Computers in Biology and Medicine 192 (2025) 110308

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/compbiomed

A study on heart data analysis and prediction using advanced machine learning methods

Serbun Ufuk Değer

Kastamonu Vocational School, Kastamonu University, 37150, Kastamonu, Turkey

ARTICLE INFO

Keywords:
Heart disease prediction
Partial derivative
Machine learning
AutoML
ClassicML
Visualization

ABSTRACT

Cardiovascular diseases comprise a diverse array of disorders impacting the cardiac structure and vascular system and rank among the predominant factors contributing to mortality on a global scale. Every day, a significant number of individuals die from various heart-related issues. Therefore, early detection of heart diseases is of critical importance. Especially following these diagnoses, providing a more accurate diagnosis for individuals at high risk and subsequent extra treatments outline an essential roadmap for preventing heart attacks. This paper compares the performance of classic machine learning (ClassicML) and automated machine learning (AutoML) models across different variations that incorporate feature engineering and balancing techniques, thereby identifying which machine learning model is more successful in the development and implementation of patient-centered systems for the early prediction of cardiovascular diseases. The models used in this study utilize a combined dataset from Swiss, Hungarian, Cleveland, and Long Beach VA universities. This dataset consists of 14 main features, of which we used 12 features to analyze the individual experiencing a heart attack. The result of this classification problem is validated by accuracy. The accuracy result obtained is supported by the F1 score, precision, and recall results. The research encompasses nine traditional machine-learning algorithms as well as seven automated machine-learning algorithms. The findings of this investigation demonstrate that the performance of AutoML tools is not necessarily superior to traditional machine learning methodologies. Moreover, they highlight that effective feature extraction, when combined with an appropriate data balancing technique and a suitable machine learning model, can yield the best performance.

1. Introduction

Cardiovascular diseases are among the serious health issues that cause millions of people worldwide to lose their lives. These diseases include many diseases that directly affect the cardiovascular system and indirectly affect other systems. In addition, deaths due to these diseases have an essential place among the leading causes of death worldwide. Pathologies such as atherosclerosis may precipitate prevalent forms of cardiac disease, including coronary artery disease. Factors such as obesity, hypertension, high blood sugar, and excessive alcohol consumption are referred to as lifestyle-related factors. These factors are the leading risk factors for heart disease. Therefore, detecting heart disease in people with these risk factors is extremely important for their lives. Physical examination, family history, angiography, electrocardiography, sonography, and blood tests are among the various tests used to detect cardiovascular disorders.

Consequently, the pervasive implementation of big data analytics in the healthcare domain renders both ClassicML and AutoML methodologies increasingly pivotal in the diagnosis and treatment of cardiovascular disorders. Thanks to important features derived from clinical data, these tools can estimate the existence or absence of cardiovascular diseases more precisely.

AutoML tools are user-friendly tools that automate machine learning processes and do not require deep technical knowledge in this field. These tools automate data preparation, model selection, and model tuning, allowing users with less technical knowledge to use powerful machine learning models efficiently. The introduction of AutoML in the detection and treatment of cardiovascular diseases provides healthcare professionals with a proficient instrument that conserves both time and resources.

However, in the detection and prediction of complex medical conditions such as heart disease, it may not be enough to rely solely on AutoML tools. ClassicML methods also make an important contribution to this process. ClassicML allows for developing more controlled and customized machine learning solutions with steps such as specific algorithm selection, feature engineering, and model tuning. Therefore, it

E-mail address: sudeger@kastamonu.edu.tr.

https://doi.org/10.1016/j.compbiomed.2025.110308
Received 8 October 2024; Received in revised form 24 March 2025; Accepted 29 April 2025
Available online 12 May 2025
0010-4825/© 2025 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

is important to use both ClassicML and AutoML approaches together to achieve more successful results in the diagnosis and prediction of heart disease.

This paper aims to evaluate the performance of AutoML and classical machine learning (ClassicML) models in heart disease diagnosis in different scenarios where feature engineering and balancing techniques are applied. The application of this method can make a significant contribution to the treatment of heart disease by giving healthcare professionals more accurate and reliable diagnostic methods.

2. Related works

ClassicML and AutoML approaches have been significant research areas in the early diagnosis and treatment of cardiovascular diseases in the last ten years. These studies are critical for the early diagnosis of diseases. As a result, there are significant opportunities to prevent irreversible problems that occur in the advanced stages of diseases, and time can be used as a treatment tool with the proper treatment methods. Data related to cardiovascular diseases are called medical data, and many studies are conducted on this medical data using ClassicML methods.

Wettschereck and Dietterich made important steps towards improving classification problems using KNN algorithms and nested generalized examples [1]. Also, Wettschereck et al. surveyed a class of weight adjustment methods for lazy learning algorithms and presented a structure for identifying differences and experimentally comparing these methods [2]. In the late 1990s, the work of Liu and Motoda laid the foundations of machine learning. This study focuses on providing datasets and feature selection techniques and displays the applicability of automatic machine learning models in predicting cardiovascular diseases [3].

In the second half of the 2000s, further progress was made in this field, with Guyon et al. developing new approaches to feature extraction and classification problems using generalized discriminant analysis and making significant progress in this area [4]. Avendano-Valencia et al. compared different dimensionality reduction methods to apply principal component analysis (PCA) to time-frequency representations (TFRs). In their study, they found that methods that represent TFRs in matrix form better discriminate heart murmurs and that the proposed approach reduces variability in the results. [5]. Das et al. obtained an accuracy of 89.01 % with the classification model they developed for the effective diagnosis of heart diseases through neural network ensembles [6].

The early 2010s focused on prediction models for specific conditions and diseases. Srinivas et al. investigated whether self-reported incidences of cardiovascular disease are elevated in the Singareni coal mining regions of Andhra Pradesh, India, compared to other locales. They employed ClassicML techniques, including Decision Trees, Naive Bayes, and Neural Networks. They evaluated the prediction of heart disease based on 15 different factors such as body mass index (BMI), number of doctors, age, ethnicity, education, and income for conditions with symptoms of heart disease such as chest pain, stroke, and heart attack [7]. Parthiban and Srivatsa developed a predictive model that forecasts the likelihood of a diabetic individual developing heart disease, achieving an impressive accuracy rate of 94.6 % by utilizing variables such as gender, age, blood pressure, and blood glucose levels, employing Naive Bayes and support vector machines (SVM) [8].

By 2013, Santhanam and Ephzibah used the principal component analysis technique in their study on heart disease prediction. They used a feedforward neural network classifier and achieved an accuracy of 95.2 % [9]. Ratnasari et al. showed that thoracic X-ray features can be extracted using PCA-based feature selection for pulmonary tuberculosis classification aims. Mahalanobis distance classifier was used to evaluate the performance of the image classification process [10]. Melillo et al. aimed to improve an automated classifier for assessing risk in patients diagnosed with congestive heart failure (CHF). They refined classifiers utilizing classification and regression trees (CART) and integrated standard long-term heart rate variability (HRV) metrics with this classifier, thereby establishing a pivotal differentiation between low-risk and high-risk patients [11].

Approaching the conclusion of the 2010s, Kumar performed a comparative evaluation of the effectiveness of various machine learning algorithms in forecasting heart disease. Furthermore, the results were also represented in a visual format through the use of graphs [12]. Rajagopal and Ranganathan employed a probabilistic neural network (PNN) classifier for the classification of cardiac arrhythmias, subsequently evaluating this classification by juxtaposing the performance of five distinct linear and nonlinear dimensionality reduction techniques (PCA, fastICA, KPCA, hNLPCA, PPA) [13].

Also, during this period, the integration of AutoML and its potential in predicting cardiovascular diseases was further investigated.

Singh et al. developed a new method for decomposing HRV signals into subspaces. Through the application of this methodology, they scrutinized the signals through the application of the multiscale wavelet packet (MSWP) transform in conjunction with entropy features that were derived from the decomposed heart rate variability (HRV) signals. With this method, CAD patients can be identified using heart rate variability (HRV) signals [14].

By 2020, G'arate-Escamila and colleagues developed a machine model for heart disease prediction using Cleveland-Hungary (CH) datasets. In their research, they implemented a chi-square and principal component analysis (CHI-PCA) in conjunction with random forests (RF), thereby attaining an exceptional accuracy rate of 99.4 % [15]. In the same year, Ghffar et al. demonstrated how semi-supervised AutoML methods can be used effectively in their study of patients undergoing transcatheter aortic valve implantation [16].

In 2021–2024, critical studies have been conducted on ClassicML and AutoML for heart attack prediction or early detection of heart disease. Some of them are as follows;

Bharti et al. achieved 94.2 % accuracy with the DL model in their work on the Prediction of Heart Disease Using Machine Combination Learning and Deep Learning [17].

Afanasieva et al. compared many AutoML models and obtained an accuracy value of 94.2 % as the best value. Mandal and Pradhan achieved an accuracy of 95 % with decision trees (DT) in their method for early detection of heart disease [18].

Paladino et al. stated that AutoML tools give better results than ClassicML tools in detecting cardiovascular diseases. They have obtained 84.78 % accuracy with AutoGluon. They also stated that due to the ease of use of AutoML tools, these tools should be implemented on web platforms for early detection [19].

Barfungpa et al. used hybrid deep, dense networks and developed an intelligent heart disease prediction system. They achieved a 99.53 % accuracy rate in their prediction using this system [20].

Küçükmanisa and Kilimci obtained an accuracy of 90.1 % with artificial neural networks in their study on heart disease prediction [21].

3. Material and method

3.1. Dataset

3.1.1. Description of the dataset

Many datasets can be used as sources for heart disease or heart attack prediction. However, for academic studies, resources such as IEEE, Kaggle (datasets), University of California, Irvine (UCI) [22] are generally used. Some datasets can be accessed for free, while others require you to apply or pay a fee. The Cleveland heart disease datasets [23] and Statlog (Heart) datasets [24] are among the few open-access heart disease datasets available from UCI.

Many studies on heart disease prediction, such as Marimuthu et al. [25], Pol et al. [26], Valarmathi et al. [27], Padmanabhan et al. [28], Hazra et al. [29] and Khan et al. [30] have also used the Cleveland heart disease dataset for training and testing the models. Some researchers

using both Cleveland and Statlog datasets at the same time are El Bialy et al. [31], Sarra et al. [24], and Ahmed [32], respectively.

The Cleveland, Hungarian, and Statlog datasets were chosen for this study because they have been reviewed by other data scientists, allow for comparison of findings, and, most importantly, are easily accessible. Paladino et al. [19] have confirmed in their study that Cleveland includes Statlog, thus indicating that using Statlog as a dataset is redundant. After removing this dataset, the remaining Cleveland and Hungarian datasets, respectively, have 303 and 294 observations. Adding the UCI, Switzerland, and Long Beach, CA datasets brings our total observations to 918. Table 1 shows the 12 main features common to all datasets we also use.

3.2. Data pre-processing

In machine learning, data preprocessing is a crucial step to achieve successful results. The ability of algorithms used in machine learning to work accurately and with high performance depends entirely on it. Typically, three primary issues (incomplete data within the dataset, class imbalance in classification tasks, and class noise) impede the effective functioning of machine learning algorithms [33].

Missing data can occur for various reasons. For example, some specific questions (sexuality, substance use, etc.) asked to the participants during data collection may not be answered or may be answered incorrectly. On the contrary, participants may not have knowledge about the questions asked.

Especially in sectors such as health, the number and variety of data is high. In such cases, data may be entered incorrectly, patients' treatment may be terminated, patients may die, or they may not want to volunteer for any study. Therefore, it is inevitable that the data obtained will be incomplete. For this reason, various methodologies have been created to resolve the missing data problem. In general, some basic statistical methods (mean imputation, mode imputation, etc.) are used to complete the missing data, and the missing data are filled in a meaningful way, or as a different solution, the missing data are deleted from the data set without disturbing the integrity of the meaning of the data set.

The dataset employed in our investigation exhibits no deficiencies in data completeness. The Min-Max normalization technique was implemented to standardize the numerical attributes, wherein a linear transformation was executed on the dataset, thereby constraining the numerical values to the interval [0,1]. This methodology enhanced the comparability of the data by harmonizing values with disparate magnitudes within the dataset. Encoding techniques were utilized to transform categorical variables, thereby converting these variables into a numerical format conducive to the modeling process.

Fig. 1 summarizes the general workflow of this study. A total of six distinct data balancing techniques were employed to rectify class imbalances inherent in the dataset. Among these methodologies, SMOTEENN, the top performer, addresses the imbalance by generating synthetic instances to bolster minority classes, concurrently mitigating the noise within the dataset, which culminates in a more refined and equilibrated data structure. Two separate feature extraction techniques were implemented to augment the representativeness of the dataset and facilitate the model's acquisition of more descriptive features. The first technique is the Average Derivative (AD) method, which aims to discern the trends of the variables and generate features imbued with greater informational depth. In addition, the Polynomial Features method was employed to encapsulate the interactions among existing features more effectively. These feature extraction endeavors were designed to enhance the generalization capability of the model by introducing additional dimensions to the dataset.

Table 1
Descriptions of features within datasets.

Features		Description	Dtype
Age	age	Indicates the age of the patient (There are 50 different data)	int64
Sex	sex	Sex of the patient, where the male is represented as 1 and the female as 0	object
ChestPainType	cp	The types of chest pain experienced by the patients are ASY (Asymptomatic) = 3, NAP (Non-Anginal Pain) = 2, ATA (Atypical Angina) = 1, and TA (Typical Angina) = 0, respectively.	object
RestingBP	trtbps	Shows resting blood pressure in mm Hg measured at hospital admission. (There are 67 different data)	int64
Cholesterol	chol	Serum cholesterol, a measurement often reported in milligrams per deciliter (mg/dL), is a blood test result used to assess the cholesterol levels in the body. (There are 222 different data)	int64
Fasting Blood Sugar	fbs	FBS is a fasting blood sugar measurement. It is usually expressed in mg/dL (milligram per deciliter). The assumption will be accepted as true = 1 when fbs>120 and false = 0 when fbs<120.	int64
RestingECG	restecg	It refers to the results of an ECG test performed at rest. N (Normal) = 0, ST (ST-T wave abnormality) = 1, LVH (Left Ventricular Hypertrophy) = 2	object
MaxHR	thalach	It refers to the highest heart rate a person can achieve during exercise. (There are 119 different data)	int64
ExerciseAngina	exang	It refers to exercise-induced angina, which is chest pain or discomfort that occurs due to physical exertion. (yes = 1, no = 0)	object
Oldpeak	oldpeak	It refers to the degree of ST-segment depression on an electrocardiogram (ECG) induced by exercise relative to the baseline or resting state. (There are 53 different data)	float64
ST_Slope	slope	It refers to the slope of the ST segment on an ECG during peak exercise. ST segment categorized as Up = 0, Flat = 1, Down = 2	object
HeartDisease	output	Indicates the likelihood of heart attack, with less chance = 0 and more chance = 1	int64

3.2.1. Synthetic Minority Over-Sampling Technique and Edited Nearest Neighbors (SMOTEENN)

SMOTEENN represents one of the most efficacious sampling methodologies to mitigate class imbalance within datasets. This methodology encompasses a sampling procedure that entails the creation of synthetic data. SMOTEENN, a combination of SMOTE and ENN methods, aims to achieve better performance by combining the advantages of these two techniques.

SMOTE, rather than merely duplicating instances from the minority class, produces synthetic instances by utilizing the k nearest neighbors of the observed instances [34].

For every instance within the minority class, the k nearest neighbors are discerned, and the disparities among these *k* nearest neighbors are computed. These disparities are amplified by a stochastic coefficient (a scalar value ranging from 0 to 1). The resultant value derived from this multiplication is incorporated into the initial minority class sample; thus, the production of a new synthetic sample is ensured. This process continues until the desired number of synthetic samples is generated.

After applying SMOTE, the generated dataset is cleaned with the ENN method [35]. For each instance, k nearest neighbors are examined. If an instance is misclassified by its neighbors (i.e., its class label differs from the majority class label of its neighbors), it is extracted from the dataset. This step helps to improve the performance of the classifiers by removing noise and borderline examples.

3.2.2. Synthetic Minority Over-Sampling Technique (SMOTE)

SMOTE denotes a sophisticated oversampling methodology that

generates innovative synthetic instances of the minority class to address the class imbalance challenge. Instead of merely replicating data points randomly, this technique significantly bolsters the model's capacity for learning by producing new instances derived from the pre-existing ones within the minority class. The SMOTE algorithm synthesizes these instances through a procedure of random interpolation between data points identified by the k-nearest neighbor algorithm. This approach enhances the capacity for generalization pertaining to the minority class and diminishes the probability of overfitting [34].

3.2.3. Adaptive Synthetic Sampling (ADASYN)

ADASYN takes into account the data distribution when sampling for the minority class, allowing hard-to-learn examples to be supplemented with more synthetic data. It produces synthetic instances akin to the SMOTE methodology yet adaptively conducts sampling. This technique enhances the overall efficacy of the model by generating supplementary data surrounding underrepresented instances and those situated at the interface between classes. Such an approach facilitates improved learning of decision boundaries by the model, particularly in datasets characterized by imbalance [36].

3.2.4. Borderline-SMOTE

Borderline-SMOTE represents an oversampling technique that endeavors to create more significant synthetic samples by concentrating on the border regions of the minority class. Unlike the traditional SMOTE technique, it exclusively produces new instances derived from minority class samples that are in proximity to the decision boundary. This strategic approach facilitates the model's ability to differentiate between classes more effectively and contributes to a more precise articulation of decision boundaries. This framework possesses the capacity to improve model efficacy, particularly in scenarios characterized by class overlap or densely populated data distributions [37].

