Analysis of Parkinson's Disease Gait using Computational Intelligence

Omid Mohamad Beigi

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science

Department of Computer Science
Faculty of Mathematics and Science
Brock University
St. Catharines, Ontario

Abstract

Millions of individuals throughout the world are living with Parkinson's disease (PD), a neurodegenerative condition whose symptoms are difficult to differentiate from those of other disorders. Freezing of gait (FOG) is one of the signs of Parkinson's disease that have been utilized as the main diagnostic factor. Bradykinesia, tremors, depression, hallucinations, cognitive impairment, and falls are all common symptoms of Parkinson's disease (PD). This research uses a dataset that captures data on individuals with PD who suffer from freezing of gait. This dataset includes data for medication in both the *"On"* and *"Off"* stages (denoting whether patients have taken their medicines or not). The dataset is comprised of four separate experiments, which are referred to as Voluntary Stop, Timed Up and Go (TUG), Simple Motor Task, and Dual Motor and Cognitive Task. Each of these tests has been carried out over a total of three separate attempts (trials) to verify that they are both reliable and accurate. The dataset was used for four significant challenges. The first challenge is to differentiate between people with Parkinson's disease and healthy volunteers, and the second task is to evaluate effectiveness of medicines on the patients. The third task is to detect episodes of FOG in each individual, and the last task is to predict the FOG episode at the time of occurrence. For the last task, the author proposed a new framework to make real-time predictions for detecting FOG, in which the results demonstrated the effectiveness of the approach. It is worth mentioning that techniques from many classifiers have been combined in order to reduce the likelihood of being biased toward a single approach. Multilayer Perceptron, K-Nearest Neighbors, Random Forest, and Decision Tree Classifier all produced the best results when applied to the first three tasks with an accuracy of more than 90% amongst the classifiers that were investigated.

Acknowledgement

Carrying out the requisite work and prevailing over one of the most arduous challenges I've faced in computer science would not have been possible without the love and support from my supervisor, collaborators, friends and family who have selflessly supported me.

Before anything, I owe a great debt of gratitude to my supervisor, Sheridan, who saw my promise as a researcher and encouraged me to dive in. She has always been there for me, offering words of support, advice, and compassion while I worked to keep my mind on the task at hand. She is a brilliant mind and does an excellent job of maintaining the positive atmosphere that has made Brock's Computer Science department so well-known and regarded. There are no words that can express my gratitude for all the weekends and lunch breaks that you had to compromise for me.

I would also like to thank our collaborators from the Federal University of Uberlândia, Brazil. Lígia Reis Nóbrega and her supervisor, Dr. Adriano de Oliveira Andrade have helped us with various experiments and building the dataset that this research would not have been in its current state without.

Above all, I shall thank my family, particularly my father and brother who went above and beyond to make me the person I am today. Thank you for your love and support during these years.

The current research was carried out with the assistance of the National Council for the Development of Scientific and Technological Research (CNPq), the Coordination for the Improvement of Higher Education Personnel (CAPES – Program CAPES/DFATD-88887.159028/2017-00, 88887.343650/2019-00, Call no 34/2017, Program CAPES/COFECUB-88881.370894/2019-01), and the Foundation for Research Support of the State of Minas Gerais. It has been determined that A. O. Andrade is a fellow of the CNPq in Brazil (304818/2018-6). Additionally, this work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Globally, more than 8 million people are suffering from the neurodegenerative brain disorder known as Parkinson's Disease (PD), in their daily activities. According to [15], this number is forecast to increase drastically by the year 2030. Stiffness, bradykinesia, tremors, instability, tremors, depression, hallucinations and abnormalities in gait, including freezing of gait, are common signs of Parkinson's disease (PD), according to [43, 48]. Being diagnosed with Parkinson's disease, patients often take shorter steps at a more sedate pace. This also leads to a lower stride amplitude which results in a slower walk, compared to healthy people [28].

There has been a growing concern towards the impairment of gait in this disease due to its impact on the patients' quality of life [28]. One of the symptoms of PD, known as Freezing of Gait (FOG), prevents patients from moving or to start walking. According to [74] and [45], FOG debilitates the ability of the patient to stand up which is induced by several factors and that adversely affects elevation and rises with falls, which may diminish an individual's independence. As an example, the absence of instruments that are able to objectively evaluate this episodic gait condition makes treatment follow-up more difficult [14]. For this purpose, devices that provide more complete data, to be used in parallel with the clinical evaluation, and that provide quantitative measures, are of the utmost importance. These devices serve as a form of patient-centered monitoring, increasing information about the individual as well as the progression of the disease. Evidently, using AI has advantages such as higher accuracy, fewer human errors, faster and more reliable results, and fewer human resources (which is more cost efficient).

In this study, the construction of the dataset built by our collaborators is elaborated. This dataset is created by taking advantage of inertial sensors. These sensors were used to collect PD patients' data in various experiments, such as performing

FOG triggering tasks before and after taking their medicines. Some examples of these tasks include narrow passage, reaching target, gait initiation, left and right 180° and 360° turn around obstacles, and 180° turn in place. It is essential to carry out the experiment both after and before taking their medicine, since FOG is less likely to happen after taking medicine [53] and freezing conditions [52].

We examine a range of classifiers by applying them to four different challenges using this newly acquired dataset. The first step is to distinguish between those who have Parkinson's disease and those who do not. The second step is to differentiate patients who have PD and who experience FOG into two groups: those who are in the OFF medication state and those who are in the ON medication state. Within this step, the effectiveness of medicine in each experiment is assessed. Finding FOG episodes is yet another challenging task which is done in the third task. In this specific task, patients who experienced FOG during the experiments were chosen and further processes were applied to them. Finally, predicting the FOG episode in a real-time manner concludes the fourth challenge. For this matter, the author proposed a new framework aligned with the data records received from the sensors, and further analysis was performed.

The remainder of the thesis is organized as follows: Chapter 2 introduces some of the previous related works. Following, a thorough explanation of the dataset is discussed in Chapter 3. Afterwards, in Chapter 4, background information regarding the methodologies used in this work is explained. Then, the experiments performed in this study are elaborated upon, as well as the new framework, in Chapter 5. Following, the results and further analysis are discussed in Chapter 6. Finally, the thesis is concluded in Chapter 7.

Chapter 2

Literature Review

Parkinson's disease (PD) has long been diagnosed by either difficulties in speech or problems with gait, such as freezing of gait (FOG). The features of the gait affected with Parkinsonism can be identified and captured by special sensors attached to different parts of the patient. In this thesis, such a dataset is used and these features are then processed with various algorithms in order to detect whether an individual has PD or not. These techniques are mainly categorized into three groups, namely, traditional algorithms, evolutionary algorithms, and artificial neural networks. It is worth mentioning that the main focus of this thesis is on two of the above categories (traditional algorithms and artificial neural networks).

2.1 General PD background

Impairment of nerve cells in the region of the brain known as the *substantia nigra* causes a degenerative condition, called Parkinson's Disease. The loss of these nerve cells reduces the capacity to create the crucial neurotransmitter dopamine [1]. At this time, Parkinson's disease cannot be treated, but only the symptoms can be managed [13]. In the mentioned study, it was observed that when the medication was taken as prescribed and consistently, the symptom variations were reduced. The primary drug used to substitute dopamine in the brain is called *Levodopa*. On the other hand, as the condition advances, Levodopa loses its ability to act at its full efficacy, which results in a rise in motor symptoms and fluctuations [13].

Patients experience a slower walking pace even while they are in the ON phase of medication (as they have taken the medicine less than a few hours earlier), which is when the drug is acting. In [42], the authors observed that Levodopa was effective on the stride length and peak gait pace. However, temporal variables such as stance phases, duration of swing, and cadence were not responsive to it [27].

2.2 Traditional Algorithms

Traditional algorithms have been utilized by researchers for decades since they provide benefits including quick implementation, giving reliable results with the least amount of data while needing the fewest features. In [5], researchers took advantage of SVM to evaluate patients' statistical and kinematic gait features in order to assess the severity of PD. In the mentioned study, two tasks of PD classification presented in an automatic and non-invasive manner. The first task classified PD diagnosis based on their gait behavior. The second task rated the severity of the disease. The data in this paper was obtained by asking volunteers to wear 16 sensors. The collected data from these sensors consisted of time-series based records from different parts of the patients' feet. A correlation-based feature selection technique was used to gather the most relevant feature set for the classification model. To begin, the data retrieved from the VGRF sensor was statistically analyzed to distinguish between healthy controls and PD patients. The subject's weight has an impact on foot plantar pressure and classifier output in statistical analysis. To prevent the biased detection, a kinematic analysis was applied. Unlike some previous ML-based systems that conduct binary classification, they intend to address the multi-class classification problem of Parkinson's disease utilizing a discriminative feature set derived from spatiotemporal characteristics in this study. In the mentioned study, supervised classifiers such as support vector machine, decision tree, ensemble classifier, and Naïve Bayes were used. Furthermore, the 5-fold cross validation approach is used to avoid data overfitting and improve classification accuracy. The region of convergence (ROC) curve and the confusion matrix were used to validate the classifier model's efficacy.

Following, Sachin Shetty *et al.* In [71], the authors applied an SVM using Gaussian RBF kernel to detect PD patients from other neurodegenerative disorders. In the past, there have been attempts made to solely classify people with Parkinson's disease from healthy subjects. However, the focus of the mentioned study was on the particular gait features that would help detect Parkinson's disease from other neurological disorders (such as amyotrophic lateral sclerosis (ALS) and Huntington's disease) and healthy controls. In this research work, a number of statistical feature vectors were assessed by making use of temporal gait feature records, which were then condensed by the use of a correlation matrix. The seven best feature vectors that were extracted were fed to a Support Vector Machine (SVM) classifier that took advantage of a Gaussian radial as the kernel. The research indicated that the seven selected features for SVM achieved a high overall accuracy of 83.33%, a high detection rate for Parkinson's disease of 75%, and a low false positive rate of 16.67%.

Furthermore, C, the penalty parameter of the error rate, and σ, which defines how far the influence of a single training example reaches, were chosen for this study by employing a grid search strategy. The results showed that C=1000 and σ=10 demonstrated the least amount of classification error. The points were scaled to have a standard deviation of one unit before to the use of the training data. The tolerance level of the Karush Kuhn Tucker(KKT) conditions was set to 1e-3 before beginning the Sequential Minimal Optimization process, were utilized to locate the separating hyperplane. The data was initially utilized for training, which consisted of the first seven samples of each class (ALS, HD, PD, and control), and then it was used for testing, which consisted of the remaining samples. This was done in such a way that half of the samples (n=7) of the data records of PD (having n=15 samples in total), which was used for training, and the remaining data, which was used for testing, comprised n=8 samples.

The authors of [18] applied K-Nearest Neighbor (KNN) to the kinematic features of the gait which led to an 86.19% accuracy. Using the Microsoft Kinect v2 camera, this research investigated linear kinematic gait analysis. This analysis was based on the skeletal positioning data of the joints in the lower body (hips, knees, and ankles). Participants who walked in front of the Kinect camera at one of three distinct speeds provided the quantitative data that was gathered (slow, normal, and fast speed walks). Instead of depending on physicians' observations, the authors acquired data via the cameras (which is more reliable), since this will result in a more detailed description of the kinematics. Mathematical analysis may be used to produce an effective comparison between the different methods by using the quantitative data as the basis. The proposed methods of [18] have been evaluated with respect to a number of different classifiers, such as the Decision Tree (DT), linear and nonlinear Support Vector Machines (SVMs), k-Nearest Neighbor (KNN), and subspace discriminant. These classifiers were utilized in the classification of gait speeds through the utilization of unmodulated and modulated signals, respectively. The first kind of analysis is known as unmodulated velocity signal analysis, while the second type refers to velocity signals that have been modulated by the use of one of the several suggested modulation methods, such as frequency modulation (FM).

In [44], researchers applied various classifiers, namely, logistic regression, gradient boosting, random forest, K-nearest neighbor, and decision tree on the gait features to identify essential tremors for PD. The study used wearable sensors to collect the data, and demonstrated the effectiveness of the methods with a variety of measurement metrics, such as accuracy, recall, precision, and F1 score. Balance and gait characteristics features were gathered from individuals with Parkinson's disease (n = 524) and those with epilepsy (n = 43) during an instrumented stand and walk test for the purpose of this retrospective research. Using F1-scores, the performance of a number of different machine learning methods, such as neural networks, K-Nearest Neighbor, Decision Tree, Support Vector Machine, Random Forests, and Gradient Boosting, was compared with Logistic Regression and Naive Bayes.

Furthermore, authors of [16] could illustrate an accuracy of 85.6% by applying specific features of a hand-writing dataset regarding the motor disorder of neurodegenerative diseases, to an SVM classifier. The PaHaw dataset was created within this study, consisting eight different handwriting tasks from patients, such as Archimedean Spiral. The researchers ascertained that the handwriting motion is comprised of both on-surface movements and the in-air movements. In this research, a digitising tablet was used to analyse the in-air and on-surface kinematic variables that occurred during the handwriting of a phrase by 37 people diagnosed with Parkinson's disease who were taking medication and 38 healthy controls of the same age and gender.

They demonstrated that assessing the in-air/on-surface hand movements led to accurate classifications in 84% and 78% of subjects, respectively, by applying feature selection algorithms and support vector machine learning methods to separate PD patients from healthy controls. To do this, they applied feature selection algorithms and support vector machine learning methods. The combination of the two modalities resulted in a prediction accuracy of 85.61%, which was sufficient for a medically appropriate diagnosis. This improvement of 1% was achieved in comparison to the examination of in-air characteristics alone.

Moreover, in [17] new pressure and movement features of the handwriting dynamics were thoroughly studied. Then, three classifiers, namely, AdaBoost, SVM, and KNN were used on the mentioned features and a total accuracy of 81% was achieved. Other demographic characteristics, such as age, education, work, and birthplace, as well as disease duration, must be used to support the handwriting test in order to provide an effective analysis of PD patients. With the use of sophisticated ML algorithms, the Archimedean spiral based scientific approach of static and dynamic spiral can be regarded as more relevant than this handwriting experiment. In [17], authors offer the PaHaW (Parkinson's disease handwriting database), which is comprised of handwriting samples from patients diagnosed with Parkinson's disease (PD) as well as handwriting samples from healthy controls. The purpose of this study was to demonstrate that kinematic data and pressure features in handwriting may both be utilised to differentiate between PD and other conditions. The database has information from a total of eight distinct handwriting activities that were completed by 37 people diagnosed with Parkinson's disease (PD) and 38 healthy controls. The assignments consist of writing a phrase, drawing an Archimedean spiral, and repeatedly writing orthographically basic syllables and sentences. They studied additional pressure characteristics based on the pressure that was applied on the writing surface in addition to the traditional kinematic features that are connected to the dynamics of handwriting. Three distinct classifiers, namely KNN, ensemble adaBoost classifier, and SVM, were evaluated and contrasted in order to classify Parkinson's disease (PD) patients and healthy participants. The SVM was the highest performing model for predicting PD based on kinematic and pressure aspects of handwriting. It had a classification accuracy of Pacc = 81.3% (sensitivity Psen = 87.4% and specificity Pspe = 80.9%). When assessed independently, pressure characteristics were shown to be useful for Parkinson's disease (PD) diagnosis, producing a Pacc value of 82.5% as opposed to a Pacc value of 75.4% when employing kinematic features. An investigation of the kinematic and pressure elements that occur during handwriting may assist evaluate the more subtle aspects of a person's writing and differentiate between people with Parkinson's disease and healthy controls, according to the findings of several experiments.

Omer Eskidere *et al.* in [19] described applications of machine learning frameworks SVM, MLPNN, LSSVM, and general regression NN to follow the PD development, remotely. A telemonitoring dataset consisting of patients' speech with 26 features was used in this study. The performance of the prediction model was evaluated using MAE, MSE, and correlation coefficients, resulting in the conclusion that LS-SVM is superior at mapping UPDRS with vocal features. It works with non-linear voice features, both classical and non-classical. Feature extraction and normalisation of speech features is a time-consuming operation that necessitates a larger normalisation range and varies in accuracy depending on the signal processing algorithm. This is a conventional method of diagnosing Parkinson's disease, and early signs may not be seen in the patient. As a result, UPDRS mapping based solely on voice features might not result in an appropriate diagnosis. The mentioned methods had two constraints of being more computationally intensive and were prone to overfitting.

In [37], it was proposed that the spiral test be enhanced by employing a digitizing tablet instead of the conventional static spiral test with pen and paper. The author proposed a new dynamic spiral test. The article concluded that digital samples created by computer system Dynamic Spiral Test (DST) drawings paired with Static Spiral Test (SST) drawings may be utilized to construct a general PD telemonitoring, diagnostic, and control system. The tablet had a resolution of 1000 pt/cm, an accuracy of 0.025 cm, and 256 levels of detectable pressure. It was connected to a computer through USB and processed spirals using proprietary software. The theoretical equivalent of spiral expressed linearity in terms of r (theta), polar coordinates were used to turn ideal and PD patient spirals into radius angle transformations. From the perspective of the paper, the construction of a generalizable non-invasive Parkinson's disease diagnostic and monitoring decision support system necessitated the collection of a sufficient number of training samples from Parkinson's disease patients at regular intervals. Therefore, the basis of the research was on self-administered and noninvasive telemonitoring programmes, which have become popular, lately. This application allowed PD patients to gather data at home and transfer it over the internet to a dedicated server.

2.3 Evolutionary Algorithms

Genetic programming (GP) One of the hot topics in classification models. Having a medium-sized dataset, GP performs and generalizes better than most of the traditional algorithms, such as linear regression [34]. The study applied features from pressure sensors to a GP model in order to introduce a descriptive symbolic nonlinear model. In order to do symbolic regression, a particular kind of GP was used for this study. The GP system was exactly the same as the one described in [17], with the exception of the settings for the system parameters. A number of enhancements were introduced into the GP system, the most notable of which were fitness predictors and an acyclic graph representation. This format has been proven to prevent overfitting and bloat, leverages the reliability of results, scales well, and has the potential to employ perhaps relevant subexpressions.

Furthermore, GP was used in [33] to develop models of human gait using data from patients' smartphones. In the mentioned work, GP was used to construct an identity gait fingerprint for two individuals, whose walking data was acquired from the accelerometer in a commercially accessible phone. This walking data was collected from the authors of this paper. Users walked freely without a predetermined routine at a regular, nonuniform speed while the phone was freely put into a pocket. The architecture of this data gathering process was one that corresponds more directly to the uses of such a technology in the actual world. For the purpose of performing symbolic regression, a highly sophisticated GP system with several modular modifications was put into operation. The effectiveness of the system in modelling each dataset with a high level of precision was proven, and it was shown to be resistant to noise. It was also shown that a model may be constructed for an entire subject's dataset using the data from only one phase of the process. In order to determine the extent to which these mathematical models are original, leading models were used to analyze the data of different subjects.

The GP can be combined with various methods to overcome the overfitting. For instance, James Hughes developed a fitness predictor for symbolic regressors to reduce the overfitting in [32].

2.4 Artificial Neural Networks (ANN)

2.4.1 Shallow Neural Networks

The advanced ANN algorithms have demonstrated a significant improvement in many different classifications. Convolutional Neural Networks, for example, are very adept at handling high dimensionality and extracting features from enormous datasets with many features. Additionally, by avoiding complex feature engineering, they have also demonstrated a strong semantic interpretation. In [35] a feedforward multilayer perceptron was used to illustrate how well ANNs work in feature extraction of gait classification.

The authors of [55] have developed a revolutionary artificial intelligence-based method to assist in the early diagnosis of Parkinson's disease. The dysphonic measurements and clinical points for 68 individuals were obtained using the UCI Machine Learning database. Weights obtained from the Multi-Layer Perceptron (MLP) served as the basis for the feature selection. The Lagrange Support Vector Machine (LSVM) classification process was then employed as an input with this condensed set of information. To elaborate more on the classification framework, the weights that were obtained from a Multi-Layer Perceptron (MLP) were used for the feature selection process, and their moduli were utilized to rank the input features according to their relative value in differentiating between healthy and pathological data patterns.

As a result, the original 27 characteristics were narrowed down to 20 carefully chosen diagnostic variables. Following that, this trimmed down feature set was fed into an LSVM so that it could be classified. Therefore, the total performance of this hybrid feature-driven algorithm, known as MLPLSVM, was measured against that of commercially available software as well as classifiers derived from research that were quite comparable. The findings showed that the MLP-LSVM had a classification accuracy of 100% overall and an area under the receiver operating characteristic curve that is equal to 100%. Additionally, the algorithm had a relatively faster convergence, which demonstrated its potential for assisting in the early diagnosis of Parkinson's disease in a clinical setting.

2.4.2 Memory-based Deep Neural Networks

The authors of [58] utilized Deep Reinforcement Learning (DRL) to detect Parkinson's Disease using smartphones. For this classification, they used sensors of each patient's cellphone to collect the data. The notion that Parkinson's patients exhibit distinct anomalies in their gait if they do not take their medicine as prescribed is the basis for this approach. While the patient kept their mobile phone in their pocket, the information was being passively collected by the cell phone. After that, the data preprocessor assisted in isolating the walking cycles that were included inside the biomarker connected to Parkinson's disease.

The integration of neural networks into reinforcement learning makes it possible to automatically abstract and extract high-level features as well as semantic interpretation directly from the input data. This eliminates the need for complex feature engineering or delicate feature hand-crafting and selection for each individual task. Gradients, and by extension the backpropagation algorithm, are essential to the great bulk of the work that is being done by the DRL. The fundamental reason for this is due to the fact that gradients, when accessible, give a powerful learning signal. Due to the fact that these gradients are only approximated in practice, whether by sampling or some other method, it is needed to design algorithms with helpful inductive biases in order for them to be tractable.

Using the state-of-the-art Long Short-Term Memory (LSTM) networks, authors of [51] differentiated both essential tremors and PD. For this matter, they used Leap Motion Controller (LMC) which collected 3D data for classification. The proposed model used 40 subjects, and the tests demonstrated a 90% of accuracy in combined tremor analysis. The medical assessment of Essential Tremor (ET) and Power Spectral Density (PSD) tremors utilizing LMC without distinguishing parameters like frequency and power spectral density was done in this study. In the study, deep neural networks brought several advantages, but one of the most significant is that they get rid of the need of manually collecting features and instead teach themselves features via training. The LMC does not have a constant frame rate, thus the tremor data that it collects should be interpreted as a non-uniform time series. As a result, differentiating between Physical Tremors (PT) and Essential Tremors (ET) is difficult since doing so requires identifying the precise frequency and magnitude parameters of tremors using LMC data. Hence, the use of deep learning, in which the characteristics of ET and PT are learnt via training rather than discovering features such as frequency and power spectral density estimates, is an extremely helpful technique for the differentiation of ET and PT. The research used data obtained from LCM recordings of hand tremors taken in both the posture and resting positions.

2.4.3 Convolutional Neural Networks

In [27], a Convolutional Neural Network (CNN) was used to detect patients from healthy people for a precise diagnosis of PD tremors and other tremors. In this study, accelerometer sensors gathered patient data to measure the right and left-hand tremor data in 3 axes. The raw data was then normalized using a standard scalar function, which scaled the data in a 0 to 1 range using mean and standard deviation. Finally, the model was trained based on the normalized data which produced 92% accuracy. More into the classification model, they proposed a classification model based on a CNN and had seven hidden layers as well as varying filter sizes for the purpose of accurately classifying patients with PSD and healthy control (HC) subjects. A flatten layer reduced the dimensions of the data from three to one which is implemented with the Tensorflow. Following, the dense layer was responsible for the output of the categorization of PSD and HC patients based on the strength of their tremors, which was done in order to detect the PSD patient's danger at an early stage.

The authors of [73] identified MR images of healthy and PD patients using deep neural networks. The CNN architecture AlexNet was used to improve Parkinson's disease diagnosis. The classification method was done after it was retrained using a pre-trained deep network using MR images. With the suggested approach, an accuracy value of 88.9% was reached. In terms of classifying MR images for Parkinson's disease in the mentioned paper, the pre-trained AlexNet model with transfer learning was taken into consideration and modified as necessary. The weights of the convolutional layers of CNN were initialized using the weights of the pre-trained *AlexNet* CNN model that has the same architectural layout. The early layers of CNN included features that are more general, and the latter levels contained features that are more particular to the Parkinson's dataset. High-level characteristics of the Parkinson's data were learned once the final fully linked layers were fine-tuned using MR pictures of patients with Parkinson's disease. This was accomplished by suitably altering the bottom most layer of a neural network that had two output neurons in order to categorize MR images as either HC or PD. In addition, the CNN would need access to a substantial quantity of Parkinson's data in order to generate and update the weights. Therefore, moving the weights from a pre-trained model to a CNN would result in an increased rate of convergence and desired performance. Employing a model that has already been trained helps decrease the amount of memory that is necessary for computation.

Chapter 3

Data

The steps towards the data collection procedure are discussed in this chapter. The acquired dataset is then used for the classification tasks as elaborated on in Chapter 5. It is worth mentioning that the construction and data gathering have been done by our collaborators and not the author of this thesis.

The data collection method that was employed has been shown to be accurate in [40]. In order to prevent any problems that may be caused by the loss of data, every signal that was gathered was visually inspected. This was done to check for discontinuities and undesired patterns.

3.1 Methods

This research was carried out at the Federal University of Uberlandia in Brazil after receiving clearance from the institution's ethics council[1]. Before beginning the experiment, every subject was provided with an in-depth explanation of what would be happening, and they all signed a permission form indicating that they were comfortable taking part in the research.

3.2 Groups

The dataset is comprised of three major groups: the control group (GC), the group of negative freezing of gait patients (GFOG-) who were diagnosed with PD but who did not have FOG episodes, and the Group of positive freezing of gait patients (GFOG+), who were the PD patients that experienced occurrences of freezing were. Each group

[1]CAAE number: 38885720.3.0000.5152

consists of 10 volunteers. Exclusion criteria included the presence of significant visual and auditory impairments in the participants, the presence of other musculoskeletal or neurological disorders, or the use of drugs that have the potential to induce these conditions.

3.3 Technology

The inertial sensors that are used to collect gait and episode information of FOG are coupled to three smartwatches, with a Movement Disorders Monitoring System (NetMD). This system was developed by the Spanish research group CAR-CSIC in order to analyze and remotely and continuously monitor movement disturbances through inertial signals [4].

The NetMD is built on the collaborative efforts of an Android mobile smartphone and wristwatch devices (the Smartwatch3 SWR50 model, manufactured by Sony). Communication between the two is carried out by Bluetooth.

It is feasible to gather data from the internal sensors of smartwatches (accelerometers and gyroscopes) with a frequency of sampling rate of up to 50 Hz by using this method (that is, with a temporal resolution of 20 ms). The system creates a text file that is kept in the Sony Android mobile device. This file contains the values of the inertial signals that are generated by each wristwatch (time in milliseconds, sensor name, battery status, and accelerometer and gyroscope on the x, y, and z axes). It is possible to resample the data at a rate of 100 Hz and a temporal resolution of 10 milliseconds after having performed a time vector interpolation on them.

3.4 Sensor Position

Three wireless inertial sensors were utilized in this dataset, two of which were placed in the pelvis [3] over the ends of the iliac spine and one of which was placed over the fibula [45][42][57]. The peroneal muscle sensor was placed on the side most affected by PD, identified by skilled and experienced healthcare professionals.

The side of the peroneal muscle sensor that was most impacted by PD was determined by knowledgeable and experienced medical practitioners to be the most appropriate placement.

3.5 Tasks

The research compares the performance of the different groups across four different activities. These are: voluntary stop; TUG (timed up and go); simple motor task; dual motor and cognitive task. On average, each individual has 2,500 timesteps (records) per trial in a task. Each record contains the obtained data from all sensors mentioned in Section 3.3.

3.5.1 Voluntary Stop

The individual is required to get up from a chair, walk three meters, and stand inside a square of tape that has been made on the floor for ten seconds before returning along the same path and sitting back down. This job is required so that the signal from the accelerometer and gyroscope that was captured during the voluntary arrest may subsequently be compared with the signal that was obtained at the involuntary halt, which is the FOG. It was carried out in the same setting as Task 2, TUG, which is described in the following paragraphs.

3.5.2 Timed Up and Go (TUG)

The current task is a clinical evaluation that was developed in 1991 [56] and consists of recording the time required for the volunteer to get up from a chair, walk 3 meters, turn (u-turn), return the same way, and sit down again [72]. In other words, the test measures how long it takes the volunteer to stand up, walk, turn, return, and sit down again [72]. It has been utilized in a number of research that have focused on FOG, including [72] [47] [2] [41] [77].

3.5.3 Simple Motor Task

In this motor circuit task, the patient engages in everyday activities like sitting down and getting up from a chair and walking as well as movements that may cause FOG, including going through doors and veering right and left to avoid obstacles. Specifically, the patient sits in a chair, rises from the chair, and walks (cones on the ground). The individual rises from the chair, walks for three meters, and then squeezes through an opening that is 67.5 cm in width. After that, they construct a route in the form of an infinity symbol by walking 1.30 meters and navigating around the two barriers. The physical therapist will determine which side of the obstacle is the most challenging for the volunteer during the clinical examination. The volunteer will then execute a 360-degree turn around the first obstacle, and then do another 360-degree turn around the second obstacle. The last step involves the volunteer completing just one complete circle around to return to the first challenge. The volunteer makes their way back to the chair, which requires them to go back through the confined space, and they sit down.

3.5.4 Dual motor and cognitive task

In this step of the experiment, the patient continues to carry out the simple motor activity while at the same time carrying out a cognitive task. The monitoring of digits is the mental component of the task (DMT). DMT involves giving each participant a number with a single digit as its representation. This volunteer has been given the task of counting out loud (without using their fingers) the number of times that the digit in question is mentioned in the audio. The audio is the same for all of the volunteers, it was transcribed so that the researcher would know the frequency of each digit, and there is a draw of the digit before collecting to ensure that the test is randomized [7]. When the trial is over, the participants report to the researcher the total number of occasions on which they were exposed to the digit [14]. In order to prevent the user's stride from being synchronized with the audio track, the auditory inter-stimulus interval is shown in a randomized fashion and may range anywhere from 100 to 1000 milliseconds. The participants were asked to continue counting the digit even if they completed the circuit before the audio ended [7]. The audio is played for a total of sixty seconds.

An illustration of a sample time series for each of the aforementioned four activities in [49].

3.6 Medicine Status - On/Off

The patient experiences the OFF phase prior to the time that has been established for the next dosage of the drug. During this time, the patient has the perception that the effects of the medication have worn off and are ready for the next dose. It is determined that the patient is in the ON state when they are feeling better while also being under the effect of medication.

In the OFF medication state, the patient's gait is formed by reduced or absent arm swing, reduced trunk rotation, forward leaning of the trunk, reduced range of motion of the hip, knee, and ankle, slowness, reduction in step amplitude, and decreased foot displacement height during the swing phase [39]. One further thing to take into consideration is known as the double-stance phase, which refers to how much longer the foot remains in touch with the ground.

In order for the people with Parkinson's disease who volunteered to take part in the study to be in the OFF phase, they were instructed to skip the first dosage of the day and go without taking any levodopa for a period of 12 hours [6], [75].

Chapter 4

Background

In this chapter, an insightful explanation of the algorithms that have been used in the experiments (elaborated on in Chapter 5), is presented.

4.1 Nearest Neighbors

An instance-based non-generalized classification, known as *Neighbors-based Classification* keeps samples of the training data rather than trying to create a general internal model. The categorization of each point is determined by a sole majority assessment of the points that are spatially closest to each center point. The data class with the greatest proportion among the query points nearest to a given query point is assigned to that center query point. The fundamental nearest neighbor classification makes use of uniform weights. This means that the value that is given to a query point is derived from the votes of the closest neighbors using a simple majority vote. There are situations in which it is preferable to weigh the neighbors in such a way that those who are closer to the neighbors contribute more to the fit. This is something that can be accomplished by using the weights keyword. When you choose the default setting of weights as uniform (all points in each neighborhood are weighted equally), you are giving each neighbor the same amount of weight. Weights are allotted in a manner proportionate to the inverse of the distance from the query point when distance is used as the value for weights. You also have the option of providing a user-defined function of the distance to be used in the computation of the weights [68].

4.1.1 K-Nearest Neighbors (KNN)

KNN is a machine learning algorithm that implements learning based on the k closest neighbors of each query point, where k is an integer number that the user specifies [65].

The KNN classification is one of the techniques that is widely and commonly used. The ideal choice of the value k is very dependent on the data. In general, a bigger k reduces the impact of noise, but it also blurs the classification boundaries. KNN is consistently ranked as one of the most efficient classification methods among the top 10 algorithms used in data mining. [76]

4.2 Decision Tree (DT)

A straightforward yet efficient supervised machine learning approach known as *decision tree* implements the model as a tree structure. This algorithm breaks down the samples into smaller subsets in the form of nodes and branches, then iteratively calculates the nonlinear relationship between the input and output [60]. In a wide variety of contexts, decision trees have been put to productive use. The power to extract descriptive decision-making information from the data that is provided is the most significant attribute that they possess. It is possible to create a decision tree using training sets [60]. The purpose of this algorithm is to develop a model capable of predicting the value of a target variable by the discovery and application of simple decision rules derived from the characteristics of the data. One way to think about a tree is as an example of a piecewise constant approximation.

Some advantages of decision trees are:

- Easy to comprehend.
- Feasible to graph trees.
- The cost of making predictions is proportional to the logarithm of the number of samples that were used to train the tree.
- Capability to work with numerical and category data simultaneously. The majority of the time, other methods are tailored specifically for the analysis of datasets that include only one kind of variable.
- Capability of handling difficulties involving several outputs.

- Utilizes a white-box model. Boolean logic can be used to depict an easy explanation for a certain circumstance in the model for various state of affairs. However, a black box model, such as an artificial neural network, produces outputs that can be more challenging to comprehend.

- Statistical testing can be used to validate a model. As a result, it is capable of explaining the model's reliability.

The disadvantages of decision trees include:

- Learners of decision trees may produce trees that are too complicated and do not generalize the input well. *Overfitting* is another name for this. In order to solve this issue, it is important to implement certain mechanisms, such as pruning, determining the minimum number of samples that must be collected at a leaf node, and determining the maximum depth of the tree

- It is possible for decision trees to be unreliable due to the fact that even minute changes in the data might lead to the generation of an entirely new tree. The use of decision trees inside an ensemble helps to alleviate the effects of this issue.

- The predictions made by decision trees are neither continuous nor smooth; rather, they are assumptions that are piecewise constant. As a result, they are not skilled in the art of extrapolation.

