction model using random forest algorithm. Informatics in Medicine Unlocked 2023 Jan; 41:101316. DOI: 10.1016/j.imu.2023.101316. Available from: https://www.sciencedirect.com/science/article/pii/S2352914823001624 [Accessed on: 2023 Aug 21]

22. Lutimath NM, Sharma N, and Byregowda BK. Prediction of Heart Disease using Random Forest. 2021 Emerging Trends in Industry 4.0 (ETI 4.0) 2021 May. DOI: 10.1109/eti4.051663.2021.9619208

23. Hossain MI, Maruf MH, Rahman A, Prity FS, Fatema S, Ejaz MS, and Sad A. Heart disease prediction using distinct artificial intelligence techniques: performance analysis and comparison. 2023 Jun. DOI: 10.1007/s42044-023-00148-7. [Accessed on: 2023 Jun 19]

24. Singh D and Singh B. Investigating the impact of data normalization on classification performance. Applied Soft Computing 2019 May; 97:105524. DOI: 10.1016/j.asoc.2019.105524

25. Amorim LB de, Cavalcanti GD, and Cruz RM. The choice of scaling technique matters for classification performance. Applied Soft Computing 2023 Jan; 133:109924. DOI: 10.1016/j.asoc.2022.109924. Available from: https://arxiv.org/pdf/2212.12343

26. Zhang Y, Diao L, and Ma L. Logistic Regression Models in Predicting Heart Disease. Journal of Physics: Conference Series 2021 Jan; 1769:012024. DOI: 10.1088/1742-6596/1769/1/012024

27. Sai Krishna Reddy V, Meghana P, Subba Reddy NV, and Ashwath Rao B. Prediction on Cardiovascular disease using Decision tree and Naïve Bayes classifiers. Journal of Physics: Conference Series 2022 Jan; 2161:012015. DOI: 10.1088/1742-6596/2161/1/012015

28. Mahesh TR, Dhilip Kumar V, Vinoth Kumar V, Asghar J, Geman O, Arulkumaran G, and Arun N. AdaBoost Ensemble Methods Using K-Fold Cross Validation for Survivability with the Early Detection of Heart Disease. Computational Intelligence and Neuroscience 2022 Apr; 2022. Ed. by Ahmad M:1–11. DOI: 10.1155/2022/9005278. [Accessed on: 2022 Jul 22]

29. Theerthagiri P. Predictive Analysis of Cardiovascular Disease using Gradient Boosting based Learning and Recursive Feature Elimination Technique. Intelligent Systems with Applications 2022 Sep :200121. DOI: 10.1016/j.iswa.2022.200121. [Accessed on: 2022 Sep 15]

30. Dharmendra D and Saravanan S. Prediction of Heart Failure using Support Vector Machine compared with Decision Tree Algorithm for better Accuracy. 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS) 2022 Apr. DOI: 10.1109/icscds53736.2022.9760989. [Accessed on: 2023 Nov 26]

31. Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters 2006 Jun; 27:861–74. DOI: 10.1016/j.patrec.2005.10.010. [Accessed on: 2019 Sep 18]

32. Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 1997 Jul; 30:1145–59. DOI: 10.1016/s0031-3203(96)00142-2

33. Saito T and Rehmsmeier M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLOS ONE 2015 Mar; 10. Ed. by Brock G:e0118432. DOI: 10.1371/journal.pone.0118432. [Accessed on: 2019 Nov 20]

34. Nygren R and Petkov A. Evaluation of hyperparameter optimization methods for Random Forest classifiers. DEGREE PROJECT COMPUTER ENGINEERING. Available from: https://www.diva-portal.org/smash/get/diva2:1593145/FULLTEXT01.pdf [Accessed on: 2023 Nov 26]

35. Bergstra J, Bardenet R, Bengio Y, and Kégl B. Algorithms for Hyper-Parameter Optimization. 2011. Available from: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7a%20b32cfd12577bc2619bc635690-Paper.pdf

36. Dewancker I, Mccourt M, and Clark S. Bayesian Optimization Primer. Available from: https://static.sigopt.com/b/20a144d208ef255d3b981ce419667ec25d8412e2/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf

37. Menze BH, Kelm BM, Masuch R, Himmelreich U, Bachert P, Petrich W, and Hamprecht FA. A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC Bioinformatics 2009; 10:213. DOI: 10.1186/1471-2105-10-213. [Accessed on: 2020 Mar 7]