3.2.5. Random Over-Sampling (ROS)

It is a fundamental oversampling technique that seeks to mitigate class imbalance by randomly duplicating existing samples within a minority class. Rather than generating novel synthetic instances, it amplifies the prevalence of the minority class within the dataset by replicating current instances. Although this method is computationally less demanding than SMOTE and its derivatives, it does not augment the informational richness of the dataset, potentially heightening the likelihood of overfitting. Random Oversampling (ROS) provides a swift remedy for smaller datasets [38].

3.2.6. SVM-SMOTE

SVM-SMOTE is a data balancing technique that fabricates synthetic instances in the vicinity of support vectors associated with the minority class utilizing support vector machines (SVM). In contrast to the conventional SMOTE approach, it distinctly identifies the support vectors residing within the minority class and constructs novel instances surrounding these critical reference points. This process enhances the model's capacity to accurately delineate the decision boundary, culminating in enhanced performance, particularly in the context of high-dimensional datasets. The intrinsic consideration of decision surfaces inherent to support vector machines significantly bolsters the model's potential to produce more stable and generalizable results [39].

3.3. Feature extraction

In this paper, Average Derivative (AD) and polynomial feature extraction methods are used to enrich the features in the dataset. Using the 5 numerical features given in Figs. 2 and 10 new features were obtained with the Average Derivative method. In addition, 55 additional features were obtained by polynomial feature extraction. Thanks to these two methods, a total of 65 new features were added to the existing dataset and used to improve the performance of the model.

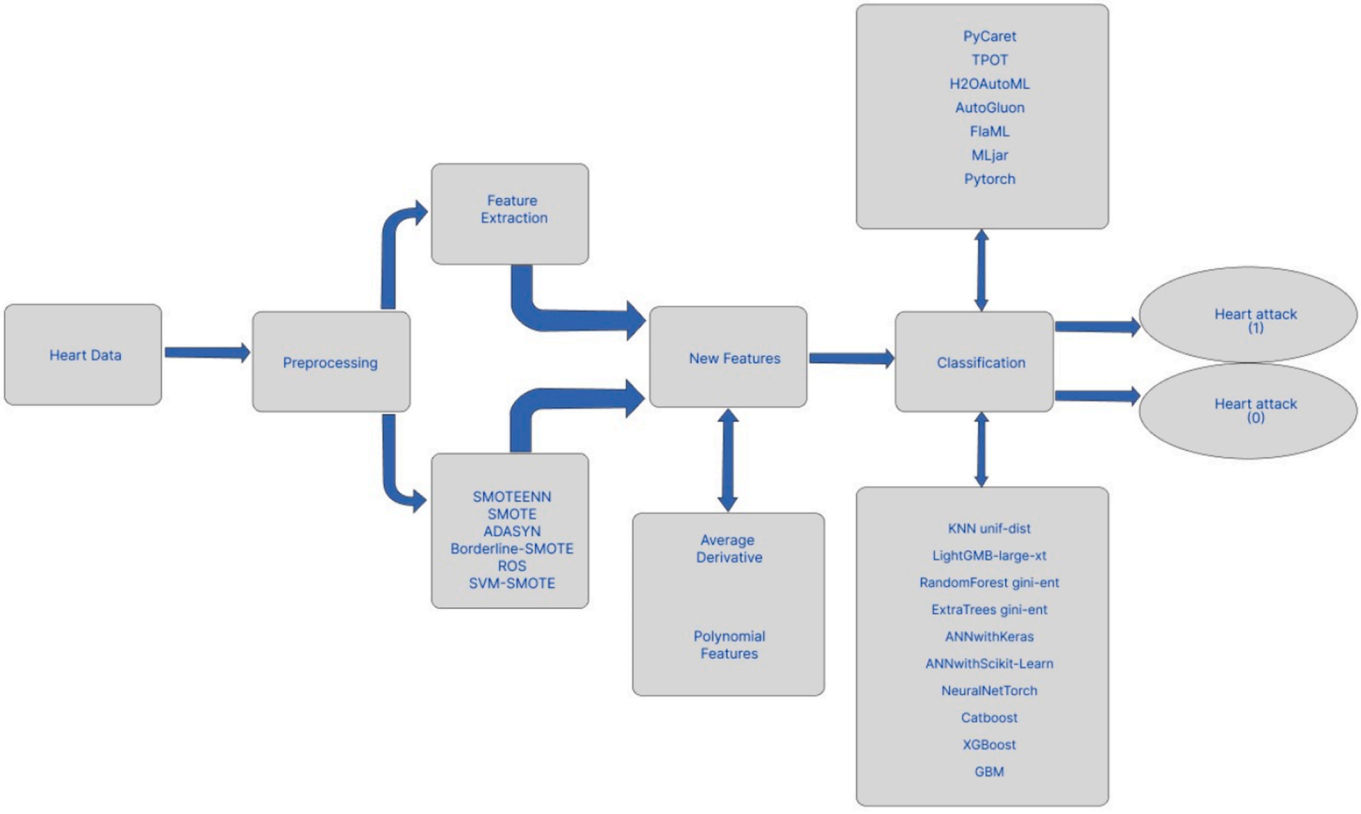

Fig. 1. General workflow.

	count	mean	std	min	25%	50%	75%	max
age	918.0	53.510893	9.432617	28.0	47.00	54.0	60.0	77.0
trtbps	918.0	132.396514	18.514154	0.0	120.00	130.0	140.0	200.0
chol	918.0	198.799564	109.384145	0.0	173.25	223.0	267.0	603.0
fbs	918.0	0.233115	0.423046	0.0	0.00	0.0	0.0	1.0
thalach	918.0	136.809368	25.460334	60.0	120.00	138.0	156.0	202.0
oldpeak	918.0	0.887364	1.066570	-2.6	0.00	0.6	1.5	6.2
output	918.0	0.553377	0.497414	0.0	0.00	1.0	1.0	1.0

Fig. 2. Statistical information.

3.3.1. Average Derivative (AD)

The Average Derivative (AD) is widely used in both analog and digital signal processing fields. This method is obtained by calculating the slope of the tangent line drawn at a specific point on the graph of a function. Additionally, the average of a data series is ascertained by dividing the aggregate of all components within the series by the total number of components in that series. We use the Average Derivative to analyze the trends of $f(x)$ values from $x = 1$ to $m - 1$, and this calculation is represented by Equation (3.1).

$$AD = \frac{1}{m-1}\sum_{x=1}^{m-1}(f(x) - f(x-1)), \tag{3.1}$$

where AD represents the Average Derivative of the data, and m refers to the length of the $f(x)$ function. The Average Derivative (AD) is a method used in machine learning to generate new features. This approach captures the trends and variations in the data to create additional features. These newly generated features, when combined with existing ones, contribute to improving the predictive performance and accuracy of machine learning models [40].

3.3.2. Polynomial Features (PF)

Polynomial features are created by raising existing features in a dataset to higher powers and taking their cross-products. This technique is used to model nonlinear relationships in the data, enhancing the model's ability to capture complex interactions between variables. Polynomial feature engineering is widely preferred, especially in regression analysis and machine learning models, to improve accuracy. By going beyond simple linear relationships, this method allows the model to detect more intricate patterns, leading to more precise predictions [41].

The degree of the polynomial plays a crucial role in determining the number of new features added to the model. As the degree of the polynomial increases, the number of input features grows significantly. In this study, third-degree polynomial features were generated. This process creates a new feature matrix that includes all polynomial combinations of features with a degree less than or equal to the specified degree, allowing for more complex interactions between the existing features to be captured.

The addition of new features expands the total number of features, increasing the dimensionality of the data to be processed by the model. This allows the model to access more information, thus enhancing the accuracy of predictions made by machine learning algorithms. In this study, we use the PolynomialFeatures () function from the Scikit-Learn library to transform the original features into higher-degree polynomials. This method helps capture non-linear relationships in the data, aiming to improve the model's performance [42].

3.4. Model interpretation

3.4.1. Shapley Additive Explanations (SHAP)

The SHAP framework facilitates the interpretation of machine learning model performance through the lens of game theory. It employs an additive feature attribution methodology to construct an interpretable model, signifying that the model's output is represented as a linear aggregation of the input variables. SHAP possesses the capability to operate with any category of machine learning model, assess the contribution of each feature by equitably distributing the variable effects across the complete dataset, and examine the interactions among variables [43].

3.4.2. Local Interpretable Model-Agnostic Explanations (LIME)

LIME is an extensively recognized model-agnostic algorithm. It concentrates solely on a particular prediction by selecting a limited number of features and is applicable across various machine learning paradigms. This algorithm produces simulated data points around a sample through stochastic perturbations. It elucidates the results by fitting a weighted sparse linear model on the predictions derived from the perturbed data. The interpretations provided by LIME are locally consistent with a specified sample, irrespective of the classifier employed [44].

3.5. Classification

Machine learning allows computers to learn from data to make automatic decisions and develop models capable of making predictions. This process relies on the use of specific algorithms. Today, machine learning offers us significant opportunities in finance, education, industry, health, and many other fields. (For example, easy processing of image-based data significantly increases a doctor's success rate of incorrect diagnosis).

Machine learning has sub-branches, such as supervised and unsupervised learning. Classification, a specialized domain within supervised learning, represents a prevalent and efficacious methodology in data mining and machine learning. This method works on labeled datasets to categorize data into specific classes or categories, which is crucial in developing predictive models.

This methodology encompasses the allocation of datasets into distinct categories or classes predicated upon specific attributes. It is instrumental in discerning features that interrelate data elements. For this purpose, the dataset is partitioned into training and testing subsets. The training data, constituting approximately 70–80 % of the dataset, is employed for model training, whereas the testing data, comprising 30-20 % of the dataset, is utilized to assess the model's accuracy. The training data is instrumental in facilitating the model's learning of a particular task and identifying patterns within the data.

The training data is used to help the model learn a specific task and recognize patterns in the data. Test data, on the other hand, is employed to assess the efficacy of the model's performance. In this paper, the dataset is partitioned into 80 % allocated for training purposes and 20 % reserved for testing across all classification methodologies. In this paper, the dataset is partitioned into 80 % for training and 20 % for testing across all classification tools.

Different classification tools are chosen according to the data structure and problem type. The classification algorithms used here are discussed under two headings. First, information about AutoML tools is given, and the AutoML tools used here are introduced. Then, information about ClassicML tools is given, and the ClassicML tools used here are described in detail [45].

3.5.1. AutoML Tools

Automatic Machine Learning (AutoML) is a framework that automates the development of machine learning models. Its primary purpose is to simplify and accelerate complex systems that require data science and machine learning expertise. Thus, it enables the creation of machine learning models while rendering them accessible to a broader spectrum of users. AutoML tools automate data preprocessing, model selection, and hyperparameter optimization. The AutoML tools utilized in this

research are as follows:

3.5.1.1. PyCaret. PyCaret is designed for data scientists and analysts. It is a machine-learning library with an open-source code structure. This Python library broadly automates data preparation, model selection, optimization, and model deployment processes. Notable for its user-friendly interface and low coding requirement, PyCaret is an excellent option for beginners in data science. Its prominent features include model comparison, hyperparameter tuning, and automatic ML pipeline creation [26].

3.5.1.2. AutoGluon. AutoGluon is developed by AWS (Amazon Web Services). It enables users to build high-quality machine-learning models quickly. Capable of working with a wide range of data types, this AutoML tool uses advanced techniques such as model merging and multi-task learning to maximize model performance [46].

3.5.1.3. FLAML. FLAML (Fast and Lightweight AutoML) is a Python-based automated machine-learning library developed by Microsoft Research. It was created to minimize computational costs. On the other hand, machine learning automates the model selection and hyperparameter tuning process. It does not require extensive manual intervention by the people doing the machine modeling. Since it minimizes computational costs, it does not require high computational resources and allows them to find the right models efficiently [47].

3.5.1.4. H2O AutoML. H2O AutoML is developed by H2O.ai. It is a comprehensive AutoML solution that automatically tests a set of algorithms to select the most successful model. This platform offers users the opportunity to work on datasets quickly and effectively. It is designed to be accessible to a broad user base and offers high-performance modeling techniques through a user-friendly interface [48].

3.5.1.5. TPOT. Developed by the open-source community, TPOT (Tree-Based Pipeline Optimization Tool) aims to find the optimal machine-learning pipeline using genetic algorithms. This Python-developed AutoML library automatically selects the best ML algorithms and preprocessing steps to achieve the best results from complex datasets. This process saves time for the user and automates the typically manual model selection process [49].

3.5.1.6. MLJAR AutoML. This library is an AutoML service developed by the MLJAR community. It works through a user-friendly web interface. It allows users to upload datasets and quickly train various machine learning models. MLJAR aims to simplify the model development process maximally and enables users to work with various algorithms [50].

3.5.1.7. Auto-PyTorch. Auto-PyTorch is a PyTorch-based AutoML solution focused on the configuration and optimization of deep learning models. It was developed by the University of Tübingen. This model provides detailed control over the model's training process and resource usage. Auto-PyTorch performs important functions such as automatic model selection and hyperparameter tuning for deep learning applications [51].

3.5.2. ClassicML Tools

Classic Machine Learning (ClassicML) tools are software used to create models capable of learning and making predictions from data. These tools attempt to detect patterns and relationships from datasets, often using statistical algorithms and mathematical techniques. ClassicML methods used here respectively:

3.5.2.1. K-Nearest Neighbors (KNN). The k-nearest neighbors (k-NN) method is considered one of the easiest methods among machine learning tools. In contrast to alternative algorithms, k-NN does not construct a model from training data; instead, it performs its learning when utilized with test data, categorizing it as lazy learning. The inputs are p-dimensional vectors represented as:

$$x_i = (x_{1i}, x_{2i}, \ldots, x_{pi}) \in X, \tag{3.2}$$

where X denotes a vector. During the training phase, both the feature vectors and their corresponding class labels are retained. In the classification phase, distances between stored vectors and a new vector are computed, and the k-nearest examples of the new data are selected. Subsequently, the classification of novel data is determined by identifying the most prevalent class among these selected vectors. Although any distance metric may be employed, the Euclidean distance is predominantly utilized (See (3.3)) [52].

$$d(x_i, x_j) = \sqrt{\sum_{j=1}^{p} (x_{ri} - x_{rj})^2}. \tag{3.3}$$

i. K-NeighborsUnif (Uniform Weights)

In this method, each of the selected K neighbors is given equal weight in their vote. That is, the influence of each neighbor is equally considered when making a prediction. Mathematically, for a classification problem, the most frequently occurring class label among the neighbors is directly chosen:

$$y = mod(y_1, y_2, y_3, \ldots, y_i) \tag{3.4}$$

where y_i is the class label of the ith nearest neighbor.

ii. K-NeighborsDist (Distance Weights)

In this approach, the influence of a neighbor is inversely related to its distance, meaning that nearer neighbors have greater weight. This guarantees that predictions are more affected by those neighbors that are closer. For classification, the weighted voting is done as follows:

$$y = \arg max \sum_{i=1}^{K} \omega_i . 1(y_i = c), \tag{3.5}$$

$$\omega_i = \frac{1}{d(x, x_i)^2}, \tag{3.6}$$

where ω_i is the weight of the ith neighbor, and 1 $(y_i = c)$ is an indicator function that is 1 if y_i is of class c and 0 otherwise.

3.5.2.2. GBM (Gradient Boosting Machine). Gradient Boosting Machine (GBM) represents a sophisticated form of ensemble learning methodology. Within this framework, feeble predictors, primarily comprised of decision trees, are consolidated into a sequential configuration to develop a formidable predictive model. GBM is employed in both regression and classification endeavors. The fundamental principle underlying GBM pertains to the incremental enhancement of the series of weak learners through a progressive refinement aimed at rectifying the errors of its antecedent models. During this process, each successive model is meticulously trained to diminish the inaccuracies present in all preceding models. Essentially, this entails the construction of a model at each iteration that aspires to minimize the discrepancy (slopes) between the actual outcomes and the aggregated predictions of all prior models up to that juncture. The main components of GBM are, respectively, weak learners (these are typically decision trees), loss function, gradient descent, and learning rate.

The mathematical model of GBM involves adding a new learner at each step in a way that minimizes the loss function. Suppose $F_m(x)$ is our model at the m-th step. At step $m+1$, we want to add a learner $(\gamma_m h_m(x))$ that will correct the errors of all previous steps. Here, $h_m(x)$ is the new

learner and γ_m measures the contribution of this learner.

The loss function denoted as $L(y, F(x))$ is utilized to evaluate the discrepancy between the observed values y and the predicted values $F(x)$. Throughout the execution of the gradient descent algorithm, the gradient of the loss function

$$g_m = \frac{\partial L(y, F(x))}{\partial F(x)}, \tag{3.7}$$

is calculated. At each step, the $h_m(x)$ learner that best addresses the errors indicated by g_m is found. The learning rate η is used to adjust the impact of the new learner added at each step. The model update is

$$F_{m+1}(x) = F_m(x) + \eta\gamma_m h_m(x), \tag{3.8}$$

where γ_m and $h_m(x)$ are chosen to minimize the loss function [40,45].

3.5.2.3. LightGBM. LightGBM is developed by Microsoft. It is referred to as a lightweight gradient-boosting framework. When processing large datasets, it uses fewer resources and can train quickly. As a tree-based learning algorithm, LightGBM is popular in data science competitions and industrial applications [53].

i. LightGBMXT

LightGBMXT is a variation of LightGBM designed to perform better when working with categorical features. It employs specialized techniques for more efficient processing of categorical features.

ii. LightGBMLarge

LightGBMLarge is another variation of LightGBM, optimized for better performance on large-scale datasets. It offers more efficient memory usage and parallelization techniques to tackle the challenges brought by large datasets.

3.5.2.4. CatBoost. CatBoost constitutes a proficient gradient-boosting algorithm that demonstrates effectiveness with datasets comprising categorical features and exhibits resistance to overfitting. Compared to the traditional Gradient Boosting algorithm, CatBoost introduces two main innovations: Ordered Boosting and Ordered Target Statistics (TS) [54].