- It is well known that the task of learning an optimal decision tree is NP-complete under a number of different characteristics of optimality, and this holds true even for basic notions. As a consequence of this, practical decision-tree learning algorithms are based on heuristic algorithms like the greedy algorithm, in which judgments are made at each node that are locally optimum. These algorithms cannot ensure that they will provide the decision tree that is globally optimal. Training several trees in an ensemble learner, in which the features and samples are randomly selected with replacement, is one way to reduce the impact of this problem.

- Some ideas, such as XOR, parity, and multiplexer problems, are challenging to grasp due to the fact that decision trees are unable to explain them in a straightforward manner.

- Learners of decision trees produce biased trees if some classes are more prevalent. Before attempting to fit the dataset to the decision tree, it is thus important to ensure that the dataset is balanced.

4.2.1 Decision Tree Classifier

A decision tree classifier is a type of classifier that can be applied for multi-class classification on a dataset.

As is the case with other classifiers, the decision tree classifier accepts as input two arrays: an array X, which can be sparse or dense and has the shape (n samples, m features), which stores the training samples, and an array Y, which stores the class labels for the training samples and also has the shape (n samples), which stores the integer values [60].

4.2.2 Decision Tree Regressor

A supervised learning issue having several outputs to predict is referred to as a multi-output problem. This kind of problem occurs when the variable Y is represented as a two-dimensional array with the form (*n_samples*, *n_outputs*) [61].

When there is no connection between the outputs, a relatively easy solution is to develop separate models, one for each output, and then use those models to forecast each and every one of the outputs individually. This is a solution that can be used when there is no correlation between the outputs. However, because it is likely that the output values related to the same input are themselves correlated, constructing a single model that is capable of simultaneously predicting all outputs is often a better way to go about things. This is because of the likelihood that the output values related to the same input are themselves correlated. To begin, the amount of time spent training is reduced since just a single estimator is developed. Second, the generalization accuracy of the estimator that is produced as a consequence may often be improved [61].

4.2.3 Comparing Decision Tree Regressor and Classifier for Classification

In this study, both Decision Tree Regressor and Decision Tree Classifier were used for the classification task. The use of both algorithms provides the advantage of model comparison, allowing for the selection of the best model for the data. Additionally, the outputs from both models can be used as inputs for ensemble methods, such as Random Forest or AdaBoost, to enhance the overall performance and accuracy of the model. This approach is particularly useful when the data has a non-linear relationship, as both Decision Tree Regressor and Decision Tree Classifier are capable of handling non-linear data and finding complex decision boundaries. Through the comparison of these two algorithms, this study aimed to identify the most suitable model for the classification task and improve the prediction accuracy.

4.3 Support Vector Machine

Support vector machines (SVMs) are a group of supervised learning algorithms that can be used for classifying data, doing regression analysis, and finding outliers [70].

The advantages of support vector machines are:

- Beneficial in the context of multi-dimensional datasets.
- Effective even when the number of dimensions exceeds the number of samples.
- It is also resource efficient since it only employs a subset of the training points (known as support vectors) in the decision function.
- The decision function may be modified to utilize a variety of kernels (such as radial basis function, linear, nonlinear, polynomial, and sigmoid), making it very versatile. It is possible to define individualized kernels, in addition to the standard ones that are given [22].

The disadvantages of support vector machines include:

- It is very essential to prevent overfitting when selecting kernel functions and regularization term if the selection of attributes is much more than the number of samples.
- SVMs do not immediately offer probability estimates. Rather, they are computed via a costly process known as five-fold cross-validation (for more details, please refer to Section 4.11).

4.3.1 Linear Support Vector Machine (LSVM):

SVM is a technique that depicts a suitable fit for bi-polar classification in a supervised way. It was first presented by Boser, Guyon, and Vapnik in the year 1992 [10]. SVM is useful because it can cope with both linear and non-linear classifications, which is a distinct benefit. In this sense, support vector machines (SVMs) make use of the fundamental concepts of the hyper plane and the margin in order to transform nonlinear inputs into a feature space that has a higher dimension utilizing kernel methods [66].

4.4 Ensemble Classifiers (EC)

The performance of classification may be improved using ensemble classifiers, which integrate a large number of decision tree classifiers that all learn the same goal function. This allows the techniques to combine the predictions made by each classifier. Combining the findings of a large number of classifiers is one of the primary advantages of EC since it lowers the likelihood of producing an inaccurate classification [62].

4.4.1 AdaBoost

Adaptive Boosting, known as *AdaBoost*, was first introduced by Yoav Freund and Robert Schapire [23]. By combining a set of classifiers, they developed an iterative ensemble algorithm with a higher accuracy than each of the included classifiers. It implements a powerful classifier by training a classifier on the initial dataset. Then, it fits further replicas of the same classifier on the same data samples, while the weights of erroneously classified instances are changed such that succeeding classifiers focus only on challenging cases.

4.5 Quadratic Discriminant Analysis (QDA)

Quadratic discriminant analysis (QDA) is a classifier created by fitting class probability distributions to the dataset and most popularly using Bayes' rule, with a quadratic decision boundary. Discriminant analysis (DA) includes techniques that are applied to both dimensionality reduction and classification. Furthermore, QDA is widely used due to its DA dual functionality and ability to separate non-linear data. Each class is presumed by QDA to have a Gaussian distribution.

4.6 Extra Tree Classifier (ETC)

Extremely randomized trees classifier, also known as *extra trees classifier*, is a form of ensemble learning approach that outputs its classification result by aggregating the outcomes of several de-correlated decision trees gathered in a *"forest"*. It is only distinct from a random forest classifier in the method in which the decision trees in the forest are constructed, but conceptually, it is very comparable to a random forest classifier.

The data from the first training sample is used to create each decision tree included inside the extra trees forest. Then, at each test node, each tree is given a random sample of k features from the feature set, and from those features, each decision tree must choose the feature that would partition the data the most effectively based on some mathematical criterion (typically the Gini index). This random sampling of characteristics results in the production of numerous decision trees that are independent of one another [63].

In order to perform feature selection utilizing the aforementioned forest structure, while the forest is being constructed, for each feature, the normalized total reduction in the mathematical criteria used in the decision of feature of split (Gini index, if the Gini index is used in the construction of the forest) is computed. This is done during the course of the construction of the forest. The Gini importance of the feature is the name given to this particular value. In order to accomplish feature selection, each feature is ranked in decreasing order according to the Gini importance of each feature, and the user chooses the top K features based on their preferences once the rankings have been determined [63].

4.7 Extra Tree Regressor

An extremely randomized tree regressor, also known as *extra tree regressor*, constructs extra-trees in a manner unlike that of traditional decision trees. When searching for the optimal way to divide the samples produced by a node into two distinct groups, random splits are first generated for each of the features that have been picked at random, and then the optimal split is chosen from among those random splits. When maximum features is set to 1, the result is the construction of a decision tree that is completely arbitrary [64].

4.8 Random Forest (RF)

Training a vast number of decision trees is the foundation of the *random forest* technique to ensemble learning, which may be used for classification, regression, and other types of problems. When used to classification problems, the output of a random forest is the category that was selected by the vast majority of trees. When doing a regression job, the mean or average prediction of the various trees is what is delivered. The researchers trained the RF using the characteristics extracted from the preprocessed data in order to categorize PD patients and healthy individuals. The findings pointed to a challenge for the model due to the fact that the characteristics displayed a tight link among themselves [69].

4.9 Bayes Classifier (BC)

The term *naive bayes methods* refers to a group of supervised learning algorithms that are based on applying Bayes' theorem with the *naive* assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem is used to classify data based on the relationship between the features [67].

The Bayes classifier is a statistical classifier that, in accordance with the Bayesian theorem, does probabilistic feature prediction and makes use of the forecast in order to classify a new test dataset. BC is a recommended option to go with if the input dimensionality is large and the prior probabilistic model is already known [46]. In probabilistic connection modeling, often known as BC, the class is determined by applying posterior probabilities to the feature vectors and the class variable. The Bayes classifier has also exhibited results that are comparable to those of the random forest.

4.10 Artificial Neural Networks (ANNs)

The current state-of-the-art technique for making prediction/classification in supervised manner is called an *artificial neural network (ANN)*, and it is defined as a set of linked nodes that are implemented in a supervised way [50].

In the world of machine learning, they form the basis of popular deep learning algorithms. Machine learning is broken down into subfields, one of which is neural networks, which are the fundamental building blocks of deep learning algorithms. Other names for these networks are artificial neural networks (ANNs) and simulated neural networks (SNNs) [78]. Their structure, which is analogous to the manner in which neurons in living organisms interact with one another, took its cues from the human brain, which served as the source of inspiration for both their name and their structure [36].

Training data are required for neural networks to learn and improve their accuracy over time. However, once these learning algorithms have been fine-tuned for accuracy, they become powerful tools in computer science and artificial intelligence, allowing us to rapidly categorize and cluster data. This opens up several research options.

Tasks requiring speech recognition or image identification may take minutes rather than hours in comparison to the time it takes human specialists to manually identify something. Google's search algorithm is one of the neural networks that has attracted the most attention [36].

Advantages of ANNs are:

- Information is not saved on a database but rather on the whole network. This is the case for information that is used in conventional programming. The inaccessibility of a few bits of information in a single location does not render the network inoperable.

- Capability to function despite the presence of missing information. Following ANN training, the data may provide output despite the presence of missing information. The severity of the performance loss in this situation is directly proportional to the significance of the information that is unavailable.

- Having fault tolerance means that even if one or more of the ANN's cells get corrupted, the network will still be able to generate output. Because of this property, the networks are more resilient to failure.

- Having a memory that is distributed: In order for ANN to be able to learn, it is important to identify the examples, and then train the network according to the output that is intended by presenting the network these instances. The success of the network is directly related to the examples that are chosen.

- The capacity for machine learning: artificial neural networks learn events and make judgments by commenting on other occurrences that are similar.

- Artificial neural networks have the numerical ability to do multi-tasks at the same time, allowing them to perform parallel processing.

Disadvantages of ANNs:

- A network will gradually slow down and experience relative deterioration as time passes due to gradual corruption. The issue with the network does not instantly deteriorate quickly.

- Because of the nature of their structure, artificial neural networks have a dependency on the underlying hardware. These networks need CPU processors that are capable of parallel processing. As a consequence of this fact, the actualization of the apparatus is dependant.

- The most significant issue with ANNs is that they cannot adequately describe the behavior of the network. When an ANN generates a probing solution, it does not provide any explanation as to why or how the answer was generated. This results in a decreased level of confidence in the network.
- Determination of the appropriate network structure. There is not one single rule that can be used to figure out the structure of artificial neural networks. Obtaining the appropriate network structure requires both expertise and learning via trial and error.
- ANNs need to deal with numerical information, which makes it difficult to demonstrate the issue to the network. Prior to being fed into an ANN system, problems must first be reduced to their numerical essence and analyzed. The display method that will ultimately be chosen in this context will have a direct impact on the efficiency of the network. This is dependent on the capabilities of the user.
- There is no way to determine how long the network will remain operational. Nevertheless, when it reaches a certain level of accuracy on a test sample, it indicates that the training phase is over. This setting does not provide ideal results.

The ANNs that were first introduced to the world in the middle of the 20th century are undergoing fast improvements. The benefits of ANNs, as well as the challenges that might arise from using them, have been analyzed and discussed at length. It is essential to keep in mind that the limitations of ANNs, are gradually being overcome as their applications continue to expand. This indicates that ANNs will eventually become an integral component of our daily lives and will become more significant.

The major classes of ANNs are described in the following sections.

4.10.1 Multi Layer Perceptron (MLP)

The simplest kind of ANN is called a multilayer perceptron (MLP), and it works by connecting all of the nodes in a given layer. It then calculates a classification probability based on the feed forward result. This study makes use of MLP to compare the efficacy of ANNs with conventional approaches. The goal is to keep things as straightforward and simple as possible.

4.10.2 Convolutional Neural Network (CNN)

CNNs are a sort of deep learning model inspired by the architecture of the animal visual cortex to analyze input that has a grid structure, such as photographs. These networks are used to process data like facial recognition data [31], [24]. CNNs are intended to learn spatial hierarchies of information automatically and in an adaptable manner, progressing from low-level patterns to higher-level ones. The convolution layers, pooling layers, and fully feedforward connected layers make up a CNN, which is a mathematical construct that is often made up of these three kinds of blocks. The first two layers, which are called convolution and pooling layers, are responsible for the process of feature extraction. The third layer, which is called a fully connected layer (or alternatively, a dense layer), is responsible for mapping the features that have been extracted into the final output, such as classification. A CNN is made up of a stack of mathematical operations, one of which is called convolution, which is a specialized kind of linear operation. The convolution layer plays an important part in CNN. Because pixel values in digital images are stored in a two-dimensional (2D) grid, also known as an array of numbers (Fig. 2), and a small grid of parameters known as a kernel, which is an optimizable feature extractor, is applied at each image position, CNNs are extremely efficient for the processing of images. This is due to the fact that a feature may appear anywhere in the image. The retrieved characteristics have the potential to increase in complexity both hierarchically and sequentially as one layer feeds its output into the next layer. Training is the process of improving parameters such as kernels, and it is done in order to reduce the difference between outputs and ground truth labels using an optimization procedure known as backpropagation and gradient descent, amongst other optimization algorithms.

A basic CNN model's architecture is composed of numerous repeats of a stack that comprises of several *convolution layers* and a pooling layer (such as *max pooling*), followed by one or more *fully connected feedforward layers.*

Convolution layer

The convolution layer is a critical component of the CNN design that is in charge of feature extraction. In general, feature extraction comprises a combination of linear and nonlinear processes, specifically the convolution operation and the activation function.

Pooling layer

A pooling layer is an act of a standard down sampling technique. This method decreases the inner dimensionality of the feature maps in order to add a translation invariance to tiny changes and deformation and to limit the number of future trainable parameters. It is important to notice that none of the pooling layers have a trainable parameter, but that filter size, stride, and padding are hyperparameters in pooling operations. This is because pooling operations are very similar to convolution operations.

Max pooling

The most common kind of pooling operation is known as max pooling, and it works by taking patches from the input feature maps, producing an output value that is equal to the largest value in each patch, and throwing away all of the other values.

Fully connected layer

The last results retrieved by the convolution or pooling layer are typically flattened, which means that they are converted into a one-dimensional (1D) array of numbers (or vector), and then connected to one or more fully connected layers (feedforward network), which are also known as dense layers, in which every input is connected to every output by a learnable weight. This process is known as *flattening*. After the features have been retrieved by the convolution layers and down sampled by the pooling layers, they are mapped by a subset of fully connected layers to the final outputs of the network. For example, in classification tasks, the probabilities for each category are derived from this mapping. Typically, the last layer to be completely linked will have the same number of output nodes as there are classes. Following each completely connected layer comes a nonlinear function, such as the rectified linear unit (ReLU) function.

4.10.3 Long Short-Term Memory (LSTM)

An analysis of error flow in existing RNNs [29] provided the impetus for the development of the LSTM architecture [30], [25]. This analysis discovered that long time lags were inaccessible to existing network models due to the fact that backpropagated error either gets destroyed or failure will occur.

The LSTM Recurrent Neural Network (RNN) is a specialized kind of RNN that was designed to meet the objective of long-term reliance and address the challenges of limited memory capacity brought on by gradient vanishing or explosion [9] in conventional RNN models. In addition, it is a sequential network, which makes it possible for information to be stored. It is possible to solve the issue of disappearing gradients that RNNs experience. When it comes to long-persistent memory, a recurrent neural network, also known as an RNN, is the way to go. The LSTM algorithm integrates long-term memory into recurrent neural networks. It solves the *vanishing gradient problem*, which is when a neural network stop learning because the updates to the different weights inside a specific neural network grow less and smaller. This issue is resolved thanks to this solution. This is accomplished via the use of a number of different *gates*. These are stored in memory blocks, which are coupled to one another via layers.

The LSTM is known for using three distinct kinds of gates, which include the *forget gate*, *the input gate*, and *the output gate*.

Forget Gate

The purpose of the forget gate is to either store the information from the previous state or remove it entirely. The first thing that has to be decided in this situation is whether or not the information from the prior timestamp should be discarded or kept. The information from the neuron cell that is not vital is filtered out by this gate. This ultimately leads to the enhancement of performance. This gate has two inputs; one is the output provided by the cell that came before it, and the other is the input value of the current cell. The value is then given an activation function (i.e., sigmoid function), after the appropriate bias and weights have been added and multiplied. produces a number between 0 and 1 and uses this to guide the decision about which pieces of information to maintain. If the value is 0, the forget gate will eliminate that information; if the value is 1, the information must be remembered since it is significant and must not be forgotten.

Input Gate

The functionality of the input gate is to exercise control over the information about the prior output status and the present input state. This information is utilized to assess how significant the newly acquired information conveyed by the input is. This gate is used not only for the purpose of regulating the information but also for the purpose of adding information to the neuron cell. It is responsible for deciding what values should be added to the cell by applying an activation function like *sigmoid*. It produces an array of information that needs to be added. This is done by employing another activation function, such as *tanh*. It produces a result that is somewhere between -1 and 1. The sigmoid function works as a filter and governs what information needs to be put in the cell.

Output Gate

The function of the output gate is to regulate outputs in a manner that is determined by the present state. This gate is in charge of picking significant data from the currently active cell and displaying it as the result of that selection as the output. By using the tanh function, it generates a vector of values with a range that goes from -1 to 1. As a regulator, it takes into account both the most recent input and the most recent output, as well as the sigmoid function, and selects which numbers should be shown on the screen.

4.10.4 BiDirectional LSTM

Bidirectional Long Short-Term Memory (BiLSTM), is a kind of recurrent neural network that is most often used in natural language processing, or time-series based applications. These may be thought of as an improvement over LSTMs. In order to differentiate amongst recurrent networks, bidirectional LSTMs exhibit each training sequence in both forward and backward directions throughout the learning process. Both sequences share the same output layer that they are linked to. BiLSTMs are equipped with comprehensive knowledge of every point in a particular sequence, including everything that came before and after it.

In contrast to a conventional LSTM, the input flows both ways, and the system is able to make use of information from both sides. In addition to this, it is an effective method for modeling the sequential dependencies that exist between words and phrases in both the forward and backward directions of the sequence. In conclusion, BiLSTM consists of the addition of one extra layer of LSTM, which inverts the normal flow of information. To put it simply, it indicates that the extra LSTM layer processes the input sequence in reverse order. After that, it aggregates the outputs of the two LSTM layers in a number of different ways, including averaging, summing, multiplying, or concatenating them.

4.11 Cross Validation (CV)

A method known as 5-fold cross validation is included as an essential component of the classification process.

The process uses k as the parameter that specifies the number of groups into which a given data sample should be divided. Hence, the process is frequently referred to as k-fold cross-validation. When a specific value for k is specified, it may be substituted for k in the model's reference, such as k=*5* for 5-fold cross-validation [11]. This process (CV) is done to help prevent overfitting and to help guarantee that the accuracy findings would be consistent over time.

Chapter 5

Methodology

The main objective of this research is to perform four main tasks of PD classification. The first task is to distinguish between a healthy person and a patient with PD (Section 5.1.1). The second task is to determine the effectiveness of medication on each individual patient (Section 5.1.2). Following, the third task detects the FOG episodes within different timestamps (windows) of a patient's activity. The fourth task, similar to the third task, predicts FOG occurrences but in a real-time manner. The first two experiments are described in Section 5.1, and the two latter ones are elaborated on in Section 5.2.

For all these tasks, the gathered data is preprocessed and cleaned as the first step. Furthermore, the data records are passed to the feature extraction module. Subsequently, multiple classifiers are trained based on the extracted features, cross validated, and are tested against the main dataset. Each of the abovementioned steps is elaborated upon below.

5.1 Initial Experiments

In this section, the initial experiments are discussed. These experiments are also published in [8].

5.1.1 Detecting PD patients from healthy volunteers

For the purpose of this classification issue, all of the data from all of the tasks that are stated in Section 3 are utilised. Only healthy volunteers and PD patients who are FOG+ and in the ON state for medication are taken into consideration. Therefore, the purpose is to differentiate between these two categories.

5.1.2 Effectiveness of Medication

Considering just PD patients who are FOG+ and in the ON state for medication, in addition to PD patients who are FOG+ and in the OFF state for medication, all of the data records from all of the tasks that are mentioned in Chapter 3 are utilised for this issue. In order to differentiate between these two groups, it is necessary to do an analysis of the impact that the drug had.

5.1.3 Preprocessing

Having the raw data samples from initial experiments, some of the volunteers struggled in their trials, or some of the sensors missed recording their movements, at certain points. Hence, as an initial preprocessing task in the framework, the data records are cleaned from missing sensor values, false signals, and out-of-range values, which is considered to be the preprocessing phase. The records are then fed to the feature extraction module. Over 200 features (for some experiments, up to 1400 features over all of the sensors) were retrieved to assist better detecting the characteristics of each sensor at a given span. The Tsfresh [21] library from Python programming language was used for this purpose. Please refer to [20] to see the complete list of features used in this study, which includes for example absolute maximum, absolute sum of changes, absolute energy etc.

Data cleaning

The data records received from sensors through Bluetooth might experience a slight disconnection, which leads to a loss of data record in the dataset. Moreover, due to the time-series type of the dataset, the length of the received signals may vary from each other by only a few milliseconds. Although the validity of the data records was verified by human visualization, the structure of the data required justification. Hence, volunteers with missing sensors were omitted, those with extra records were trimmed to the average record length, and sets with a few missing records are then padded. The padded sets are those with all the sensors but missing a few milliseconds (which is still processable). However, in the case of missing sensors, it is not possible to make a classification or prediction, and hence they need to be excluded from the classification process.

Feature Extraction

In order to examine more persistent and accurate results, it is crucial to extract a set of fundamental features from the data. By doing so, solving various challenges and tasks in time-series-based datasets becomes easier. In order to deal with the high dimensionality of data and redundant records, the useful features are extracted to only deal with parts of the data which can make a positive difference in the classifications. For this purpose, by taking advantage of the Python library "tsfresh", over 200 features were extracted. Then the feature extraction was applied to all the experiments and trials. Moreover, various combinations of trials and experiments were gathered and processed. As an example of such combinations, the task of distinguishing people with PD and healthy volunteers requires features from a concatenation of all experiments and trials from both types of medicine status and a combination of all the healthy volunteers doing all the experiments.

Dataset generation

As the last preprocessing step, all of the extracted features are stored in CSV files in a data folder using the Pandas library [54]. At this step, all the *Not a Number (Nan)* records are then removed.

5.1.4 Classifiers

In this section, a thorough description of the settings and parameters of the classifiers used in this study is presented. To read more about the classifiers and how they work in general, please refer to Section 4.

K-Nearest Neighbors Classifier

In this part, the specific settings used for the KNN (described in Section 4.1.1) are articulated. This study takes advantage of the extracted features from the preprocessing phase, and by considering the cross-validation method, they are fed into the KNN input layers using 2 neighbors. For this study, the weights of the KNN are set to uniform. In the uniform mode, the weight distribution of each point is done equally. Moreover, the leaf size is set to 30. This affects how quickly the tree is created, queried, and stored, as well as how much memory is required. Finally, the metric used to compute the distance is minkowski. It is worth mentioning that for the purpose of this study, Scikit Learn library in Python was used [65].

Tree Classifiers and Regressors

In this part, the specific parameters used for the tree classifiers (described in Section 4.2) is elaborated. To calculate the quality of a split, Gini function is used [59], for which the results can be observed in the next paragraph. Regarding the depth of the tree, the nodes are expanded until all of the leaves are single. For the purpose of this study, Scikit Learn library in Python was used for all the Tree classifiers [60][61][63].

In this study, Decision Tree Classifier(DTC) showed an exceptional result in various scenarios, compared to other methods. Hence, the analysis of how DTC performed on different tasks is illustrated below (see further analysis in Section 6.3).

The DTC analysis (settings) for a single trial of the Voluntary Stop, Dual Task, and Timed Up and Go (TUG) experiments are shown in Figures 5.1, 5.2, and 5.3, respectively.

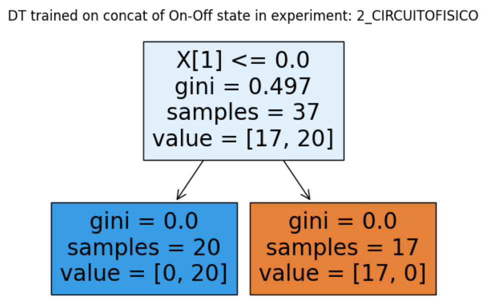


Figure 5.1: Voluntary Stop.

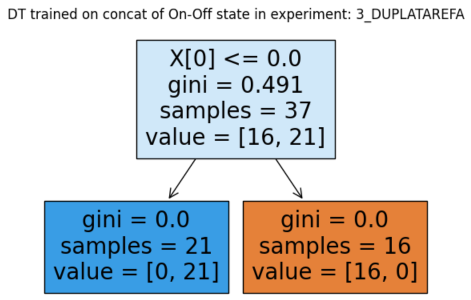


Figure 5.2: Dual Task Motor.

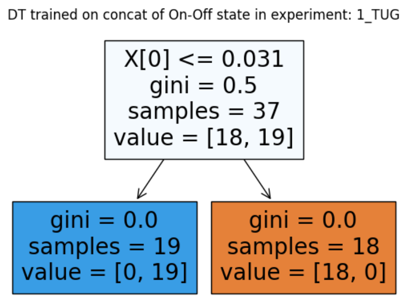


Figure 5.3: Timed Up and Go (TUG)

The settings used for decision tree classifier analysis of all three trials combined (single experiment based) for the Voluntary Stop, Dual Task, and Timed Up and Go (TUG) experiments are shown in Figures 5.4, 5.5, and 5.6, respectively.

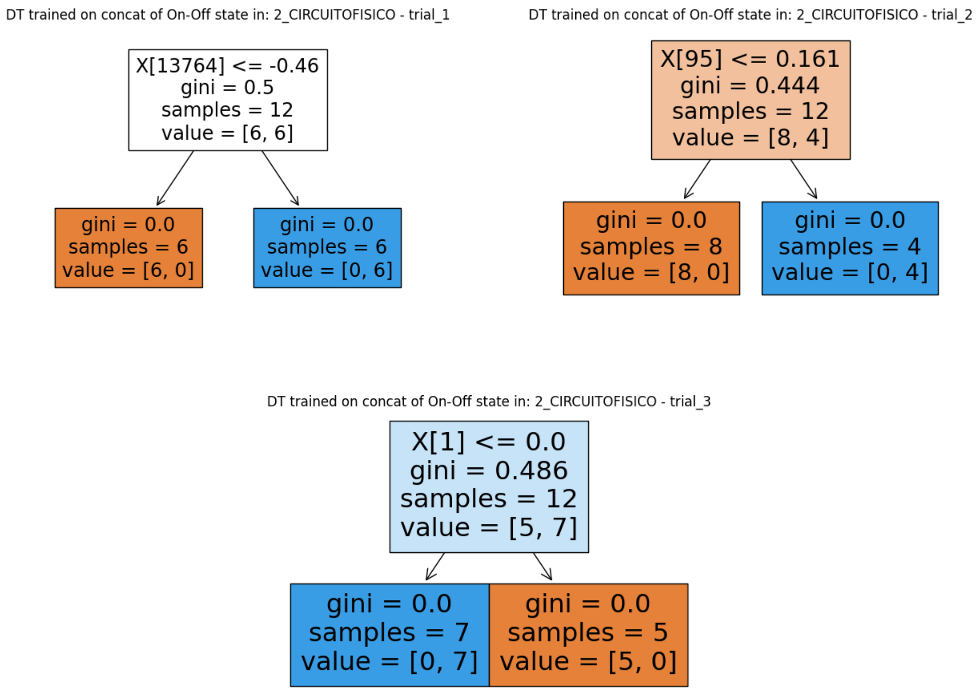


Figure 5.4: Trials of Voluntary Stop

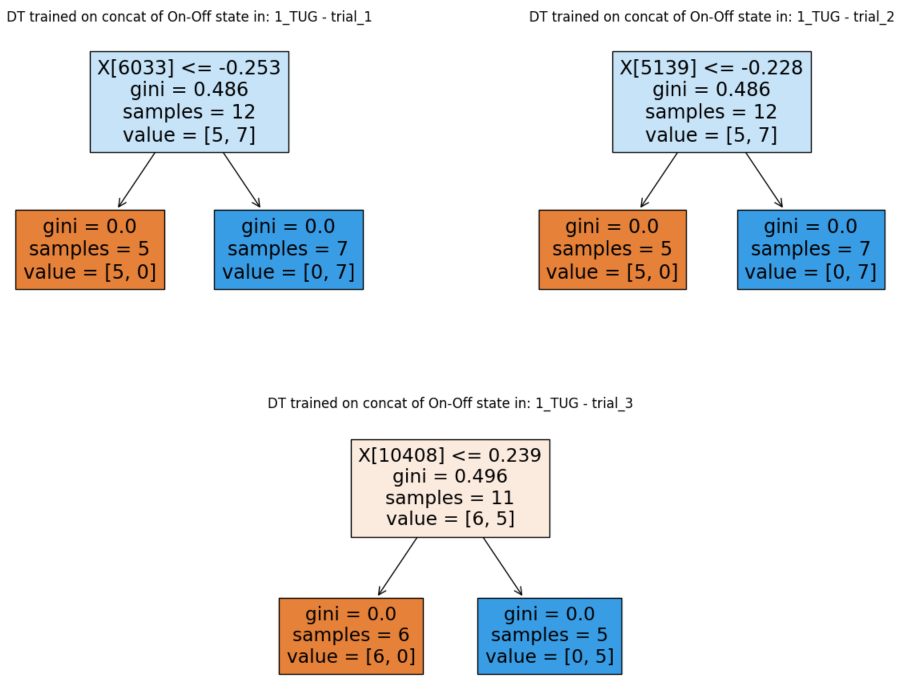


Figure 5.5: Trials of Timed Up and Go

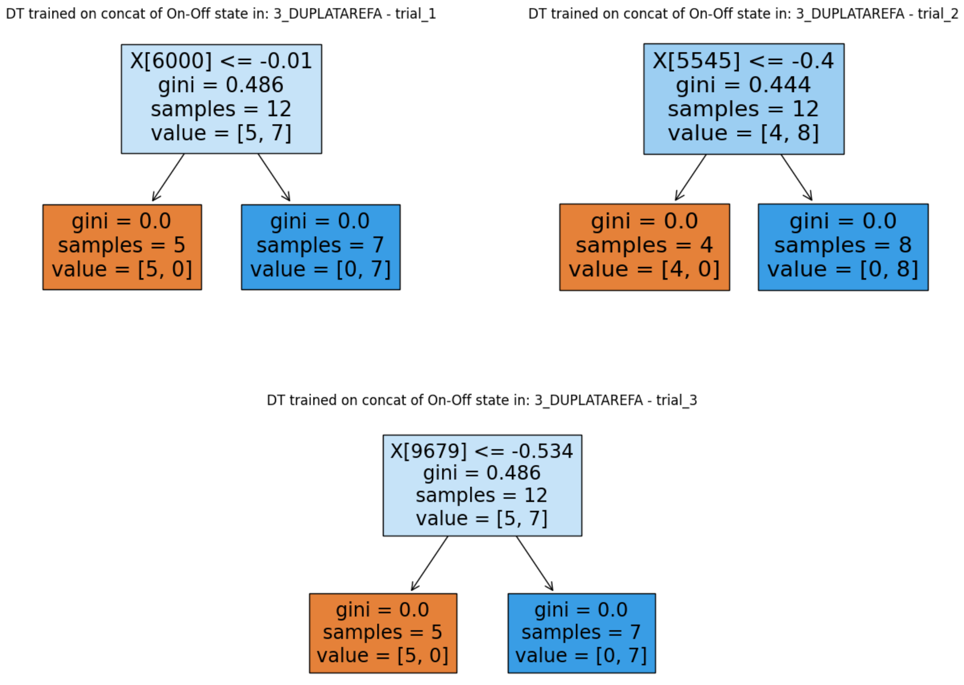


Figure 5.6: Trials of Dual Simple Motor

And finally, the settings used for DTC analysis of all experiments combined (Voluntary Stop, Dual Task, and Timed Up and Go (TUG)), are shown in Figure 5.7.

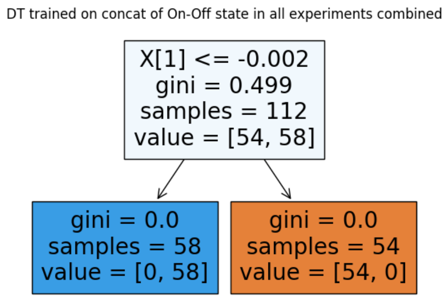


Figure 5.7: Trials of Dual Simple Motor

Support Vector Machine (SVM)

In this section, the detailed settings used for *SVM* (described in Section 4.3) are presented. In this study, both Radial Basis Function (RBF) [12] and Linear kernel are taken into consideration. For the sake of penalization, L2 penalty (as the standard choice) is used. Furthermore, *hinge* function is used as the loss function (which is a standard loss function in SVMs [70]).

AdaBoost

This section elaborates the kernel and the parameters used for *AdaBoost* (described in Section 4.4.1). In this study, *Stagewise Additive Modeling (SAMME)* was used as a multi-class AdaBoost function which performs slightly better than the base AdaBoost [26]. The estimator (classifier) used in this research is Decision Tree Classifier. And the number maximum estimators were set to 50. The learning rate is set to 1 (which has a trade-off with the number of estimators). At every boosting iteration, weight is assigned to each classifier. A greater learning rate boosts each classifier's contribution.

Random Forest

This study used Random Forest algorithm to analyze gait in Parkinson's disease. As elaborated in Section 4.8, Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to make a final prediction. Each decision tree in the Random Forest is constructed by randomly selecting a subset of features from the data and using them to split the data into multiple regions. The prediction of each tree is based on the majority vote of the samples in the region that the test sample falls into. By combining the predictions of multiple trees, the Random Forest algorithm can provide improved accuracy and stability compared to a single decision tree. In our study, we used the default parameters of the Random Forest algorithm, which includes 100 trees to be used in the ensemble, no limited depth of each tree (in which, the nodes are expanded until all leaves have less than 2 samples), and 2 samples required to split a node.

Naive Bayes

In the initial experiment of this research, we examined Parkinson's disease-related gait using a Naive Bayes algorithm. Naive Bayes, a probabilistic algorithm, bases its predictions on the highest likelihood of a collection of provided characteristics. The probability of each class based on the features is calculated using Bayes' Theorem under the assumption that the features are independent of one another. In this study, Naive Bayes was employed using algorithm's default settings, which include the distribution of the data and the smoothing parameter of 1e-9 employed to prevent overfitting.