38. Gao L and Ding Y. Disease prediction via Bayesian hyperparameter optimization and ensemble learning. BMC Research Notes 2020 Apr; 13. DOI: 10.1186/s13104-020-05050-0. [Accessed on: 2020 May 18]

39. Kaufman S, Rosset S, Perlich C, and Stitelman O. Leakage in data mining. ACM Transactions on Knowledge Discovery from Data 2012 Dec; 6:1–21. DOI: 10.1145/2382577.2382579

40. Strobl C, Boulesteix AL, Kneib T, Augustin T, and Zeileis A. Conditional variable importance for random forests. BMC Bioinformatics 2008 Jul; 9. DOI: 10.1186/1471-2105-9-307

41. Turner R, Eriksson D, McCourt M, Kiili J, Laaksonen E, Xu Z, and Guyon I. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. arXiv.org, 2021 Aug. DOI: 10.48550/arXiv.2104.10201. Available from: https://arxiv.org/abs/2104.10201 [Accessed on: 2023 Jun 5]

42. Darst BF, Malecki KC, and Engelman CD. Using recursive feature elimination in random forest to account for correlated variables in high dimensional data. BMC Genetics 2018 Sep; 19. DOI: 10.1186/s12863-018-0633-8. [Accessed on: 2021 Jun 24]

43. Zhang A, Xing L, Zou J, and Wu JC. Shifting machine learning for healthcare from development to deployment and from models to data. Nature Biomedical Engineering 2022 Jul. DOI: 10.1038/s41551-022-00898-y

44. Saini A. Gradient Boosting Algorithm: A Complete Guide for Beginners. Analytics Vidhya, 2021 Sep. Available from: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Appendix

1.1 Code Availability

Raw files was extracted from Kaggle, such as linkedin here 'Heart.csv'. Cleaned CSV is desposited on Github. The data preprocessing, exploratory data analysis model development and hyperparameter tuning with Bayesian optimisation are available in my Github repository (AI4BH).

- EDA was partitioned into 2 parts as linked: 1) 'AI4BH_EDA_part1_real.ipynb' 2) AI4BH_EDA_part2_real.ipynb'
- Model Development had 6 different model evaluated with the respective model packages from sklearn as shown here, AI4BH_model_development.ipynb.
- Hyperparameter tuning of RF is as linked here: AI4BH_hyperparameter_tuning.ipynb

1.2 5 ML Classifiers for Binary Classification- how do they work?

Logistic Regression

The logistic regression model [26] is a commonly used statistical method to predict the probability of a binary outcome. In the context of the heart failure dataset, the logistic regression model can be represented as follows:

$$\text{prob}(Y = 1) = \frac{e^z}{1 + e^z} \tag{1}$$

where Y refers to the binary dependent variable representing the occurrence of heart failure (with $Y = 1$ if the event of heart failure happens, and $Y = 0$ otherwise). The term e stands for the base of natural logarithms, and z is the linear combination of predictors given by

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p \tag{2}$$

Here, β_0 is the constant term, β_j are the coefficients that represent the contribution of each predictor X_j towards the outcome, for j predictors ($j = 1, 2, \ldots, p$). Each predictor X_j corresponds to a different feature in the heart failure dataset, such as age, blood pressure, cholesterol levels, and other clinical measurements that could influence the risk of heart failure.

Gaussian Naïve Bayes

Gaussian Naïve Bayes [27] is a probabilistic classifier that extends the Naïve Bayes algorithm to continuous data; it is particularly useful when the features of a dataset are not binary. This approach assumes that the continuous values associated with each feature are distributed according to a Gaussian distribution. In the context of heart failure prediction, Gaussian Naïve Bayes can classify patient outcomes based on continuous health indicators such as blood pressure, cholesterol levels, and heart rate.

The classification process involves the following steps:

1. Calculate the mean and standard deviation for each feature within the dataset.

2. Apply the Gaussian probability density function to each feature value for a given class, which is defined as:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{3}$$

   where x_i is a feature value, μ_y is the mean of the feature for class y, and σ_y^2 is the variance of the feature for class y.

3. Multiply the probabilities of each feature value for a given class, assuming feature independence, and then use the class's prior probability to find the class probability.

4. Predict the class with the highest probability as the outcome for heart failure.

By applying these steps, Gaussian Naïve Bayes can effectively classify instances in the heart failure dataset and predict whether a new patient is likely to experience heart failure based on their health indicators.