3.5.2.5. XGBoost. XGBoost is a more advanced version of the Gradient Boosting algorithm, fine-tuned with numerous optimizations. Key attributes of the algorithm include its ability to deliver robust predictive performance, curb overfitting, handle missing data, and perform these tasks swiftly [55].

3.5.2.6. Random forest. Random Forest (RF) represents a sophisticated algorithm utilized for classification and regression tasks, fundamentally grounded in decision tree structures and employing the principles of ensemble learning techniques. This method uses multiple decision trees to make predictions and combines the predictions of each tree to produce an overall prediction. Random Forest can create trees using two different criteria: gini impurity and entropy [56,41] These two methods are known as RandomForestGini and RandomForestEntropy, respectively:

i. RandomForestGini

RandomForestGini creates trees using the Gini impurity criterion. Gini impurity measures the total product probability that a randomly chosen sample from a node belongs to a particular class and the probability of it being misclassified. The formula for Gini impurity is:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i \log_2(p_i), \tag{3.9}$$

where D is the set of samples in a node, m is the number of classes, and p_i is the proportion of the ith class within D. A diminished Gini impurity value signifies that the node exhibits greater purity, indicating a substantial predominance of a singular class. Random Forest selects features with the lowest Gini impurity value as the branching criteria while creating trees.

ii. RandomForestEntr

RandomForestEntropy creates trees using the entropy criterion. Entropy measures the disorder or uncertainty in a system. The entropy of a node indicates how disorganized the distribution of samples into certain classes at that node is. The formula for entropy is:

$$Entropy(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \tag{3.10}$$

where D is the set of samples in a node, m is the number of classes, and p_i is the proportion of the ith class within D. A higher entropy value indicates that the samples in the node are more mixed. Random Forest selects the best branching feature to reduce entropy while creating trees.

3.5.2.7. ExtraTrees. These classification models are a variation of the RF, a tree-based modeling technique. These models introduce extra randomness in branching criteria and feature selection while creating decision trees, offering more diversity and faster training times than Random Forest. Based on the criteria for branching the trees, there are two main variants: ExtraTreesGini and ExtraTreesEntr [57].

i. ExtraTreesGini

The ExtraTreesGini model utilizes the Gini impurity criterion to determine how trees are branched. Gini impurity measures how mixed the classes within a node are; its value ranges between 0 and 1, where 0 denotes a node exhibiting complete homogeneity and 1 signifies a node characterized by total heterogeneity. The aim is to minimize this criterion when selecting a feature and split point. The Gini impurity for a node is calculated with the formula:

$$Gini(t) = 1 - \sum_{i=1}^{C} p(i|t), \tag{3.11}$$

where $p(i|t)$ is the proportion of class i within node t and C represents the number of classes.

ii. ExtraTreesEntr

The ExtraTreesEntr model uses the Entropy criterion. Entropy measures the uncertainty or disorder of the classes within a node, indicating how disordered the class distribution is. Minimizing entropy maximizes information gain, which in turn helps the model make better separations. Entropy for a node is calculated with the formula:

$$Entropy(t) = -\sum_{i=1}^{C} p(i|t) log_2\, p(i|t), \tag{3.12}$$

where $p(i|t)$, is the proportion of class i within node t and C represents the number of classes.

3.5.2.8. NeuralNetTorch. NeuralNetTorch uses the PyTorch framework to build and train neural networks. PyTorch provides flexibility and powerful tools for researching and developing deep learning models

[18].

3.5.2.9. ANN (Artificial Neural Network). Artificial neural networks (ANN) constitute a sophisticated mathematical framework. They emulate the cognitive processes employed by the human brain to assimilate information. Such models are employed to discern intricate relationships and patterns inherent within datasets [58,59].

i. *ANNwithScikit-Learn*

Scikit-Learn is a library frequently used in many machine-learning applications in Python. Using simple neural network architectures to build deep learning networks, Scikit-Learn offers limited support through the MLPClassifier and MLPRegressor modules.

ii. ANNwithKeras

Keras is integrated with TensorFlow as tf.keras and is designed to build and train deep learning models. It provides a more intuitive and flexible platform for creating complex neural networks.

3.6. Statistical analysis

Statistical analysis plays a crucial role in comparing the performance of machine learning models by going beyond simple averages to assess whether the observed differences are statistically significant. In this context, the one-way ANOVA test is a powerful method used to determine whether there are meaningful differences in performance metrics across multiple models evaluated on different data folds. ANOVA helps to distinguish whether the observed variations are due to random chance or reflect actual performance differences, thereby enhancing the reliability of the comparison. As a result, models with high accuracy, those that produce consistent and stable outcomes across different data splits, can be scientifically validated.

3.6.1. One-way ANOVA and Statistical significance with p-value

One-way ANOVA is a parametric statistical method used to test whether there is a significant difference in the mean values of a dependent variable concerning a single independent (or factor) variable at different levels. This analysis effectively tests whether there is a difference between the means of three or more groups [60,61]. One-way ANOVA splits the total variance into two components:

- Between-group variance: Represents the variance between the means of different groups.
- Within-group variance: Represents the deviation of individuals within the same group from the mean.

The F-statistic is computed using the Sum of Squares (SS), Mean Squares (MS), and degrees of freedom (df), and is defined as follows:

$$F = \frac{MS_{between}}{MS_{within}} = \frac{SS_{between}/df_{between}}{SS_{within}/df_{within}}. \tag{3.13}$$

The p-value expresses the probability that the F-statistic obtained would occur if the null hypothesis (all group means are equal) is accepted as accurate.

- If $p < 0.05$, then the null hypothesis is rejected, i.e., there is a statistically significant difference between the groups.
- If $p \geq 0.05$, it is concluded that the difference between the groups is not statistically significant.

4. Analysis of result

Correlation analysis constitutes a statistical methodology employed to ascertain the association between variables alongside the orientation and intensity of this association. The correlation analysis between the variables we used in our study is shown in Fig. 3. Fig. 3, the most influential variables on output are slope, exang, and cp, while the highest correlation between variables is between (slope, oldpeak), (slope, exang) and (cp, exang) respectively. This tells us that slope, exang, cp, and oldpeak are the variables that we should pay attention to when analyzing the data. At the same time, other variables that affect slope and exang can also be effective in classification. As evidenced in Fig. 3, the correlation coefficients among various features are illustrated through a heat map representation.

The dataset comprises two distinct classes. The allocation of these classes within the dataset is illustrated in Fig. 4. Despite the absence of a significant imbalance between the classes presented in the dataset (Fig. 4-a), the inadequacy of data quantity constitutes a significant problem. In modeling with an insufficient amount of data, the class with more data will naturally be dominant, and therefore, the minority class will be weak in classification.

Balancing methods were used to solve this problem. Thanks to balancing methods, data imbalance between classes is minimized. Among these methods, the number of data was increased from 918 to 1827 by using the SMOTEENN method, which is the method that provides the best results in machine learning models with feature extraction (Fig. 4-b).

4.1. Performance evolution

The 2x2 confusion matrix employed for assessing model efficacy delineates the actual and anticipated outcomes of binary classification. Various evaluative metrics, including accuracy, precision, recall, F1-score, and Area Under the Curve (AUC), are derived from the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values encapsulated within the confusion matrix. Here's the structure of a confusion matrix for a binary classification problem:

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix}, \tag{4.1}$$

metrics obtained from the confusion matrix:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.2}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{4.3}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{4.4}$$

$$\text{F1} - \text{Score} = 2\frac{(precision \times recall)}{(precision + recall)}. \tag{4.5}$$

4.2. Classification results

In this section, we perform a comprehensive analysis to investigate the effectiveness of AutoML and traditional machine learning methodologies. We identify the best-performing methodologies and analyze the results obtained from these methodologies in detail. To perform these analyses, in addition to the standard (raw) dataset, we applied feature engineering techniques, including Average Derivative (AD) and polynomial feature extraction approaches, to increase the amount and variability of data. We also used six different balancing methods to address the class imbalance problem. Using these techniques, we expanded the dataset from an initial 12 to 77 features. We transformed the dataset from 918 samples to a sample size ranging from 1827 to 2016, depending on the methodologies applied. To ensure a standardized evaluation, we allocated 80 % of each dataset for model training

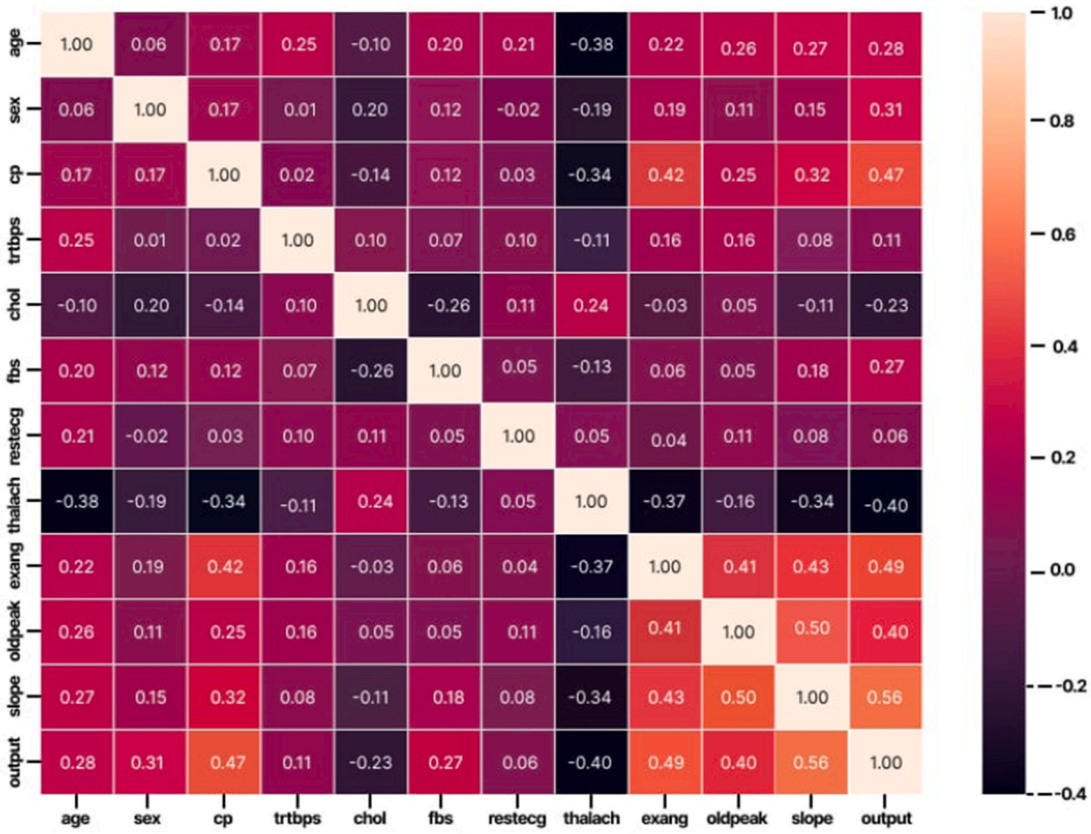

Fig. 3. Correlation matrix for relationships between attributes.

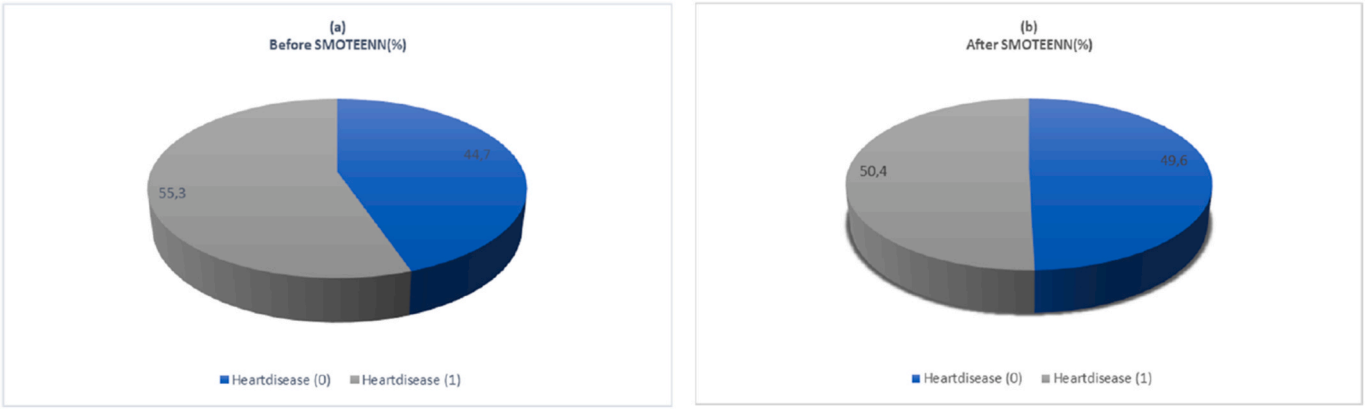

Fig. 4. Distribution of the classes in Smoteenn.

and the remaining 20 % for model testing. Thanks to this method, we systematically analyzed model performances within a specific standard framework.

The model performance evaluation in classification problems is usually based on metrics such as accuracy, F1 score, precision, and recall. While the accuracy metric is frequently utilized as a performance indicator in such contexts, the accuracy alone may not fully reflect the model's success. It is important to assess not only the accuracy metric but also supplementary metrics such as the F1 score, precision, and recall to gain a holistic understanding of the model's overall performance and reliability. In the domain of healthcare modeling, beyond the metric of accuracy, recall emerges as a salient evaluation criterion. In particular, a high recall value plays a critical role in accurately predicting positive classes.

In this study, the performance of a total of seven AutoML and fifteen Classic ML models was analyzed. Default settings were used for AutoML tools, while traditional steps were followed in Classic ML modeling to construct the most effective machine learning models. To evaluate the performance of all classical machine learning models and the artificial neural network (ANN) model, 10-fold cross-validation was applied. Additionally, the optimal hyperparameters for each model were determined using the GridSearchCV method. The hyperparameter settings such as learning_rate, number of iterations, and depth for tree-based models; learning_rate, n_estimators and max_depth for boosting-based models; n_neighbors, weights: uniform/distance and distance metric (p: Euclidean/Manhattan) for K-Nearest Neighbors (KNN) algorithm were optimized. For artificial neural networks (ANN), activation functions (ReLU, sigmoid), hidden_layer_sizes, and optimization algorithms

(Adam optimizer) were used. The results obtained under four different conditions were compared using four distinct evaluation metrics. These comparisons were presented through tables, which were subsequently visualized using graphical representations. The conditions considered in the analysis are as follows:

- The case where neither feature extraction nor balancing techniques are used (S),
- The case where only the balancing technique is applied (B),
- The case where only feature extraction is applied (F),
- The case where balancing techniques and feature extraction are applied together (F&B).

As seen from Table 2, Table 3, Table 4, and Fig. 5, the combination of feature extraction and balancing techniques significantly improves the stability and overall performance of the model. The use of balancing techniques generally led to improvements in the accuracy and recall metrics. Especially in the context of health data, a high recall value is a critical factor, as it indicates that the model is less likely to miss diseases. In this context, SMOTEENN and ROS methods performed well in terms of accuracy and recall in both traditional machine learning (ClassicML) and automated machine learning (AutoML) models.

However, the Borderline-SMOTE and ADASYN methods underperformed in some traditional machine learning models (Random Forest- Entropy-Gini, GBM, and Neural Networks with Keras) compared to the cases without feature extraction and balancing techniques.

As a result of these analyses, a detailed model performance comparison will be conducted at the end of this section, focusing on the condition under which the best-performing AutoML and Classic ML models were obtained. In the subsequent Discussion and Conclusion sections, a more in-depth evaluation and interpretation will be provided by focusing on the models that achieved the highest performance under the identified condition.

In contrast to traditional machine learning methods, AutoML models using feature extraction and balancing techniques generally outperformed, except for one model (PyCaret), compared to scenarios without these techniques.

Table 2 and Fig. 6 show that among the traditional machine learning models, only XGBoost, KNN (dist), and Extra Tree (Ent) models maintained or increased their recall. In the other models, feature extraction caused a significant decrease in the recall value.

In terms of AutoML models, feature extraction only led to a significant performance improvement in the PyTorch model. In contrast, the H2O AutoML model did not cause any change in the accuracy and recall metrics. In contrast, in the other AutoML models, feature extraction led to a significant decrease in accuracy and recall. This suggests that, in some cases, feature extraction may weaken the model's ability to detect positive classes.

As seen in Table 2, Table 4, and Fig. 7, in traditional machine learning (ClassicML) models, the ADASYN method works well with Extra Tree (Entropy-Gini) models. At the same time, it exhibits poor compatibility with Artificial Neural Networks (ANN) models. In general, balancing techniques significantly increase model performance, and this effect is observed to be more pronounced, especially in Artificial Neural Networks (ANN) models implemented with KNN (dist-unif), Extra Tree (Entropy-Gini), and Scikit-Learn.

In contrast to feature extraction, balancing methods other than ADASYN significantly reduce the likelihood of models missing diseases. In particular, ROS and SMOTEENN methods improve model performance by providing better recognition of low-frequency classes. On the other hand, when feature extraction and balancing techniques are not used, model performance is generally low, and there is a significant performance loss due to unbalanced data distribution.

In terms of AutoML models, unlike traditional machine learning models, all AutoML models with balancing techniques performed better than the models without balancing techniques. Therefore, the application of balancing techniques in AutoML models provides a significant advantage in terms of model performance, which is clearly seen in Fig. 7.

From Table 2, Table 3, Table 4, and Fig. 8, it is clearly shown that applying balancing techniques to the dataset (i.e., removing class imbalance) yields better results than adding new features to the dataset. This trend is generally observed regardless of the model type, with ROS and SMOTEENN methods achieving the best performance for both traditional and automated machine learning models.