5.1.5 Post Processing

Cross Validation (CV)

A generic explanation of *CV* is articulated in Section 4.11. In this section, the usage of CV in this thesis is presented. For this purpose, the dataset has been divided into 5 separate subsets, and one record at a time is trained and evaluated for accuracy. CV is performed on each of the subsets. Similar to the previous iteration, the first eighty percent of the dataset is used as training data, and the remaining twenty percent is used as test data. In the end, the findings are summarized by their mean.

5.2 FOG Prediction

5.2.1 Distinguishing Frozen episodes from Unfrozen episodes

In this section, a set of experiments were conducted to classify the frozen episodes from the unfrozen episodes. The experiments are based on the number of extracted features and how determining the right features can affect the results. Furthermore, the data is divided into separate windows with different window lengths. For the sake of preprocessing (which leads to the next step, processing), a window (episode) of records is taken (see more details in the next paragraph). Then preprocessing steps are applied to the window, and the output is stored into a separate file to be used as the input of the next step (classification process). Finally, the extracted features from the previous step are passed to the same set of classifiers, as in Sections 5.1.2 and 5.1.1.

The duration window of FOG occurrence were manually annotated in the dataset (overall 149 FOG episodes). Depending on how long each FOG episode took, the window length varies. The statistics for the windows are presented in Table 5.1. For this study, a median window size of 4150 ms is used.

All extracted features

features of that dataset is different fr

In this set of experiments, over 1400 features have been used to determine the results (see more details in Section 5.1.3). It is worth mentioning that the prepro-

Table 5.1: Window Statistics over 149 FOG episodes

Name	Duration (milliseconds)
Mean	10746.9
Maximum	208112
Minimum	336
Median	4150

cessing time (time to extract the features) was tremendously high, and with the help of parallelism, concurrency and GPU processing it took more than 20 hours to achieve the results for the preprocessing phase.

Single feature

To demonstrate the effectiveness of taking advantage of multiple features, a set of experiments was performed using only one single feature to compare with the multi-feature experiments. It is evident that with using a single feature, the classifiers are facing more challenges than with multi-feature datasets. However by compromising the accuracy, this model performs the fastest in terms of feature extracting speed. In this study, absolute energy is used as the only feature for classification. Using any single feature from Section 5.2.1 depicts a similar result. However, the result from absolute energy was chosen randomly as a candidate for the single feature classification to demonstrate the performance and metric measurements of this type of classification.

Subset of extracted features

A hybrid of the two approaches described above can help to solve the issues of long processing time and lack of sufficient accuracy. For this purpose, only a subset of features that performed best, were chosen. This includes 6 features, namely, length, absolute energy, absolute maximum, AR coefficient, FFT aggregated, and Fourier entropy (based on various trials and errors). For this matter, other experiments with various features and lengths were taken into consideration. These experiments consisted of a feature combination of length, absolute energy, absolute maximum, AR coefficient, FFT aggregated, Fourier entropy, change quantiles, count above the mean, count below the mean, and Friedrich coefficients, using combination counts of 4, 5, and 6. However, the abovementioned setup of the 6 features performed best.

5.2.2 Predicting freezing of gait steps

Another useful application in this field is real-time prediction for the FOG, prior to/at the same time of its occurrence. For this purpose, removing the preprocessing phase, as one of the most time-consuming steps, is the first task. However, since preprocessing plays a crucial role in the end result of the prediction, it will be transferred to the actual classification process. To increase the performance, a deep neural network is designed to fulfil this task, and the preprocessing part is transferred to a CNN subnetwork of the model. Furthermore, we proposed an efficient framework to predict the freezing of gait prior to its occurrence, and to demonstrate its effectiveness, the framework is compared with other DNNs.

5.2.3 Predicting Freezing of Gait using Naive CNN

For the first task of the prediction, a deep neural network is used which is popular for its thorough feature extraction before the prediction. The network is built from a layer of input, a layer of CNN, followed by a batch normalization. Finally, the result is passed to a dense layer (a simple feedforward MLP) to get the results.

5.2.4 Predicting Freezing of Gait using the proposed framework

The proposed framework for predicting Parkinson's disease timestep by timestep includes a network of state-of-the-art DNN models, aligned with the characteristics of the sensors and the dataset. The purpose of this framework is to predict the FOG at the time of occurrence in a real-time manner. Hence, the feature extraction and prediction parts are done at the same time, which leads to a real-time result for the patient without the need to distinguish any parts from each other. The framework consists of four major parts, i.e., the input layer, the feature extraction layer, the memory layer, and the dense layer. An illustration of the architecture is depicted in Figure 5.8. For more details regarding each individual cell, input and output size of the network, and the model architecture, please refer to Figures 6.1 and 6.2. The input layer receives the data in the shape of 18 (3 sensors * 6 axes) and passes it to the feature extraction layer. In the feature extraction layer, 6 cells are deployed to extract the features for each set of axes independently. The first cell is responsible for extracting features in a *unigram* manner (extract features only based on one single axis of a sensor at a time). The second cell extracts features from the same data, without any prior knowledge of the unigram but in a *bigram* manner (which takes two axis of a sensor at a time and extracts the features based on those two directions together). It goes on for all the axes, after which the last cell extracts the features based on all the information retrieved from all six axes of a sensor (in other words, feature extraction based on full information retrieved from each sensor).

Furthermore, each cell consists of 5 layers, as described in the following subsections.

Convolutional layer

The filter size for each cell is 256 which refers to the dimension of the output space, or the number of output filters in the convolution. The first cell (*the unigram cell*) has a kernel size of 1, *the bigram* cell has a kernel size of 2, etc. The kernel size, the length of the convolution window, depends on the cell. The input is always collected straight from the input layer in this case, while ReLu is the activation layer utilized in each of the convolutional layers.

Batch Normalization

The output from the convolutional layer is passed to the batch normalization layer. The logic behind it is that instead of normalizing the raw data, a neural network's layers may use a technique called *Batch Norm*. In this method, work is done in chunks of data rather than all at once. Training may be completed more quickly, and higher learning rates can be put to use, simplifying the learning process.

Flatten

Before making a prediction for the desired features using a dense layer, one should pass the data to the *flatten layer* to get a one-dimensional vector from the same output. Now, the flattened output can be used for prediction.

Dense Layer

This is the prediction layer of the feature extraction phase. In this step, all the output from the flatten step is passed to a feedforward neural network and the result is collected.

Batch Normalization

Finally, the prediction from the dense layer is normalized. Hence, it could be used in the concatenation layer, along with the other cells. As the last step of feature extraction phase, all the retrieved results from the cells are concatenated, and the result is flattened to be used in the final prediction stage.

In the memory layer, the input is retrieved straight from the input layer and passed to the memory cells, individually. Hence, each memory cell is independent from the other, so the result is more consistent. A memory cell consists of a BiLSTM layer, having two LSTMs with 128 units in each LSTM cell. Each unit demonstrated the dimensionality of the output space. In this framework, two memory cells are used, which are considered to be the bottlenecks of the framework in terms of computation time. However, without having these cells, the network would have an impaired idea of the patients' history, which leads to a wrongful prediction. This is mostly because the output of the sensors from the unfrozen parts of the dataset can be similar to the frozen parts of the records, which confuses the whole network.

In the last layer of the framework, the output of the memory cells is concatenated with the output of the feature extraction model, and the result is passed to a dense layer. For this MLP layer (the dense layer), only one probability output is set, and the decision is made by using the sigmoid activation function. The total number of trainable parameters in this framework is 1,890,049 and it has a total number of 0 non-trainable parameters.

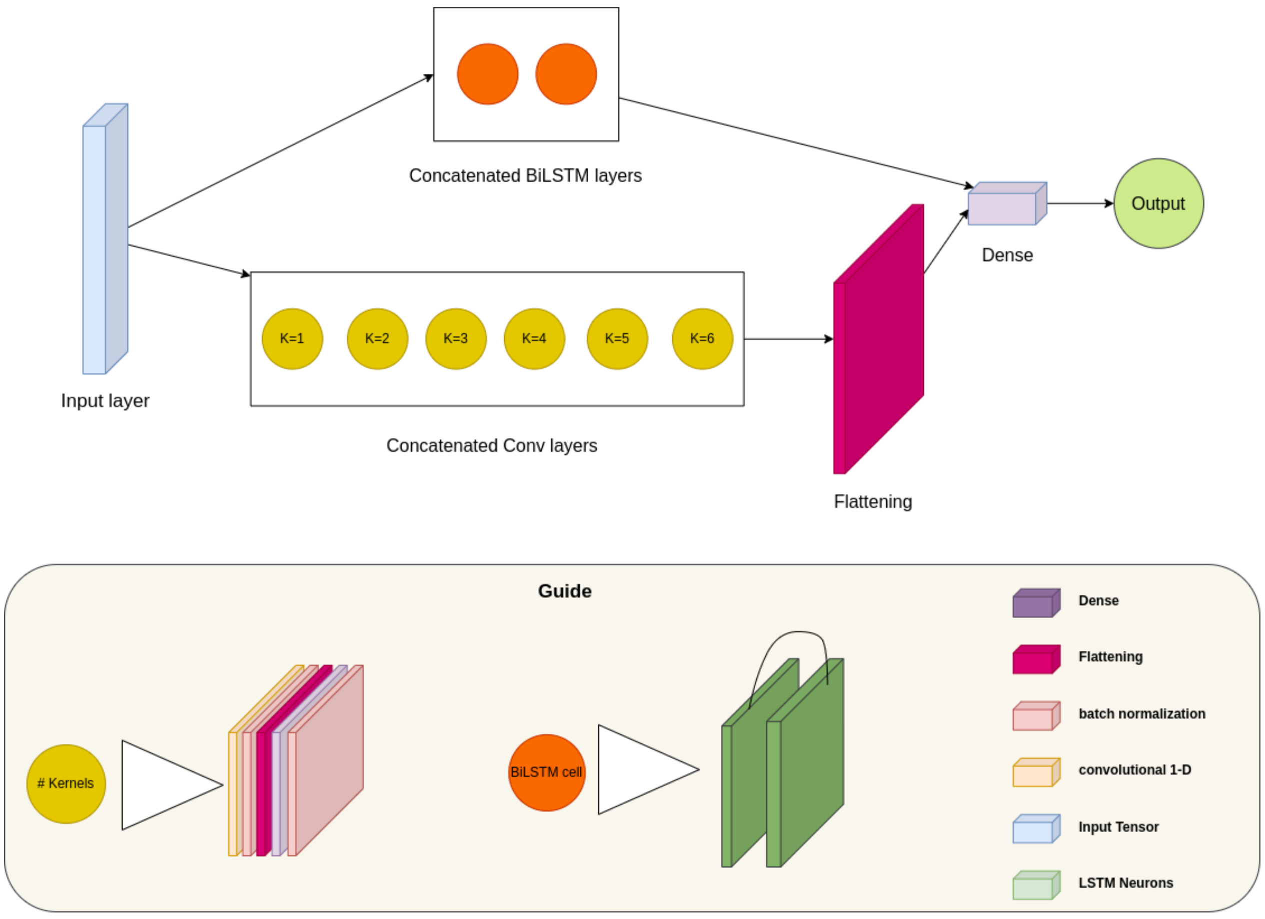


Figure 5.8: Schema of proposed framework

Chapter 6

Results

In this chapter, the results of the experiments described in Chapter 5 are presented. Experiments are undertaken with the primary goals of differentiating healthy persons from those who are Parkinson's disease (PD) patients and determining whether or not PD patients are taking their medication. Moreover, a set of experiments are conducted to detect and predict the FOG episodes within a patient's routine activity. Every one of the algorithms have been outlined in Section 5.

Accuracy, F1-score, and precision are the assessment criteria that are used for this research. The proportion of volunteers who are properly categorised in relation to the total number of volunteers is the definition of *accuracy*:

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

Here, we referred to *True Positives* (abbreviated *TP*) as *properly anticipated positive observations*. *FP* is an abbreviation for *False Positive*, which refers to an incorrect forecast of a positive result (i.e. the actual label is negative). In conclusion, the term *False Negative* (*FN*) refers to the number of positive labels that were incorrectly marked as negative, while the term *True Negative* (abbreviated as *TN*) describes the number of negative labels that were properly predicted.

Precision refers to the proportion of accurately anticipated positive observations relative to the total number of expected positive observations:

$$Precision = \frac{TP}{TP + FP}$$

The *F1-score* is calculated as the weighted harmonic mean of precision and recall scores, and is calculated as follows:

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

6.1 Initial Experiments Results

The findings point to a challenge for several classical algorithms due to the fact that the characteristics had a strong link among one another. Examples of such algorithms were random forest and Bayes classifier, both of which displayed results that were quite similar to one another. Due to the fact that the models are working with high-dimensional features, this is regarded to be one of the bottlenecks of the classical approaches.

6.1.1 Detecting PD patients from healthy volunteers

The findings of the experiment described in Section 5.1.1 are summarized in Table 6.1. The results presented in Tables 6.2-6.5 are generated from 27 healthy subjects and 27 PD subjects, each containing 200 features. For more details on feature extraction, please refer to Section 5.1.3. The 27 subjects consist of 9 individuals who performed the same task three times in different time frames to ensure the validity of the results. For more elaboration on the tasks, please refer to Section 3.5. For this experiment, the first 67% of the data was used for training and the remaining 33% was used for validation. Table 6.1 demonstrates the results from all of the tasks combined. In this regard, a total of 108 subjects were used as the healthy class and a total of 108 subjects were used as the PD class.

It is clear that while most of the classifiers show decent result, MLP delivers superior results than others with a score of 96% for F1-score, 96% for accuracy, and 99% for recall. It is important to point out that the ANN model has discriminated the feature dimensions more effectively than the previous approaches, especially when taking into account the number of features that were collected from the data.

Tables 6.2-6.5 provide the results of the accuracy, F1-score, precision, and recall tests conducted on each of the separate tasks outlined in Section 5.1. We are able to evaluate the overall performance of the classifiers for each of the tasks by dividing the data in this way and analysing the results separately.

Table 6.1: Accuracy, F1 score, Precision, and Recall for PD patients vs healthy volunteers on all tasks

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	86%	87%	90%	84%
Decision Tree	96%	94%	94%	99%
Linear Support Vector	94%	95%	93%	96%
Random Forest	76%	75%	87%	65%
Naive Bayes	74%	74%	81%	68%
Multilayer Perceptron	96%	96%	94%	99%

Table 6.2: Accuracy, F1 score, Precision, and Recall - PD patients vs healthy volunteers - Task: Voluntary Stop

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	72%	66%	100%	49%
Decision Tree	100%	100%	100%	100%
Linear Support Vector Machine	72%	72%	80%	66%
Random Forest	63%	60%	75%	49%
Naive Bayes	60%	60%	75%	49%
Multilayer Perceptron	63%	66%	66%	66%

Table 6.3: Accuracy, F1 score, Precision, and Recall - PD patients vs healthy volunteers - Task: TUG

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	100%	100%	100%	100%
Decision Tree	100%	100%	100%	100%
Linear Support Vector Machine	100%	100%	100%	100%
Random Forest	81%	82%	87%	77%
Naive Bayes	56%	69%	57%	%88
Multilayer Perceptron	100%	100%	100%	100%

Table 6.4: Accuracy, F1 score, Precision, and Recall - PD patients vs healthy volunteers - Task: Simple Motor Task

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	50%	33%	100%	19%
Decision Tree	93%	94%	100%	90%
Linear Support Vector Machine	81%	82%	100%	70%
Random Forest	68%	66%	100%	49%
Naive Bayes	68%	66%	100%	49%
Multilayer Perceptron	75%	74%	100%	59%

Table 6.5: Accuracy, F1 score, Precision, and Recall - PD patients vs healthy volunteers - Task: Dual motor and cognitive task

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	93%	90%	83%	100%
Decision Tree	93%	90%	83%	100%
Linear Support Vector Machine	93%	90%	83%	100%
Random Forest	68%	61%	50%	80%
Naive Bayes	56%	58%	41%	100%
Multilayer Perceptron	93%	90%	83%	100%

It is evident that according to Table 6.2, Decision Tree depicted a decent classification output on the Voluntary Stop. Since the Voluntary Stop task is the most simple task compared to the others, patients demonstrated very similar behavior to the healthy volunteers, which made the remaining classifiers struggle with detection.

It is clear to see that, in general, the Timed Up and Go (TUG) task, which is shown in Table 6.3, yielded the best possible outcomes for the participants. The Decision Tree, KNN, LSVM, and MLP classifiers each obtained a score of 100% across the board for every metric. Therefore, using any of these classifiers to differentiate between people with PD and healthy ones ought to provide excellent results for a TUG test.

Regarding the basic motor task that is outlined in Table 6.4, all of the classifiers achieved a precision of 100%, which means that they were able to accurately detect true positives each and every time. Taking into account all of the metrics, the Decision Tree approach produced the greatest results for the voluntary stop task.

The classifiers struggled with the task that required both motor and cognitive ability, as seen in Table 6.5. It is probable that the data for this task have higher variability, which might cause problems for some of the classifiers. Having said that, it is important to highlight the fact that four of the classifiers still had an accuracy and f1-score of at least 90%, and five of them had a recall of 100%.How selected?

6.1.2 Effectiveness of Medication

The findings of the experiment described in Section 5.1.2 are presented in Table 6.6. For this experiment, a total number of 108 subjects were used on the OFF status (refer to Section 3.6) and the same amount of subjects were used on the ON status. For each subject, 200 features were extracted from their movement. In this study, the first 70% of the data was designated for training and the remaining 30% was allocated

Table 6.6: Accuracy, F1 score, Precision, and Recall - Effectiveness of medicine on patients

Classifier	Accuracy	F1-Score	Precision	Recall
K-Nearest Neighbors	95%	95%	99%	92%
Decision Tree	99%	99%	99%	99%
Linear Support Vector	87%	88%	80%	100%
Random Forest	57%	58%	55%	62%
Naive Bayes	61%	59%	60%	58%
Multilayer Perceptron	96%	95%	92%	99%

for validation.

The classifiers utilised are the same ones that were used in the previous experiment. Because it has a score of 99% in all four categories of accuracy, F1-score, precision, and recall, the Decision Tree seems to be the most effective method of categorization for this issue. However, a number of other methods, most notably MLP and KNN, also produced very good outcomes, with scores above 90% for every metric, in most of the tasks. Each of these (KNN, DT, MLP) could be used with a high degree of confidence to assess the effectiveness of medications.

6.2 FOG Prediction

6.2.1 Distinguishing Frozen episodes from Unfrozen episodes

In this section, the results from the experiments of Section 5.2.1 are presented. The experiments were performed on the features of 149 FOG episodes against the features of 149 episodes in which no freezing occurred. For more details on statistics of the episodes, please refer to Table 5.1. The number of features vary for each experiment and is described in details in Section 5.2.1. In the present study, 70% of the data was utilized for training purposes and the remaining 30% was assigned for validation.

It is worth noting that the results shown in Table 6.7 have the highest scores in all the measurement metrics, and this score is reduced when lowering the feature dimension. Although using more features in this process increases the scores, doing so also uses more resources (computational power), which leads to an expensive time complexity. Hence, providing both priorities (higher scores and fewer resources), it is concluded that Table 6.9, which considers a subset of features, satisfies the requirements the most.

Tables 6.7, 6.8, 6.9 demonstrate the results for the experiments described in Sec-

Table 6.7: Accuracy, F1 score, Precision, and Recall - FOG Episode detection - All features

Classifier	Accuracy	F1-Score	Precision	Recall
Decision Tree Classifier	100%	100%	100%	100%
Decision Tree Regressor	100%	100%	100%	100%
Extra Tree Classifier	100%	100%	100%	100%
Extra Tree Regressor	100%	100%	100%	100%
K Neighbors Classifier	98%	98%	96%	99%
Linear SVM	100%	100%	100%	100%
RBF SVM	48%	65%	48%	100%
Gaussian Process	51%	0%	0%	0%
Random Forest	91%	90%	92%	89%
Neural Net	100%	100%	100%	100%
AdaBoost	100%	100%	100%	100%
Naive Bayes	100%	100%	100%	100%
QDA	60%	65%	56%	78%

tion 5.2.1. It is evident that the results shown in Table 6.9 (subset of features elaborated in Section 5.2.1) are comparable with the results shown in Table 6.7 (all features), while having an estimated time of 30 seconds for each of processes, in comparison to an average time of over 20 hours when using all features.

6.2.2 Predicting Freezing of Gait using Naive CNN

Table 6.10 demonstrates the results from using naive CNN for the experiment described in Section 5.2.3. In this table, the accuracy and f1-score of the training data (first two columns after the epochs) are presented, as well as the validation accuracy and the validation f1-score. The dataset used for this experiment consists of 395,073 unfrozen records and 160,267 frozen records. Each record represents sensors information of a timestep from the volunteers. Hence, the 160,267 frozen records indicates the timesteps in which the patients experienced FOG in their tasks. For this experiment and the following experiment described in Section 6.2.3, the CNN layers are responsible for the feature extraction, so only the raw sensor data is utilized. The training data consists of 70% of the whole dataset, while the validation dataset consists of the remaining 30% of the whole dataset. In increasing the epochs, it can be observed that the algorithm does not learn from its previous mistakes. Hence, it cannot extract useful information based on the limitations of the network. Moreover, the f1-score shows a huge gap with the accuracy score (30% difference), making it obvious that the network is also biased towards one class. This concludes that even

Table 6.8: Accuracy, F1 score, Precision, and Recall - FOG Episode detection - Single feature

Classifier	Accuracy	F1-Score	Precision	Recall
Decision Tree Classifier	96%	96%	93%	100%
Decision Tree Regressor	98%	98%	96%	100%
Extra Tree Classifier	91%	91%	90%	93%
Extra Tree Regressor	93%	93%	96%	90%
K Neighbors Classifier	98%	98%	100%	96%
Linear SVM	93%	93%	96%	90%
RBF SVM	91%	90%	100%	83%
Gaussian Process	98%	98%	100%	96%
Random Forest	93%	93%	93%	93%
Neural Net	93%	93%	93%	93%
AdaBoost	98%	98%	100%	96%
Naive Bayes	91%	91%	90%	93%
QDA	50%	0%	0%	0%

Table 6.9: Accuracy, F1 score, Precision, and Recall - FOG Episode detection - Selective Features

Classifier	Accuracy	F1-Score	Precision	Recall
Decision Tree Classifier	100%	100%	100%	100%
Decision Tree Regressor	100%	100%	100%	100%
Extra Tree Classifier	100%	100%	100%	100%
Extra Tree Regressor	100%	100%	100%	100%
K Neighbors Classifier	100%	100%	100%	100%
Linear SVM	98%	98%	100%	96%
RBF SVM	51%	67%	50%	100%
Gaussian Process	100%	100%	100%	100%
Random Forest	100%	100%	100%	100%
Neural Net	98%	98%	100%	96%
AdaBoost	100%	100%	100%	100%
Naive Bayes	100%	100%	100%	100%
QDA	81%	82%	78%	86%

Table 6.10: Accuracy, F1 score, Precision, and Recall - Real-time FOG detection - Naive CNN

#	Train Acc	Train F1	Val Acc	Val F1
Epoch 1	71%	41%	71%	41%
Epoch 2	71%	41%	71%	41%
Epoch 3	71%	41%	71%	41%
Epoch 4	71%	41%	71%	41%
Epoch 5	71%	41%	71%	41%
Epoch 6	71%	41%	71%	41%
Epoch 7	71%	41%	71%	41%
Epoch 8	71%	41%	71%	41%
Epoch 9	71%	41%	71%	41%
Epoch 10	71%	41%	71%	41%

Table 6.11: Accuracy, F1 score, Precision, and Recall - Proposed Framework - Training dataset

#	Accuracy	f1-score	Recall	Precision
Epoch 1	98%	98%	96%	98%
Epoch 2	100%	100%	100%	100%
Epoch 3	100%	100%	100%	100%
Epoch 4	100%	100%	100%	100%
Epoch 5	100%	100%	100%	100%
Epoch 6	100%	100%	100%	100%
Epoch 7	100%	100%	100%	100%
Epoch 8	100%	100%	100%	100%
Epoch 9	100%	100%	100%	100%
Epoch 10	100%	100%	100%	100%

though the network is extracting some features for prediction, it is not using all the essential information required to make a judgement. Hence, a more generalized network is required to check the data from different perspectives, which is one of the motivations for the proposed framework.

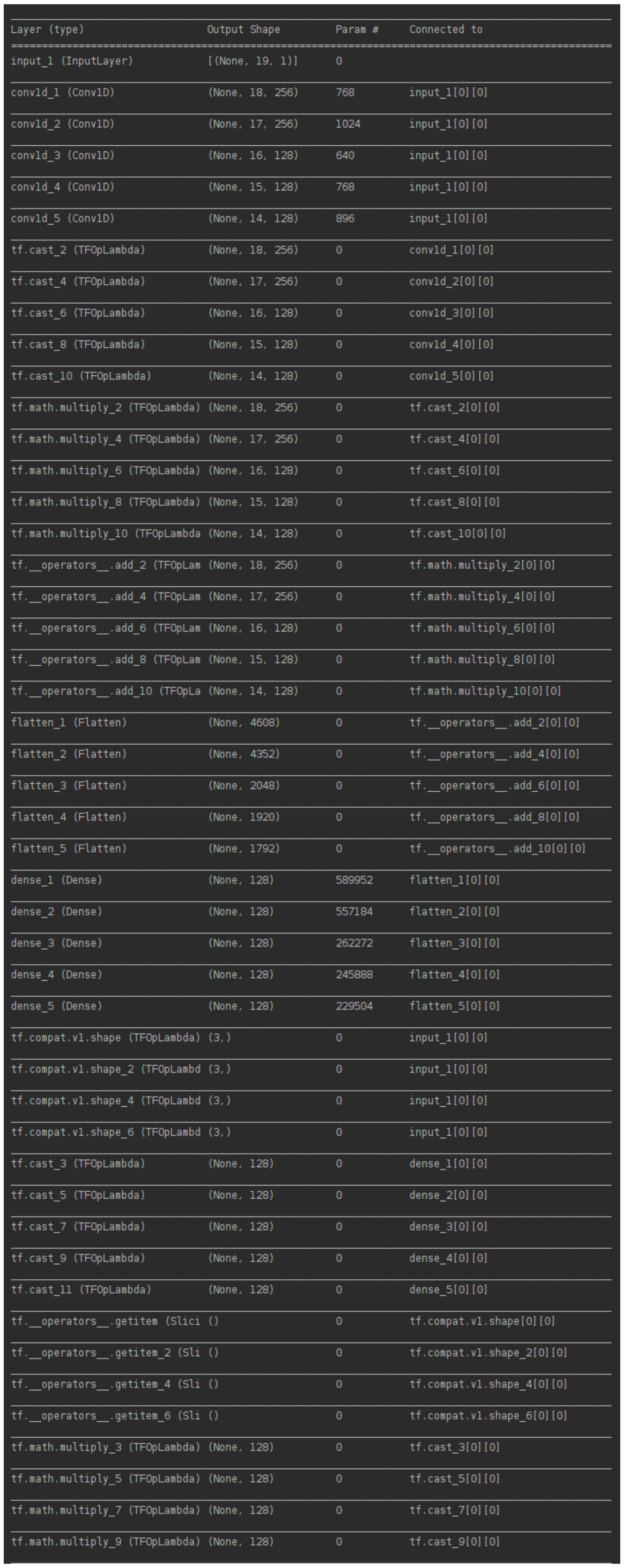
6.2.3 Predicting Freezing of Gait using the proposed framework

The dataset used for this experiment is the same dataset as described in Section 6.2.2. Hence, a total number of 395,073 unfrozen records and 160,267 frozen records were used for this experiment. The training data comprises 70% of the entire dataset, and the validation dataset consists of the remaining 30%. The framework was challenged

Table 6.12: Accuracy, F1 score, Precision, and Recall - Proposed Framework - Validation dataset

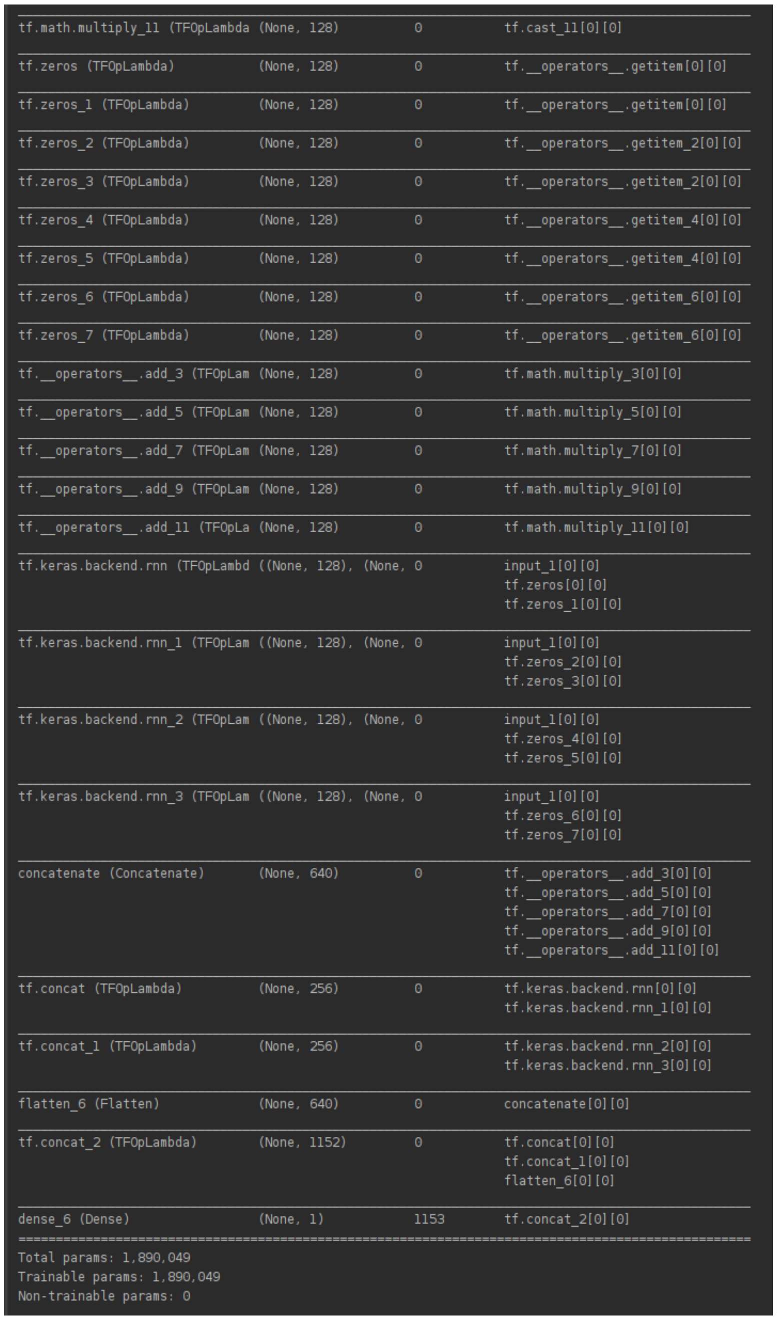
#	Val Accuracy	Val F1	Val Recall	Val Precision
Epoch 1	100%	100%	100%	100%
Epoch 2	100%	100%	100%	100%
Epoch 3	100%	100%	100%	100%
Epoch 4	100%	100%	100%	100%
Epoch 5	100%	100%	100%	100%
Epoch 6	100%	100%	100%	100%
Epoch 7	100%	100%	100%	100%
Epoch 8	100%	100%	100%	100%
Epoch 9	100%	100%	100%	100%
Epoch 10	100%	100%	100%	100%

by selecting an unbalanced dataset that mimics real-world scenarios where FOG occurrences are less frequent compared to other activities. This choice ensures robust results and showcases the framework's consistency in predicting FOG accurately, even with limited instances. Table 6.11 presents the measurement metrics on the training dataset. In this table, it can be observed that after the first epoch (with more than 95% score in all the metrics), the scores all reach 100%. Moreover, in increasing the epochs, the algorithm learns from its previous flaws and minimizes the loss, demonstrating the absence of negative learning. It is worth mentioning that the 100% is the output of the Tensorflow model. However, the actual numbers (according to the analysis in 6.12 which are made by Tensorboard) are slightly different (more precise). However, the difference is only a small amount. Most significantly, the results of the validation dataset, as demonstrated in Table 6.12, highlight the consistency of the framework and indicate the absence of overfitting. For further analysis, see Section 6.3.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 19, 1)]      0
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 18, 256)      768         input_1[0][0]
__________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 17, 256)      1024        input_1[0][0]
__________________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 16, 128)      640         input_1[0][0]
__________________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 15, 128)      768         input_1[0][0]
__________________________________________________________________________________________________
conv1d_5 (Conv1D)               (None, 14, 128)      896         input_1[0][0]
__________________________________________________________________________________________________
tf.cast_2 (TFOpLambda)          (None, 18, 256)      0           conv1d_1[0][0]
__________________________________________________________________________________________________
tf.cast_4 (TFOpLambda)          (None, 17, 256)      0           conv1d_2[0][0]
__________________________________________________________________________________________________
tf.cast_6 (TFOpLambda)          (None, 16, 128)      0           conv1d_3[0][0]
__________________________________________________________________________________________________
tf.cast_8 (TFOpLambda)          (None, 15, 128)      0           conv1d_4[0][0]
__________________________________________________________________________________________________
tf.cast_10 (TFOpLambda)         (None, 14, 128)      0           conv1d_5[0][0]
__________________________________________________________________________________________________
tf.math.multiply_2 (TFOpLambda) (None, 18, 256)      0           tf.cast_2[0][0]
__________________________________________________________________________________________________
tf.math.multiply_4 (TFOpLambda) (None, 17, 256)      0           tf.cast_4[0][0]
__________________________________________________________________________________________________
tf.math.multiply_6 (TFOpLambda) (None, 16, 128)      0           tf.cast_6[0][0]
__________________________________________________________________________________________________
tf.math.multiply_8 (TFOpLambda) (None, 15, 128)      0           tf.cast_8[0][0]
__________________________________________________________________________________________________
tf.math.multiply_10 (TFOpLambda (None, 14, 128)      0           tf.cast_10[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_2 (TFOpLam (None, 18, 256)      0           tf.math.multiply_2[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_4 (TFOpLam (None, 17, 256)      0           tf.math.multiply_4[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_6 (TFOpLam (None, 16, 128)      0           tf.math.multiply_6[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_8 (TFOpLam (None, 15, 128)      0           tf.math.multiply_8[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_10 (TFOpLa (None, 14, 128)      0           tf.math.multiply_10[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 4608)         0           tf.__operators__.add_2[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 4352)         0           tf.__operators__.add_4[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 2048)         0           tf.__operators__.add_6[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 1920)         0           tf.__operators__.add_8[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 1792)         0           tf.__operators__.add_10[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          589952      flatten_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          557184      flatten_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          262272      flatten_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 128)          245888      flatten_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 128)          229504      flatten_5[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape (TFOpLambda) (3,)                 0           input_1[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_2 (TFOpLambd (3,)                 0           input_1[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_4 (TFOpLambd (3,)                 0           input_1[0][0]
__________________________________________________________________________________________________
tf.compat.v1.shape_6 (TFOpLambd (3,)                 0           input_1[0][0]
__________________________________________________________________________________________________
tf.cast_3 (TFOpLambda)          (None, 128)          0           dense_1[0][0]
__________________________________________________________________________________________________
tf.cast_5 (TFOpLambda)          (None, 128)          0           dense_2[0][0]
__________________________________________________________________________________________________
tf.cast_7 (TFOpLambda)          (None, 128)          0           dense_3[0][0]
__________________________________________________________________________________________________
tf.cast_9 (TFOpLambda)          (None, 128)          0           dense_4[0][0]
__________________________________________________________________________________________________
tf.cast_11 (TFOpLambda)         (None, 128)          0           dense_5[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem (Slici ()                   0           tf.compat.v1.shape[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_2 (Sli ()                   0           tf.compat.v1.shape_2[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_4 (Sli ()                   0           tf.compat.v1.shape_4[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem_6 (Sli ()                   0           tf.compat.v1.shape_6[0][0]
__________________________________________________________________________________________________
tf.math.multiply_3 (TFOpLambda) (None, 128)          0           tf.cast_3[0][0]
__________________________________________________________________________________________________
tf.math.multiply_5 (TFOpLambda) (None, 128)          0           tf.cast_5[0][0]
__________________________________________________________________________________________________
tf.math.multiply_7 (TFOpLambda) (None, 128)          0           tf.cast_7[0][0]
__________________________________________________________________________________________________
tf.math.multiply_9 (TFOpLambda) (None, 128)          0           tf.cast_9[0][0]
__________________________________________________________________________________________________
```

Figure 6.1: Proposed Framework Architecture (1)