Adaboost Classifer

AdaBoost, or Adaptive Boosting [28], is a powerful ensemble technique used for classification tasks, including heart failure prediction. It iteratively trains weak classifiers, typically decision stumps, with each subsequent classifier focusing more on the training instances that previous classifiers misclassified. The final model aggregates the weak classifiers into a weighted sum that represents the final output of the strong classifier. This process is mathematically captured by the following equation for the classifier weight:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{P_{+1}}{P_{-1}} \right), \tag{4}$$

where α_t is the weight for the classifier at iteration t, and P_{+1} and P_{-1} are the probabilities of the correctly classified instances and misclassified instances, respectively. The output of AdaBoost is a combination of the weak classifiers' decisions, each weighted by its corresponding α_t, which together form a strong classifier that is used to predict heart failure events.

GradientBoost Classifier

The Gradient Boosting Classifier [44] is an advanced machine learning technique that is particularly effective for predicting outcomes like heart failure. It uses the concept of boosting weak learners, typically decision trees, to create a strong predictive model. The classifier optimizes a log-likelihood loss function, which for binary classification is given by:

$$L = -\sum_{i=1}^{n} [y_i \log(p) + (1 - y_i) \log(1 - p)] \tag{5}$$

Here, y_i represents the true outcome, and p is the predicted probability. The loss function is minimized using the gradient descent method, where residuals are computed as:

$$r = -\frac{\partial L}{\partial \log(\text{odds})} \tag{6}$$

These residuals are then used to fit new learners to predict the log(odds) of heart failure, which are then scaled by a factor α_t and added to update the model:

$$\alpha_t = \frac{1}{2} \log \left(\frac{P_{+1}}{P_{-1}} \right) \tag{7}$$

The final prediction is made by summing the contributions of all the trees, with each tree's output being a function of the residuals from the previous tree, adjusted by the learning rate. This additive model allows the classifier to focus on the most challenging cases in the heart failure dataset, thus improving its predictive performance with each iteration.

Support Vector Machine

SVM [30] has shown effectiveness in distinguishing patients with and without heart failure by finding the optimal hyperplane that maximizes the margin between classes, offering a high level of accuracy in heart disease.

1.3 Details of Evaluation Metrics

These are the descriptions of the metrics, obtained from [21] used in our study.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Accuracy indicates the number of samples that were accurately corrected for a given total number of samples.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Precision indicates the number of samples that were accurately corrected for a given total number of samples.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

Recall indicates the prediction of positive values to the actual positive value.

$$\text{F1 score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{11}$$

F1 score is the harmonic mean of precision and recall.

Where:

- TP (True Positive) is the number of correct predictions that an instance is positive,

- TN (True Negative) is the number of correct predictions that an instance is negative,
- FP (False Positive) is the number of incorrect predictions that an instance is positive,
- FN (False Negative) is the number of incorrect predictions that an instance is negative.

AUC, which stands for Area Under the Receiver Operating Characteristic (ROC) Curve, is a performance measurement for classification problems at various threshold settings. The ROC is a probability curve that plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at different classification thresholds. The AUC represents the degree to which the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. An AUC of 0.5 suggests no discrimination (no better than random guessing), while an AUC of 1.0 denotes perfect prediction.

1.4 Hyperparameter Optimization of Random Forest Model

In [7]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from bayes_opt import BayesianOptimization
import pandas as pd
df= pd.read_csv('heart_num_filtered.csv')
X = df.iloc[:, :-1]  # all columns except the last one
y = df.iloc[:, -1]   # the last column

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [8]:
```
def objective(n_estimators, max_depth, min_samples_split, max_features):
    model = RandomForestClassifier(n_estimators=int(n_estimators),
                                  max_depth=int(max_depth),
                                  min_samples_split=int(min_samples_split),
                                  max_features=min(max_features, 0.999),  # Fraction, must be <= 1.0
                                  random_state=42)

    return -1.0 * cross_val_score(model, X_train, y_train, cv=3, scoring="neg_mean_squared_error").mean()
```

In [18]:
```
optimizer = BayesianOptimization(f=objective, pbounds=param_bounds, random_state=42)
optimizer.maximize(init_points=5, n_iter=15)
```