When Fig. 9 is examined, it is seen that balancing techniques such as ADASYN and Borderline SMOTE used with feature extraction in some traditional machine learning models (Random Forest (Entropy-Gini) and GBM) do not provide the expected improvement. On the contrary, these

Table 2
Accuracy, F1 score, Precision and Recall results in AutoML and ClassicML models.

	The case where neither feature extraction nor balancing techniques are used (S)				The case where only feature extraction is applied (F)			
AutoML Models	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
PyCaret	0.8693	0.8831	0.8735	0.8958	0.862900	0.877600	0.861900	0.895300
H2OAutoML	0.889502	0.888354	0.901645	0.889503	0.889502	0.889455	0.889626	0.889503
TPOT	0.891304	0.891304	0.891304	0.891304	0.853261	0.853813	0.855633	0.853261
AutoGluon	0.891304	0.906542	0.906542	0.906542	0.885870	0.886149	0.886904	0.885870
FlaML	0.885869	0.886148	0.886904	0.885869	0.880434	0.880952	0.883352	0.880434
Mljar	0.885870	0.882076	0.884644	0.880022	0.880435	0.876706	0.878287	0.875349
PyTorch	0.869565	0.866907	0.865036	0.869644	0.875000	0.871815	0.871190	0.872496
ClassicML Models	**Accuracy**	**F1-Score**	**Precision**	**Recall**	**Accuracy**	**F1-Score**	**Precision**	**Recall**
Knn_dist	0.858695	0.875	0.900990	0.850467	0.887681	0.903426	0.923566	0.884146
Knn_unif	0.869565	0.8875	0.910256	0.865853	0.840579	0.903426	0.923566	0.884146
lightGBM_xt	0.907609	0.920188	0.924528	0.915888	0.905797	0.919753	0.931250	0.908537
lightGBM_large	0.896739	0.909953	0.923077	0.897196	0.880435	0.897196	0.917197	0.878049
lightGBM	0.880435	0.894231	0.920792	0.869159	0.869565	0.884615	0.910891	0.859813
XGBoost	0.880435	0.895238	0.912621	0.878505	0.898551	0.913043	0.930380	0.896341
Catboost	0.902174	0.915888	0.915888	0.915888	0.858696	0.877743	0.903226	0.853659
RandomForest_ent	0.891304	0.905660	0.914286	0.897196	0.858695	0.878504	0.898089	0.859756
RandomForest_Gini	0.885870	0.900474	0.913462	0.887850	0.862319	0.882716	0.893750	0.871951
ExtraTrees_ent	0.891304	0.903846	0.930693	0.878505	0.885870	0.901408	0.905660	0.897196
ExtraTrees_Gini	0.891304	0.904762	0.922330	0.887850	0.875000	0.889952	0.911765	0.869159
GBM	0.887681	0.904615	0.913043	0.896341	0.880435	0.895899	0.928105	0.865854
ANNwithKeras	0.858696	0.872549	0.917526	0.831776	0.831522	0.851675	0.872549	0.831776
ANNwithScikit-Learn	0.864130	0.881517	0.894231	0.869159	0.842391	0.859903	0.890000	0.831776
NeuralNetTorch	0.875	0.886699	0.9375	0.841121	0.864130	0.878048	0.918367	0.841121

Table 3
Accuracy, F1 score, Precision, and Recall results in AutoML models.

		The case where only the balancing technique is applied (B)				The case where both balancing techniques and feature extraction are applied together (F&B)			
Balances techniques	AutoML Models	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
ADAYSN	PyCaret	0.9107	0.9097	0.9183	0.9025	0.891200	0.888800	0.899600	0.879200
	H2OAutoML	0.924479	0.924490	0.924614	0.924479	0.919271	0.919276	0.920291	0.919271
	TPOT	0.949622	0.949651	0.950114	0.949622	0.929471	0.929488	0.929552	0.929471
	AutoGluon	0.939547	0.939587	0.940390	0.939547	0.914358	0.914414	0.915213	0.914358
	FlaML	0.944584	0.944620	0.945424	0.944584	0.944584	0.944608	0.944824	0.944584
	Mljar	0.939547	0.939377	0.939377	0.939377	0.924433	0.924433	0.926676	0.926894
	PyTorch	0.921914	0.921864	0.921983	0.923165	0.899244	0.899192	0.899467	0.900565
SMOTE	PyCaret	0.9430	0.9433	0.9393	0.9479	0.936700	0.937000	0.935400	0.939500
	H2OAutoML	0.944723	0.944724	0.944724	0.944724	0.982490	0.982484	0.982774	0.982490
	TPOT	0.941032	0.941072	0.941158	0.941032	0.945946	0.945983	0.946065	0.945946
	AutoGluon	0.945946	0.945983	0.946065	0.945946	0.943489	0.943508	0.943539	0.943489
	FlaML	0.945945	0.945945	0.945945	0.945945	0.941031	0.941108	0.941369	0.941031
	Mljar	0.945946	0.944619	0.946920	0.942759	0.955774	0.954951	0.954360	0.955591
	PyTorch	0.941031	0.939676	0.941099	0.938448	0.931203	0.929623	0.931013	0.928423
SMOTEENN	PyCaret	0.9828	0.9826	0.9906	0.9751	0.975000	0.974900	0.976600	0.973500
	H2OAutoML	0.983099	0.983099	0.983163	0.983099	0.983099	0.983099	0.983163	0.983099
	TPOT	0.978142	0.978142	0.978142	0.978142	0.983607	0.983614	0.983857	0.983607
	AutoGluon	0.978142	0.978156	0.978695	0.978142	0.972677	0.973544	0.994594	0.953367
	FlaML	0.972677	0.972690	0.972936	0.972677	0.983606	0.983614	0.983857	0.983606
	Mljar	0.978142	0.978100	0.977760	0.978676	0.986339	0.986316	0.985955	0.987047
	PyTorch	0.975409	0.975378	0.975089	0.976384	0.953551	0.953523	0.953714	0.955060
ROS	PyCaret	0.9571	0.9572	0.9573	0.9578	0.959200	0.959200	0.959600	0.959200
	H2OAutoML	0.957286	0.957290	0.957403	0.957286	0.967337	0.967338	0.967352	0.967337
	TPOT	0.975430	0.975462	0.975685	0.975430	0.958231	0.958341	0.959397	0.958231
	AutoGluon	0.965602	0.965625	0.965696	0.965602	0.972973	0.973000	0.973134	0.972973
	FlaML	0.968058	0.968090	0.968229	0.968058	0.975429	0.975461	0.975684	0.975429
	Mljar	0.980344	0.979922	0.980646	0.979249	0.982801	0.982444	0.982796	0.982106
	PyTorch	0.965601	0.964913	0.964913	0.964913	0.958230	0.957537	0.956228	0.959150
BORDERLİNE	PyCaret	0.9367	0.9363	0.9418	0.9310	0.913500	0.911800	0.930200	0.894500
	H2OAutoML	0.939698	0.939650	0.940363	0.939698	0.909548	0.909557	0.909750	0.909548
	TPOT	0.933661	0.933727	0.933896	0.933661	0.931204	0.931369	0.932260	0.931204
	AutoGluon	0.931204	0.931402	0.932731	0.931204	0.918919	0.919048	0.919458	0.918919
	FlaML	0.918918	0.918889	0.918872	0.918918	0.936117	0.936237	0.936768	0.936117
	Mljar	0.936118	0.934930	0.934365	0.935542	0.938575	0.937390	0.937094	0.937697
	PyTorch	0.914004	0.912987	0.911056	0.916847	0.904176	0.903145	0.901297	0.907524
SVM	PyCaret	0.9374	0.9376	0.9347	0.9409	0.933200	0.932600	0.943600	0.922600
	H2OAutoML	0.942211	0.942199	0.943628	0.942211	0.934673	0.934673	0.934673	0.934673
	TPOT	0.963145	0.963182	0.963325	0.963145	0.945946	0.946076	0.946943	0.945946
	AutoGluon	0.953317	0.953440	0.954504	0.953317	0.943489	0.943610	0.944295	0.943489
	FlaML	0.943488	0.943610	0.944294	0.943488	0.945945	0.946047	0.946564	0.945945
	Mljar	0.950860	0.950012	0.948937	0.951281	0.948403	0.947612	0.945983	0.949828
	PyTorch	0.948402	0.947612	0.945982	0.949827	0.926289	0.925293	0.923423	0.928325

techniques did not create a significant change in accuracy values but caused a decrease in recall values (number of true positives).

In AutoML models, it was observed that only the PyCaret model using ADASYN reduced the true positive detection rate. In all other cases, feature extraction combined with balancing techniques yielded better results than feature extraction alone.

Table 3, Table 4 and Fig. 10 show that the highest accuracy values in ClassicML models are obtained in KNN (dist-unif) and Neural-Net models where balancing techniques and feature extraction are used together. On the other hand, the best recall values were obtained with the ExtraTrees Gini model trained using only balancing techniques. In AutoML models, the highest accuracy and recall values are obtained with MLjar, which is trained with balancing techniques and feature extraction. This situation indicates that while feature extraction combined with balancing techniques can lead to significant performance improvements in some models, it may negatively impact the ability of certain models to detect positive classes.

Especially in decision tree-based models such as Random Forest and ExtraTrees and neural network-based models such as ANN (Keras and Scikit-Learn) trained with SMOTE, SMOTEENN, ROS, and SVM SMOTE methods in CLasicML, it has been observed that feature extraction causes a decrease in accuracy and recall values.

In contrast, in Table 4 and the KNN (unif-dist) and Neural-Net models, feature extraction increases accuracy and recall. In general, feature extraction decreases the accuracy and recall values in ADASYN and Borderline SMOTE methods. This result shows the importance of evaluating the effect of the balancing techniques used in feature extraction. For example, while some balancing techniques, such as ROS, may give good results with feature extraction, some techniques, such as Borderline SMOTE may have a negative effect.

If the goal is to maintain or increase recall, ADASYN, and Borderline SMOTE may be negatively affected by feature extraction. On the other hand, in terms of accuracy, it has been observed that feature extraction usually leads to small losses, but improvement is achieved with the ROS method. In AutoML models, feature extraction with ADASYN, Borderline SMOTE, and SVM SMOTE methods decreased the accuracy and recall values. In contrast, feature extraction with ROS and SMOTEENN methods increased the accuracy and recall values.

When Tables 3 and 4 are analyzed, the average accuracy rate of the models trained with data without feature engineering is 95.01 %. In comparison, the accuracy rate of the models with feature engineering and data balancing is 94.52 %. Although the difference is small, the accuracy of the models without feature engineering is higher. Similarly, the average recall of the model without feature engineering is 94.98 %, while the recall of the model with feature engineering is 94.40 %. This reveals that the models without feature engineering perform better in detecting true positives. Overall, the models without feature engineering perform slightly better in all metrics.

Table 4
Accuracy, F1 score, Precision, and Recall results in ClassicML models.

		The case where only the balancing technique is applied (B)				The case where both balancing techniques and feature extraction are applied together (F&B).			
Balances Techniques	ClassicML Models	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
ADAYSN	Knn_dist	0.942065	0.942355	0.989473	0.899521	0.931989	0.933333	0.964285	0.904306
	Knn_unif	0.942065	0.942355	0.989473	0.899521	0.934508	0.935	0.979057	0.894736
	lightGBM_xt	0.891688	0.895377	0.910891	0.880383	0.919395	0.921569	0.944724	0.899522
	lightGBM_large	0.899244	0.901478	0.928934	0.875598	0.916877	0.919315	0.940000	0.899522
	lightGBM	0.899244	0.901478	0.928934	0.875598	0.886650	0.889435	0.914141	0.866029
	XGBoost	0.909396	0.910299	0.931973	0.889610	0.919395	0.922705	0.931707	0.913876
	Catboost	0.924433	0.926829	0.945274	0.909091	0.934509	0.936585	0.955224	0.918660
	RandomForest_ent	0.931990	0.934940	0.941748	0.928230	0.866499	0.868486	0.902062	0.837321
	RandomForest_Gini	0.937028	0.939467	0.950980	0.928230	0.866499	0.868486	0.902062	0.837321
	ExtraTrees_ent	0.952141	0.953995	0.965686	0.942584	0.934509	0.937198	0.946341	0.928230
	ExtraTrees_Gini	0.954660	0.956311	0.970443	0.942584	0.931990	0.934940	0.941748	0.928230
	GBM	0.902685	0.903333	0.928082	0.879870	0.885906	0.887043	0.908163	0.866883
	ANNwithKeras	0.831234	0.836186	0.855000	0.818182	0.871537	0.875306	0.895000	0.856459
	ANNwithScikit-Learn	0.916877	0.917706	0.958333	0.880383	0.909320	0.910891	0.943590	0.880383
	NeuralNetTorch	0.846347	0.852300	0.862745	0.842105	0.876574	0.875318	0.934782	0.822966
SMOTE	Knn_dist	0.950859	0.956521	0.964912	0.948275	0.960687	0.965065	0.977876	0.952586
	Knn_unif	0.958230	0.962962	0.973568	0.952586	0.960687	0.965065	0.977876	0.952586
	lightGBM_xt	0.931204	0.939394	0.943478	0.935345	0.950860	0.956710	0.960870	0.952586
	lightGBM_large	0.936118	0.944681	0.932773	0.956897	0.923833	0.933333	0.931330	0.935345
	lightGBM	0.923833	0.933333	0.931330	0.935345	0.936118	0.944206	0.940171	0.948276
	XGBoost	0.936066	0.941529	0.942943	0.940120	0.938575	0.946004	0.948052	0.943966
	Catboost	0.953317	0.958606	0.969163	0.948276	0.953317	0.958606	0.969163	0.948276
	RandomForest_ent	0.948403	0.954839	0.952790	0.956897	0.909091	0.920771	0.914894	0.926724
	RandomForest_Gini	0.948403	0.954839	0.952790	0.956897	0.916462	0.927350	0.919492	0.935345
	ExtraTrees_ent	0.950860	0.956710	0.960870	0.952586	0.950860	0.956710	0.960870	0.952586
	ExtraTrees_Gini	0.950860	0.956710	0.960870	0.952586	0.953317	0.958785	0.965066	0.952586
	GBM	0.939344	0.944858	0.940653	0.949102	0.934426	0.940653	0.932353	0.949102
	ANNwithKeras	0.936118	0.944206	0.940171	0.948276	0.904177	0.915401	0.921397	0.909483
	ANNwithScikit-Learn	0.941032	0.947368	0.964286	0.931034	0.914005	0.924078	0.930131	0.918103
	NeuralNetTorch	0.896805	0.909871	0.905982	0.913793	0.916461	0.927659	0.915966	0.939655
SMOTEENN	Knn_dist	0.986338	0.986876	1	0.974093	0.989071	0.989528	1	0.979274
	Knn_unif	0.981785	0.982269	0.985765	0.978798	0.989071	0.989528	1	0.979274
	lightGBM_xt	0.974499	0.974820	0.992674	0.957597	0.978142	0.978417	0.996337	0.961131
	lightGBM_large	0.963570	0.964158	0.978182	0.950530	0.967213	0.967626	0.985348	0.950530
	lightGBM	0.959016	0.960836	0.968421	0.953368	0.969945	0.970976	0.989247	0.953368
	XGBoost	0.961749	0.962433	0.974638	0.950530	0,965392	0,965889	0,981752	0,95053
	Catboost	0.972678	0.972973	0.992647	0.954064	0.978142	0.978339	1	0.957597
	RandomForest_ent	0.976321	0.976827	0.985612	0.968198	0.941712	0.942857	0.953069	0.932862
	RandomForest_Gini	0.972678	0.973958	0.979058	0.968912	0.943534	0.945133	0.946809	0.943463
	ExtraTrees_ent	0.981785	0.982014	1	0.964664	0.972678	0.973070	0.989051	0.957597
	ExtraTrees_Gini	0.983607	0.984211	1	0.968912	0.967213	0.967742	0.981818	0.954064
	GBM	0.965392	0.966132	0.974820	0.957597	0.963570	0.963899	0.985240	0.943463
	ANNwithKeras	0.980874	0.981627	0.994681	0.968912	0.964481	0.966057	0.973684	0.958549
	ANNwithScikit-Learn	0.983607	0.984211	1	0.968912	0.969945	0.971429	0.973958	0.968912
	NeuralNetTorch	0.975409	0.976623	0.979166	0.974093	0.989071	0.989528	1	0.979274
ROS	Knn_dist	0.968058	0.971677	0.982378	0.961206	0.977886	0.980561	0.982683	0.978448
	Knn_unif	0.960655	0.964179	0.961309	0.967065	0.970491	0.973134	0.970238	0.976047
	lightGBM_xt	0.958231	0.963283	0.965368	0.961207	0.970516	0.973913	0.982456	0.965517
	lightGBM_large	0.975430	0.978448	0.978448	0.978448	0.977887	0.980477	0.986900	0.974138
	lightGBM	0.972973	0.976035	0.986784	0.965517	0.972973	0.976242	0.978355	0.974138
	XGBoost	0.962295	0.965517	0.966967	0.964072	0.980344	0.982684	0.986957	0.978448
	Catboost	0.977887	0.980645	0.978541	0.982759	0.977887	0.980477	0.986900	0.974138
	RandomForest_ent	0.972973	0.976139	0.982533	0.969828	0.970516	0.973913	0.982456	0.965517
	RandomForest_Gini	0.977887	0.980562	0.982684	0.978448	0.970516	0.973913	0.982456	0.965517
	ExtraTrees_ent	0.980344	0.982759	0.982759	0.982759	0.972973	0.976344	0.974249	0.978448
	ExtraTrees_Gini	0.982801	0.984946	0.982833	0.987069	0.972973	0.976344	0.974249	0.978448
	GBM	0.954098	0.957831	0.963636	0.952096	0.963934	0.967164	0.964286	0.970060
	ANNwithKeras	0.936118	0.942731	0.963964	0.922414	0.948403	0.954048	0.968889	0.939655
	ANNwithScikit-Learn	0.970516	0.974026	0.978261	0.969828	0.958231	0.962637	0.982063	0.943966
	NeuralNetTorch	0.923832	0.931567	0.954751	0.909482	0.950859	0.952586	0.960869	0.956709
BORDERLİNE	Knn_dist	0.933660	0.941684	0.943722	0.939655	0.931203	0.939655	0.939655	0.939655
	Knn_unif	0.933660	0.941684	0.943722	0.939655	0.931203	0.939655	0.939655	0.939655
	lightGBM_xt	0.916462	0.925439	0.941964	0.909483	0.918919	0.927790	0.942222	0.913793
	lightGBM_large	0.921376	0.930736	0.934783	0.926724	0.891803	0.898148	0.926752	0.871257
	lightGBM	0.882064	0.893805	0.918182	0.870690	0.899263	0.910284	0.924444	0.896552
	XGBoost	0.894349	0.905495	0.923767	0.887931	0.914005	0.923414	0.937778	0.909483
	Catboost	0.938575	0.945770	0.951965	0.939655	0.931204	0.938865	0.951327	0.926724
	RandomForest_ent	0.928747	0.936264	0.955157	0.918103	0.862408	0.873874	0.915094	0.836207
	RandomForest_Gini	0.926290	0.934211	0.950893	0.918103	0.864865	0.875847	0.919431	0.836207
	ExtraTrees_ent	0.941032	0.948052	0.952174	0.943966	0.928747	0.936819	0.947137	0.926724
	ExtraTrees_Gini	0.941032	0.948052	0.952174	0.943966	0.931204	0.938596	0.955357	0.922414
	GBM	0.916393	0.921899	0.943574	0.901198	0.883607	0.890601	0.917460	0.865269