```
__________________________________________________________________________________________________
tf.math.multiply_11 (TFOpLambda (None, 128)          0           tf.cast_11[0][0]
__________________________________________________________________________________________________
tf.zeros (TFOpLambda)           (None, 128)          0           tf.__operators__.getitem[0][0]
__________________________________________________________________________________________________
tf.zeros_1 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem[0][0]
__________________________________________________________________________________________________
tf.zeros_2 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_2[0][0]
__________________________________________________________________________________________________
tf.zeros_3 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_2[0][0]
__________________________________________________________________________________________________
tf.zeros_4 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_4[0][0]
__________________________________________________________________________________________________
tf.zeros_5 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_4[0][0]
__________________________________________________________________________________________________
tf.zeros_6 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_6[0][0]
__________________________________________________________________________________________________
tf.zeros_7 (TFOpLambda)         (None, 128)          0           tf.__operators__.getitem_6[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_3 (TFOpLam (None, 128)          0           tf.math.multiply_3[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_5 (TFOpLam (None, 128)          0           tf.math.multiply_5[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_7 (TFOpLam (None, 128)          0           tf.math.multiply_7[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_9 (TFOpLam (None, 128)          0           tf.math.multiply_9[0][0]
__________________________________________________________________________________________________
tf.__operators__.add_11 (TFOpLa (None, 128)          0           tf.math.multiply_11[0][0]
__________________________________________________________________________________________________
tf.keras.backend.rnn (TFOpLambd ((None, 128), (None, 0           input_1[0][0]
                                                                 tf.zeros[0][0]
                                                                 tf.zeros_1[0][0]
__________________________________________________________________________________________________
tf.keras.backend.rnn_1 (TFOpLam ((None, 128), (None, 0           input_1[0][0]
                                                                 tf.zeros_2[0][0]
                                                                 tf.zeros_3[0][0]
__________________________________________________________________________________________________
tf.keras.backend.rnn_2 (TFOpLam ((None, 128), (None, 0           input_1[0][0]
                                                                 tf.zeros_4[0][0]
                                                                 tf.zeros_5[0][0]
__________________________________________________________________________________________________
tf.keras.backend.rnn_3 (TFOpLam ((None, 128), (None, 0           input_1[0][0]
                                                                 tf.zeros_6[0][0]
                                                                 tf.zeros_7[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 640)          0           tf.__operators__.add_3[0][0]
                                                                 tf.__operators__.add_5[0][0]
                                                                 tf.__operators__.add_7[0][0]
                                                                 tf.__operators__.add_9[0][0]
                                                                 tf.__operators__.add_11[0][0]
__________________________________________________________________________________________________
tf.concat (TFOpLambda)          (None, 256)          0           tf.keras.backend.rnn[0][0]
                                                                 tf.keras.backend.rnn_1[0][0]
__________________________________________________________________________________________________
tf.concat_1 (TFOpLambda)        (None, 256)          0           tf.keras.backend.rnn_2[0][0]
                                                                 tf.keras.backend.rnn_3[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 640)          0           concatenate[0][0]
__________________________________________________________________________________________________
tf.concat_2 (TFOpLambda)        (None, 1152)         0           tf.concat[0][0]
                                                                 tf.concat_1[0][0]
                                                                 flatten_6[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 1)            1153        tf.concat_2[0][0]
==================================================================================================
Total params: 1,890,049
Trainable params: 1,890,049
Non-trainable params: 0
```

Figure 6.2: Proposed Framework Architecture (2)

6.3 Further Analysis

To demonstrate the effectiveness of the above models, further analysis is required to measure how well each model performed. For this purpose, the following have been taken into consideration: *Confusion Matrix*, *Partial Dependence*, and *Calibration*

Curve. Furthermore, solely the best analysis of each experiment is discussed below. To see all the figures, please refer to the Appendix A.1.

6.3.1 Confusion Matrix

A confusion matrix (or mistake matrix) is a kind of table used in machine learning to visualize the results of an algorithm, usually one that employs supervised learning. The results that belong to a given real class are represented along the rows of the matrix, while the instances that belong to a given predicted class are shown along the columns. The term comes from the fact that it's simple to see whether the program is conflating two classes, i.e. commonly mislabeling one class as another.

In Figure 6.3, four of the most interesting confusion matrices with respect to a single-trial classification, are presented. These were chosen due to having maximum number of TNs and TPs in Tables 6.2-6.5. Furthermore, the confusion matrix is applied on the test cases (vividly, as the prediction data is needed to make the analysis). In this scenario, 30% of the data is used as test cases which leaves us with 6 test-cases since the total number of patients is 20. It is evident that the Decision Tree Classifier performed the best amongst the others, which demonstrates its effectiveness in the highest accuracy and f1 score. Quadratic Discriminant Analysis (QDA) stays at the second best amongst the others with a total number of 5 correct predictions. In the third rank, the Neural Network showed a more biased tendency towards the *"False"* class, which shows the lack of data for this type of classifier. Finally, Linear SVM with the most diverse labeling predicted half of the correct labels.

Figure 6.4 presents the confusion matrix from all of the trials combined. It can be observed in Figure 6.4 that more TPs and TNs are placed. As a result of having more data (more features) to lead the classifiers towards the right path. As explained in the previous paragraph, the number of patients seen in the chart is the number of our test cases. However, as described in Chapter 3, not all of the volunteers made it to the feature extraction level because of lack of sensors, lack of sufficient data, etc. In this regard, Decision Tree has the highest number of TPs and TNs with total wrong prediction of 0. It is obvious that the data is more biased towards the *"True"* class, rather than the *"False"* class. With more data to train the network, the Neural Network takes the second position with a total number of 15 correctly predicted labels. Although LSVM performed the worst within the single trial, it was less biased towards a specific label and had only four wrong labels out of 16 predictions.