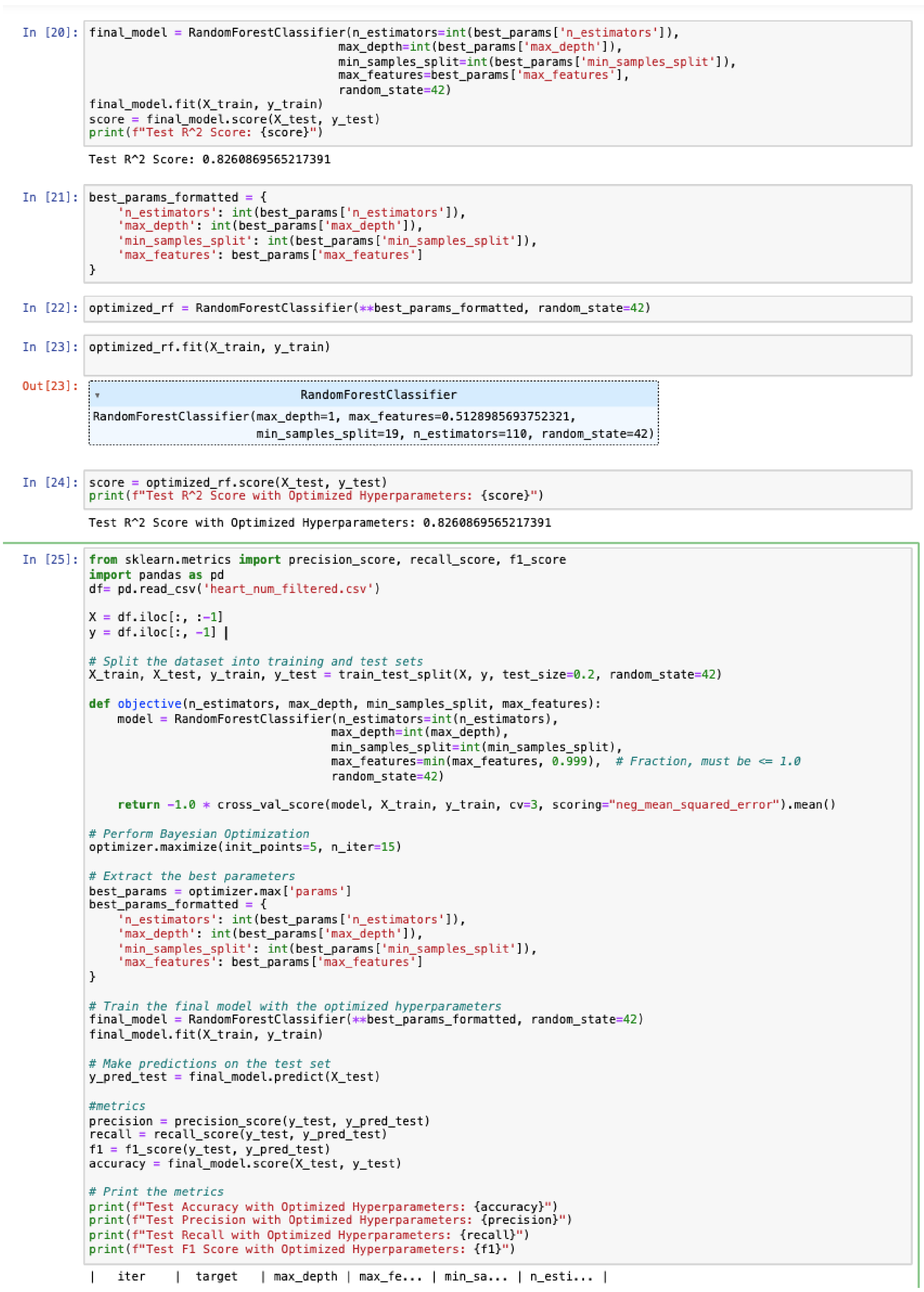
```
|   iter    |  target   | max_depth | max_fe... | min_sa... | n_esti... |
-------------------------------------------------------------------------
| 1         | 0.1485    | 19.35     | 0.9547    | 15.18     | 123.7     |
| 2         | 0.1336    | 8.645     | 0.2402    | 3.046     | 174.6     |
| 3         | 0.1485    | 30.45     | 0.7366    | 2.371     | 194.3     |
| 4         | 0.1376    | 41.79     | 0.2909    | 5.273     | 44.85     |
| 5         | 0.1444    | 15.91     | 0.5718    | 9.775     | 65.33     |
| 6         | 0.1363    | 20.09     | 0.4191    | 15.12     | 123.6     |
| 7         | 0.1471    | 3.412     | 0.1166    | 12.81     | 147.5     |
| 8         | 0.1662    | 1.219     | 0.1363    | 10.4      | 76.9      |
| 9         | 0.1444    | 48.57     | 0.7379    | 19.89     | 81.05     |
| 10        | 0.1267    | 24.1      | 0.2443    | 16.44     | 28.34     |
| 11        | 0.1458    | 46.98     | 0.735     | 10.1      | 199.7     |
| 12        | 0.1499    | 38.98     | 0.893     | 6.089     | 80.76     |
| 13        | 0.139     | 10.48     | 0.6585    | 6.468     | 155.9     |
| 14        | 0.1785    | 1.309     | 0.5129    | 19.58     | 110.1     |
| 15        | 0.1485    | 37.39     | 0.5721    | 7.406     | 13.74     |
| 16        | 0.1444    | 44.46     | 0.5774    | 6.624     | 110.9     |
| 17        | 0.1567    | 11.1      | 0.972     | 2.258     | 23.2      |
| 18        | 0.1295    | 22.97     | 0.179     | 2.931     | 169.0     |
| 19        | 0.1417    | 5.07      | 0.8178    | 18.6      | 194.6     |
| 20        | 0.139     | 18.86     | 0.3578    | 14.46     | 136.8     |
=========================================================================
```