(*continued on next page*)

Table 4 (*continued*)

		The case where only the balancing technique is applied (B)				The case where both balancing techniques and feature extraction are applied together (F&B).			
Balances Techniques	ClassicML Models	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
	ANNwithKeras	0.872236	0.884956	0.909091	0.862069	0.874693	0.888403	0.902222	0.875000
	ANNwithScikit-Learn	0.918919	0.927473	0.946188	0.909483	0.916462	0.925110	0.945946	0.905172
	NeuralNetTorch	0.877149	0.891774	0.895652	0.887931	0.891891	0.903930	0.915929	0.892241
SVM	Knn_dist	0.963144	0.967320	0.977973	0.956896	0.975429	0.978260	0.986842	0.969827
	Knn_unif	0.963144	0.967320	0.977973	0.956896	0.975429	0.978260	0.986842	0.969827
	lightGBM_xt	0.896806	0.907080	0.931818	0.883621	0.945946	0.951542	0.972973	0.931034
	lightGBM_large	0.928747	0.935982	0.959276	0.913793	0.931204	0.938053	0.913793	0.963636
	lightGBM	0.918919	0.926174	0.962791	0.892241	0.921376	0.930131	0.942478	0.918103
	XGBoost	0.924590	0.930931	0.933735	0.928144	0.931204	0.939130	0.947368	0.931034
	Catboost	0.958231	0.962637	0.982063	0.943966	0.941032	0.947137	0.968468	0.926724
	RandomForest_ent	0.948403	0.953846	0.973094	0.935345	0.899263	0.909890	0.928251	0.892241
	RandomForest_Gini	0.945902	0.949309	0.974763	0.925150	0.891803	0.900000	0.911043	0.889222
	ExtraTrees_ent	0.958231	0.962963	0.973568	0.952586	0.953317	0.958606	0.969163	0.948276
	ExtraTrees_Gini	0.960688	0.965217	0.973684	0.956897	0.945946	0.951754	0.968750	0.935345
	GBM	0.909836	0.916793	0.926606	0.907186	0.914754	0.920732	0.937888	0.904192
	ANNwithKeras	0.867322	0.877273	0.927885	0.831897	0.886978	0.896396	0.938679	0.857759
	ANNwithScikit-Learn	0.938575	0.946467	0.940426	0.952586	0.938575	0.945295	0.960000	0.931034
	NeuralNetTorch	0.874692	0.884353	0.933014	0.840517	0.906633	0.915178	0.949074	0.883620

In this context, we can begin to evaluate the results obtained by examining in detail the machine learning models used on the dataset obtained by applying feature extraction and balancing techniques.

If we begin analyzing AutoML models, the results indicate that MLjar, FlaML, TPOT, PyCaret, and H2O AutoML achieved the highest accuracy values, respectively. These findings are presented in detail in Table 3 and Fig. 11, where the performance comparisons are visualized.

Mljar has shown the best performance with approximately 98.6 % accuracy, 98.6 % F1 score, 98.6 % precision, and 98.7 % recall. TPOT follows MLjar with high values of 0.983607 accuracy, 0.983614 F1 score, 0.983857 precision, and 0.983607 recall. This makes TPOT the model with the second-highest performance. On the contrary, Pytorch has the lowest accuracy with 0.953551. As shown in Table 3, the accuracy values of AutoML tools range from 0.953551 to 0.986339.

To facilitate a comparative analysis between the performance of ClassicML models and those constructed using AutoML tools, we utilized Python libraries, including sklearn, Seaborn, Pandas, and Matplotlib, alongside employing visual representations. Among the ClassicML models created by executing the traditional steps, Knn_unif, Knn_dist, and NeuralNetTorch showed the highest performance metrics with the same accuracy values (Table 4).

Knn_unif, Knn_dist, and NeuralNetTorch have demonstrated the best performance metrics with an accuracy of 0.989071, an F1 score of 0.989528, a precision of 1, and a 0.979274 recall value. As we can see in Tables 3 and 4, this performance is better than that of the AutoML tools. On the contrary, RandomForest_ent showed the worst performance with an accuracy of 0.941712.

An examination of the performance metrics pertaining to ClassicML tools reveals, as illustrated in Table 4 that the accuracy values range from 0.941712 to 0.989071, F1 score values fluctuate between 0.942857 and 0.989528, precision values extend from 0.946809 to 1, and recall values vary between 0.932862 and 0.979274. The findings indicate that all metrics exhibit a complementary relationship (Fig. 14).

To compare the model performances, a one-way ANOVA test was performed on the accuracy, precision, recall, and F1-score values obtained for each fold at the end of the k-fold cross-validation process. The F-statistic and p-value for the results of this test are presented in Table 5.

5. Discussion

Machine learning can fundamentally be characterized as a rudimentary artificial intelligence paradigm that facilitates the formulation and advancement of mathematical models to enhance data comprehension [62]. Nowadays, creating a resilient predictive model utilizing machine learning methodologies is critically important for the swift detection of diverse diseases and the execution of efficacious interventions.

This process, which started in the 1990s with computers in disease detection and diagnosis, continues today with artificial intelligence. Therefore, artificial intelligence has become an important research topic as a constantly evolving intelligence simulation in all areas of life.

In general, the data collected is raw data, and data science is used to process this raw data and make it useable. A prevalent challenge within health data is the insufficiency of the number of data available. The insufficiency of the number of data available leads to underperformance in learning, or vice versa, to overfitting.

Another problem is missing data. The lack of data appears to be a critical factor in the failure of machine learning algorithms. Therefore, within the existing literature, missing data are frequently either disregarded or imputed utilizing various assignment techniques such as mean, median, and regression imputation methods [13; 40; 45].

Our study differs from other studies' innovative approaches to data preprocessing and comprehensive data analysis. Within the scope of the study, new features were added to the heart disease data, imbalances between classes were minimized, and the dataset was made more balanced. Then, these data were applied to AutoML and Classic ML models, and a comprehensive performance comparison was made. In addition to the correlation analysis performed on the raw data, SHAP and LIME analyses were performed on the AutoML and ClassicML models with the highest accuracy rates, and the effects of the variables used after normalization, data balancing, and feature extraction on the models were observed. The analyses presented in Figs. 17 and 18 show the contributions of the variables to the prediction process in detail.

According to Fig. 17, the KNN-dist and KNN-unif relied heavily on exang_1 (chest pain during exercise) and slope_1 (ST slope after exercise) variables in the prediction process. The KNN-dist performed a more precise evaluation in the prediction process, giving more weight to close neighbors. In contrast, the KNN-unif performed a more balanced prediction process by assigning equal weight to neighbors.

The MLjar model distributed the variable contributions more evenly in the prediction process and used the variable exang_1 = 1 (Chest pain during exercise) in the positive prediction direction. Moreover, the MLjar is considered the most flexible model and includes an uncertainty of 1 % in its negative predictions. The NeuralNetTorch is the most stable model compared to the other models and uses the variables sex_1 (individual being male) and cp_1 (type of chest pain) as the strongest predictors for positive prediction. In particular, being male is a factor that increases the risk of disease. In addition, the variables slope_1 and

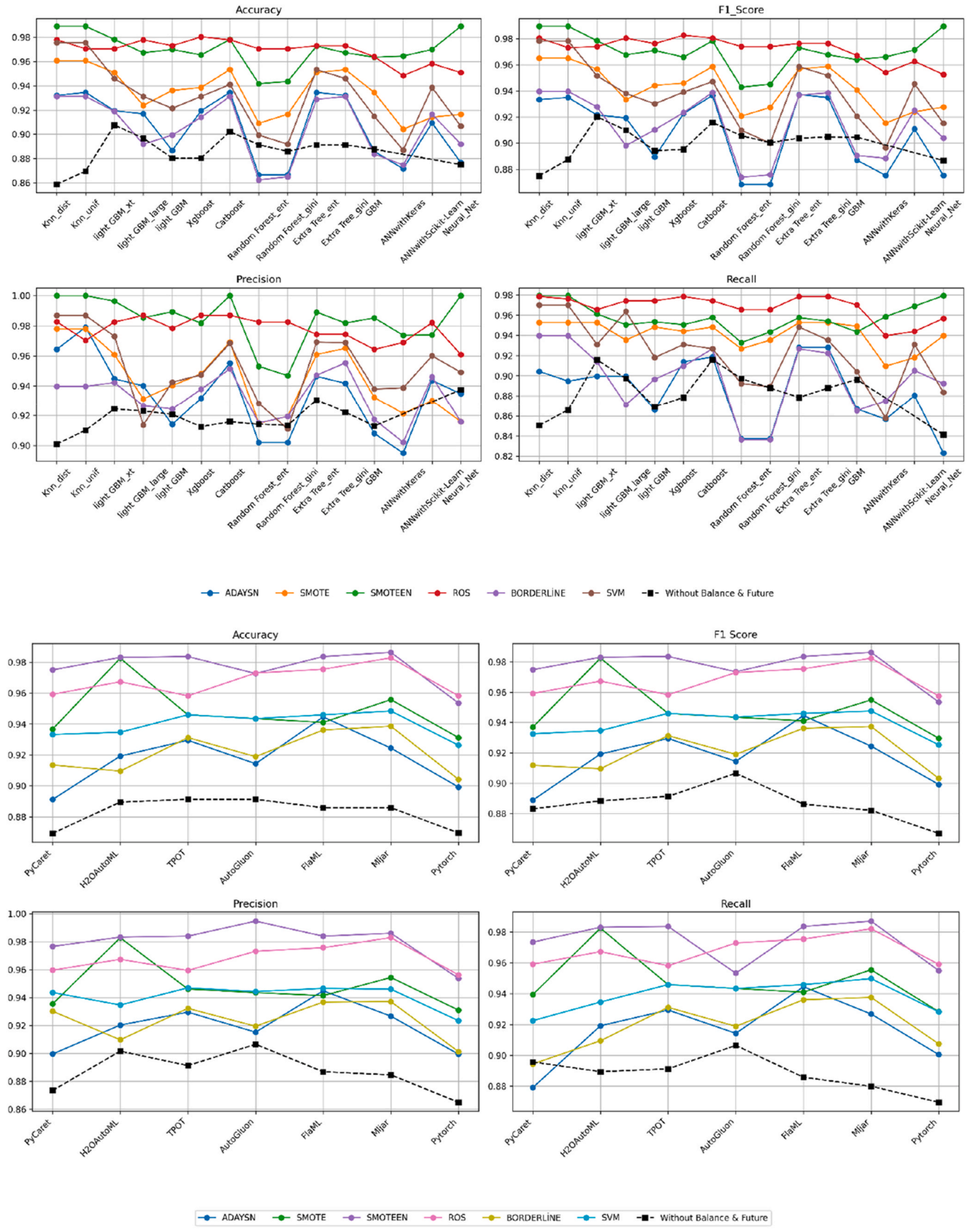

Fig. 5. Comparison between the case **F&B** and the case **S** for ClassicML and AutoML.

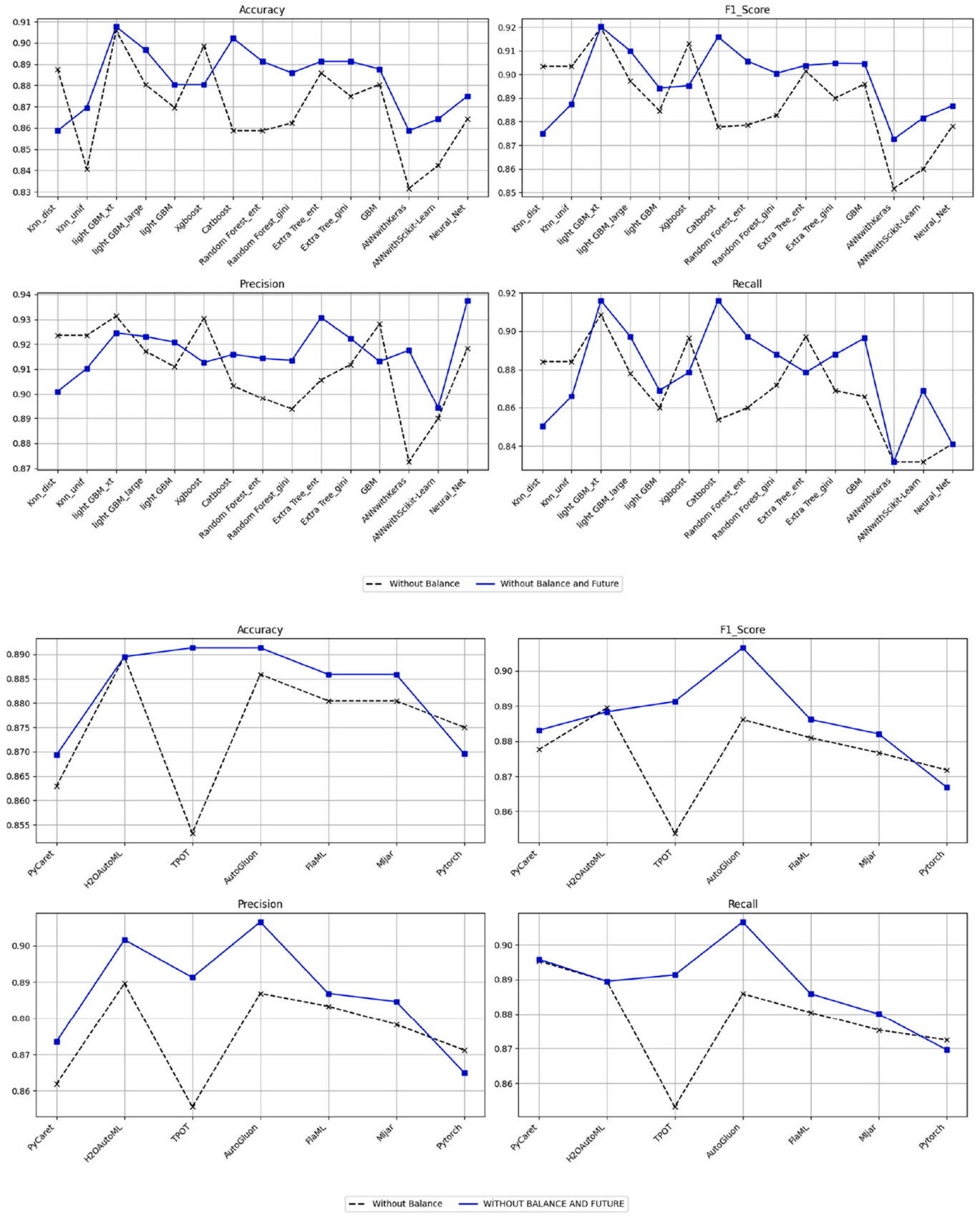

Fig. 6. Comparison between the case **S** and the case **F** for ClassicML and AutoML.