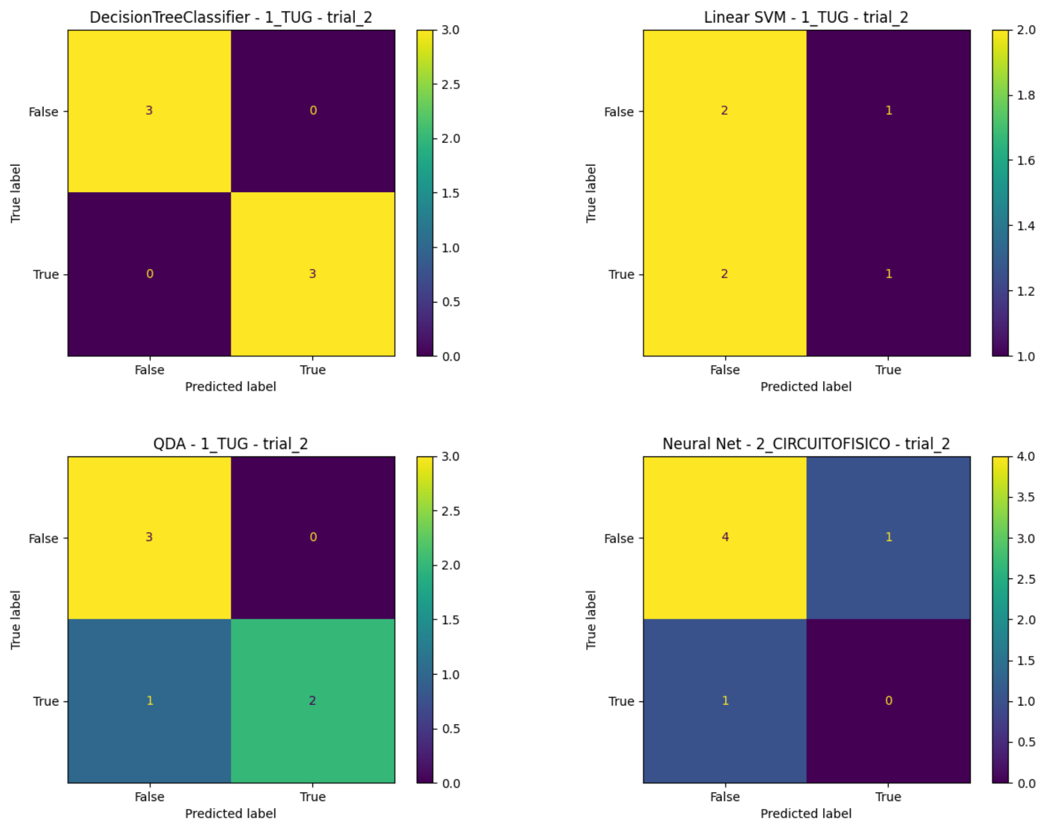


Figure 6.3: Confusion Matrices of Single Trials

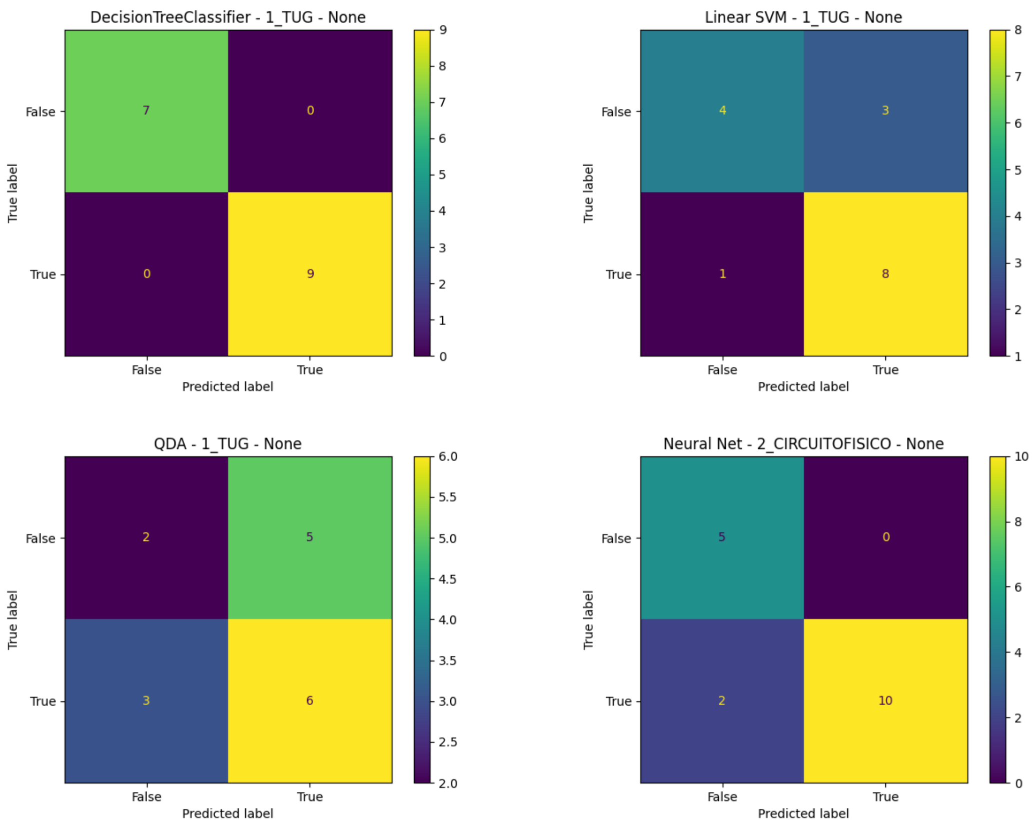


Figure 6.4: Confusion Matrices of Single Trials

6.3.2 Partial Dependence

A plot known as a *partial dependency* may be used to illustrate the functional link that exists between a few input parameters and predictions. They demonstrate how the predictions are dependent on the values of the variables that are considered to be relevant inputs. This might be seen, for instance, in a partial dependency plot, if there is a linear increase in the likelihood of FOG as one moves from experiment to experiment.

In the analysis shown in Figure 6.5, Gaussian Process demonstrated a correlation between the volunteers with FOG and the healthy individuals in the first trial of the TUG test. However, it can be observed that in the Decision Tree Classifier, dependency on the label “0” (patients without FOG) varies from the patients with FOG. Furthermore, the dependency on label “0” starts to plummet into an L-shape, while the label “1” holds its position. For more information on Partial Dependence, please refer to [38].

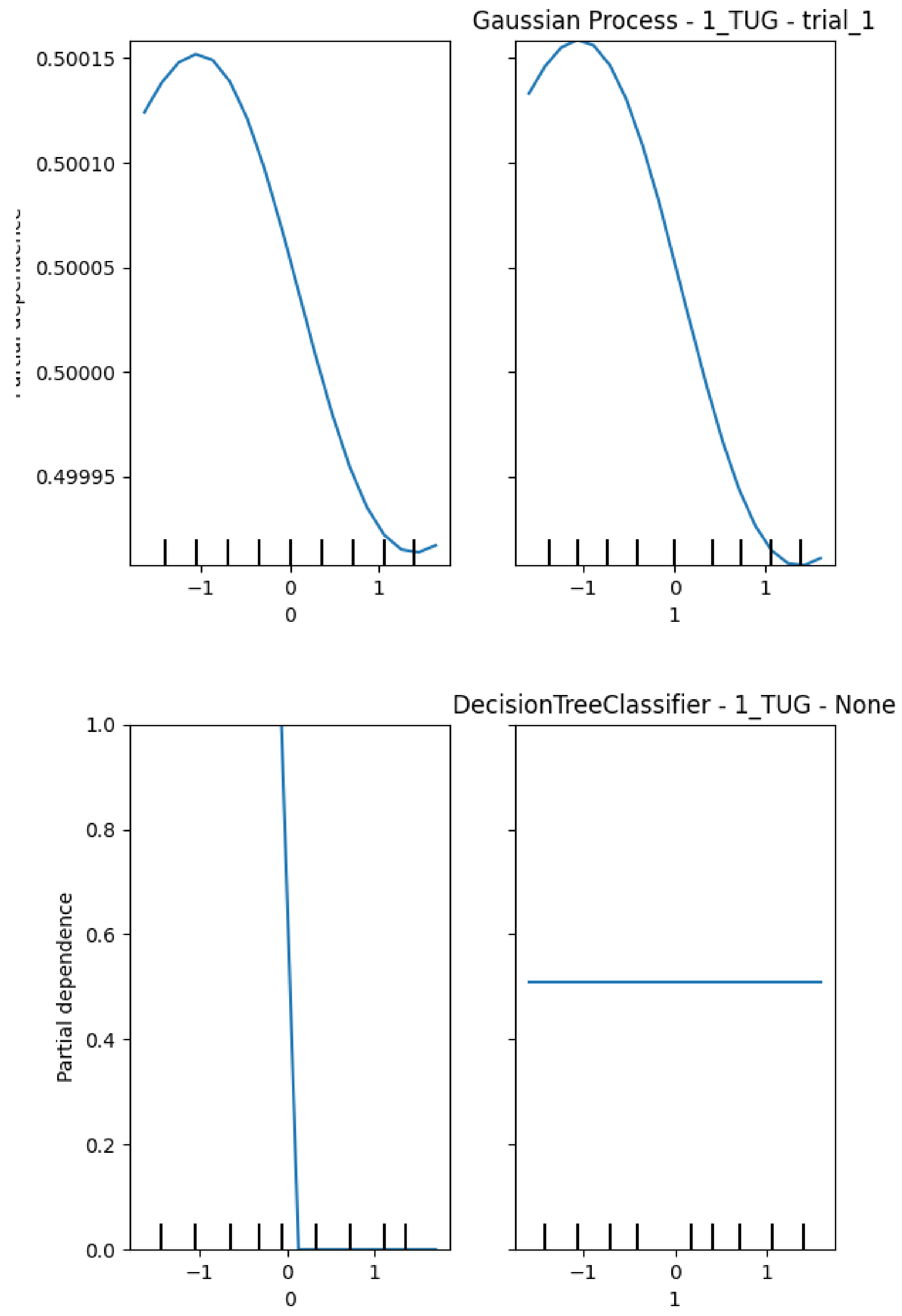


Figure 6.5: Partial Dependence

6.3.3 Calibration Curve

Comparisons between the calibrated probabilistic predictions of a binary classifier are made using *calibration curves*, commonly referred to as *reliability diagrams*. In order to make binned predictions, it displays the actual frequency of the positive label against the likelihood of that label. Each bin's average expected probability is shown on the x-axis.

In Figure 6.6, the first two figures on top are the analysis regarding the Ada Boost Classifier and Decision Tree Classifier for only one trial. In these two figures, it can be seen that the line regarding the mean predicted probability and the fraction of positives has a divergence from the calibrated line (dotted-line). However, when increasing the number of samples, the consistency on the alignment of both lines increases, and depicts the two bottom figures. In these bottom figures, it can be seen that the Decision Tree Classifier had shown a perfect calibration along with the predicted probability of the outputs, which shows that the expected outputs (produced from the DTC model) are aligned with the actual classes.

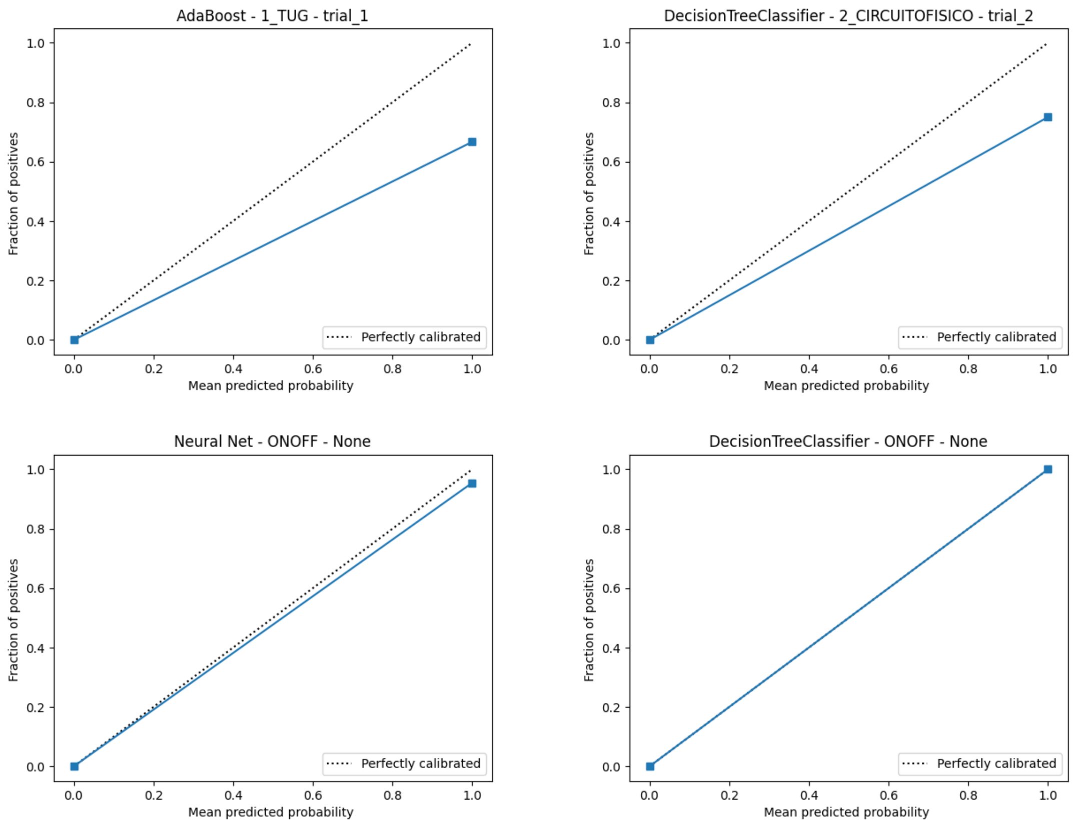


Figure 6.6: Calibration Curve

6.3.4 Analysis on the proposed framework

For the proposed framework, there exist a few more analyses to take overfitting and underfitting into consideration. First, Figure 6.7 shows the chart of epoch-accuracy. In this chart, in increasing the epochs, the accuracy goes close to 1, and after the 10th epoch, the accuracy gets closest to 1. For this matter, the orange line demonstrates the training phase, and the blue line demonstrates the validation phase. Hence, it can be seen that during the validation phase, the accuracy stays consistent, and the model is not overfitted by the training data.

Using the same analogy, it can be observed that the F1-score and precision measurement, shown in Figures 6.8 and 6.10, respectively, have also depicted consistent behavior, and the validation on both of them stayed persistent. Furthermore, In Figure 6.9 the error rate (loss) drops close to 0 after the 8th epoch. In Figure 6.11 the chart shows a plummet in loss based on the iterations. Note that the number of iterations is the number of data batches that the algorithm has observed, or simply the number of passes that the programme has done on the dataset, while epoch refers to the number of times that the algorithm has passed through the whole dataset.

6.4 Computational Power

The preprocessing phase of the classifications in this study was the bottleneck for the computational power and an expensive task regarding the processing time. It required a great deal of CPU resources to be able to extract the features, mostly when the features exceeded 20. Also, the extracted features needed to be stored in a cache prior to being stored in a file on the hard drive, as the history of features is needed for the new feature extraction. Of course, with a limited number of features, this should not be a problem, but not when it comes to features exceeding 1,000. Moreover, the process of training the proposed framework requires not just a significant amount of CPU and RAM, but also a GPU. For this purpose, this study has been done using a machine with CPU of AMD Ryzen 9 5900X 12-Core Processor (24 virtual cores), 64 GB of DDR4 RAM, and a GPU of Nvidia 3080 ti. It is worth noting that the processes still have to go through a concurrency mechanism, and many batch computations have been applied.

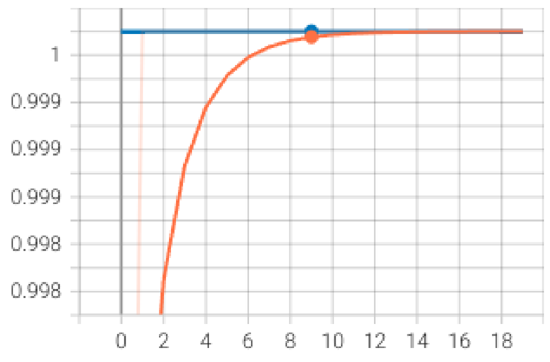


Figure 6.7: Epoch - Accuracy

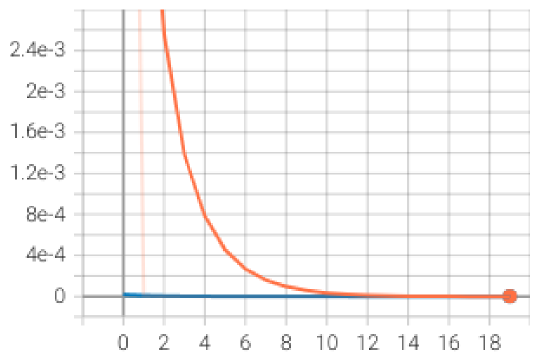


Figure 6.9: Epoch - Loss

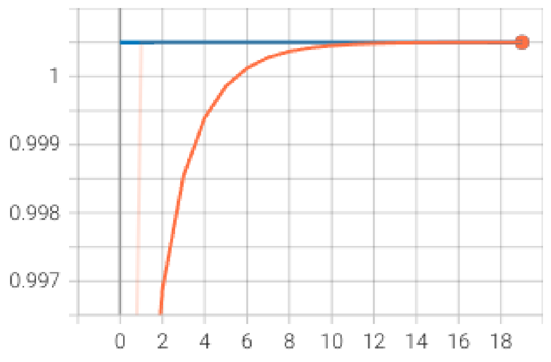


Figure 6.8: Epoch - F1 score

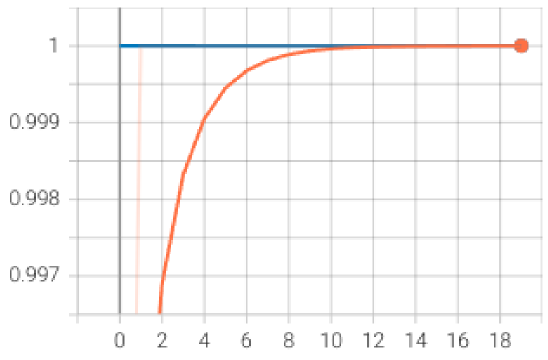


Figure 6.10: Epoch - Precision

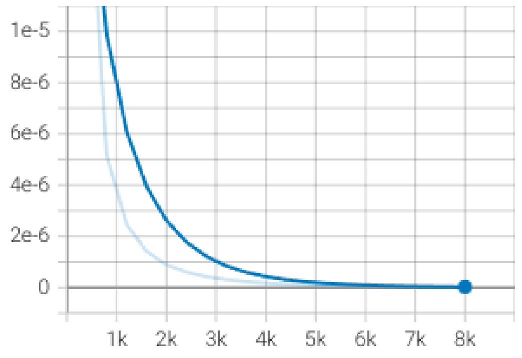


Figure 6.11: Loss - Iteration

Figure 6.12: Framework Analysis

Chapter 7

Conclusion

7.1 Conclusion

This study used a dataset, described in Chapter 3, that contains records from various experiments and trials that were conducted on both PD patients and healthy volunteers. The purpose of this was to illustrate the distinguishing characteristics of PD from those of healthy individuals, including with regard to the ON and OFF states for medication that treats FOG. It was claimed in [13] that there are many unsolved concerns that remain concerning the ON state of FOG. This supplied the motivation for the compilation of such a dataset, which was used.

When analysing this data, the initial experiments that were addressed were concerned with distinguishing people with Parkinson's disease from healthy persons and assessing the degree to which medicine was helpful. Several different classifiers were created and studied so that the aforementioned challenges may be solved. Despite the fact that the majority of the classifiers did a good job using the extracted features, the multilayer perceptron exhibited a more consistent outcome for both challenges. This was because it was able to cope with the dimensions of the features. Notably, the decision tree approach succeeded well for both of the issues. When looking at the outcomes of individual tasks, the Timed Up and Go (TUG) challenge produced the best overall results. Four of the classifiers were able to get perfect scores on each and every one of the metrics in this work.

Moreover, further experiments were conducted on distinguishing FOG episodes in a FOG+ individual from normal activities. Three experimental evaluations were performed to assess the efficacy of classifiers in the context of FOG classification, with a focus on determining the impact of feature quantity on classifier performance and accuracy. The first experiment utilized all the extracted features, resulting in the highest scores and the lowest processing speed. The second experiment employed a single feature, resulting in the lowest scores and the highest processing speed. Finally, the last experiment employed a subset of features, yielding a balance of favorable scores and processing speed. These experiments were initiated with traditional methods whilst using feature extraction techniques and various classifiers, and led to a framework to predict the FOG episodes in real time. In this framework, a state-of-the-art approach was proposed that aligns with the records received from the sensors and makes the prediction based on them. The results and analysis depicted a promising solution for detecting FOG in real time for each individual patient. Hence, this led the author to further challenges.

7.2 Future Work

In the current research, real-time prediction of FOG has been studied. However, the same prediction can also be made prior to the occurrence of FOG. By detecting FOG features and alerting patients ahead of time, the patients can prevent further accidents that may face them due to the freezing, such as falling down, car accidents, crossing the road, etc.