In [17]:
```
# Bounds for hyperparameters
param_bounds = {
    'n_estimators': (10, 200),  # 0 is not a valid value, and 200 is typically sufficient
    'max_depth': (1, 50),       # Depths greater than 50 are rarely needed and can cause overfitting
    'min_samples_split': (2, 20),  # Negative values are not valid, and values too high can lead to underfitting
    'max_features': (0.1, 0.999),  # This range is usually fine
}
```

In [19]:
```
best_params = optimizer.max['params']
best_params
```

Out[19]:
```
{'max_depth': 1.3088786382010564,
 'max_features': 0.5128985693752321,
 'min_samples_split': 19.579876635267876,
 'n_estimators': 110.08221214378727}
```

Figure 8: Hyperparamter tuning with Bayesian Optimization on Random Forest Classifier part 1



```
In [20]: final_model = RandomForestClassifier(n_estimators=int(best_params['n_estimators']),
                                              max_depth=int(best_params['max_depth']),
                                              min_samples_split=int(best_params['min_samples_split']),
                                              max_features=best_params['max_features'],
                                              random_state=42)
         final_model.fit(X_train, y_train)
         score = final_model.score(X_test, y_test)
         print(f"Test R^2 Score: {score}")

         Test R^2 Score: 0.8260869565217391

In [21]: best_params_formatted = {
             'n_estimators': int(best_params['n_estimators']),
             'max_depth': int(best_params['max_depth']),
             'min_samples_split': int(best_params['min_samples_split']),
             'max_features': best_params['max_features']
         }

In [22]: optimized_rf = RandomForestClassifier(**best_params_formatted, random_state=42)

In [23]: optimized_rf.fit(X_train, y_train)

Out[23]:                          RandomForestClassifier
         RandomForestClassifier(max_depth=1, max_features=0.5128985693752321,
                                min_samples_split=19, n_estimators=110, random_state=42)

In [24]: score = optimized_rf.score(X_test, y_test)
         print(f"Test R^2 Score with Optimized Hyperparameters: {score}")

         Test R^2 Score with Optimized Hyperparameters: 0.8260869565217391

In [25]: from sklearn.metrics import precision_score, recall_score, f1_score
         import pandas as pd
         df= pd.read_csv('heart_num_filtered.csv')

         X = df.iloc[:, :-1]
         y = df.iloc[:, -1] |

         # Split the dataset into training and test sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         def objective(n_estimators, max_depth, min_samples_split, max_features):
             model = RandomForestClassifier(n_estimators=int(n_estimators),
                                            max_depth=int(max_depth),
                                            min_samples_split=int(min_samples_split),
                                            max_features=min(max_features, 0.999),  # Fraction, must be <= 1.0
                                            random_state=42)

             return -1.0 * cross_val_score(model, X_train, y_train, cv=3, scoring="neg_mean_squared_error").mean()

         # Perform Bayesian Optimization
         optimizer.maximize(init_points=5, n_iter=15)

         # Extract the best parameters
         best_params = optimizer.max['params']
         best_params_formatted = {
             'n_estimators': int(best_params['n_estimators']),
             'max_depth': int(best_params['max_depth']),
             'min_samples_split': int(best_params['min_samples_split']),
             'max_features': best_params['max_features']
         }

         # Train the final model with the optimized hyperparameters
         final_model = RandomForestClassifier(**best_params_formatted, random_state=42)
         final_model.fit(X_train, y_train)

         # Make predictions on the test set
         y_pred_test = final_model.predict(X_test)

         #metrics
         precision = precision_score(y_test, y_pred_test)
         recall = recall_score(y_test, y_pred_test)
         f1 = f1_score(y_test, y_pred_test)
         accuracy = final_model.score(X_test, y_test)

         # Print the metrics
         print(f"Test Accuracy with Optimized Hyperparameters: {accuracy}")
         print(f"Test Precision with Optimized Hyperparameters: {precision}")
         print(f"Test Recall with Optimized Hyperparameters: {recall}")
         print(f"Test F1 Score with Optimized Hyperparameters: {f1}")

         |   iter    |  target   | max_depth | max_fe... | min_sa... | n_esti... |
```