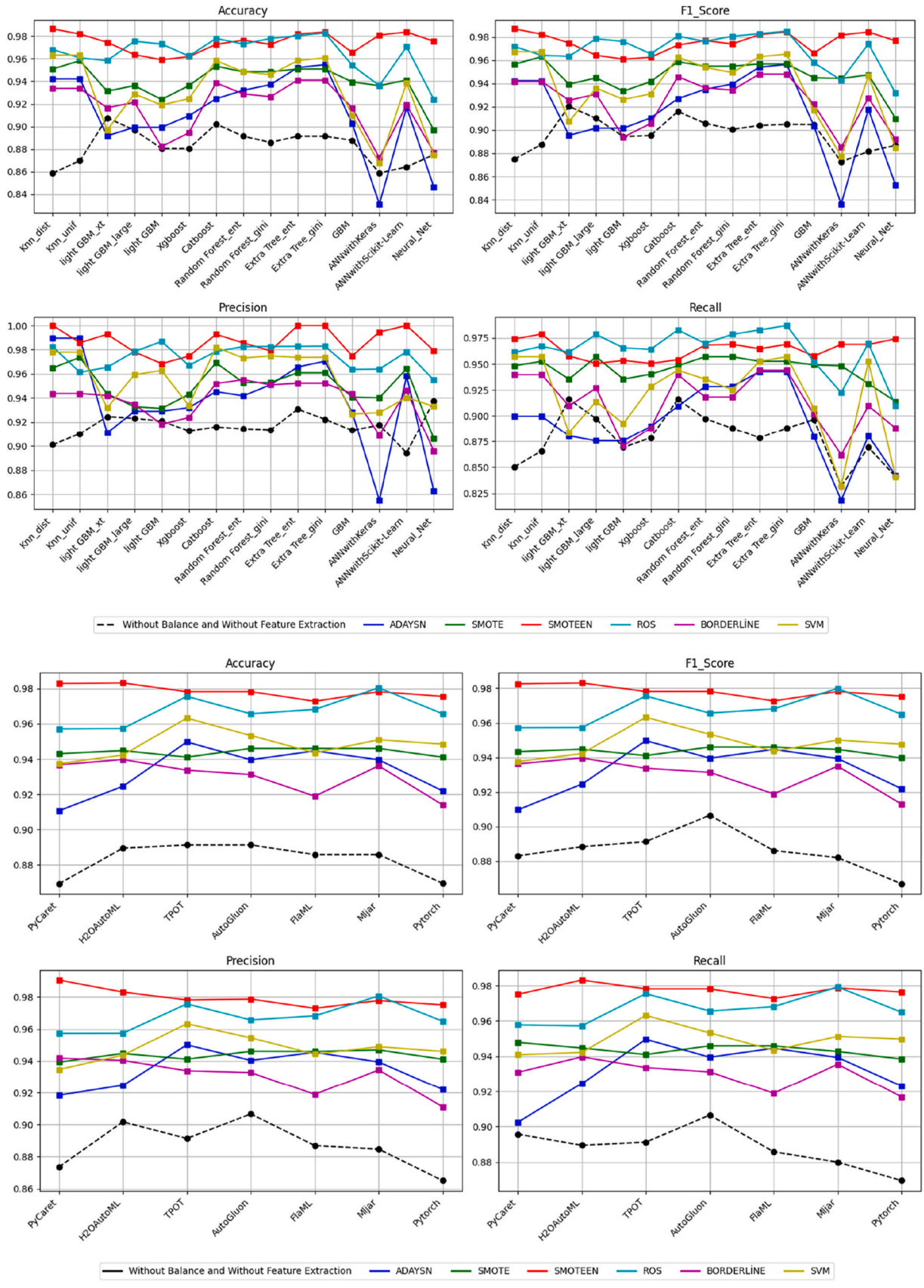

Fig. 7. Comparison between the case **S** and the case **B** for ClassicML and AutoML.

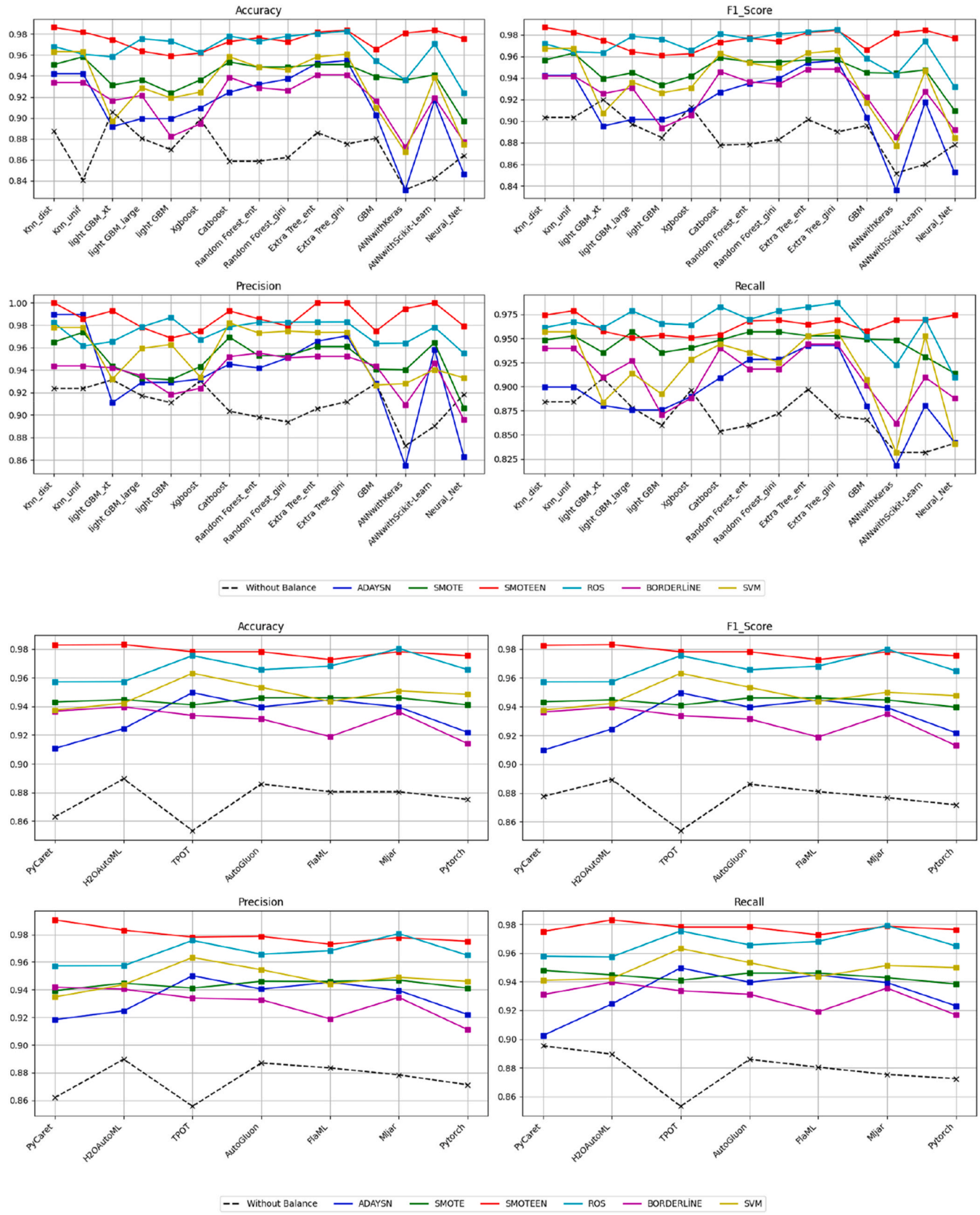

Fig. 8. Comparison between the case **B** and the case **F** for ClassicML and AutoML.

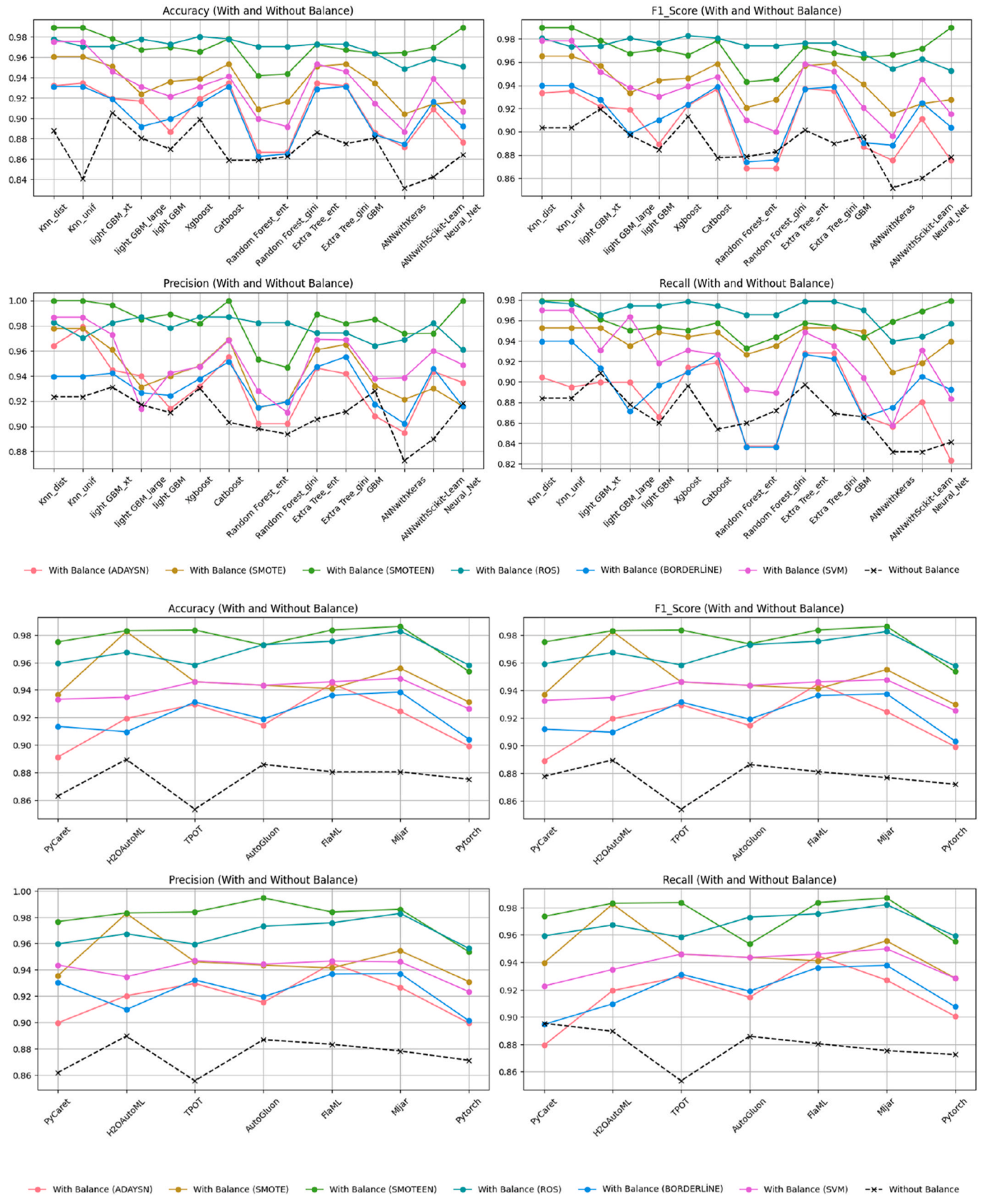

Fig. 9. Comparison between the case **F&B** and the case **F** for ClassicML and AutoML.

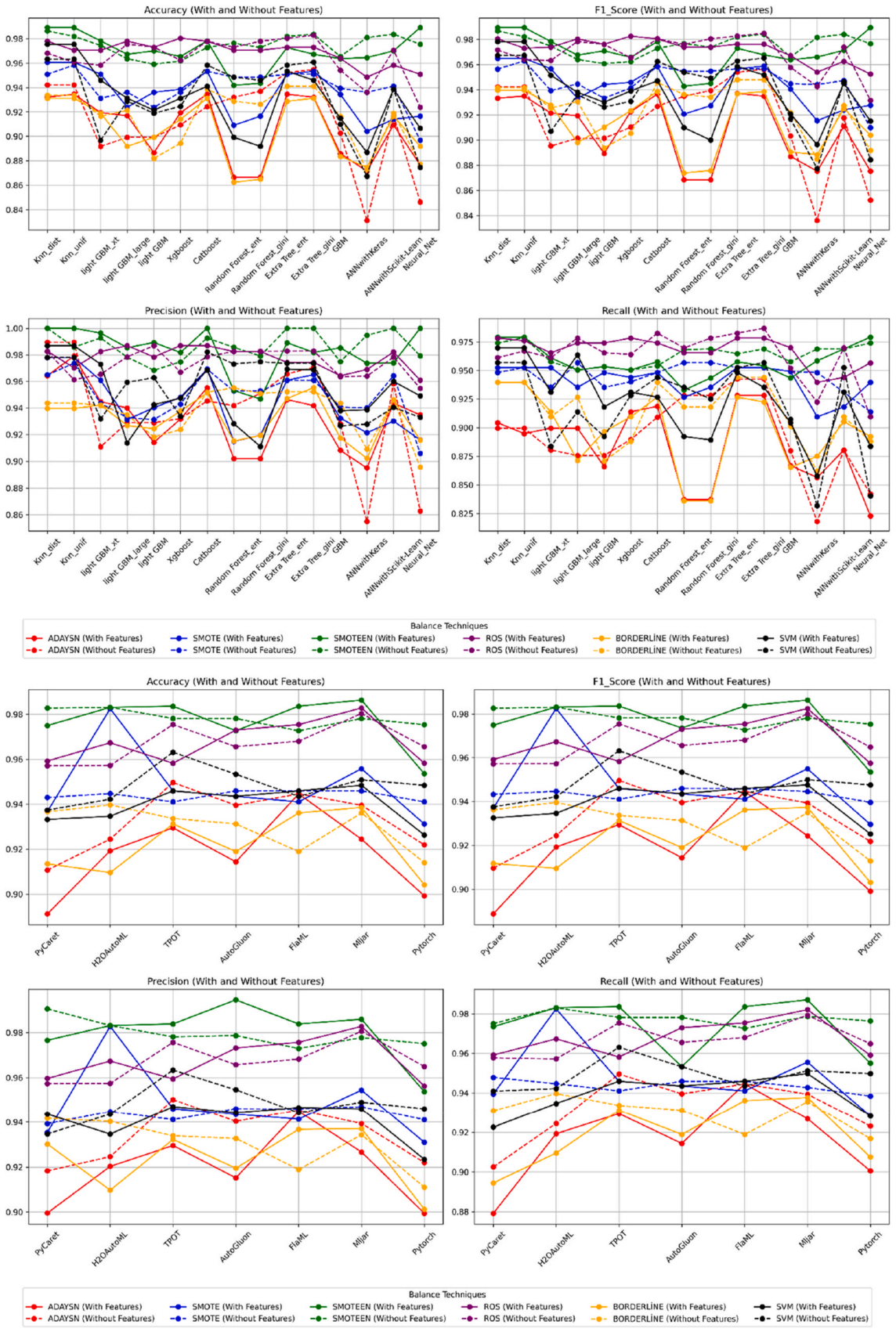

Fig. 10. Comparison between the case F&B and the case B for ClassicML and AutoML.

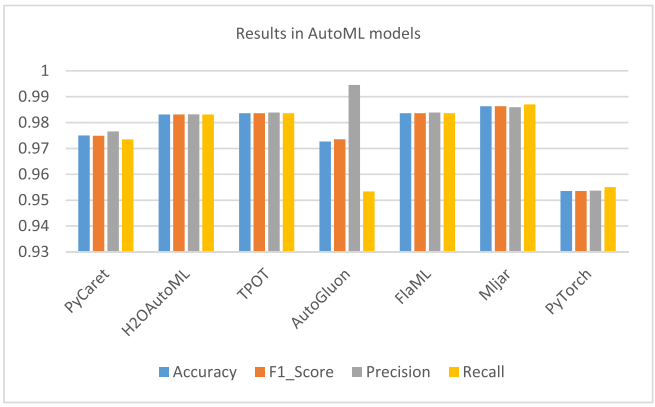

Fig. 11. Model comparison for auto ML

exang_1 are also modeled as strong predictors. In general, the strongest determinants in all models were slope_1 (ST slope after exercise) and exang_1 (chest pain during exercise).

According to the results of SHAP analysis, slope_1, exang_1, and sex_1 stand out as the most determinant variables in all models. Chest pain types (cp_1, cp_3, cp_2) have a moderate effect. In MLjar and NeuralNetTorch, the effect of variables such as blood pressure (trtbps) and ST depression (oldpeak) is more pronounced compared to KNN models.

Since the KNN-dist is a distance-based method, some observations have higher impact values and carry a greater weight on the prediction. In contrast, in the KNN-unif, the influence of variables is more evenly distributed, resulting in a more homogeneous SHAP distribution.

According to the results of SHAP analysis, slope_1, exang_1, and sex_1 stand out as the most determinant variables in all models. The MLjar highlights the most dominant variables while grouping the other variables to reflect their total influence. This approach allows for a more

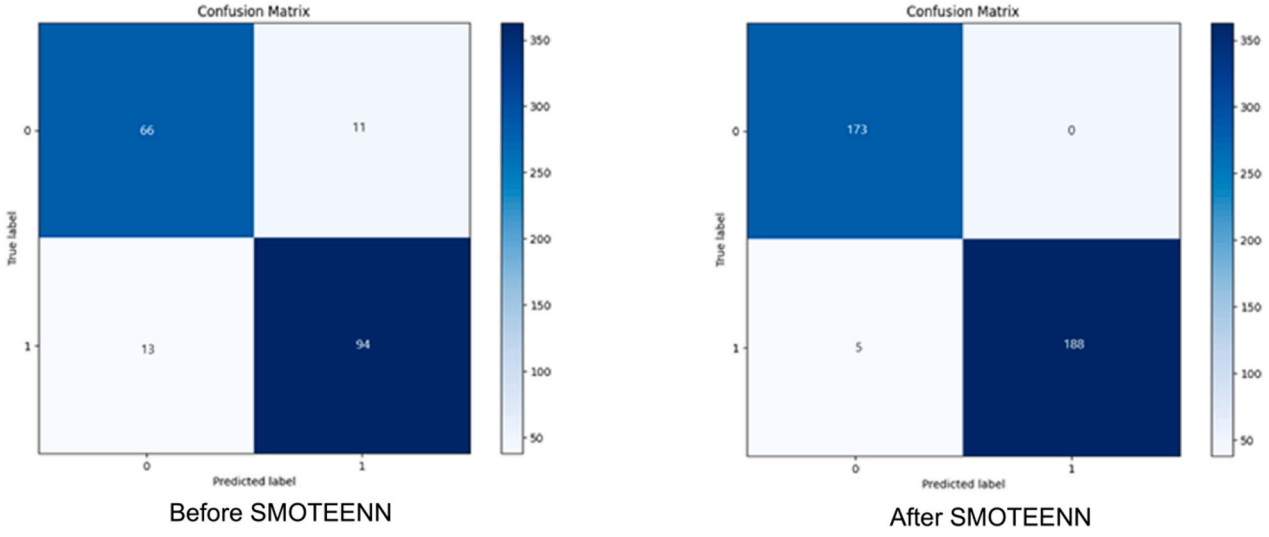

Fig. 12. Confusion matrices for MLjar: Classification performance of the AutoML model.