Additionally, conducting a patient-wise analysis could provide valuable insights into the individual variability of Parkinson's disease symptoms and their response to treatment. By focusing on each patient separately, the unique gait characteristics and the effectiveness of the different medications could be evaluated and compared. This information could assist in tailoring personalized treatment plans and enhancing the overall management of the disease. A patient-wise analysis may also give an opportunity to further understand the patient's symptoms, leading to more effective treatment strategies in the future.

Furthermore, feature importance can be calculated to better understand gait impact in Parkinson's disease. This can be achieved through methods such as feature selection, dimensionality reduction, and model-based methods. The results can provide valuable insight into the most effective features for analysis and the underlying biological mechanisms of the disease.

Moreover, further comprehensive analysis on subsets of features in detecting FOG episodes can be applied. Furthermore, the above analysis, methods, and the proposed framework can be applied to other datasets which are based on time-series records. These datasets can be applied against PD speech records, sensors on various parts of the body, etc. Moreover, the current parameters and settings used for the classifiers can be tuned more. Since the number of classifiers and their parameters are too large to be fine-tuned, the best classifiers (based on their scores) can be chosen and further updates on their calculations can be done. Using the same methodology, it could also be applied to other diseases with the same data type, such as heart attack.

These are a number of the challenges that can be addressed in future works.

Bibliography

[1] AANS. Parkinson's disease. `https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Parkinsons-Disease`. Accessed: 07-09-2022.

[2] Q.J Almeida and C.A Lebold. Freezing of gait in parkinson's disease: A perceptual cause for a motor impairment? *J. Neurol. Neurosurg. Psychiatry*, 81:513–518, 2010.

[3] A.O. Andrade et al. Pelvic movement variability of healthy and unilateral hip joint involvement individuals. *Biomed. Signal Process. Control*, 32:10–19, 2017.

[4] M.C Ariana and d.O.A Adriano. NIATS Cheat Sheet: Sistema de Monitoramento de Distúrbios do Movimento. *Zenodo*, mar 2020.

[5] E. Balaji, D. Brindha, and R. Balakrishnan. Supervised machine learning based gait classification system for early detection and stage classification of parkinson's disease. *Applied Soft Computing*, 94:106494, 2020.

[6] A.L. Bartels et al. Relationship between freezing of gait (fog) and other features of parkinson's: Fog is not correlated with bradykinesia. *Clin. Neurosci*, 10(10):584–588, 2003.

[7] E.N. Beck, K.A. Ehgoetz Martens, and Q.J. Almeida. Freezing of gait in parkinson's disease: an overload problem? *PloS one*, 10(12):e0144–986, 2015.

[8] O.M Beigi, L.Reis Nóbrega, S. Houghten, A. de Oliveira Andrade, and A.A Pereira. Classification of parkinson's disease patients and effectiveness of medication for freezing of gait. In *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8, 2022.

[9] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[10] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[11] J. Brownlee. A gentle introduction to k-fold cross-validation. `https://machinelearningmastery.com/k-fold-cross-validation/`. Accessed: 07-09-2022.

[12] M. D. Buhmann. Radial basis functions. *Acta Numerica*, 9:1–38, 2000.

[13] R. Chen. Paradoxical worsening of gait with levodopa in parkinson disease. *Neurology*, 78(7):446–447, 2012.

[14] T.T da Costa Capato, J.M.M Domingos, and L.R.S de Almeida. Versão em português da diretriz europeia de fisioterapia para a doença de parkinson: Desenvolvida por vinte associações profissionais europeias e adaptada para português europeu e do brasil. *Editora e Eventos Omini Farma*, 2015.

[15] E.R. Dorsey et al. Projected number of people with Parkinson disease in the most populous nations, 2005 through 2030. *Neurology*, 68(5):384–386, 2007.

[16] P. Drotár et al. Analysis of in-air movement in handwriting: A novel marker for parkinson's disease. *Computer methods and programs in biomedicine*, 117(3):405–411, 2014.

[17] P. Drotár, J. Mekyska, I. Rektorová, L. Masarová, Z. Smékal, and M. Faundez-Zanuy. Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease. *Artificial intelligence in Medicine*, 67:39–46, 2016.

[18] A. Elkurdi et al. Gait speeds classifications by supervised modulation based machine-learning using kinect camera. *Medical Research and Innovations*, 2(4):1–6, 2018.

[19] Ö. Eskidere, F. Ertaş, and C. Hanilçi. A comparison of regression methods for remote tracking of parkinson's disease progression. *expert systems with applications*, 39(5):5523–5528, 2012.

[20] M. Christ et al. Overview on extracted features. `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`. Accessed: 07-09-2022.

[21] M. Christ et al. Tsfresh. `https://tsfresh.readthedocs.io/`. Accessed: 07-09-2022.

[22] Data Flair. Kernel functions-introduction to svm kernel and examples. `https://data-flair.training/blogs/svm-kernel-functions`. Accessed: 07-09-2022.

[23] Y. Freund and R.E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[24] K. Fukushima. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol, Cybern*, 36:193–202, 1980.

[25] F.A Gers, N.N Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3(Aug):115–143, 2002.

[26] T. Hastie, Sa. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[27] J.J. Hathaliya et al. Parkinson and essential tremor classification to identify the patient's risk based on tremor severity. *Computers and Electrical Engineering*, 101:107946, 2022.

[28] J.M. Hausdorff et al. Impaired regulation of stride variability in parkinson's disease subjects with freezing of gait. *Experimental Brain Research*, 149:187–194, 2003.

[29] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A field guide to dynamical recurrent neural networks IEEE Press In*, 2001.

[30] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[31] D.H Hubel and T.N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[32] J. Hughes. Symbolic regression gp system made in java. `https://github.com/convergencelab/jGP`. Accessed: 07-09-2022.

[33] J.A. Hughes, J.A. Brown, and A.M. Khan. Smartphone gait fingerprinting models via genetic programming. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 408–415. IEEE, 2016.

[34] J.A. Hughes, S. Houghten, and J.A. Brown. Descriptive symbolic models of gaits from parkinson's disease patients. In *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. IEEE, 2019.

[35] J.A. Hughes, S. Houghten, and J.A. Brown. Models of parkinson's disease patient gait. *IEEE Journal of Biomedical and Health Informatics*, 24(11):3103–3110, 2019.

[36] IBM. What are neural networks? https://www.ibm.com/cloud/learn/neural-networks. Accessed: 07-09-2022.

[37] M. Isenkul, B. Sakar, O. Kursun, et al. Improved spiral test using digitized graphics tablet for monitoring parkinson's disease. In *Proc. of the Int'l Conf. on e-Health and Telemedicine*, pages 171–5, 2014.

[38] I.K Kabul. Interpret model predictions with partial dependence and individual conditional expectation plots. https://blogs.sas.com/content/subconsciousmusings/2018/06/12/interpret-model-predictions-with-partial-dependence-and-individual-conditional-expectation-plots/. Accessed: 07-09-2022.

[39] C.M Leite and S.R.F Rosa. Novas tecnologias aplicadas à saúde: integração de áreas transformando a sociedade. 2017.

[40] R. López-Blanco et al. Essential tremor quantification based on the combined use of a smartphone and a smartwatch: The netmd study. *Journal of Neuroscience Methods*, 303:95–102, 2018.

[41] Q.T. Ly et al. Detection of turning freeze in parkinson's disease based on s-transform decomposition of eeg signals. *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 81:3044–3047, 2017.

[42] I. Mileti et al. Gait partitioning methods in parkinson's disease patients with motor fluctuations: A comparative analysis. *IEEE International Symposium on Medical Measurements and Applications*, page 402–407, 2017.

[43] A. Mirelman et al. Gait impairments in parkinson's disease. *The Lancet Neurology*, 18(7):697–708, 2019.