Figure 9: Hyperparamter tuning with Bayesian Optimization on Random Forest Classifier part 2

Mathematical Intuition behind Bayesian Optimisation

Bayesian Optimization (BO) for hyperparameter tuning in Random Forest models seeks to maximize an objective function, typically the negative mean squared error (MSE), to enhance model performance. The objective function f in BO, represented by a Gaussian Process (GP), is given by:

$$f(\mathbf{x}) = -\mathrm{MSE}(\mathbf{x}) \tag{12}$$

where $\mathbf{x}$ represents the hyperparameters. The GP provides a probabilistic estimate of f and is updated iteratively using observed data. The acquisition function, such as Expected Improvement (EI), is employed to select the next hyperparameters to evaluate, aiming to find the maximum of f:

$$\mathbf{x}_{\text{next}} = \arg\max EI(\mathbf{x}) \tag{13}$$

This process systematically improves the Random Forest model by focusing the search on areas of the hyperparameter space that are likely to yield a lower negative MSE.

1.5 Additional Figures

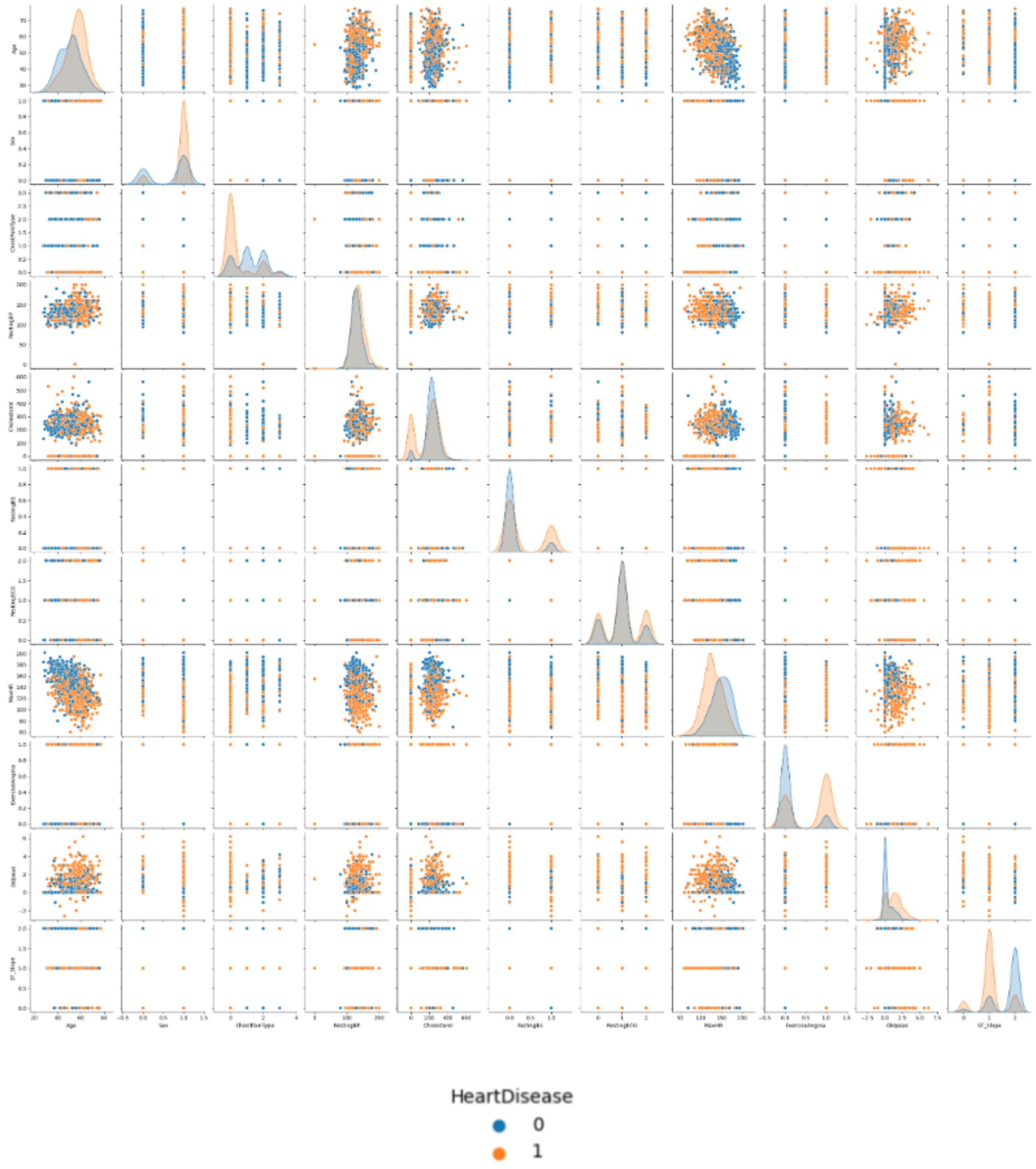


Figure 10: Relationship between each attribute against each other in heart.csv dataset