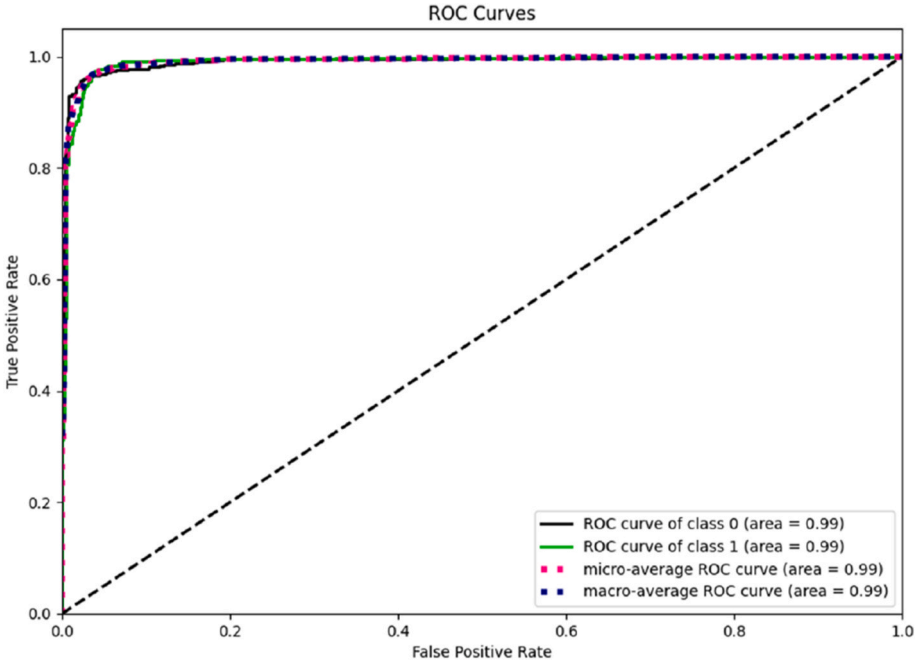

Fig. 13. ROC curves for Mljar: Evaluating classification performance.

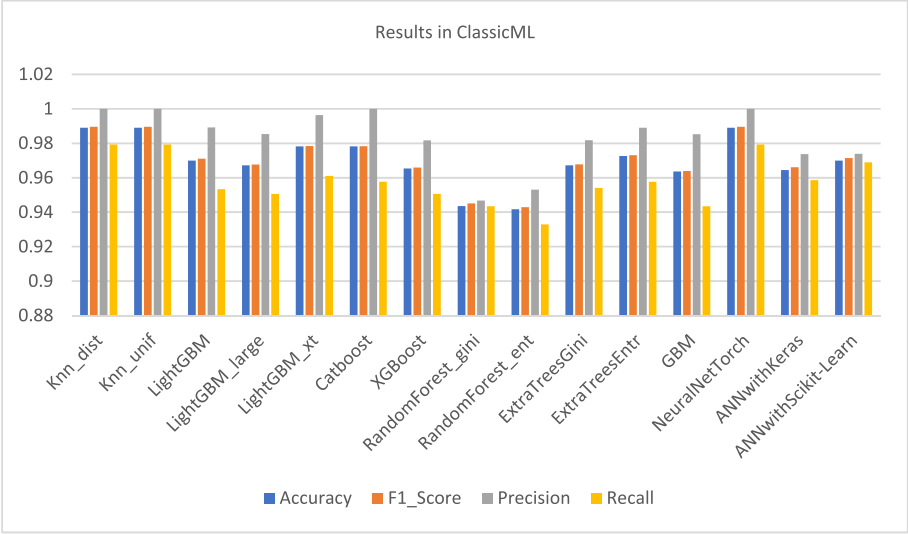

Fig. 14. Model comparison for ClassicML.

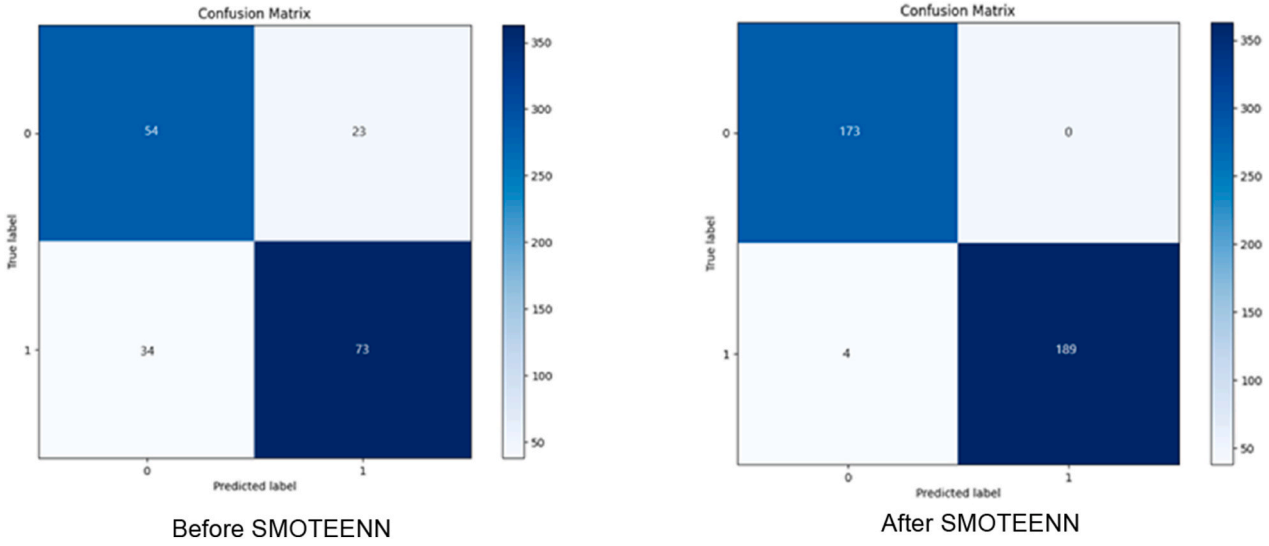

Fig. 15. Confusion matrices for KNN: Classification performance of the ClassicML model.

minimal and more precise model. The NeuralNetTorch evaluates the most extensive set of variables; however, the most important variables largely overlap with the other models. This difference is because the model considers more polynomial and derived variables.

Thus, we see the impact of the variables on the models in detail and have the opportunity to analyze how the weights given to the variables by different models shape the forecasting processes.

Another important approach we take in this study is to compare model performances using statistical methods. Table 5 for the comparison using the one-way ANOVA test shows that the performance of the models is similar and stable, i.e., they give consistent results not only randomly but also across different data splits. Such stability is a desirable property in machine learning applications.

In particular, PyTorch, KNN_dist, TPOT, MLjar, and Neural_Net, which are the models with the highest average performance values, not only achieved high success according to the k-fold cross-validation results but also were found to be strong in terms of stability with low F-statistic and high p-value in terms of ANOVA test (Fig. 19).

In addition, we have created Table 6. In Table 6, a comparative analysis is conducted with respect to studies focused on predicting heart disease, myocardial infarction, or cardiovascular conditions. Singh et al. introduced an innovative methodology for the detection of patients with coronary artery disease (CAD) through the utilization of heart rate variability (HRV) signals. They have shown that this novel methodology enhances classification accuracy, achieving an impressive 99.72 % accuracy employing Support Vector Machine (SVM) techniques [14]. Yilmaz and Yagın examined a predictive classification framework capable of identifying risk factors associated with myocardial infarction. Radial Basis Function (RBF) neural networks and Multilayer Perceptron (MLP) neural networks were employed to juxtapose the outcomes of classification predictions. In accordance with the findings related to myocardial infarction prediction, they attained an accuracy of 0.911 utilizing the MLP neural network and an accuracy of 0.797 with the model derived from the RBF neural network [63].

Bharti et al. accomplished a commendable accuracy of 94.2 % by applying a deep learning methodology for predicting heart disease [17]. Valarmathi and Sheela proposed three Hyperparameter Optimization (HPO) strategies (Grid Search, Random Search, and Genetic Programming (TPOT Classifier)) aimed at enhancing the efficacy of Random Forest and XG Boost classifier models for the detection of heart disease. In their investigation, the TPOT Classifier with Random Forest achieved 97.52 % accuracy [27]. Paladino et al. claimed that AutoML is more

Table 5
ANOVA results for model performance comparison based on F-statistics and P-values.

Models	F-statistic	P-value
PyCaret	0.0325	0.9920
H2O	0.7661	0.5296
TPOT	0.0068	0.9992
FlaML	0.0003	1
AutoGluon	0.0010	1
MLjar	0.0025	0.9998
PyTorch	0.0001	1
Knn_dist	0.0001	1
Knn_unif	0.0007	1
light GBM_xt	0.0003	1
light GBM_large	0.0008	1
light GBM	0.0032	0.9997
Xgboost	0.0023	0.9998
Catboost	0.0004	1
Random Forest_ent	0.0045	0.9996
Random Forest_gini	0.0025	0.9998
Extra Tree_ent	0.0013	0.9999
Extra Tree_gini	0.0006	1
GBM	0.0009	1
ANNwithKeras	0.0677	0.9767
ANNwithScikit-Learn	0.7197	0.5468
Neural_Net	0.0006	1

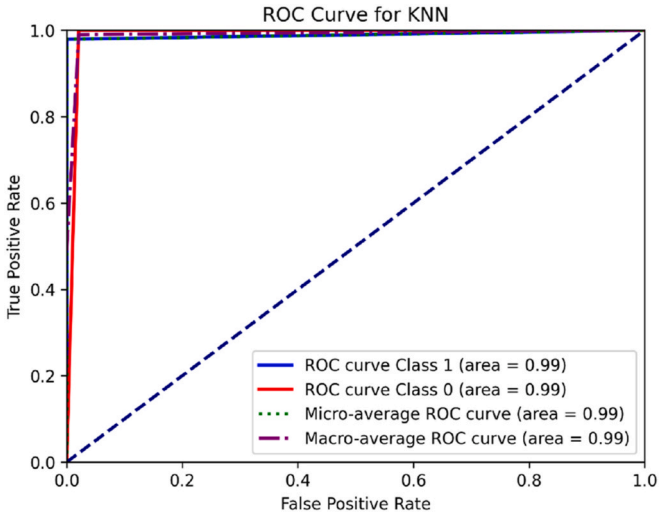

Fig. 16. ROC curves for KNN: Evaluating classification performance.

successful than Classic ML in detecting cardiovascular diseases, and AutoGluon achieved 84.78 % accuracy. They also demonstrated the ease of use of AutoML and stated that it should be implemented on web platforms for early detection [19].

Afanasieva et al. compared several AutoMLs for heart disease prediction and obtained the best value of 94.2 % accuracy with S2_WeightedEnsemble_L2. Barfungpa et al. devised an intelligent heart disease prediction system utilizing a hybrid deep, dense architecture, resulting in an accuracy of 99.53 % [18].

This investigation introduces a distinctive methodology by evaluating the efficacy of traditional machine learning (ClassicML) models in conjunction with automated machine learning (AutoML) frameworks across various iterations that incorporate feature engineering and balancing methodologies. Furthermore, it elucidates which machine learning paradigm demonstrates superior performance in the creation and execution of patient-centric systems designed to forecast cardiovascular ailments.

6. Conclusion

Data science constitutes a comprehensive interdisciplinary domain that encompasses a multitude of fields, many disciplines such as mathematics, statistics, and software. It solves complex problems and transforms data into useful and meaningful information. It represents a framework capable of making predictions based on datasets while simultaneously learning the inherent structure and functionality of the data.

In numerous sectors, including healthcare, the roles of data science and data visualization are increasingly becoming indispensable. This field enables doctors to analyze medical data in-depth, providing important insights into the early diagnosis and treatment of diseases. This process aims to make medical decisions based on fewer determinants but with higher accuracy [64]. Within this framework, the proficient application of data science and visualization methodologies is crucial for the advancement of clinical decision support systems and the enhancement of patient care [65].

The timely identification of cardiovascular disease, a prevalent contributor to mortality on a global scale, is an area of active research aimed at developing robust and dependable medical decision support systems. This research draws conclusions that assist doctors in determining the optimal strategies for predicting cardiovascular disease. Given the prominence of heart disease as a leading cause of mortality globally, it is imperative to formulate predictive models employing appropriate machine learning methodologies.

In this paper, the dataset, which initially had 918 observations, is analyzed under four different conditions. In the scenarios where data balancing was used, six different balancing methods were applied. Average Derivative (AD)and polynomial feature extraction approaches were used for feature engineering. In this study, a comprehensive analysis was performed using seven different AutoML tools and fifteen classical machine learning models to determine the most appropriate approach and the most accurate machine learning technique for heart

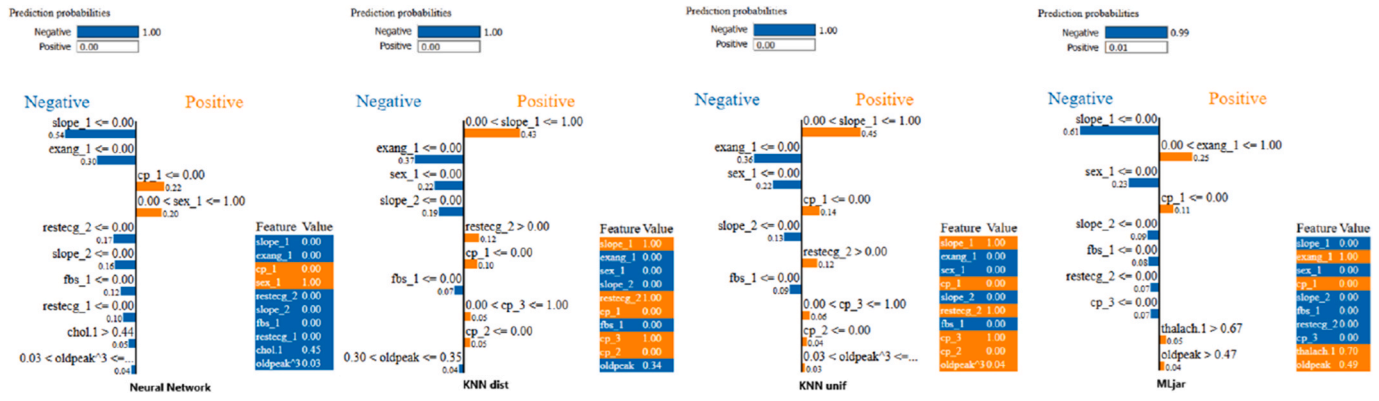

Fig. 17. LIME analysis results for NeuralNetTorch, KNN dist-unif, and MLjar.

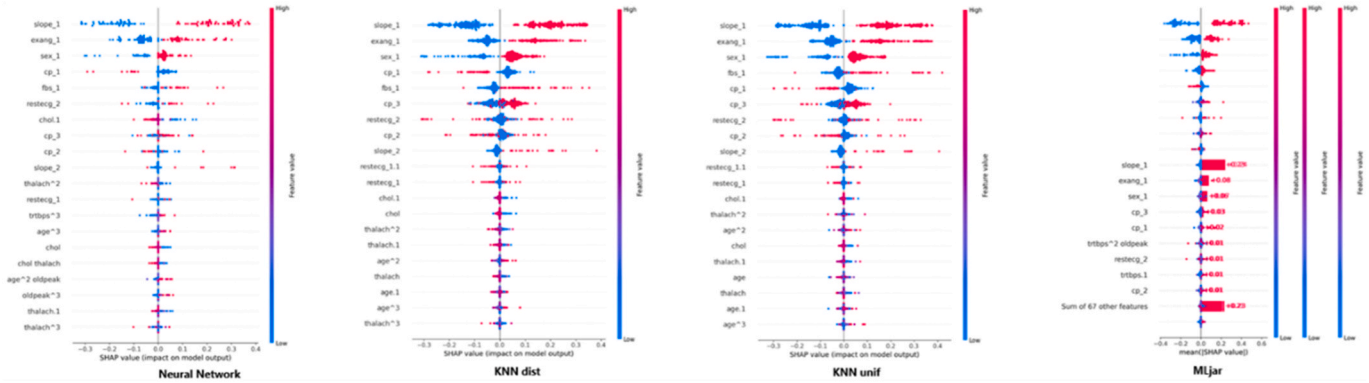

Fig. 18. SHAP analysis results for NeuralNetTorch, KNN dist-unif and MLjar.

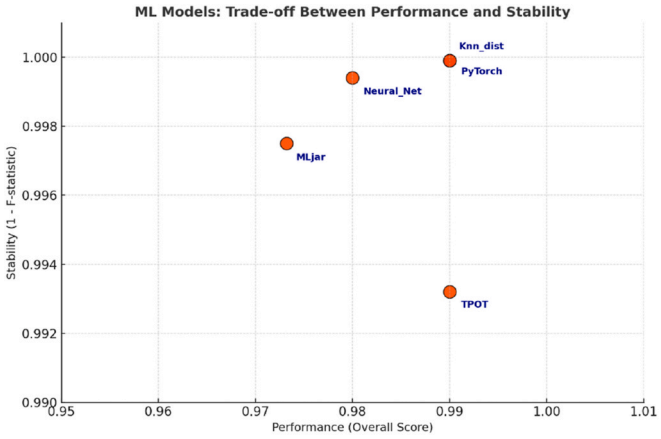

Fig. 19. Statistical results for NeuralNetTorch, KNN dist, MLjar, TPOT, and PyTorch.

disease prediction. The obtained results were comparatively analyzed using four different evaluation criteria.