[44] S. Moon et al. Classification of parkinson's disease and essential tremor based on balance and gait characteristics from wearable motion sensors via machine learning techniques: a data-driven approach. *Journal of NeuroEngineering and Rehabilitation*, 17(1):1–8, 2020.

[45] S.T Moore, H.G MacDougall, and W.G Ondo. Ambulatory monitoring of freezing of gait in parkinson's disease. *Neurosci. Methods*, 167:340–348, 2008.

[46] Y. Amirat L. Oukhellou S. Mohammed N. Khoury, F. Attal. Data-driven based approach to aid parkinson's disease diagnosis. *Sensors 19*, 2:1–27, 2019.

[47] M.H Nilsson and P. Hagell. Freezing of gait questionnaire: Validity and reliability of the swedish version. *Acta Neurol. Scand*, 120:331–334, 2009.

[48] R.L. Nussbaum and C.E. Ellis. Alzheimer's disease and parkinson's disease. *New england journal of medicine*, 348(14):1356–1364, 2003.

[49] Lígia Reis Nóbrega. Example of dataset - gyroscope - z axis. `https://zenodo.org/record/6800545`. Accessed: 07-09-2022.

[50] L. Hardesty — MIT News Office. Explained: Neural networks. `https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414`. Accessed: 07-09-2022.

[51] A.B Oktay and A. Kocer. Differential diagnosis of parkinson and essential tremor with convolutional lstm networks. *Biomedical Signal Processing and Control*, 56:101683, 2020.

[52] Y. Okuma. Freezing of gait and falls in parkinson's disease. *J. Parkinsons. Dis.*, 4(2):255–260, 2014.

[53] Y. Okuma. Practical approach to freezing of gait in parkinson's disease. *Practical neurology*, 14(4):222–230, 2014.

[54] Pandas. Pandas. `https://pandas.pydata.org/`. Accessed: 07-09-2022.

[55] L. Parisi, N. RaviChandran, and M.L. Manaog. Feature-driven machine learning to improve early diagnosis of parkinson's disease. *Expert Systems with Applications*, 110:182–190, 2018.

[56] D. Podsiadlo and S. Richardson. The timed "up & go": a test of basic functional mobility for frail elderly persons. *Journal of the American geriatrics Society*, 39(2):142–148, 1991.

[57] A. Saad, F. Guerin, I. Zaarour, M. Ayache, and D. Lefebvre. Sensoring and features extraction for the detection of freeze of gait in parkinson disease. *2014 IEEE 11th International Multi-Conference on Systems, Signals Devices*, 2014.

[58] A.J.P Salgueiro, Y. Shichkina, A. García, and L.G Rodríguez. Parkinson's disease classification and medication adherence monitoring using smartphone-based gait assessment and deep reinforcement learning algorithm. *Procedia Computer Science*, 186:546–554, 2021.

[59] Idan Schatz. Using the gini coefficient to evaluate the performance of credit score models. https://towardsdatascience.com/using-the-gini-coefficient-to-evaluate-the-performance-of-credit-score-models-59fe13ef420. Accessed: 07-09-2022.

[60] ScikitAuthorTeam. Decision tree classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`. Accessed: 07-09-2022.

[61] ScikitAuthorTeam. Decision tree regressor. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html`. Accessed: 07-09-2022.

[62] ScikitAuthorTeam. Ensemble. `https://scikit-learn.org/stable/modules/ensemble.html#ensemble`. Accessed: 07-09-2022.

[63] ScikitAuthorTeam. Extra tree classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeClassifier.html`. Accessed: 07-09-2022.

[64] ScikitAuthorTeam. Extra tree regressor. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html`. Accessed: 07-09-2022.

[65] ScikitAuthorTeam. K-nearest neighbors. `https://scikit-learn.org/stable/modules/neighbors.html?highlight=knn`. Accessed: 07-09-2022.

[66] ScikitAuthorTeam. Linear svc. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html?highlight=svm`. Accessed: 07-09-2022.

[67] ScikitAuthorTeam. Naive bayes. `https://scikit-learn.org/stable/modules/naive_bayes.html?highlight=naive+bayes`. Accessed: 07-09-2022.

[68] ScikitAuthorTeam. Nearest neighbors. `https://scikit-learn.org/stable/modules/neighbors.html`. Accessed: 07-09-2022.

[69] ScikitAuthorTeam. Random forest classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=ensemble`. Accessed: 07-09-2022.

[70] ScikitAuthorTeam. Support vector machines. `https://scikit-learn.org/stable/modules/svm.html#svm`. Accessed: 07-09-2022.

[71] S. Shetty and Y.S. Rao. Svm based machine learning approach to identify parkinson's disease using gait analysis. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–5. IEEE, 2016.

[72] B. Sijobert, C. Azevedo, D. Andreu, C. Verna, and C. Geny. Effects of sensitive electrical stimulation-based somatosensory cueing in Parkinson's disease gait and freezing of gait assessment. *Artif. Organs*, 41:E222–E232, 2017.

[73] S Sivaranjini and CM Sujatha. Deep learning based diagnosis of parkinson's disease using convolutional neural network. *Multimedia Tools and Applications*, 79(21):15467–15479, 2020.

[74] S. Sveinbjornsdottir. The clinical symptoms of parkinson's disease. *Neurochem*, page 318–324, 2016.

[75] Y. Wang et al. Freezing of gait detection in Parkinson's disease via multimodal analysis of eeg and accelerometer signals. *Proc. Annu. Int. Conf. IEEE Eng. Med Biol. Soc. EMBS*, page 847–850, 2020.

[76] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J McLachlan, A. Ng, B. Liu, P.S Yu, et al. Top 10 algorithms in data mining. *Knowledge and Information System*, 14:1–37, 2007.

[77] E. Zirek, B. Ersoz Huseyinsinoglu, Z. Tufekcioglu, B. Bilgic, and H. Hanagasi. Which cognitive dual-task walking causes most interference on the timed up and go test in parkinson's disease: a controlled study. *Neurol. Sci*, 39:2151–2157, 2018.

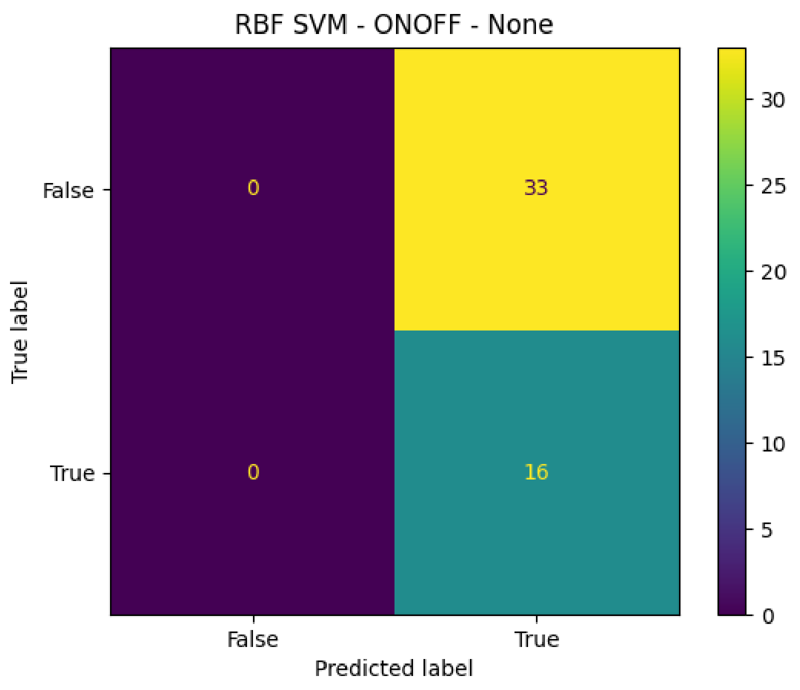
[78] Z. Yang Z.R. Yang. Artificial neural network. `https://www.sciencedirect.com/topics/neuroscience/artificial-neural-network`. Accessed: 07-09-2022.

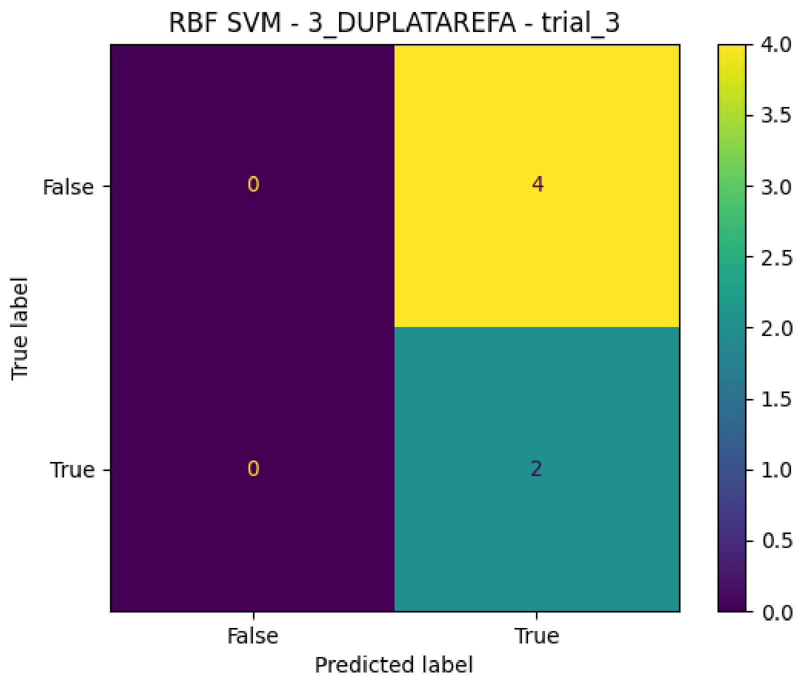
Appendix A

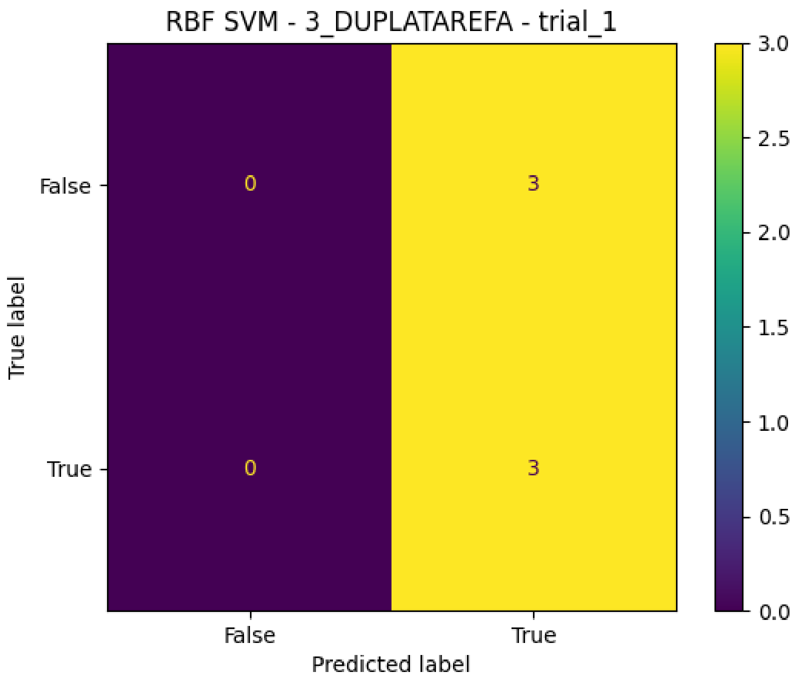
Additional Experimental Analysis

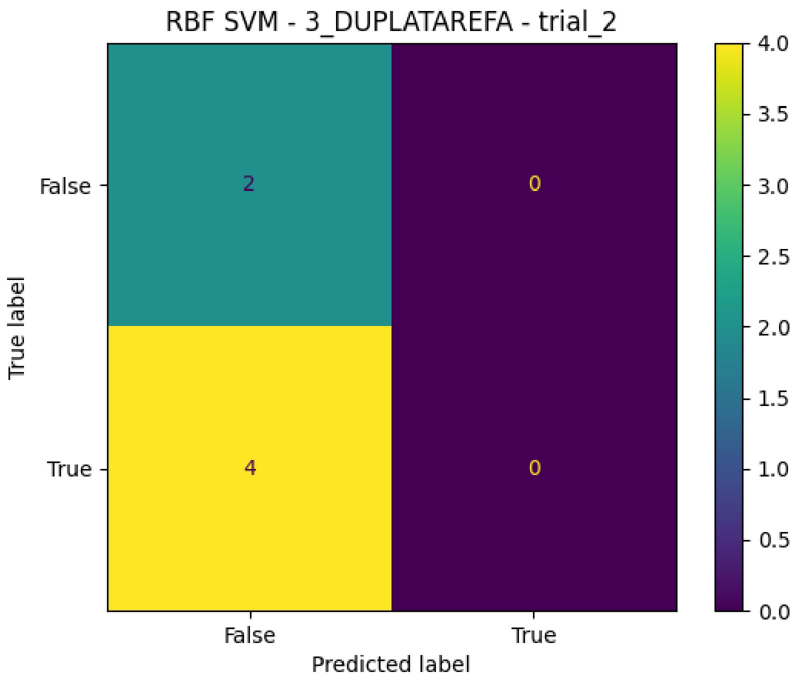
Additional figures of Confusion Matrices, Calibration Curve, and Partial Dependency. A full illustration of the Confusion Matrix analysis is depicted in Section A.1. The figures related to the Partial Dependency is shown in Section A.3. Finally, Calibration Curve is analyzed and shown in Section A.2. It is worth noting that the caption of the figures follows the *"{Classifier's name} + "-" + {Experiment's number} + {Experiment's name} + {trial number}"* convention. Please note that *None* refers to the combination of all of the trials.

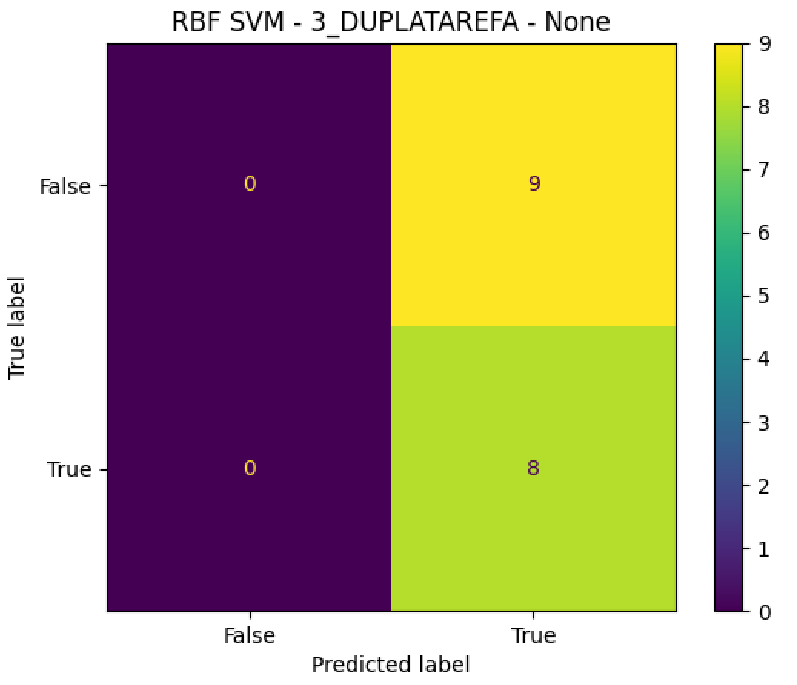
A.1 Confusion Matrix Analysis

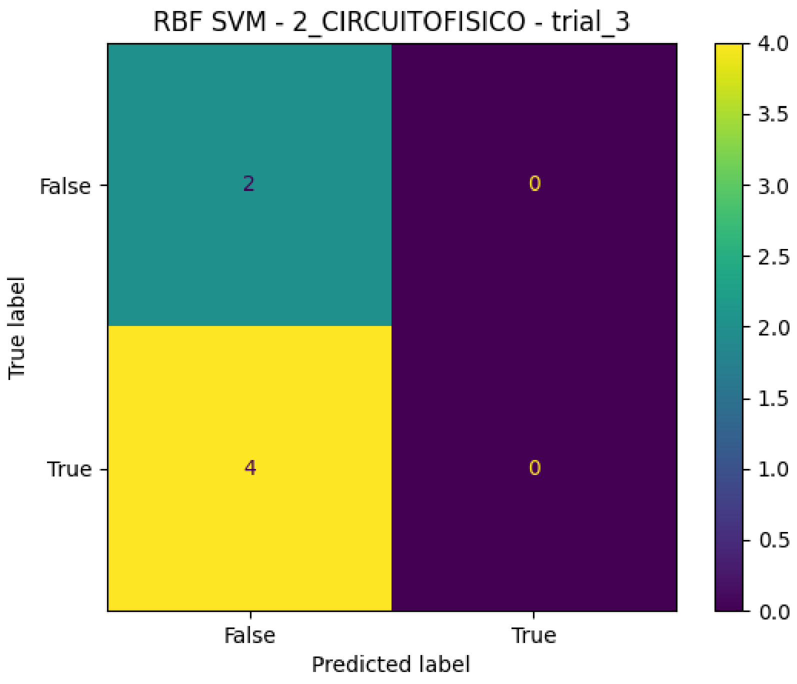


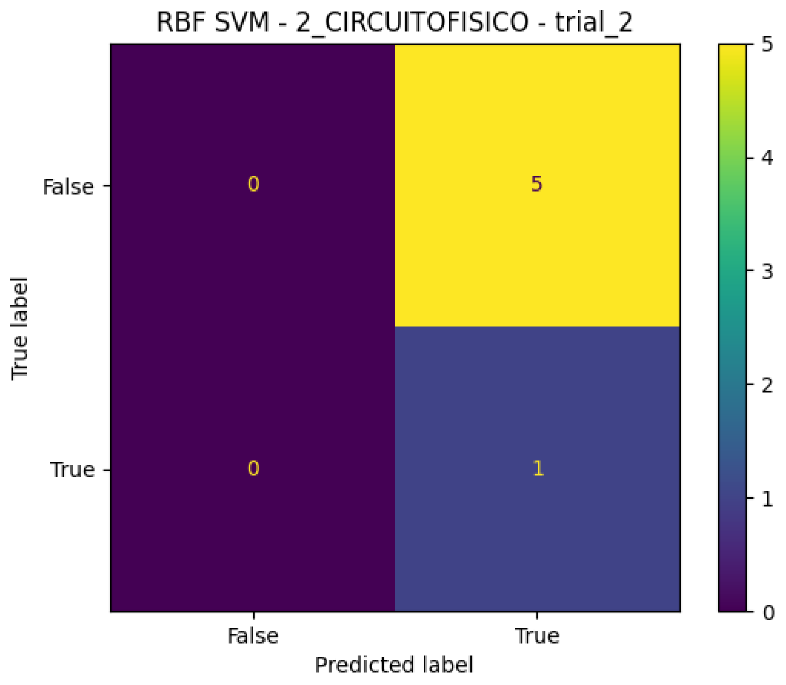


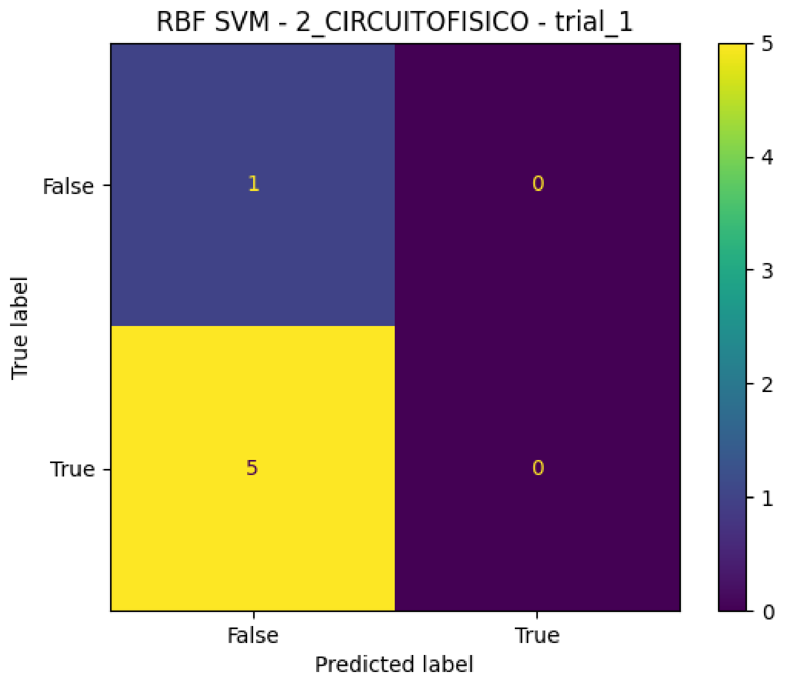

RBF SVM - 3_DUPLATAREFA - trial_1
True label
False
True
0
3
0
3
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

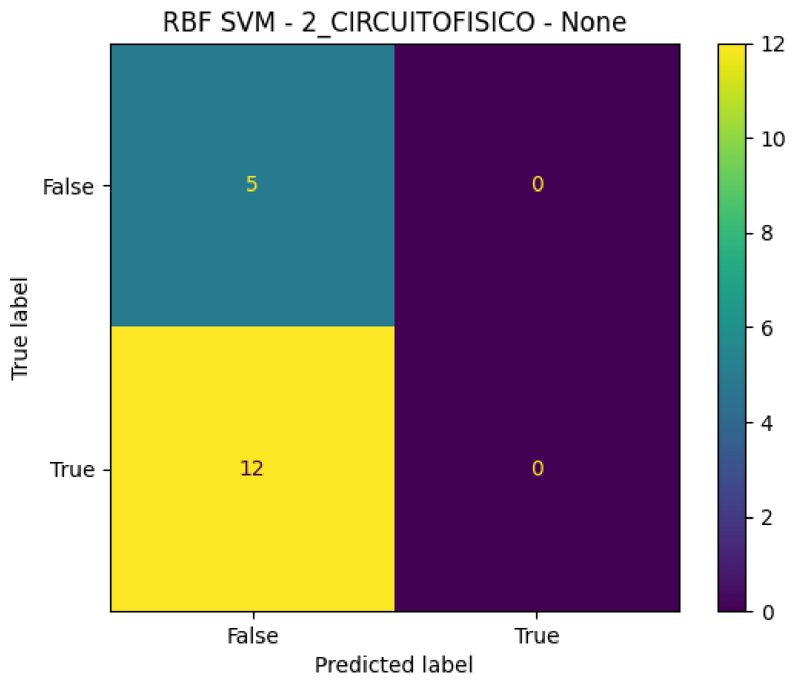

RBF SVM - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

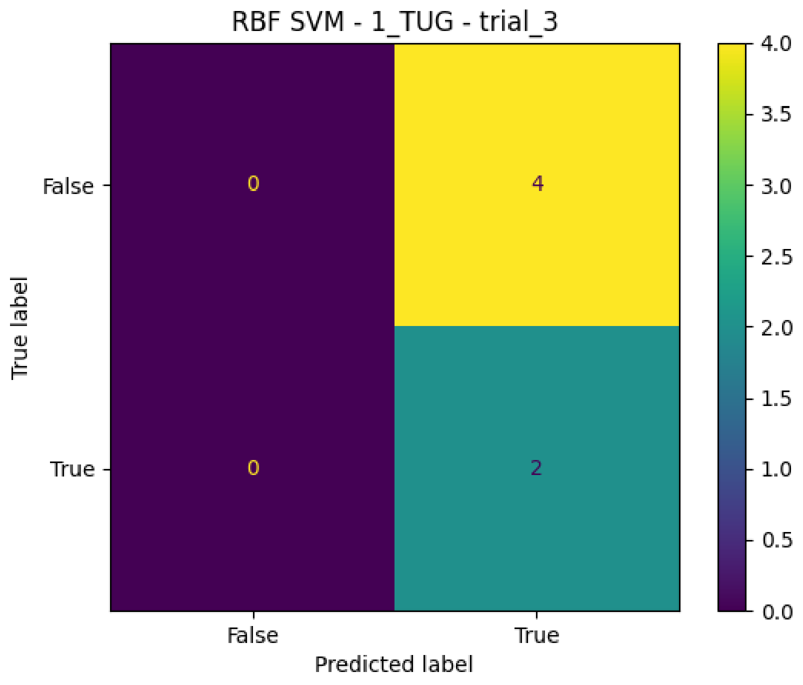

RBF SVM - 3_DUPLATAREFA - None
True label
False
True
0
9
0
8
False
True
Predicted label
9
8
7
6
5
4
3
2
1
0

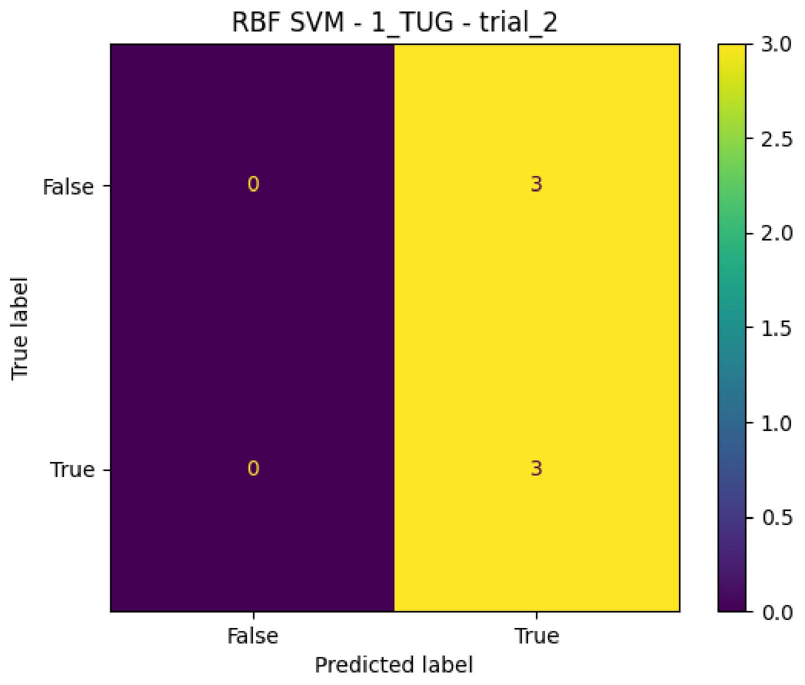

RBF SVM - 2_CIRCUITOFISICO - trial_3
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

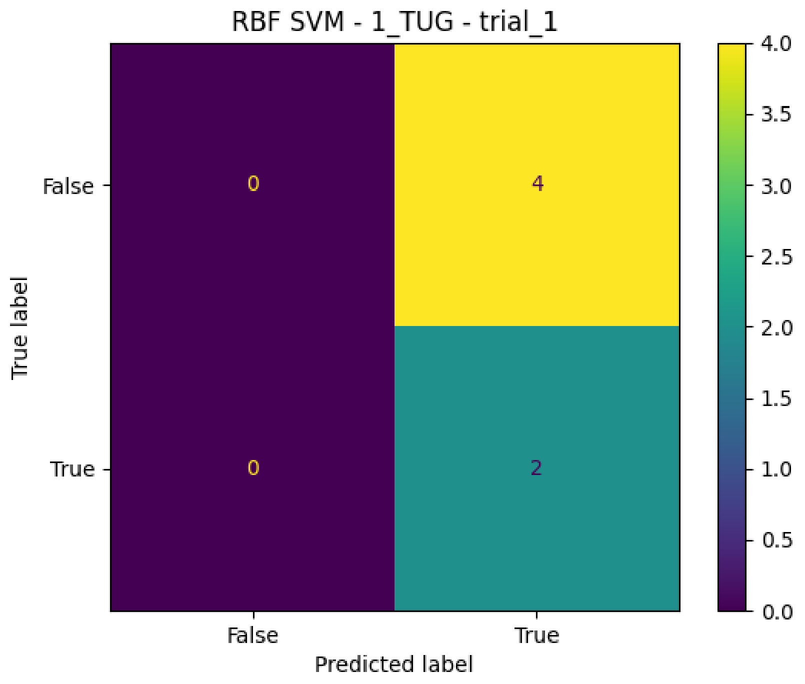

RBF SVM - 2_CIRCUITOFISICO - trial_2
True label
False
True
Predicted label
0
5
0
1
5
4
3
2
1
0

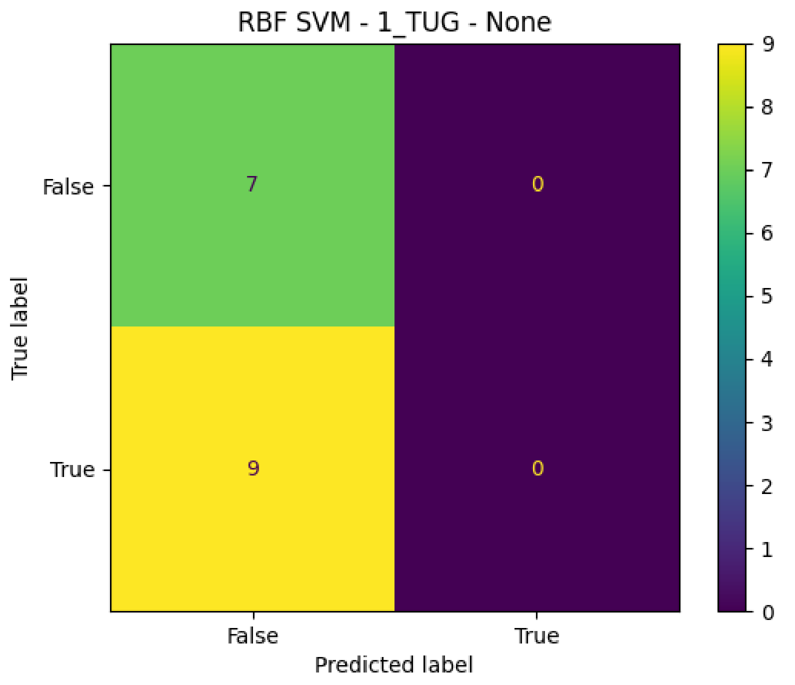

RBF SVM - 2_CIRCUITOFISICO - trial_1
True label
False
True
Predicted label
1
0
5
0
5
4
3
2
1
0

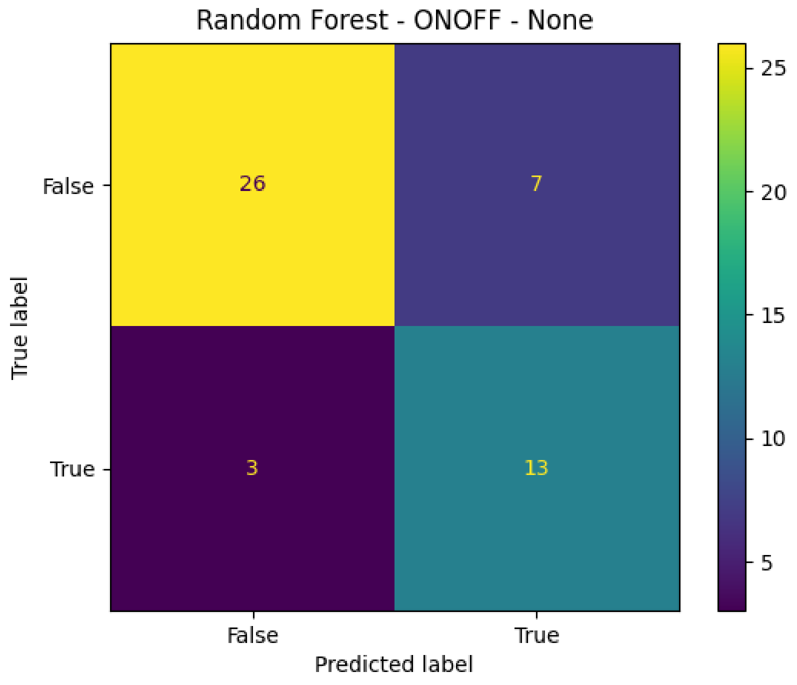

RBF SVM - 2_CIRCUITOFISICO - None
True label
Predicted label
False
True
5
0
12
0
12
10
8
6
4
2
0

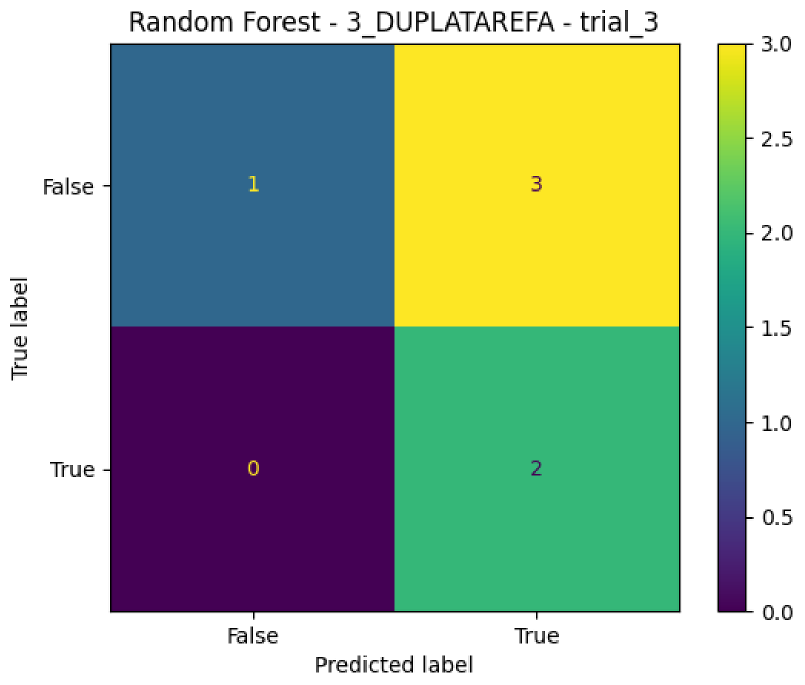

RBF SVM - 1_TUG - trial_3
True label
Predicted label
False
True
0
4
0
2
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

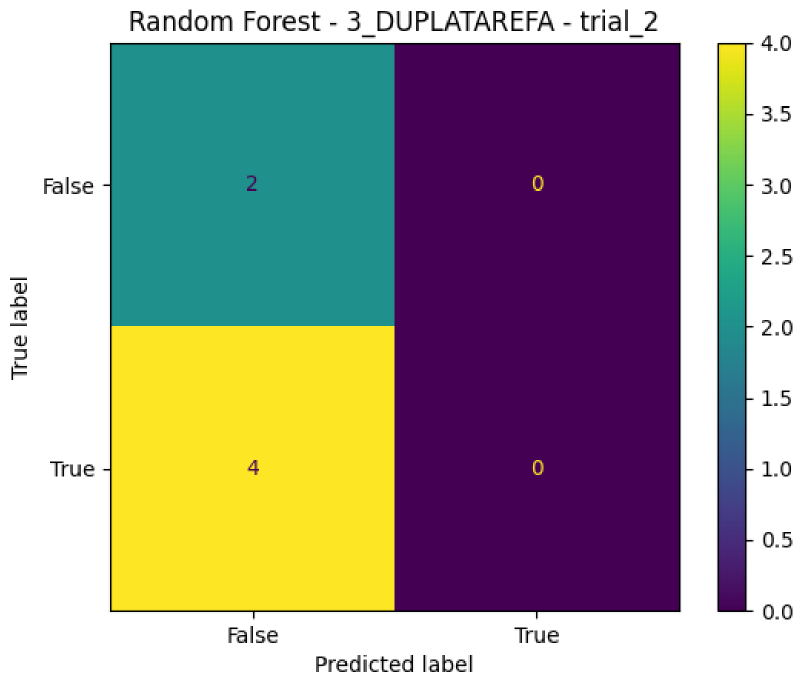

RBF SVM - 1_TUG - trial_2
True label
False
True
Predicted label
False
True
0
3
0
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

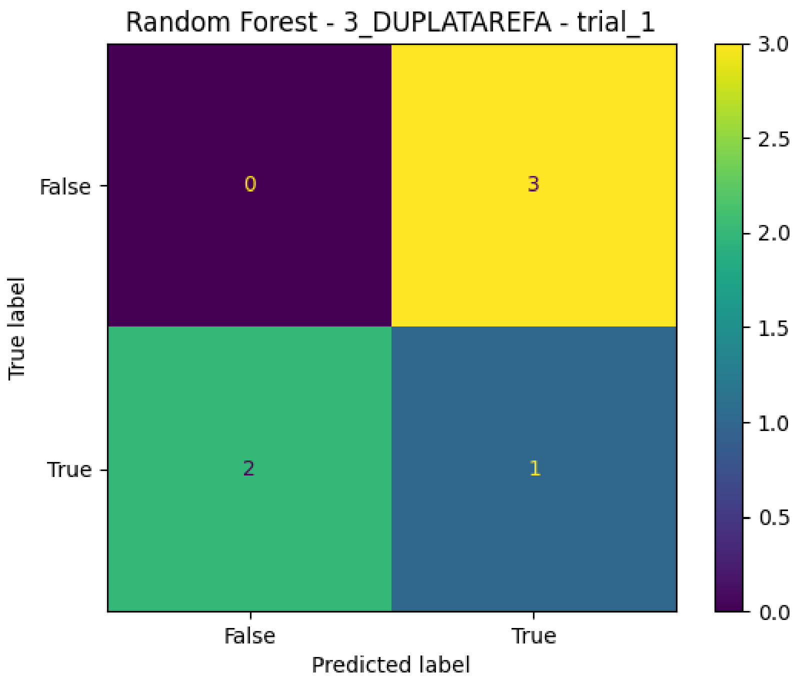

RBF SVM - 1_TUG - trial_1
True label
False
True
Predicted label
False
True
0
4
0
2
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

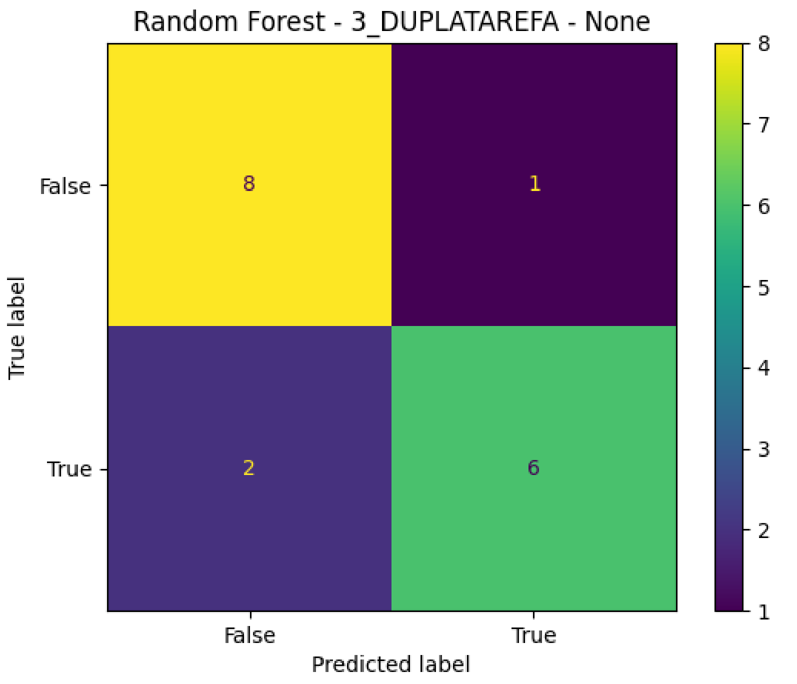

RBF SVM - 1_TUG - None
True label
False
True
7
0
9
0
Predicted label
0
1
2
3
4
5
6
7
8
9

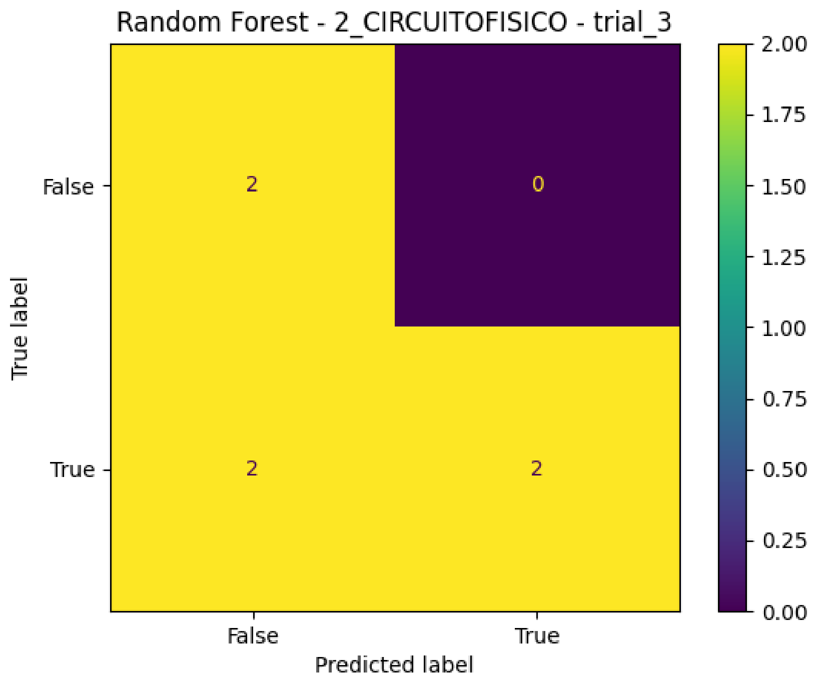