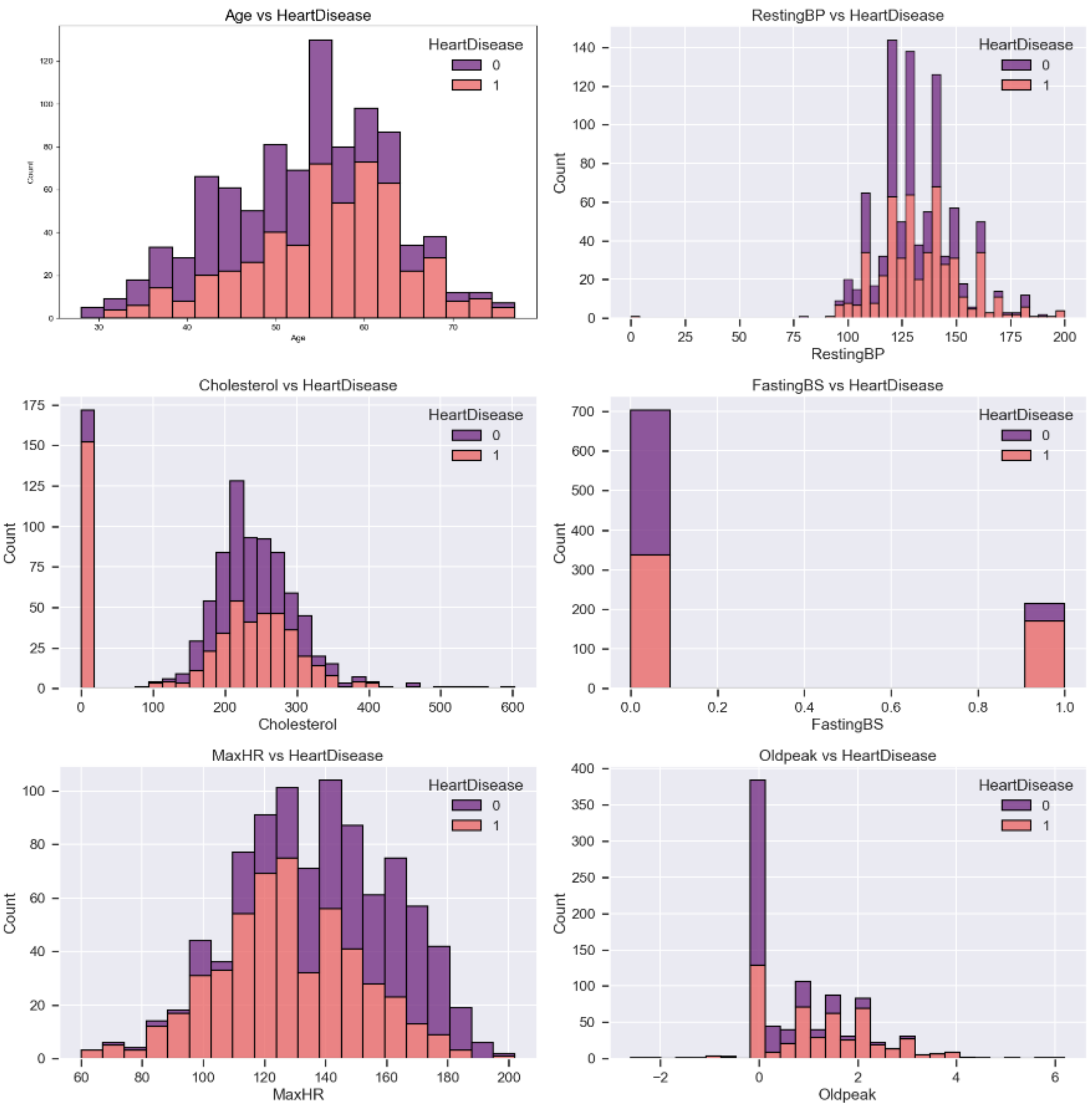


Figure 11: Distribution of numerical and continuous variables in heart.csv dataset against target variable, HeartDisease

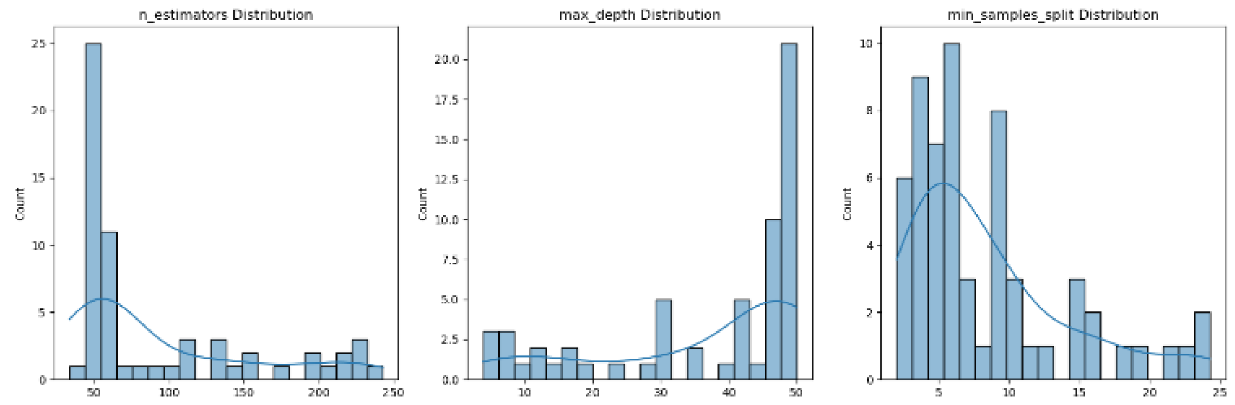