The analyses show that, in contrast to feature extraction techniques, data balancing methods, in general, significantly reduce the likelihood of models missing diseases. In particular, ROS and SMOTEENN methods provide a significant improvement in model performance by enabling better recognition of low-frequency classes. In some traditional machine learning models, balancing techniques such as ADASYN and Borderline SMOTE used with feature extraction do not provide the expected performance improvement. On the contrary, these techniques lead to a decrease in recall (the number of true positives) while not significantly changing the accuracy values.

In scenarios where only feature extraction is used, model performance is generally negatively affected, but when feature extraction is combined with data balancing techniques, performance varies depending on the balancing method chosen. That is, in some models, feature extraction in combination with data balancing techniques provides significant performance improvements. In contrast, in others, it has a negative impact on the model's ability to detect positive classes.

In ClassicML models, the highest accuracy value is obtained when balancing techniques and feature extraction are used together (Fig. 15 and Fig. 16). On the other hand, the best recall values were obtained when trained using only balancing techniques. In AutoML models, the highest accuracy values are obtained when balancing techniques and feature extraction are used together (Fig. 12 and Fig. 13).

In this context, when we evaluate the results of the scenario where we obtained the highest accuracy and recall values in general:

The analysis results showed that among AutoML tools, MLjar and TPOT stand out with high accuracy, F1 scores, precision, and recall values and provide reliable results. Among ClassicML models, Knn-unif, Knn-dist, and NeuralNetTorch performed exceptionally well, achieving the highest values in all metrics. In addition to the correlation analysis performed on the raw data, SHAP and LIME analyses were performed on the AutoML (MLjar) and ClassicML models (Knn-unif, Knn-dist, and NeuralNetTorch) with the highest accuracy rates, and the effects of the variables used after normalization, data balancing and feature extraction on the models were observed.

The findings of this study show that the performance of AutoML tools is not necessarily superior to classical methodologies and that, in appropriate scenarios, models developed with traditional methods can provide better results. As stated in Paladino et al. AutoML models are feasible and give very good results. But ClassicML models give better

Table 6
Performance comparison.

Author	Predict	Method	Metric	Result
Singh et al.: (2018)	coronary artery disease	MSWP transform	Accuracy	99.72 % (SVM)
Yılmaz and Yağın (2021)	Heart Attack Risk	ClassicML	Accuracy	0.911 (MLP) 0.797 (RBF)
Bharti et al. (2021)	Heart Disease	DL model Feature Selection	Accuracy	94.2 % (DL)
Valarmathi and Sheela (2021)	Heart Disease	AutoML	Accuracy	97.2 % (TPOT)
Paladino et al. (2023)	Heart Disease	Datapreprocessing ClassicML AutoML	Accuracy	84.78 % (AutoGluon)
Afanasieva et al. (2023)	Cardiovascular Disease	AutoML Datapreprocessing	Accuracy	92.31 % (S2_WeightedEnsemble_L2)
Barfungpa et al. (2023)	Heart Disease	Hybrid Deep Dense Aquila network	Accuracy	99.53 % (AOA)
Küçükmanisa and Kilimci (2024)	Heart Disease	ClassicML	Accuracy	90.1 % (Artificial neural networks)
This Manuscript	Heart Disease	Datapreprocessing, Feature extraction, ClassicML, AutoML	Accuracy	98.63 % (MLjar) 98.36 % (TPOT) 98.90 % (Knn_unif-dist) 98.90 % (NeuralNetTorch)

results.

Overall, it is observed that the models trained without feature engineering perform slightly better in all metrics. However, it should be noted that the highest metric values in both ClassicML and AutoML methods are obtained with feature-engineered models. These results suggest that feature extraction can perform best when combined with appropriate data balancing techniques and the right machine-learning model.

To evaluate the performance of the models in different variations and to make comparisons between scenarios as a result of these evaluations, in addition to the raw data set, different scenarios were created in which feature engineering and balancing techniques were applied. Thus, a comprehensive study was carried out in which all variations were tested, and the best results among these variations were evaluated. Furthermore, the performance values and stability of all the considered models were statistically compared with a one-way ANOVA test, and it was concluded that the performance of the models with the best metric values was similar and stable. This study differs from previous studies by clearly demonstrating the effects of feature engineering and balancing techniques (calculated separately for six different balancing methods) on model performance.

Furthermore, this study not only focuses on AutoML or ClassicML models but also provides a comprehensive comparison by evaluating both modeling approaches together. In addition to the performance analysis of the machine learning methods themselves, the comparison of AutoML and ClassicML models with each other is another important element that distinguishes this study from similar studies in the literature. Moreover, comparing the scenarios here is critical to determine whether AutoML can be used in clinical applications without additional review or manual tuning. In this context, the study provides a significant contribution that can strengthen the decision support processes of doctors and researchers in the prediction of heart diseases.

Our research offers important implications for using artificial intelligence in clinical applications and healthcare. In particular, the fact that our study contributes to early diagnosis processes helps to identify disease risk factors. It reveals the importance of data-driven approaches in health systems, which increases the practical potential of this research in healthcare. In this respect, integrating artificial intelligence-supported models into clinical decision-making processes can play an important role in patient management and treatment planning by providing more accurate and faster diagnoses in healthcare services.

In subsequent research endeavors, various methodologies for feature extraction (for instance, Clustering-Based Feature Extraction and Principal Component Analysis) alongside advanced deep learning techniques may be employed to enhance both the efficiency and precision of the models. With these methods, by selecting the components that explain the highest variance in the data set, more meaningful and fewer features can be worked with. Thus, the training time of the model can be shortened, saving time while at the same time increasing the chance of avoiding overfitting.

Ethics statement

This study uses publicly available data from the UCI Machine Learning Repository (Heart Disease dataset). The dataset is publicly available, does not contain any personally identifiable information, and therefore ethical approval is not required.

The authors confirm that this manuscript is original, has not been submitted elsewhere, and was prepared solely for submission to Computers in Biology and Medicine. No funding was received for this study.

The authors declare no conflicts of interest related to this study.

AI tools were used only for language editing and text clarity improvement, and the originality or integrity of the research content was not affected.

Declaration of generative AI and AI-enabled technologies in the writing process

While preparing this work, the author used Chatgpt and Scispace for the language structure and paragraphs of the article. After using this tool, the author reviewed and edited the content as needed and took full responsibility for the publication's content.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. Wettschereck, T.G. Dietterich, An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, Mach. Learn. 19 (1) (1995) 5–27.
[2] D. Wettschereck, et al., A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Lazy Learning 11 (1/5) (1997) 273–314.
[3] H. Liu, H. Motoda, Feature Extraction Construction and Selection, Springer, Cham, Switzerland, 1998.
[4] I. Guyon, et al., Feature Extraction: Foundations and Applications, Springer, Cham, Switzerland, 2006.
[5] D. Avendaño-Valencia, et al., TFR-based feature extraction using PCA approaches for discrimination of heart murmurs, in: Proceedings of the 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, Minneapolis MN USA, September 2009, pp. 5665–5668, https://doi.org/10.1109/IEMBS.2009.5333772.
[6] R. Das, I. Turkoglu, A. Sengur, Effective diagnosis of heart disease through neural networks ensembles, Expert Syst. Appl. 36 (4) (2009) 7675–7680, https://doi.org/10.1016/j.eswa.2008.09.013.
[7] K. Srinivas, et al., Analysis of coronary heart disease and prediction of heart attack in coal mining regions using data mining techniques, in: Proceedings of 2010 5th International Conference on Computer Science & Education, IEEE, Hefei, China, August 2010, pp. 1344–1349, https://doi.org/10.1109/ICCSE.2010.5593711.
[8] G. Parthiban, S.K. Srivatsa, Applying machine learning methods in diagnosing heart disease for diabetic patients, Int. J. Appl.Infomation Sys. 3 (7) (2012) 25–30.
[9] T. Santhanam, E.P. Ephzibah, Heart disease classification using PCA and feed forward neural networks. Mining Intelligence and Knowledge Exploration, Springer, Cham, Switzerland, 2013, https://doi.org/10.1007/978-3-319-03844-5_10.
[10] N.R. Ratnasari, et al., Thoracic X-ray features extraction using thresholding-based ROI template and PCA-based features selection for lung TB classification purposes, in: Proceedings of the 2013 3rd International Conference on Instrumentation Communications Information Technology and Biomedical Engineering (ICICIBME), IEEE, Bandung, Indonesia, November 2013, pp. 65–69, https://doi.org/10.1109/ICICI-BME.2013.6698466.
[11] P. Melillo, et al., Classification tree for risk assessment in patients suffering from congestive heart failure via long-term heart rate variability, IEEE J. Biomed. Health. Inf. 17 (3) (2013) 727–733, https://doi.org/10.1109/JBHI.2013.2244902.
[12] S. Kumar, Predicting and diagnosing of heart disease using machine learning algorithms, Int. J. Eng. Comput. Sci. 6 (6) (2017) 2319–7242, https://doi.org/10.18535/ijecs/v6i6.14.
[13] R. Rajagopal, V. Ranganathan, Evaluation of effect of unsupervised dimensionality reduction techniques on automated arrhythmia classification, Biomed. Signal Process Control 34 (2017) 1–8, https://doi.org/10.1016/j.bspc.2016.12.017.
[14] R.S. Singh, et al., Detection of coronary artery disease by reduced features and extreme learning machine, Med. Pharmacy Rep. 91 (2) (2018) 166–175, https://doi.org/10.15386/cjmed-882.
[15] A.K. Garate-Escamila, et al., Models for heart disease prediction using feature selection and PCA, Inform. Med. Unlocked 19 (2020), https://doi.org/10.1016/j.imu.2020.100330. Article ID 100330.
[16] Y.A. Ghffar, et al., Usefulness of semisupervised machine-learning-based phenogrouping to improve risk assessment for patients undergoing transcatheter aortic valve implantation, Am. J. Cardiol. 136 (2020) 122–130.
[17] R. Bharti, et al., Prediction of heart disease using a combination of machine learning and deep learning, Comput. Intell. Neurosci. (2021), https://doi.org/10.1155/2021/8387680.
[18] T.V. Afanasieva, A.P. Kuzlyakin, A.V. Komolov, Study on the effectiveness of AutoML in detecting cardiovascular disease, arXiv preprint arXiv:2308.09947 (2023), https://doi.org/10.48550/arXiv.2308.09947.
[19] L.M. Paladino, et al., Evaluating the performance of automated machine learning (AutoML) tools for heart disease diagnosis and prediction, AI, 4(4), 1036-1058, https://doi.org/10.3390/ai4040053, 2023.
[20] S.P. Barfungpa, et al., An intelligent heart disease prediction system using hybrid deep dense Aquila network, Biomed. Signal Process Control 84 (2023) 104742.
[21] A. Küçükmanisa, Z.H. Kilimci, Heart disease prediction with machine learning-based approaches, Sakarya University J. Sci. 28 (1) (2024) 101–107.

[22] UCI Machine Learning Repository, Data sets, UCI.edu, https://doi.org/10.24432/C52P4X, 2009.
[23] A. Pathare, et al., Comparison of tabular synthetic data generation techniques using propensity and cluster log metric, Int. J. Information Manag. Data Insights 3 (2023) 100177, https://doi.org/10.1016/j.jjimei.2023.100177.
[24] R.R. Sarra, et al., Enhanced heart disease prediction based on machine learning and χ2 statistical optimal feature selection model, Designs 6 (5) (2022) 87, https://doi.org/10.3390/designs6050087.
[25] M. Marimuthi, et al., A review on heart disease prediction using machine learning and data analytics approach, Int. J. Comput. Applications 181 (2018) 20–25.
[26] U.R. Pol, T.U. Sawant, AutoML: building a classification model with PyCaret, Ymer 20 (2021) 547–552.
[27] R. Valarmathi, T. Sheela, Heart disease prediction using hyperparameter optimization (HPO) tuning, Biomed. Signal Process Control (2021), https://doi.org/10.1016/j.bspc.2021.103033.
[28] M. Padmanabhan, et al., Physician-friendly machine learning: a case study with cardiovascular disease risk prediction, J. Clin. Med. 8 (2019) 1050, https://doi.org/10.3390/jcm8071050.
[29] A. Hazra, S.K. Mandal, A. Gupta, A. Mukherjee, A. Mukherjee, Heart disease diagnosis and prediction using machine learning and data mining techniques: a review, Adv. Comput. Sci. Technol. 10 (7) (2017) 2137–2159.
[30] Y. Khan, et al., Machine learning techniques for heart disease datasets: a survey, in: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, 2019, pp. 27–35, https://doi.org/10.1145/3318299.3318343.
[31] R. El-Bialy, et al., Feature analysis of coronary artery heart disease data sets, Procedia Comput. Sci. 65 (2015) 459–468, https://doi.org/10.1016/j.procs.2015.09.132.
[32] I. Ahmed, A Study of Heart Disease Diagnosis Using Machine Learning and Data Mining, 2022.
[33] S. García, et al., Data Preprocessing in Data Mining, Springer International Publishing, 2015.
[34] N.V. Chawla, K.W. Bowyer, L.O. Hall, Kegelmeyer W.P. Smote, Synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357, https://doi.org/10.1613/jair.953.
[35] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 3 (1972) 408–421.
[36] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: IEEE International Joint Conference on Neural Networks (IJCNN), 2008, pp. 1322–1328.
[37] H. Han, W. Wang, B. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. International Conference on Intelligent Computing, 2005, pp. 878–887.
[38] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 20–29.
[39] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, vol. 10, Springer, 2018.
[40] J. Friedman, The Elements of Statistical Learning: Data Mining Inference and Prediction, Springer, 2009.
[41] T. Hastie, et al., The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, 2009.
[42] A.C. Müller, S. Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists, O'Reilly Media, 2016.
[43] S. Mangalathu, S.H. Hwang, J.S. Jeon, Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach, Eng. Struct. 219 (2020) 110927.
[44] M.R. Zafar, N. Khan, Deterministic local interpretable model-agnostic explanations for stable explainability, Machine Learning and Knowledge Extraction 3 (3) (2021) 525–541.
[45] A. Natekin, A. Knoll, Gradient Boosting Machines: A Tutorial, Frontiers in Neurorobotics, 2013, https://doi.org/10.3389/fnbot.2013.00021.
[46] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A.J. Smola, AutoGluon-Tabular: robust and accurate AutoML for structured data, arXiv preprint arXiv:2003.06505, (2020).
[47] C. Wang, Q. Wu, M. Weimer, E. Zhu, Flaml: a fast and lightweight AutoML library, in: Proceedings of Machine Learning and Systems 3, 2021, pp. 434–447.
[48] E. LeDell, S. Poirier, H2o AutoML: scalable automatic machine learning, in: Proceedings of the AutoML Workshop at ICML, vol. 2020, ICML, San Diego, CA, USA, 2020.
[49] R.S. Olson, J.H. Moore, TPOT: a tree-based pipeline optimization tool for automating machine learning, in: Workshop on Automatic Machine Learning, PMLR, 2016, pp. 66–74.
[50] A. Truong, A. Walters, J. Goodsitt, K. Hines, C.B. Bruss, R. Farivar, Towards automated machine learning: evaluation and comparison of AutoML approaches and tools. Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2019, pp. 1471–1479.
[51] L. Zimmer, M. Lindauer, F. Hutter, Auto-PyTorch: multi-fidelity metalearning for efficient and robust AutoDL, IEEE Trans. Pattern Anal. Mach. Intell. 43 (9) (2021) 3079–3090.
[52] M.T. Garcia-Ordas, et al., Heart disease risk prediction using deep learning techniques with feature augmentation, Multimed. Tool. Appl. 82 (20) (2023) 31759–31773.
[53] K. Guolin, et al., Lightgbm: a highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017).
[54] L. Prokhorenkova, et al., CatBoost: unbiased boosting with categorical features, Adv. Neural Inf. Process. Syst. 31 (2018).
[55] T. Chen, Guestrin C. Xgboost, A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[56] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[57] P. Geurts, et al., Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[58] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media Inc., 2022.
[59] D. Zhang, et al., Integrating feature selection and feature extraction methods with deep learning to predict clinical outcome of breast cancer, IEEE Access 6 (2018) 28936–28944, https://doi.org/10.1109/ACCESS.2018.2837654.
[60] P. Drotár, J. Gazda, Z. Smékal, An experimental comparison of feature selection methods on two-class biomedical datasets, Comput. Biol. Med. 66 (2015) 1–10, https://doi.org/10.1016/j.compbiomed.2015.08.010.
[61] A. Field, Discovering Statistics Using IBM SPSS Statistics, fourth ed., SAGE Publications, 2013.
[62] J. VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media Inc., 2016.
[63] R. Yılmaz, F.H. Yağın, A comparative study for the prediction of heart attack risk and associated factors using MLP and RBF neural networks, J. Cognitive Sys. 6 (2) (2021) 51–54.
[64] J. Smith, A. Doe, P. Roe, Data science in healthcare: a review, J. Med.Informatics. 14 (3) (2021) 123–134.
[65] H. Lee, J. Park, S. Kim, Clinical decision support systems: the role of data science, J. Clin. Comput. 29 (1) (2022) 45–58.

Serbun Ufuk Değer is working in Kastamonu University Kastamonu Vocational School Computer Programming Program (2009–2024). After completing his doctorate in 2018, he has been teaching courses on differential equations, higher mathematics, cryptology, machine learning, artificial intelligence and data science at this institution. In addition to his various articles on stability published in international journals, he also serves as a referee in international journals such as Cogent Mathematics and statistics, Fundamental Journal of Mathematics and Applications, Journal of Mathematical Sciences, Communications Faculty of Sciences University of Ankara Series A1 Mathematics and Statistics and Modeling.