Figure 12: Distribution of best hyperparameters of Random Forest Classifier algorithm after 20 iterations

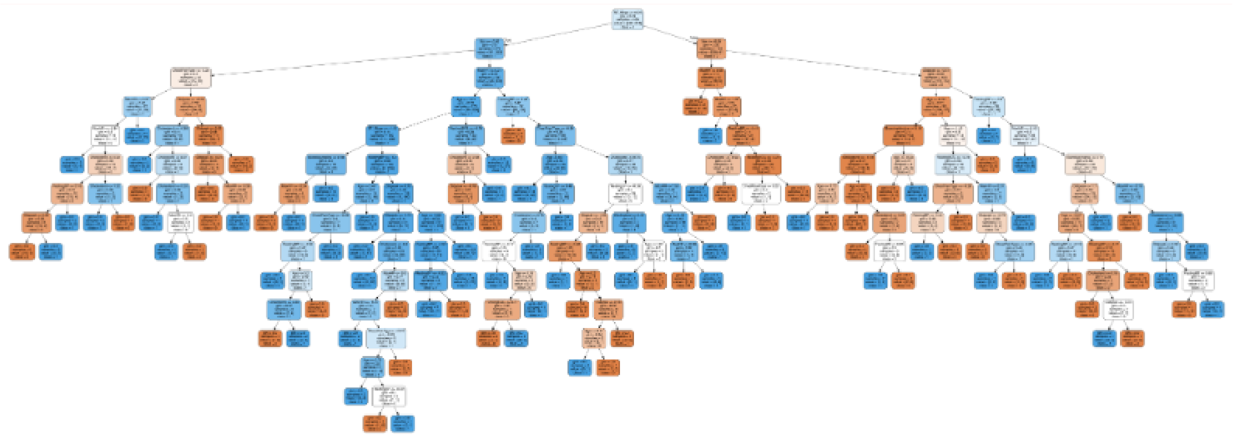


Figure 13: Visualisation of decision trees in random forest implemented on heart_num_filtered.csv