Random Forest - ONOFF - None
True label
False
True
26
7
3
13
Predicted label
5
10
15
20
25

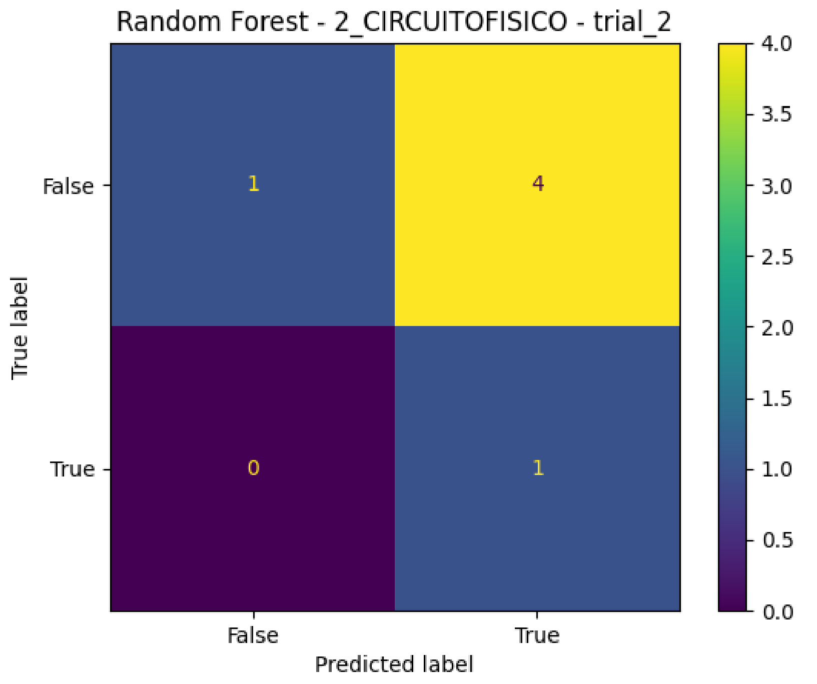

Random Forest - 3_DUPLATAREFA - trial_3
True label
False
True
1
3
0
2
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

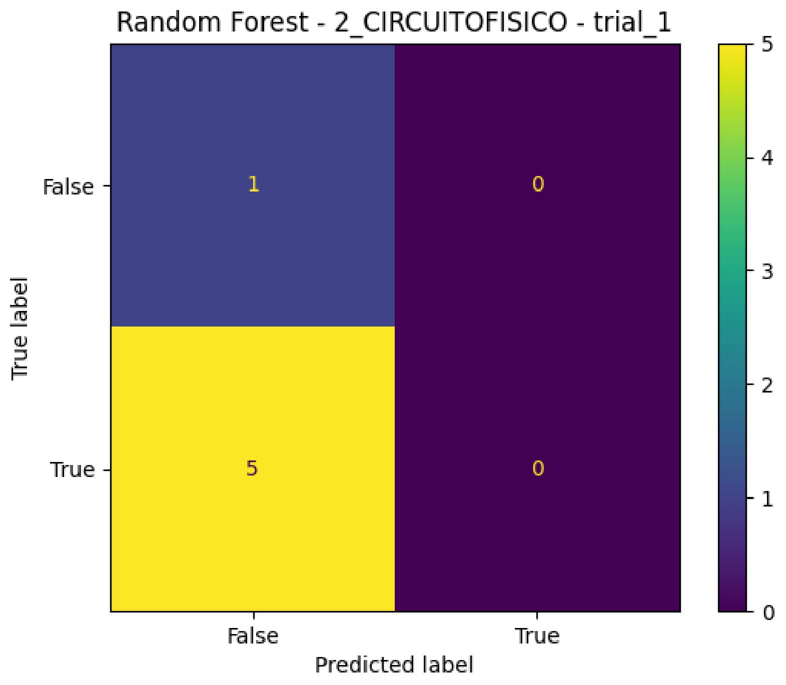

Random Forest - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
4
0
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

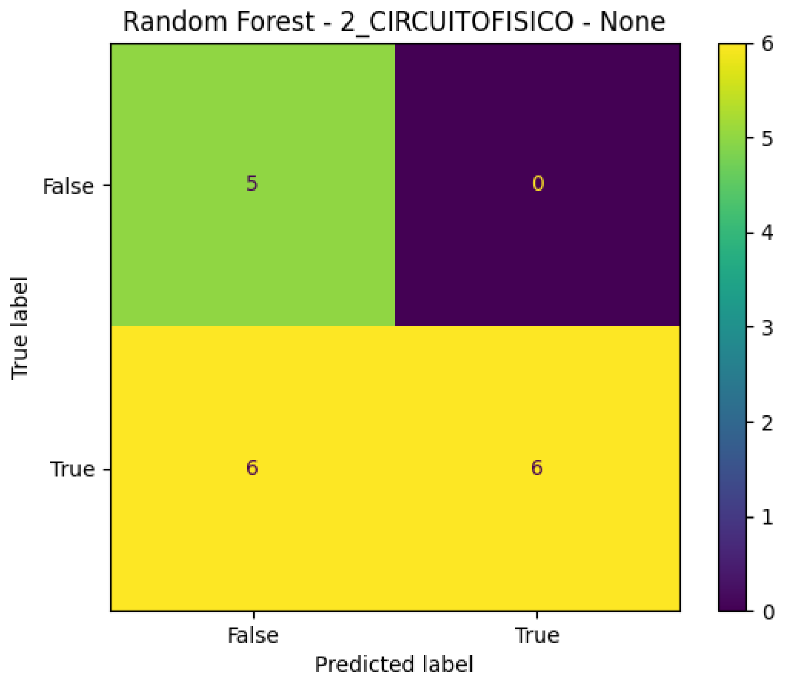

Random Forest - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
0
3
2
1
0.0
0.5
1.0
1.5
2.0
2.5
3.0

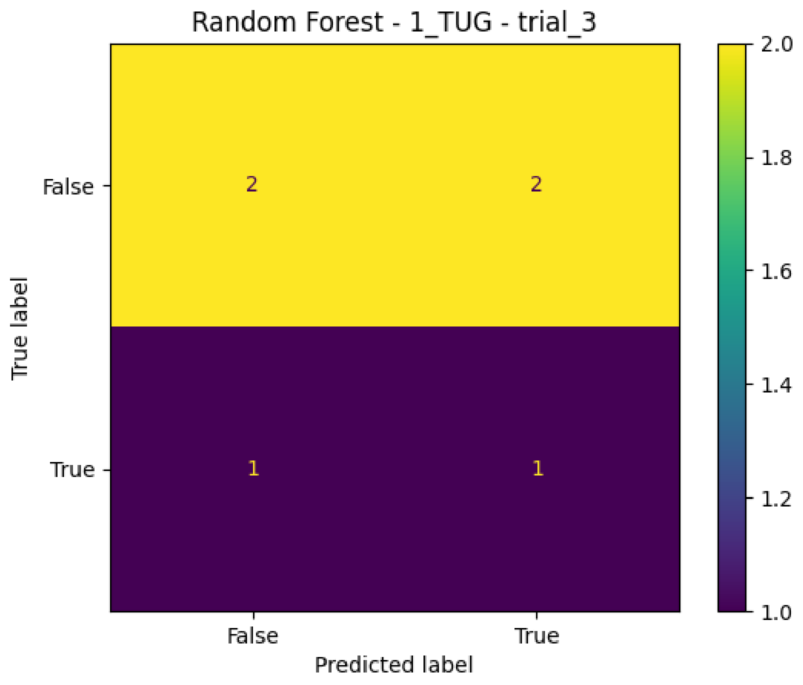

Random Forest - 3_DUPLATAREFA - None
True label
False
True
Predicted label
8
1
2
6
1
2
3
4
5
6
7
8

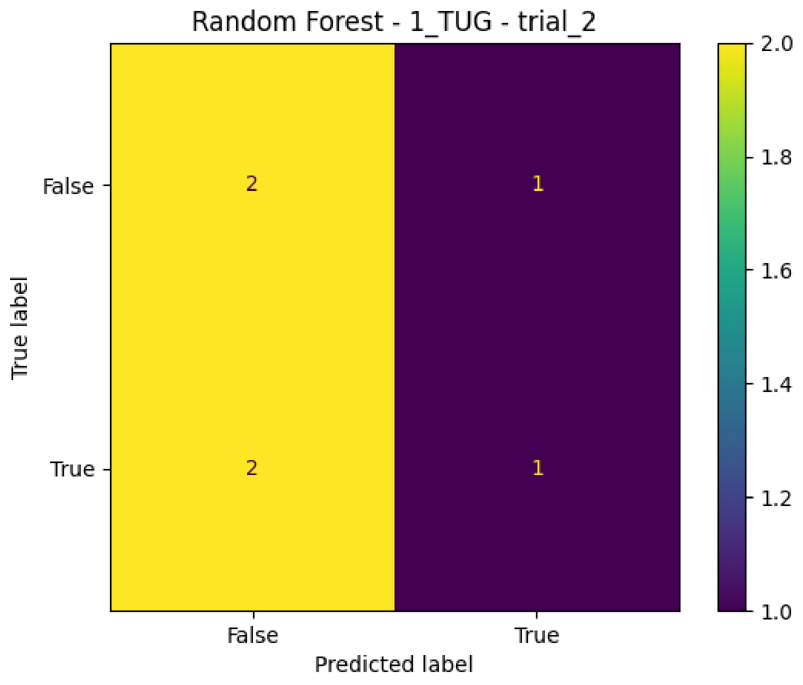

Random Forest - 2_CIRCUITOFISICO - trial_3
True label
Predicted label
False
True
2
0
2
2
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

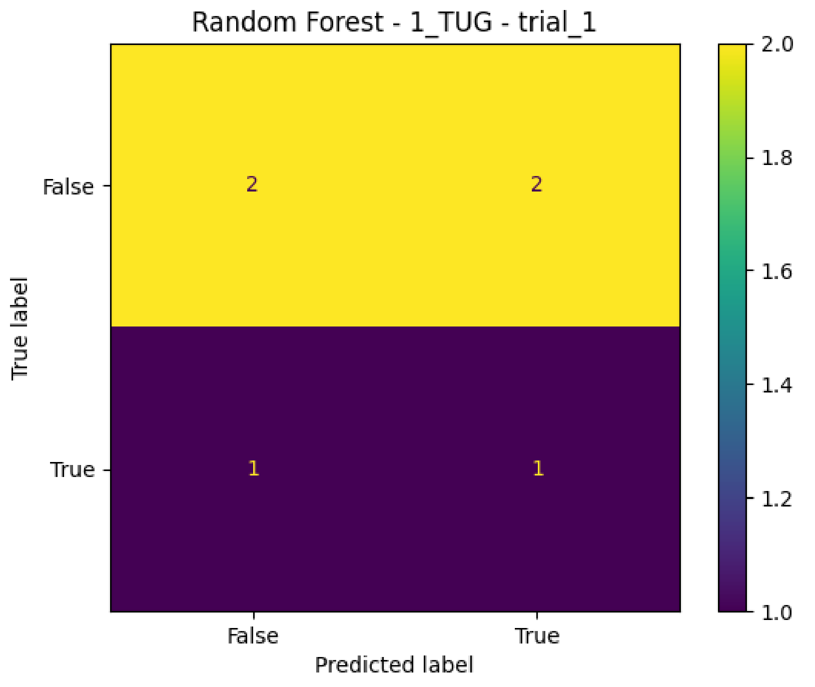

Random Forest - 2_CIRCUITOFISICO - trial_2
True label
Predicted label
False
True
1
4
0
1
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

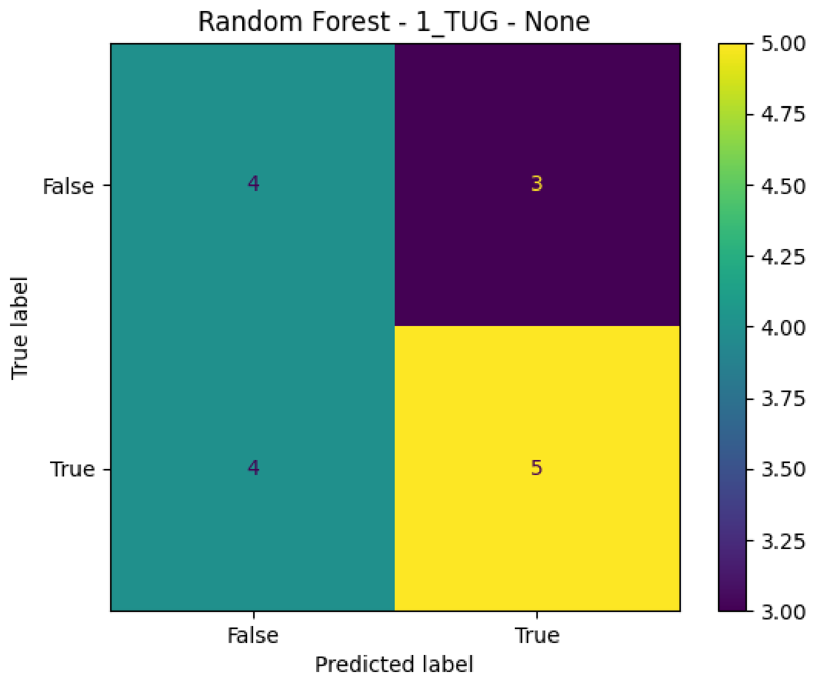

Random Forest - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
5
0
False
True
Predicted label
5
4
3
2
1
0

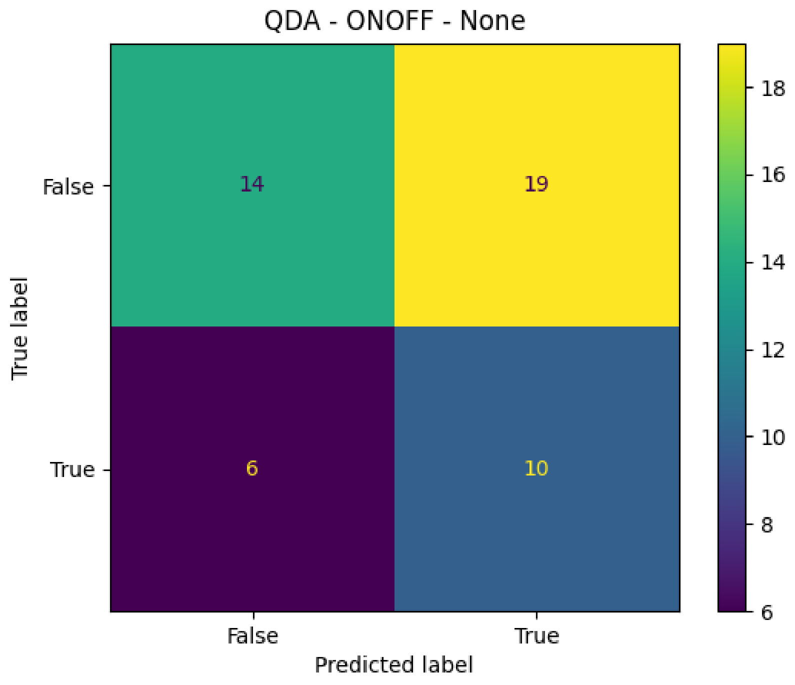

Random Forest - 2_CIRCUITOFISICO - None
True label
False
True
5
0
6
6
False
True
Predicted label
6
5
4
3
2
1
0

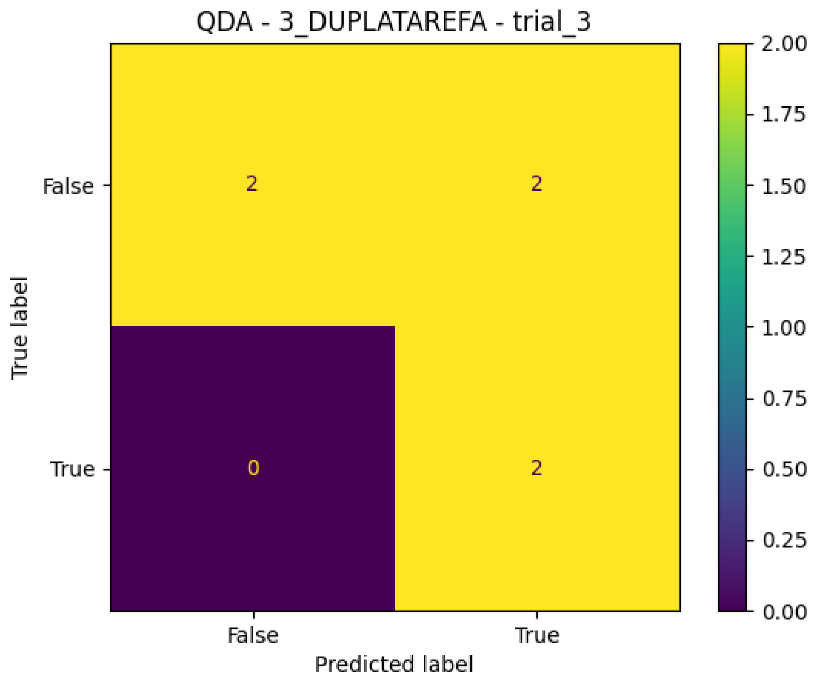

Random Forest - 1_TUG - trial_3
True label
False
True
2
2
1
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

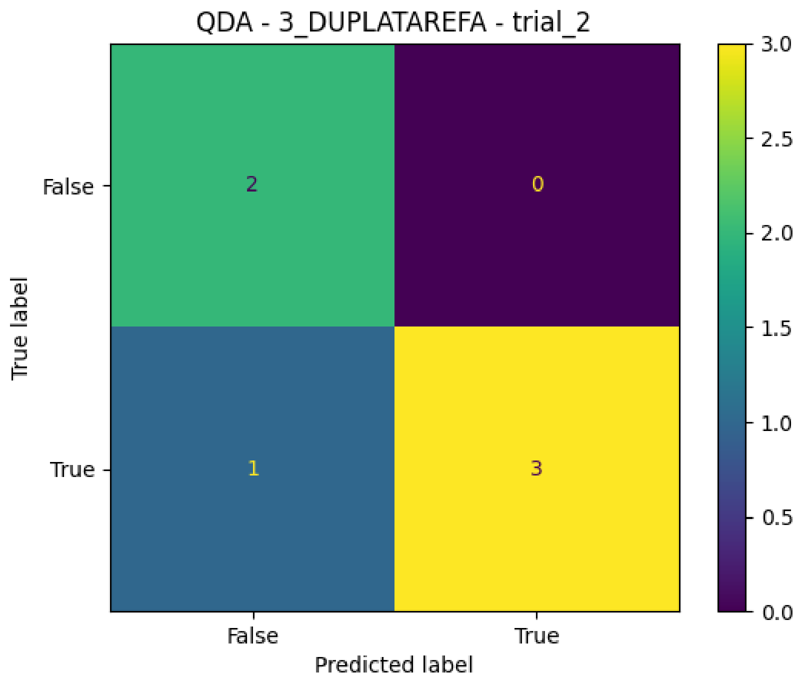

Random Forest - 1_TUG - trial_2
True label
False
True
2
1
2
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

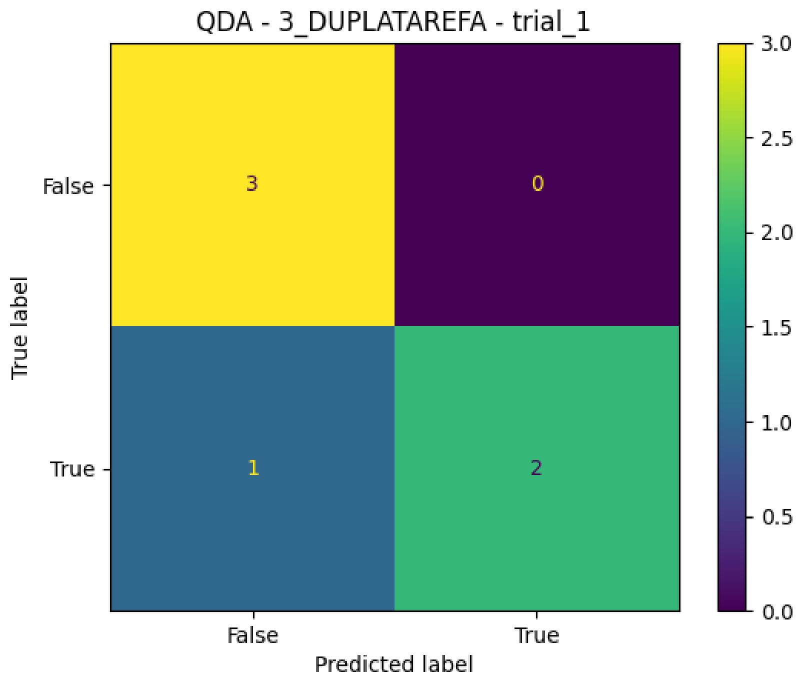

Random Forest - 1_TUG - trial_1
True label
False
True
2
2
1
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

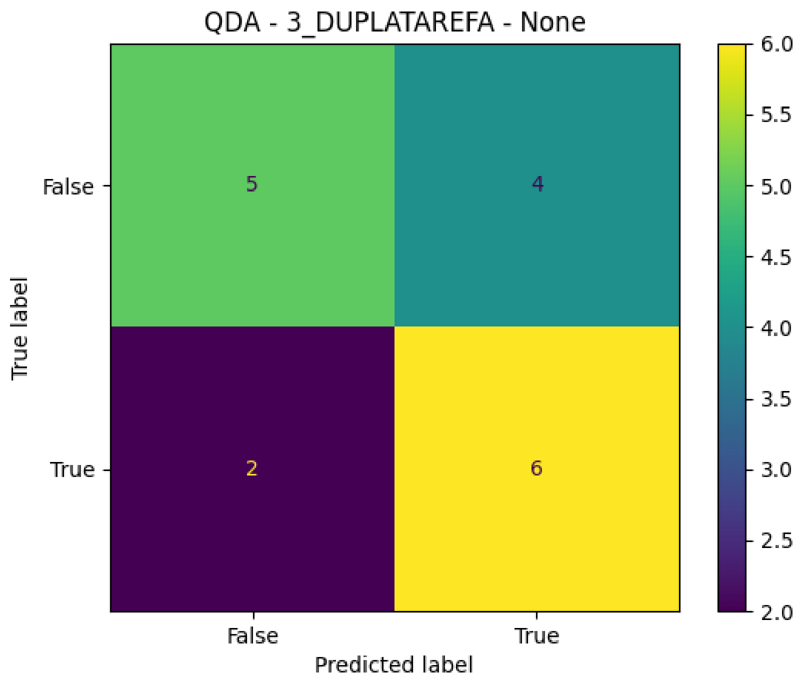

Random Forest - 1_TUG - None
True label
False
True
4
3
4
5
False
True
Predicted label
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00

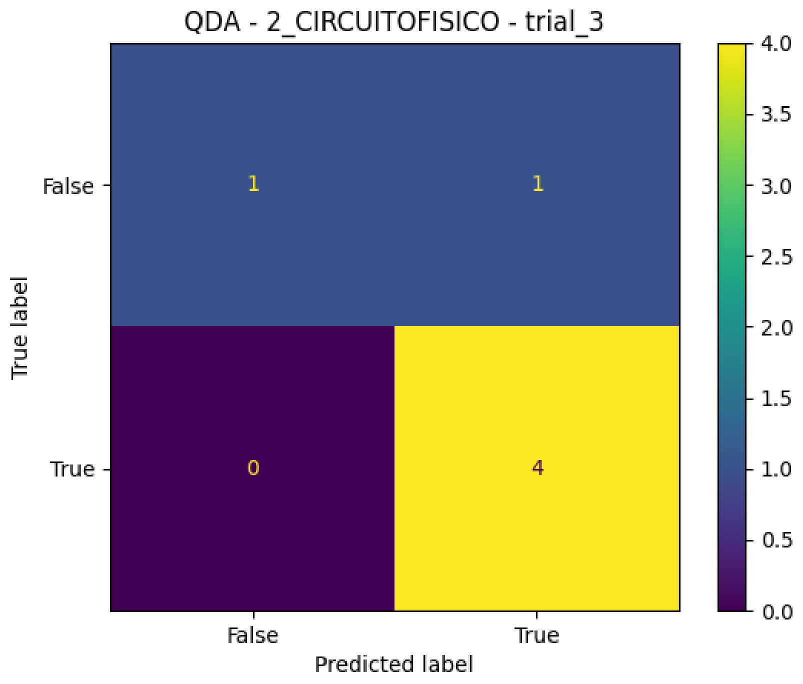

QDA - ONOFF - None
True label
False
True
14
19
6
10
Predicted label
False
True
18
16
14
12
10
8
6

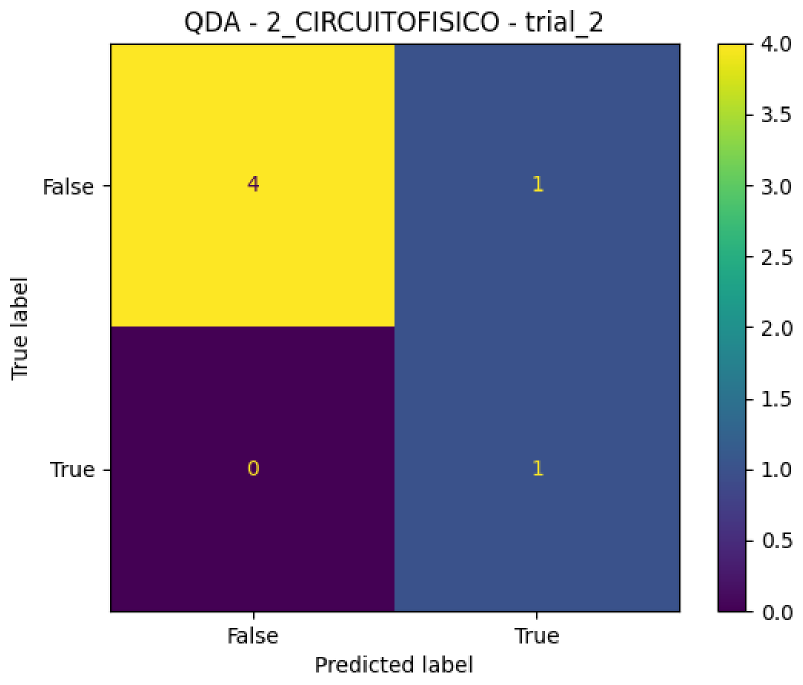

QDA - 3_DUPLATAREFA - trial_3
True label
False
True
2
2
0
2
Predicted label
False
True
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

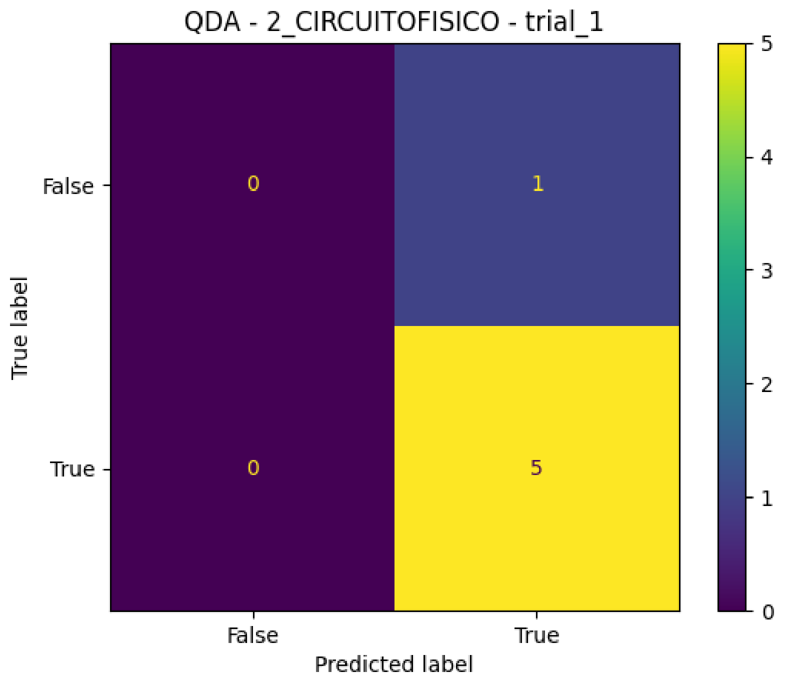

QDA - 3_DUPLATAREFA - trial_2
True label
False
True
Predicted label
False
True
2
0
1
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

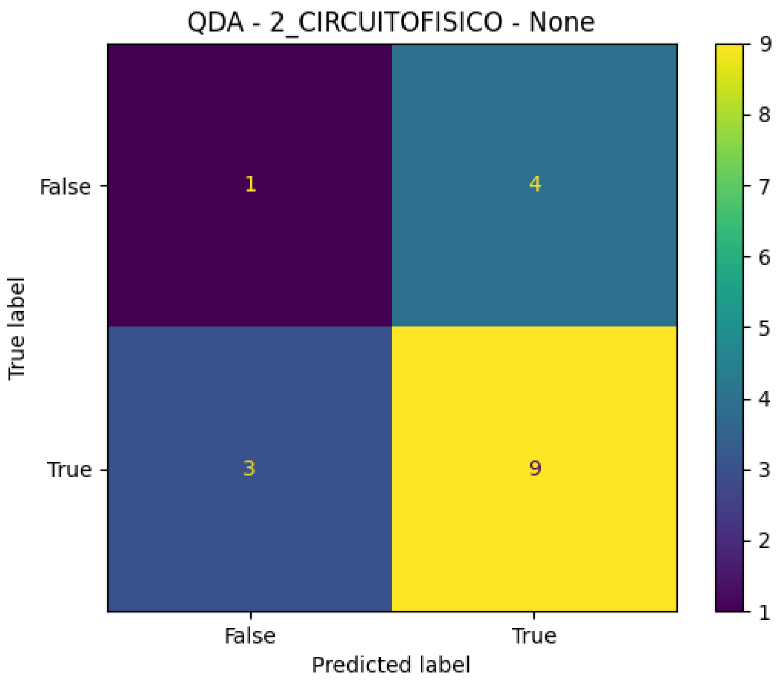

QDA - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
False
True
3
0
1
2
3.0
2.5
2.0
1.5
1.0
0.5
0.0

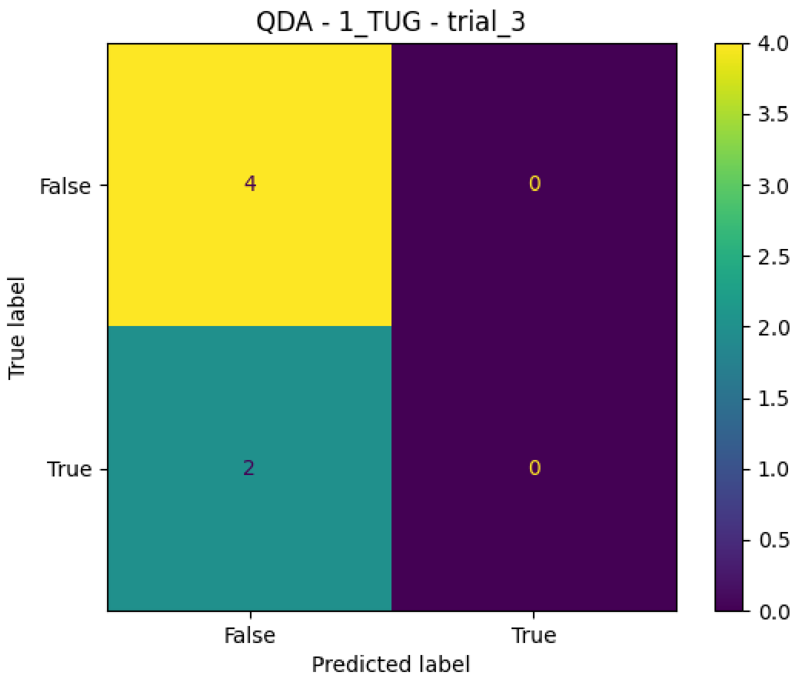

QDA - 3_DUPLATAREFA - None
True label
False
True
5
4
2
6
Predicted label
False
True
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0

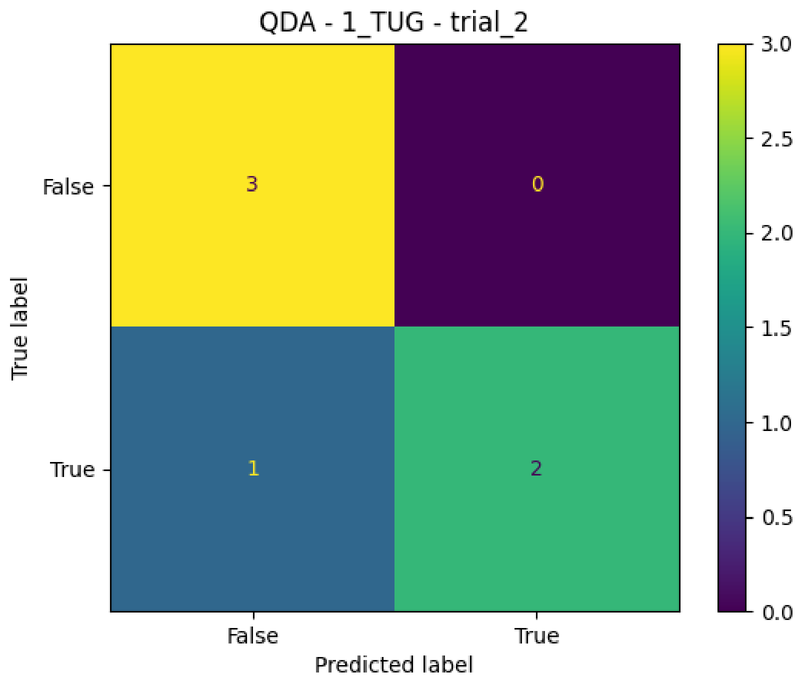

QDA - 2_CIRCUITOFISICO - trial_3
True label
False
True
1
1
0
4
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

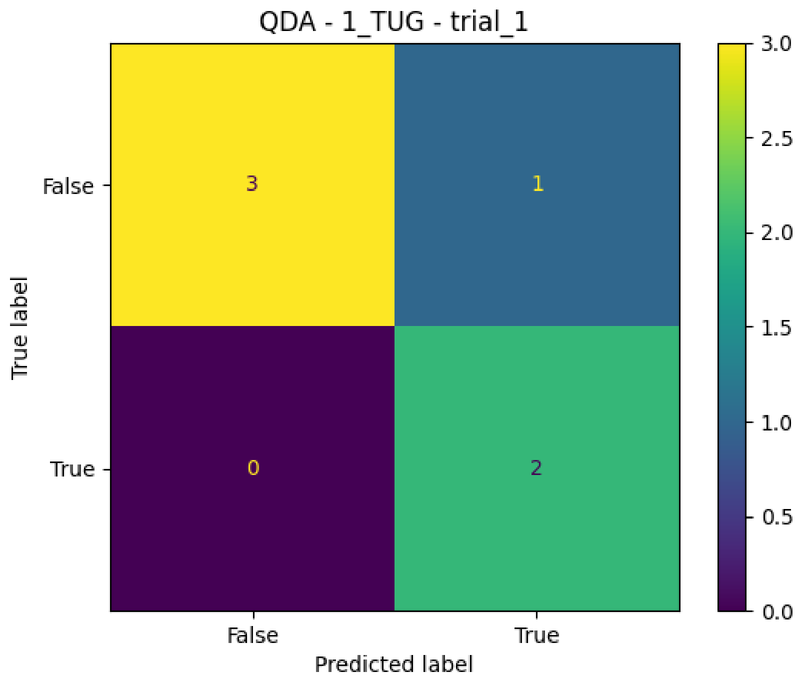

QDA - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
0
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

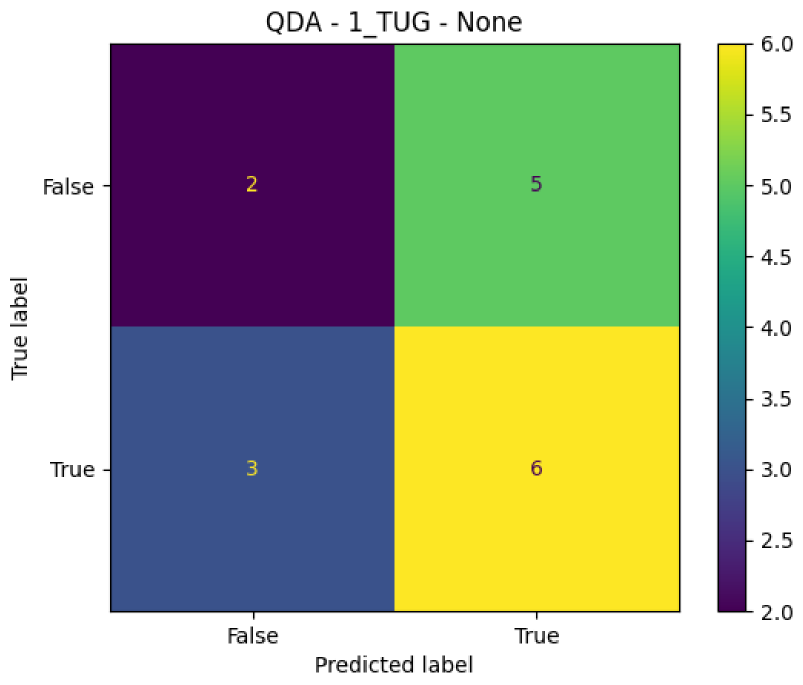

QDA - 2_CIRCUITOFISICO - trial_1
True label
False
True
0
1
0
5
False
True
Predicted label
5
4
3
2
1
0

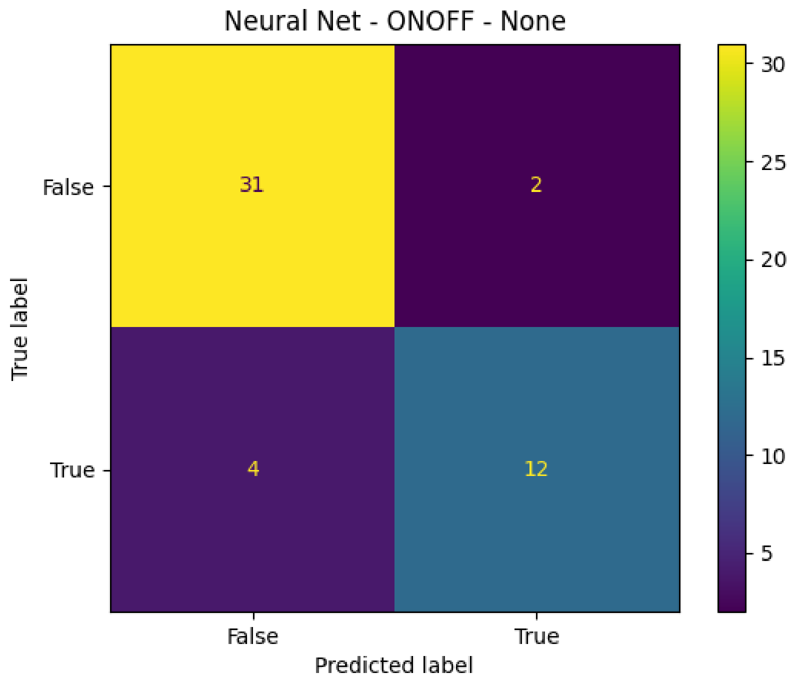

QDA - 2_CIRCUITOFISICO - None
True label
False
True
1
4
3
9
Predicted label
False
True
9
8
7
6
5
4
3
2
1

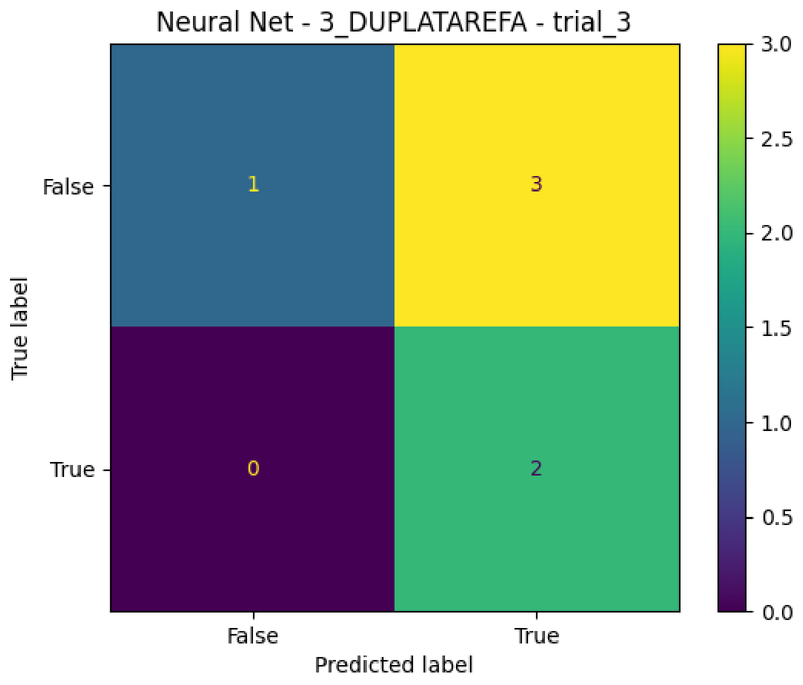

QDA - 1_TUG - trial_3
True label
False
True
4
0
2
0
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

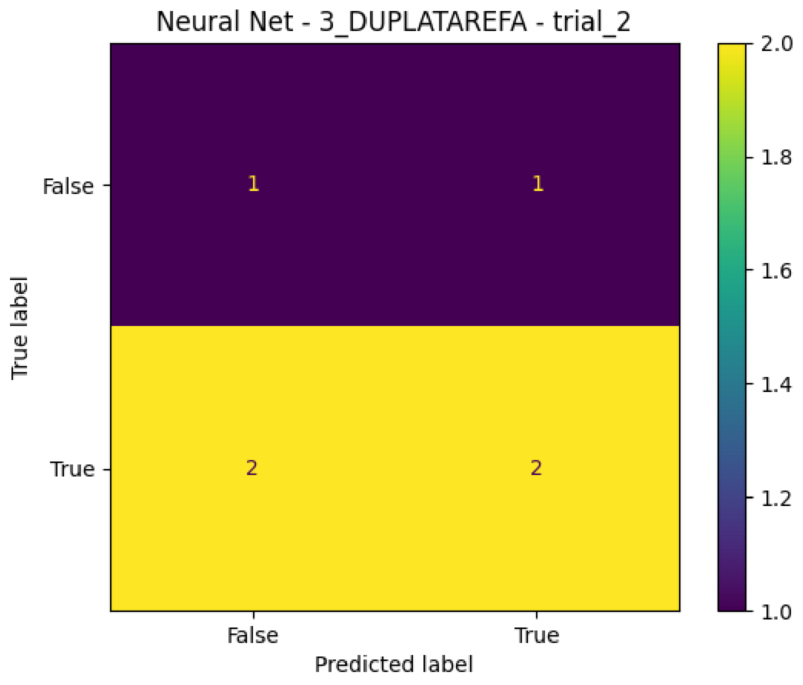

QDA - 1_TUG - trial_2
True label
False
True
3
0
1
2
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

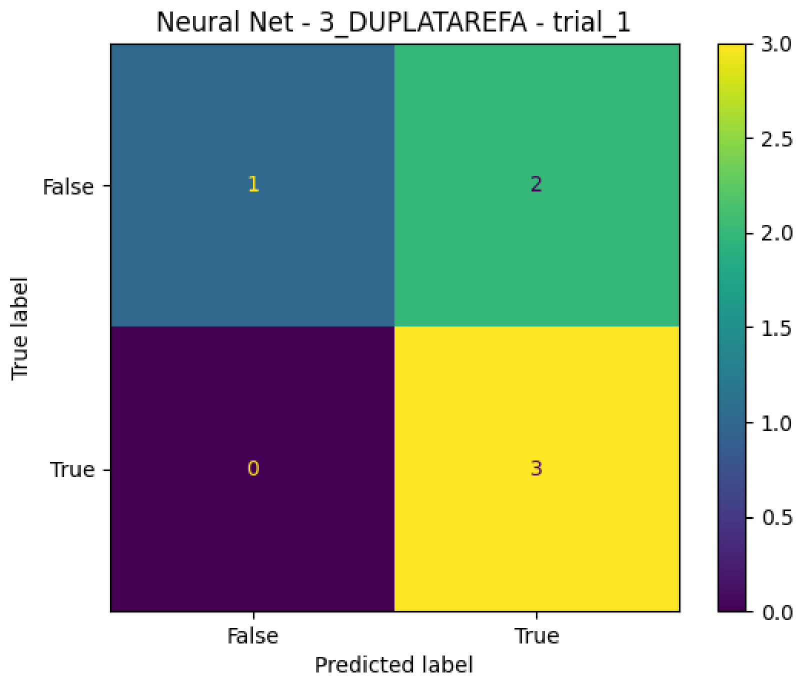

QDA - 1_TUG - trial_1
True label
False
True
3
1
0
2
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

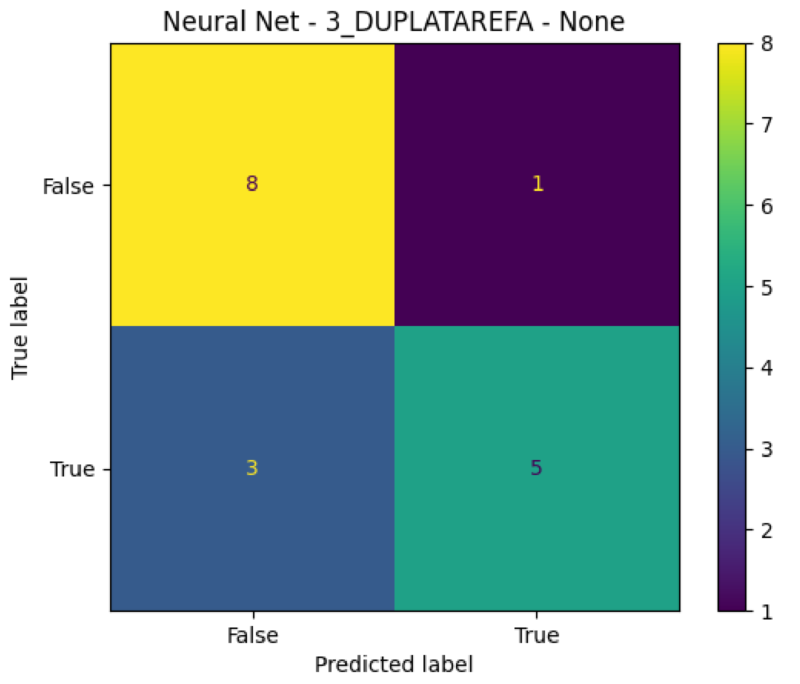

QDA - 1_TUG - None
True label
False
True
2
5
3
6
False
True
Predicted label
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0

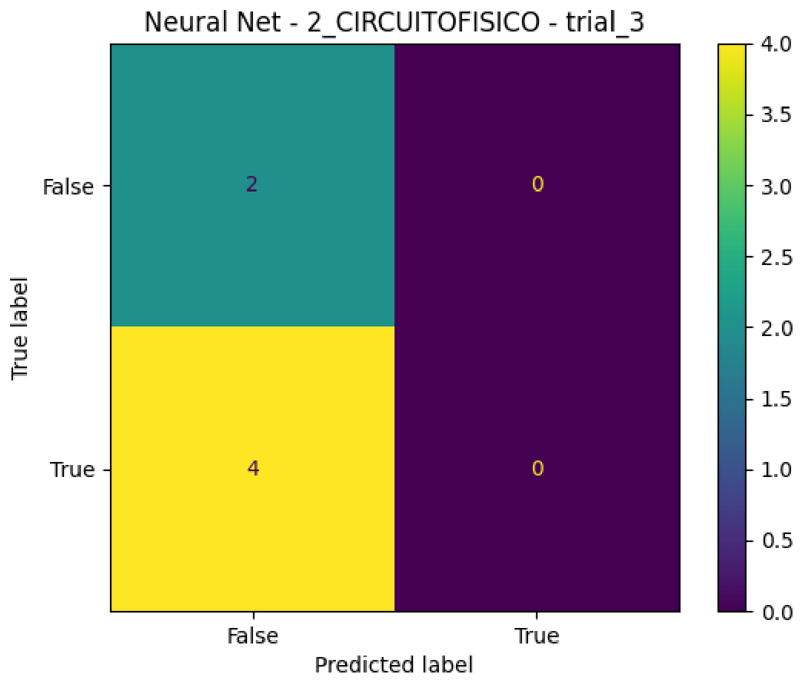

Neural Net - ONOFF - None
True label
False
True
31
2
4
12
False
True
Predicted label
30
25
20
15
10
5

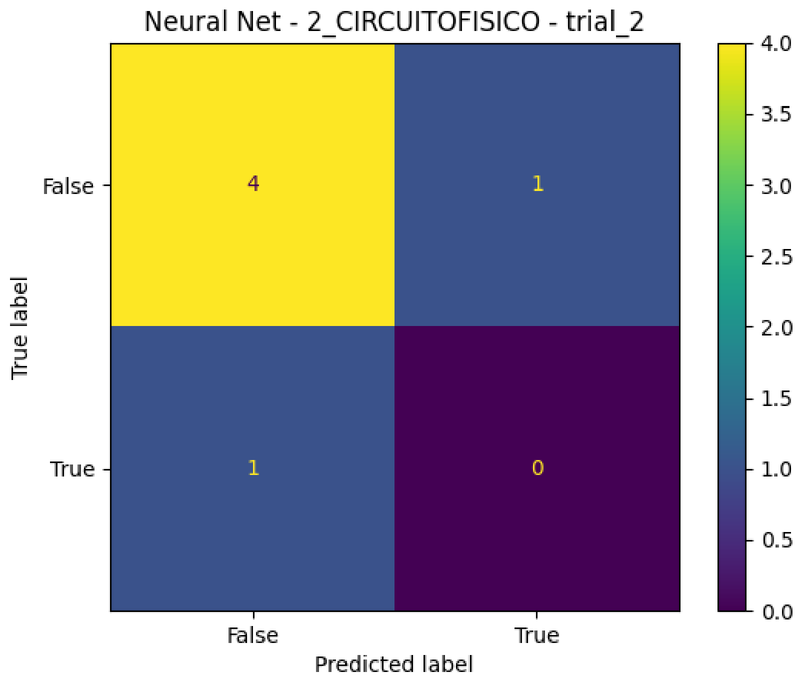

Neural Net - 3_DUPLATAREFA - trial_3
True label
False
True
1
3
0
2
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

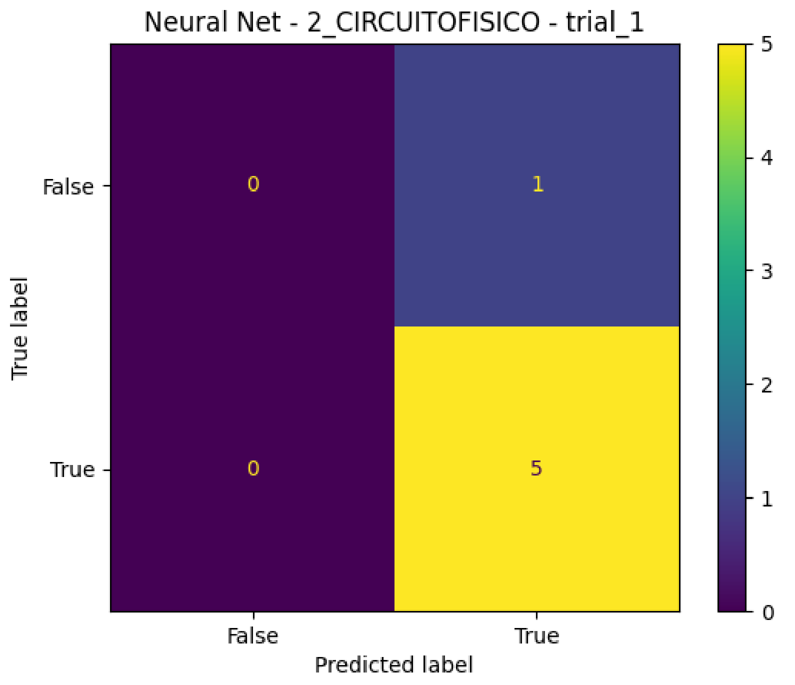

Neural Net - 3_DUPLATAREFA - trial_2
True label
False
True
1
1
2
2
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

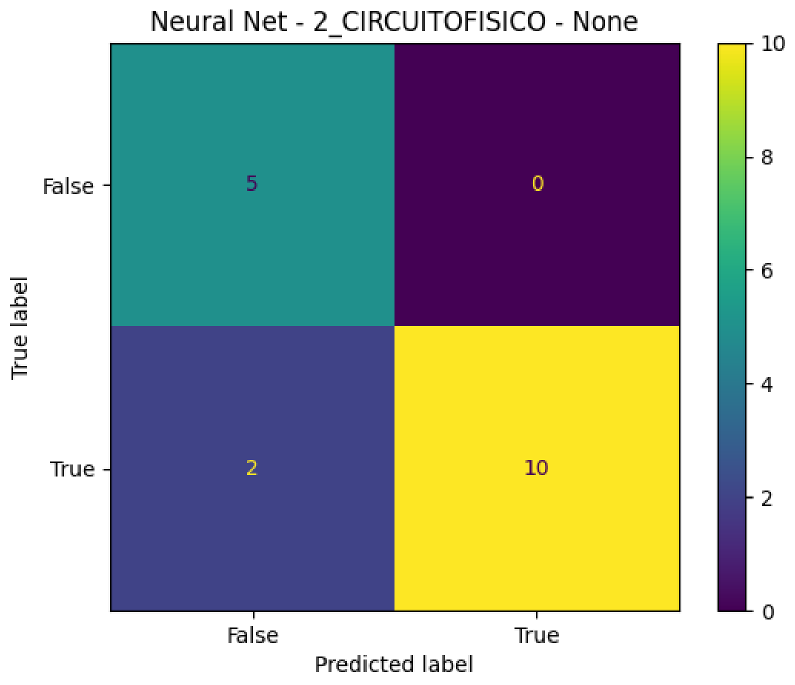

Neural Net - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
1
2
0
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

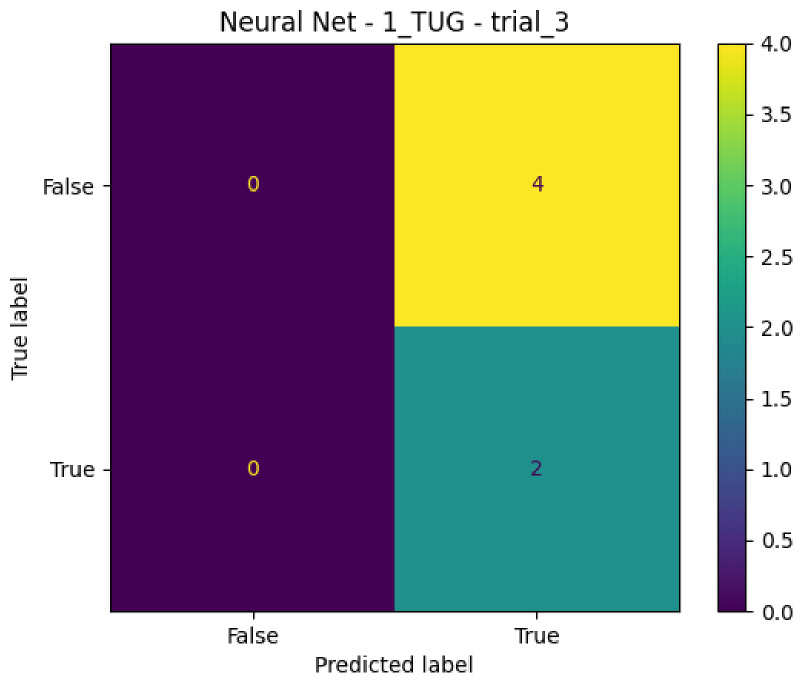

Neural Net - 3_DUPLATAREFA - None
True label
False
True
Predicted label
8
1
3
5
8
7
6
5
4
3
2
1

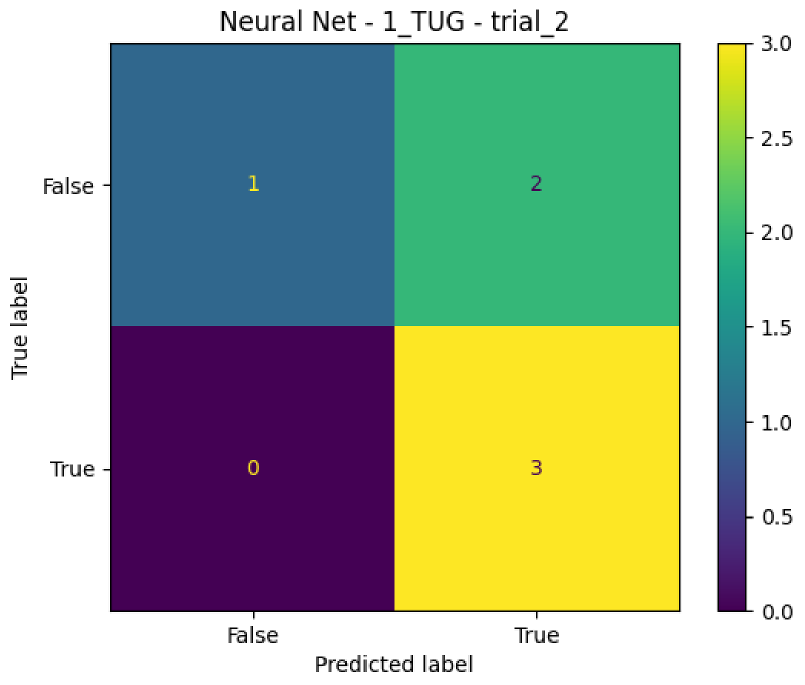

Neural Net - 2_CIRCUITOFISICO - trial_3
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

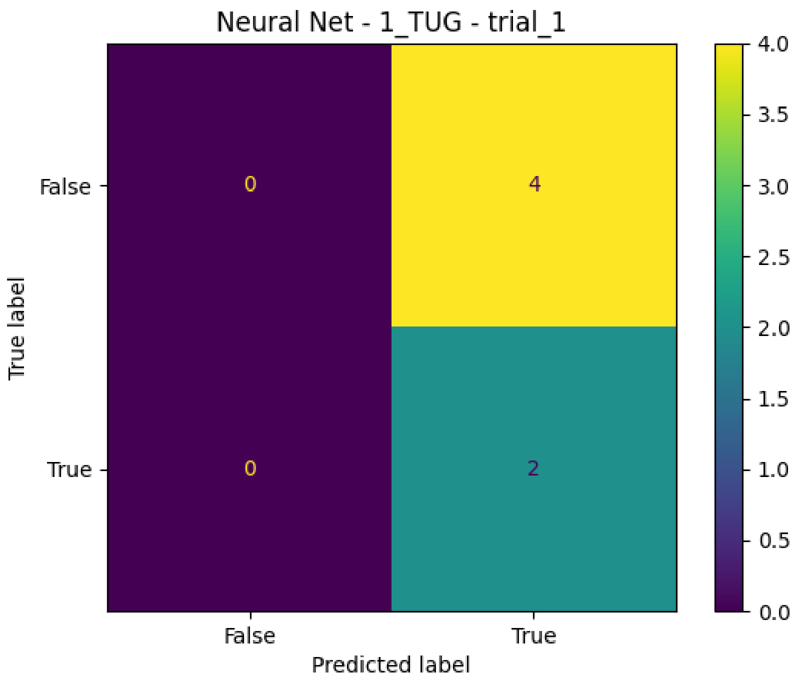

Neural Net - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
1
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

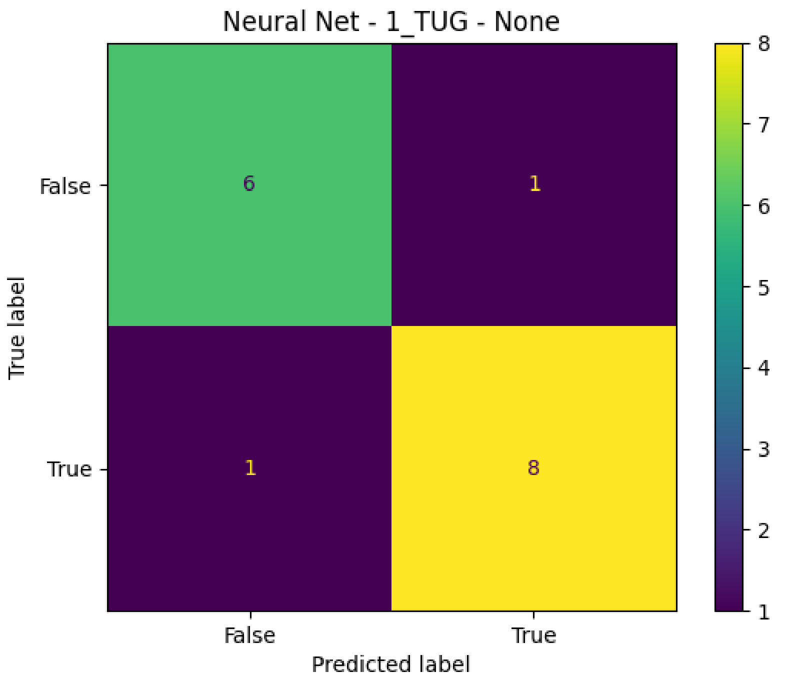

Neural Net - 2_CIRCUITOFISICO - trial_1
True label
False
True
0
1
0
5
False
True
Predicted label
0
1
2
3
4
5

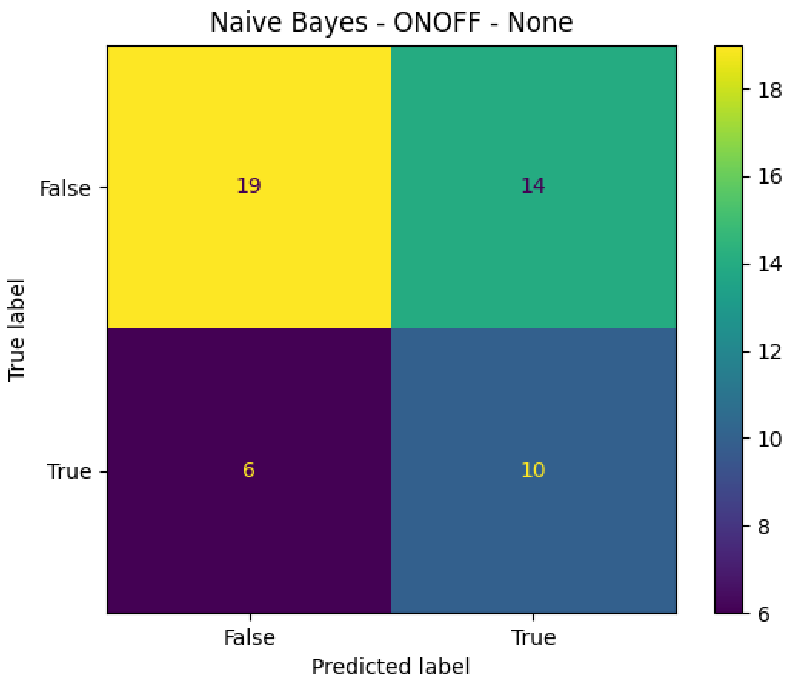

Neural Net - 2_CIRCUITOFISICO - None
True label
False
True
5
0
2
10
False
True
Predicted label
0
2
4
6
8
10

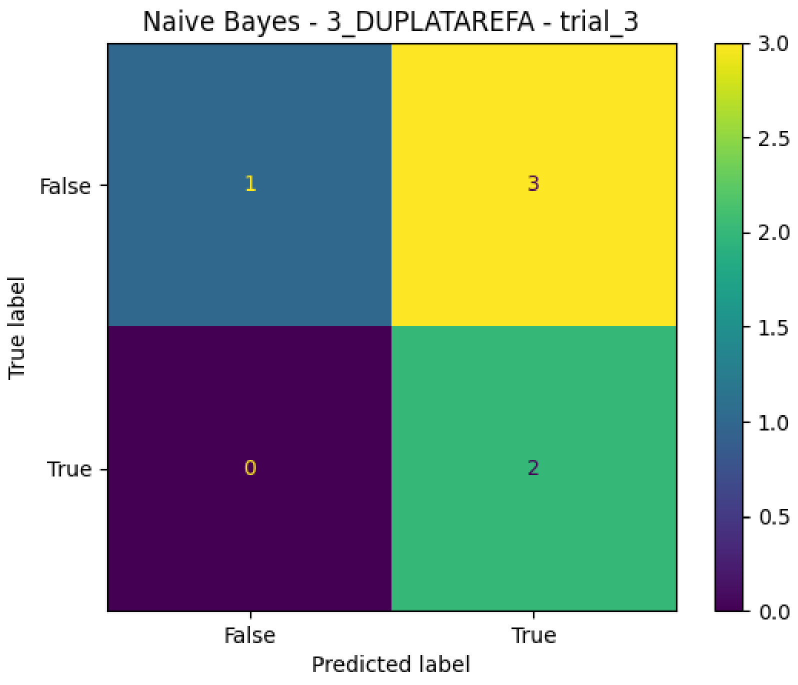

Neural Net - 1_TUG - trial_3
True label
False
True
Predicted label
False
True
0
4
0
2
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

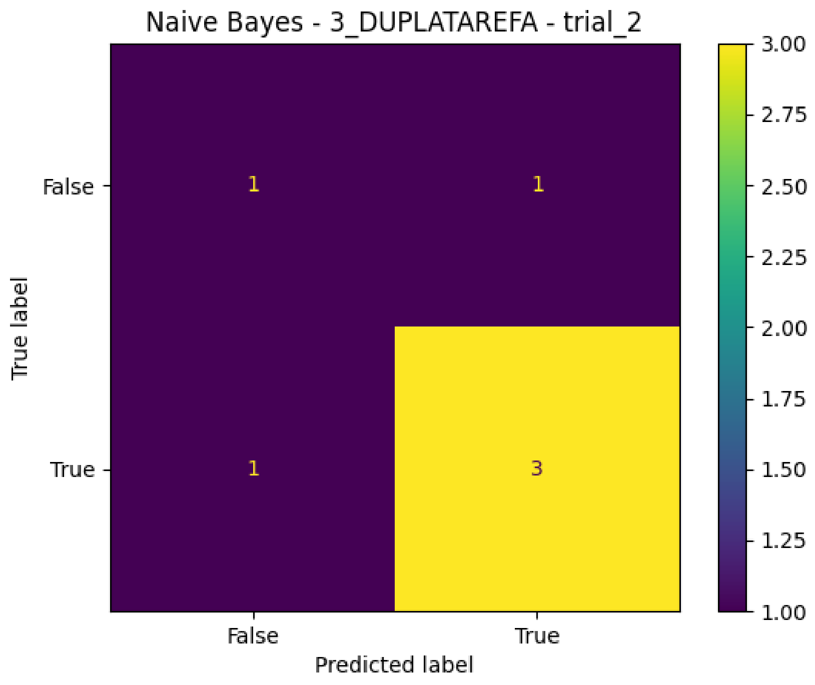

Neural Net - 1_TUG - trial_2
True label
False
True
Predicted label
False
True
1
2
0
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

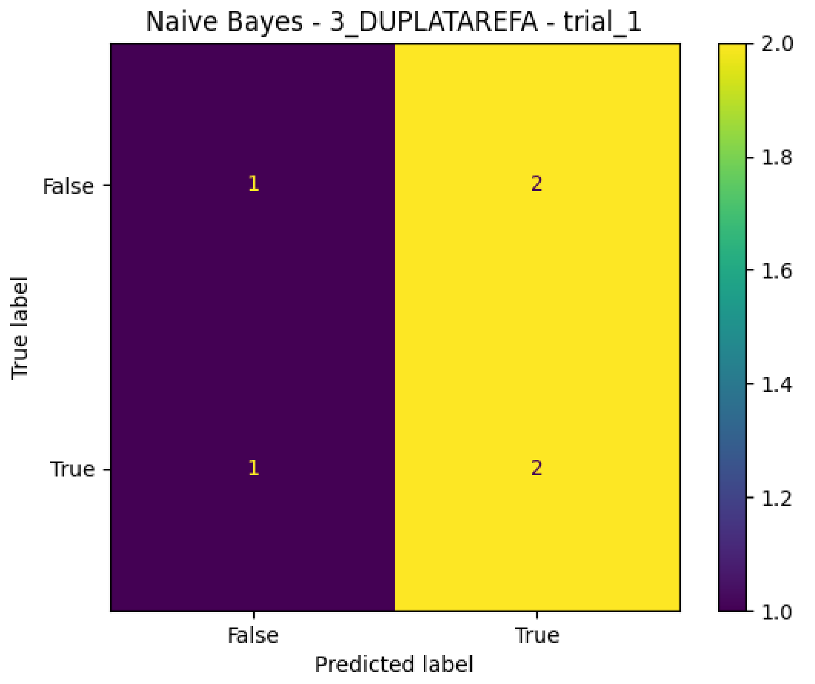

Neural Net - 1_TUG - trial_1
True label
False
True
0
4
0
2
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

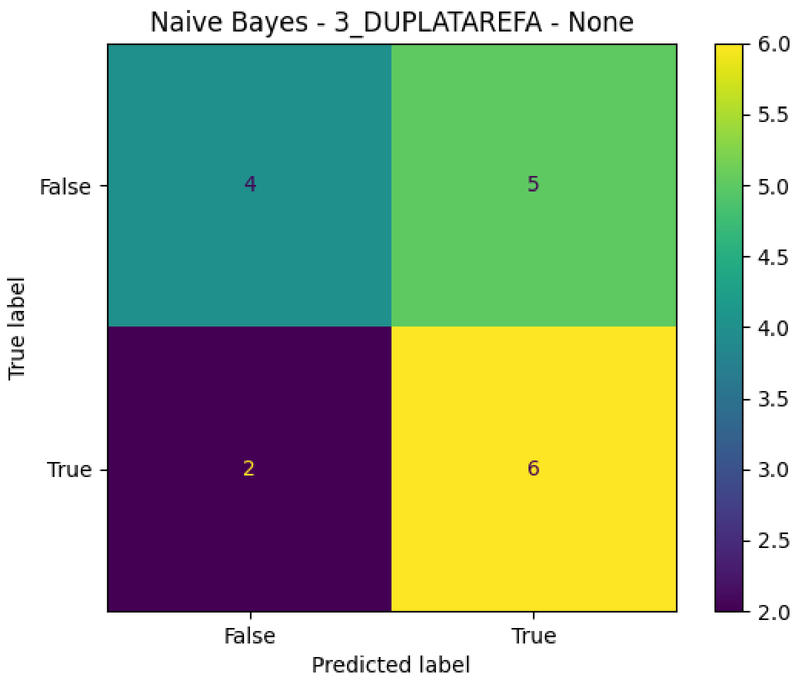

Neural Net - 1_TUG - None
True label
False
True
6
1
1
8
False
True
Predicted label
8
7
6
5
4
3
2
1

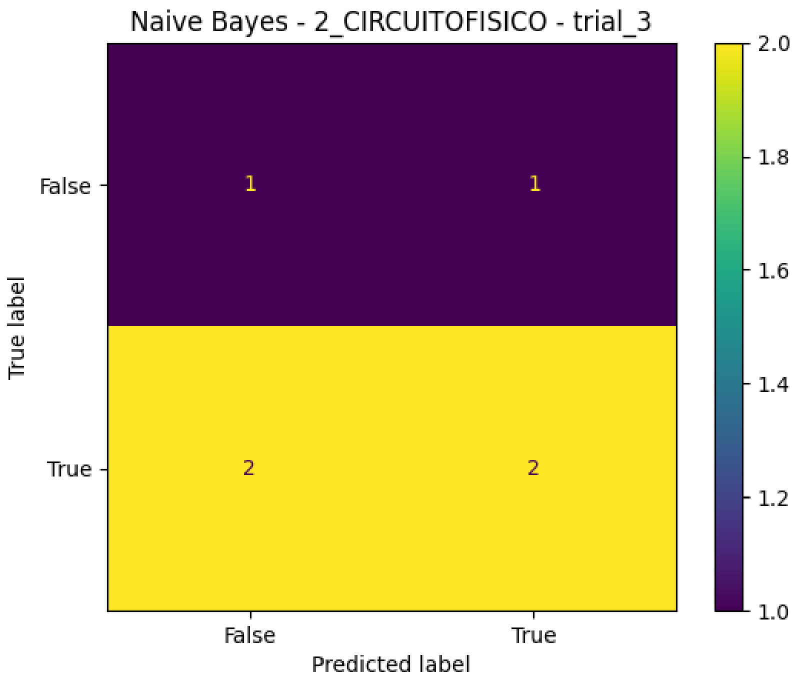

Naive Bayes - ONOFF - None
True label
False
True
19
14
6
10
False
True
Predicted label
18
16
14
12
10
8
6

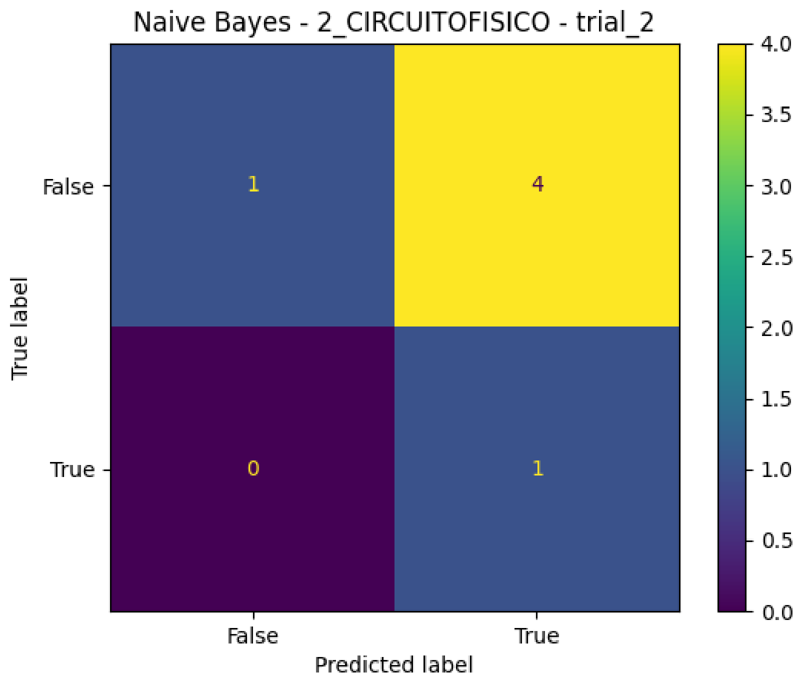

Naive Bayes - 3_DUPLATAREFA - trial_3
True label
False
True
1
3
0
2
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

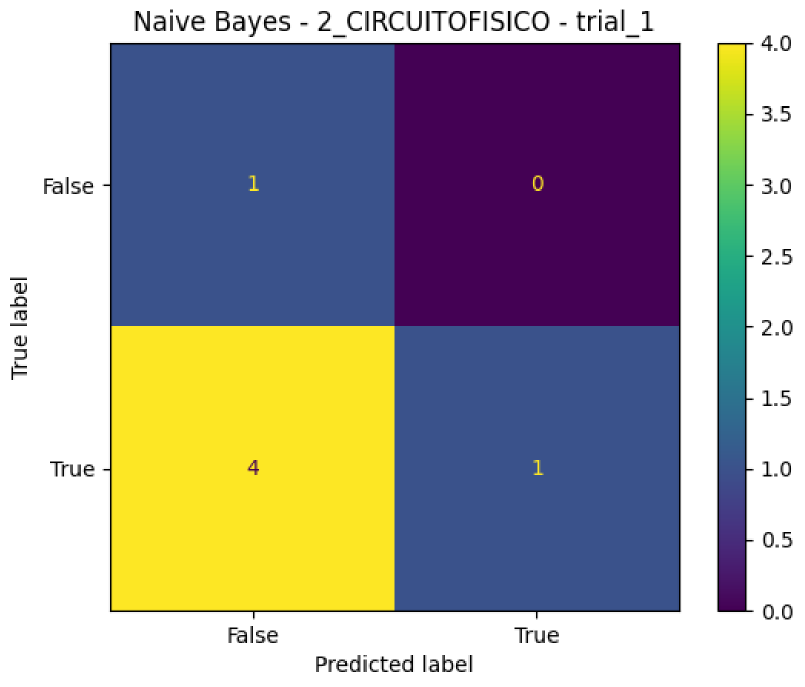

Naive Bayes - 3_DUPLATAREFA - trial_2
True label
False
True
1
1
1
3
False
True
Predicted label
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

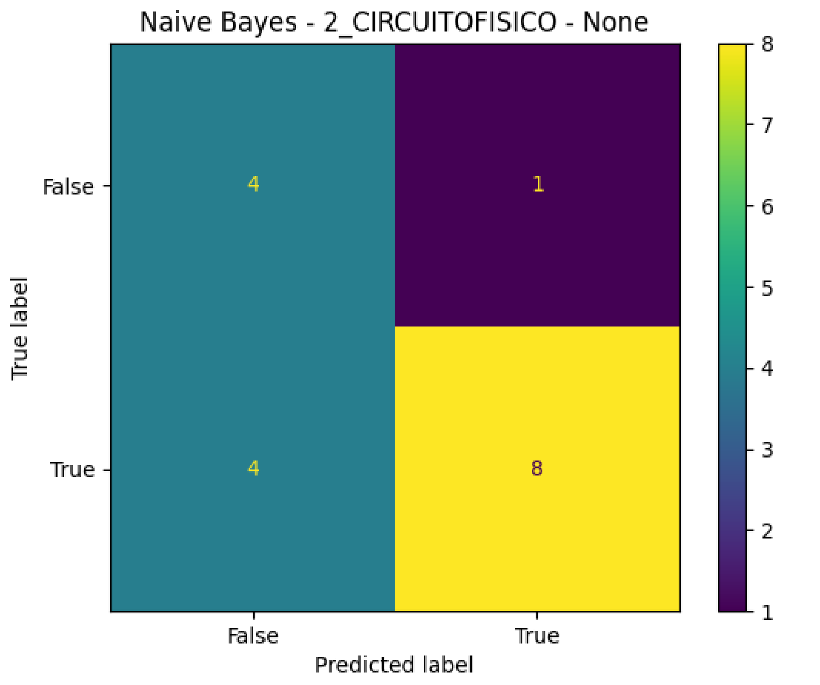

Naive Bayes - 3_DUPLATAREFA - trial_1
True label
False
True
1
2
1
2
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

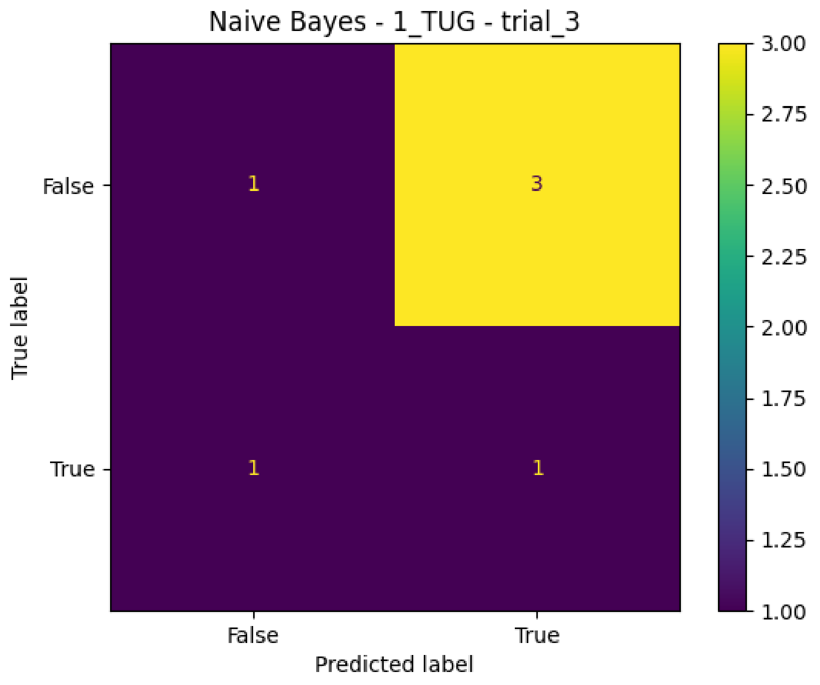

Naive Bayes - 3_DUPLATAREFA - None
True label
False
True
4
5
2
6
Predicted label
False
True
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0

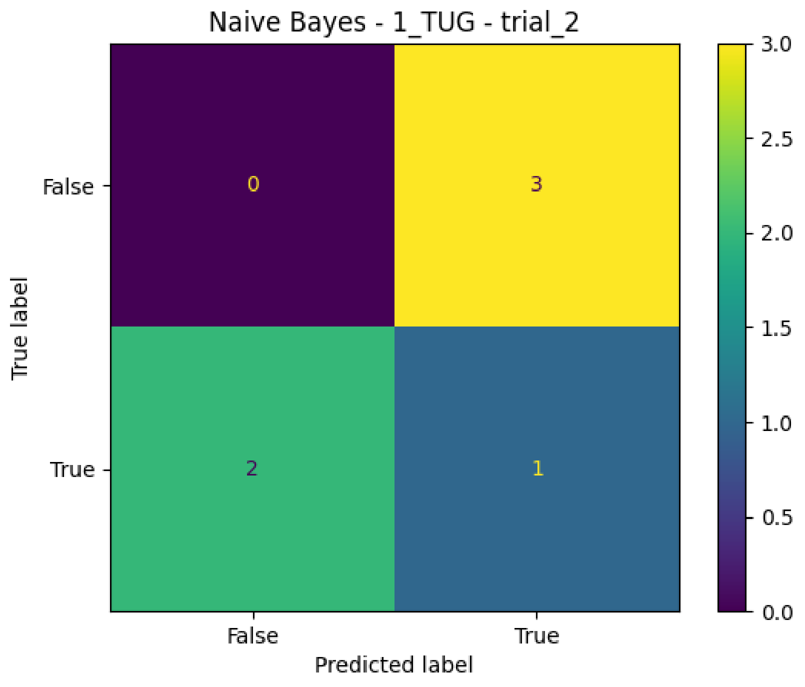

Naive Bayes - 2_CIRCUITOFISICO - trial_3
True label
False
True
1
1
2
2
Predicted label
False
True
2.0
1.8
1.6
1.4
1.2
1.0

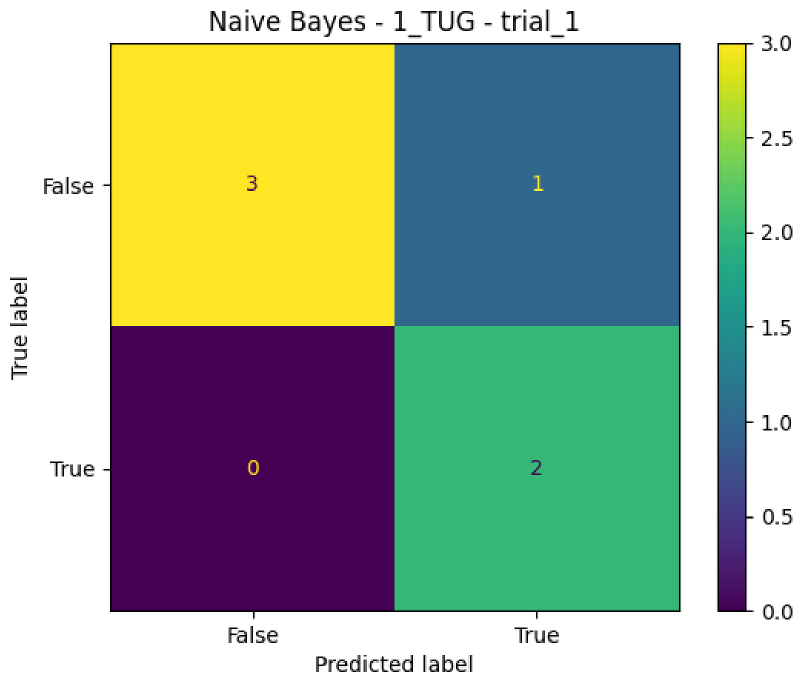

Naive Bayes - 2_CIRCUITOFISICO - trial_2
True label
False
True
1
4
0
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

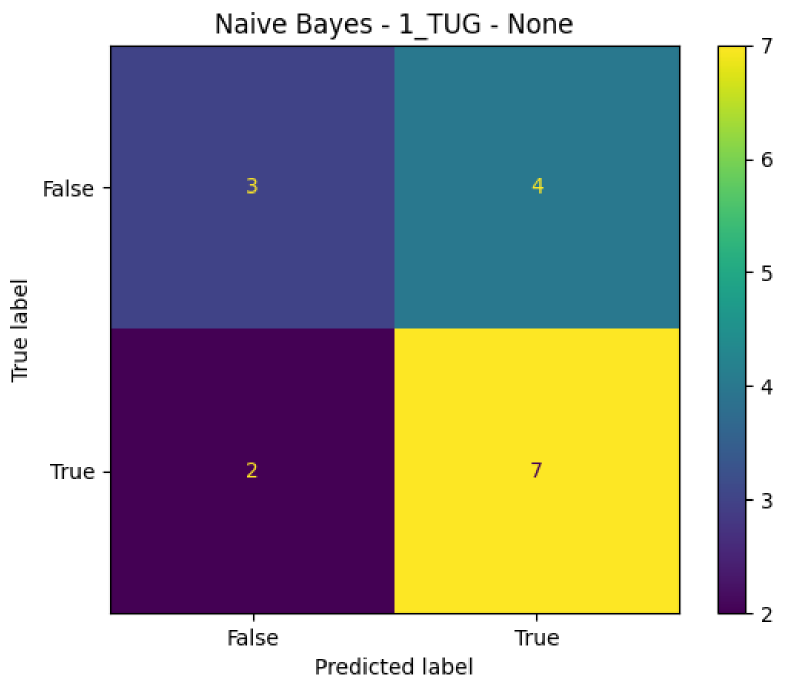

Naive Bayes - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
4
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

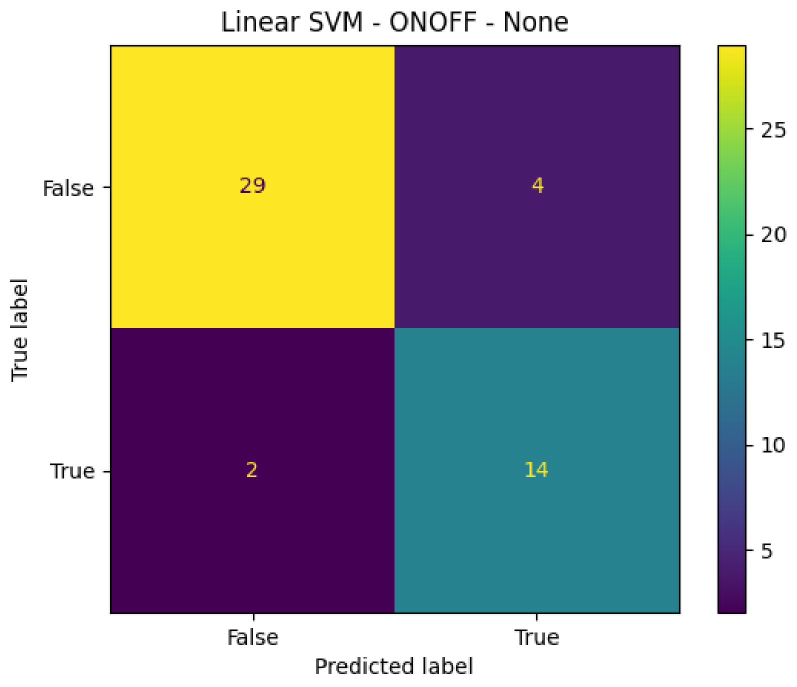

Naive Bayes - 2_CIRCUITOFISICO - None
True label
False
True
Predicted label
4
1
4
8
8
7
6
5
4
3
2
1

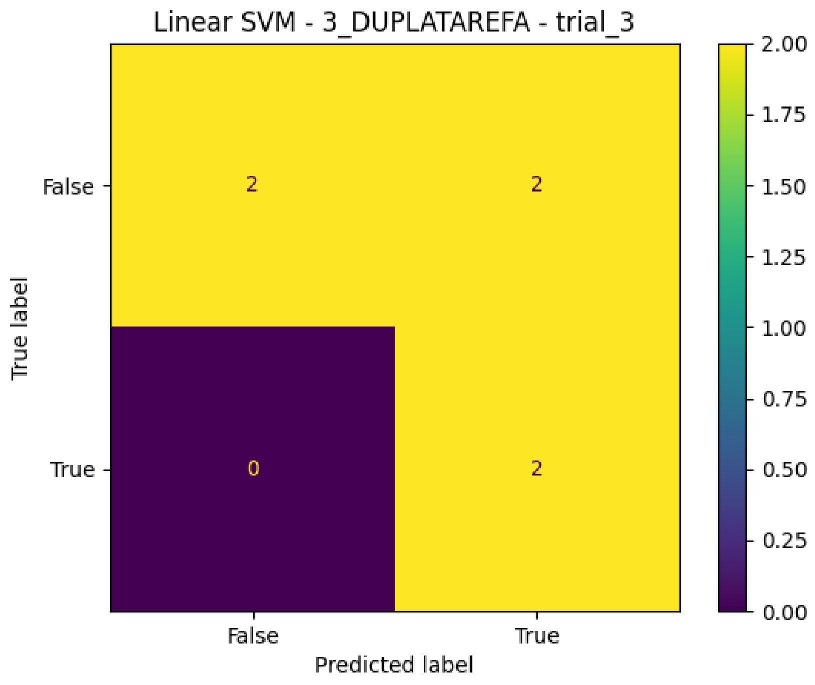

Naive Bayes - 1_TUG - trial_3
True label
False
True
Predicted label
1
3
1
1
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

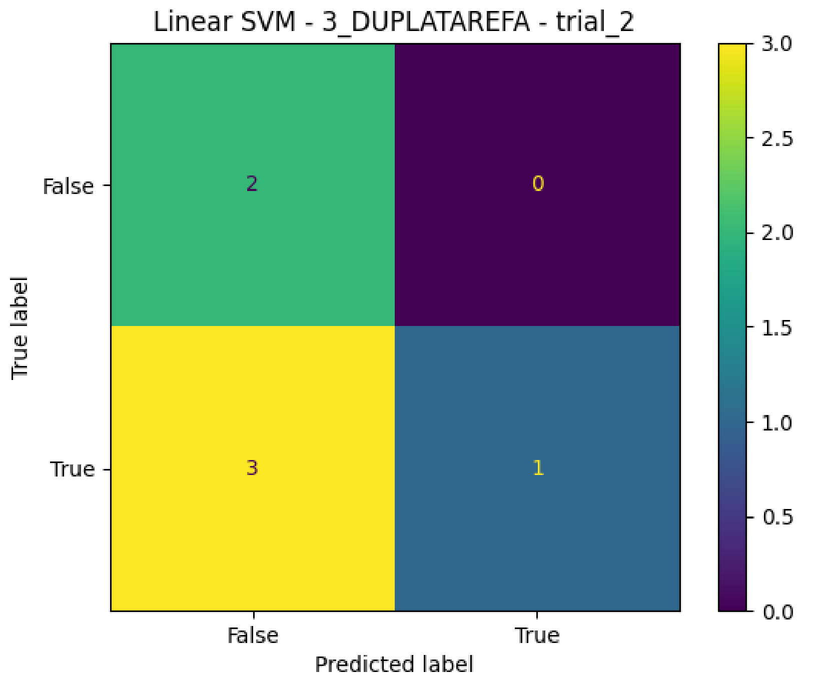

Naive Bayes - 1_TUG - trial_2
True label
False
True
0
3
2
1
False
True
Predicted label
0.0
0.5
1.0
1.5
2.0
2.5
3.0

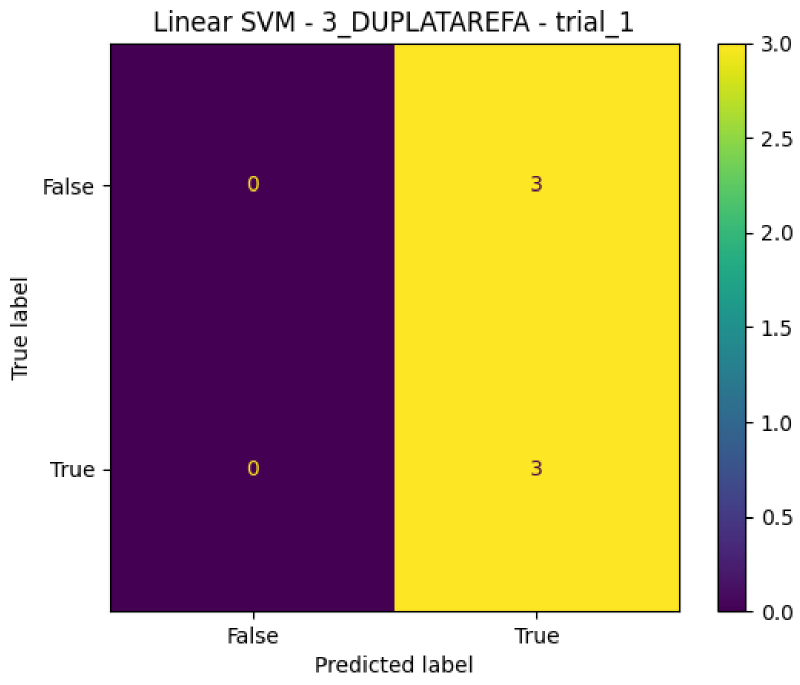

Naive Bayes - 1_TUG - trial_1
True label
False
True
3
1
0
2
False
True
Predicted label
0.0
0.5
1.0
1.5
2.0
2.5
3.0

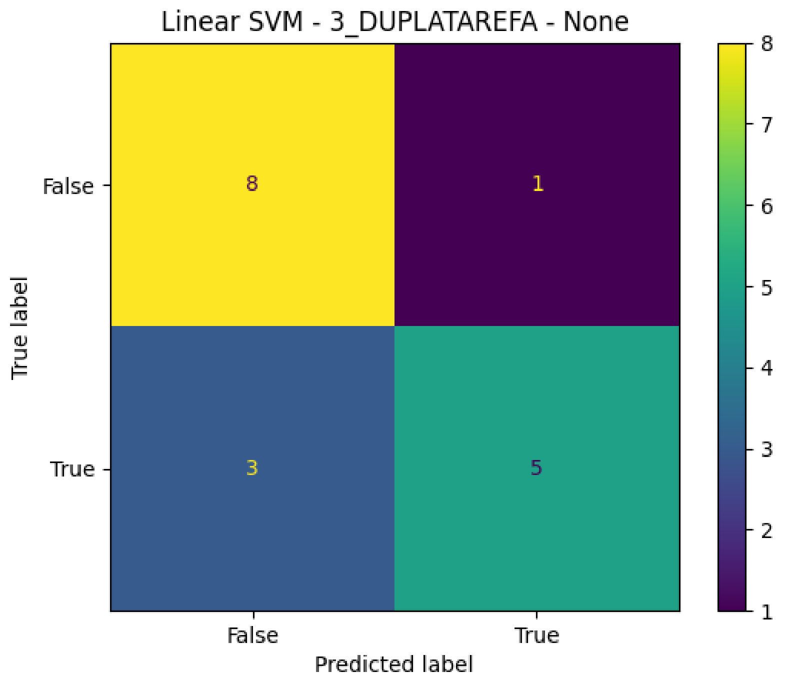

Naive Bayes - 1_TUG - None
True label
False
True
Predicted label
3
4
2
7
2
3
4
5
6
7

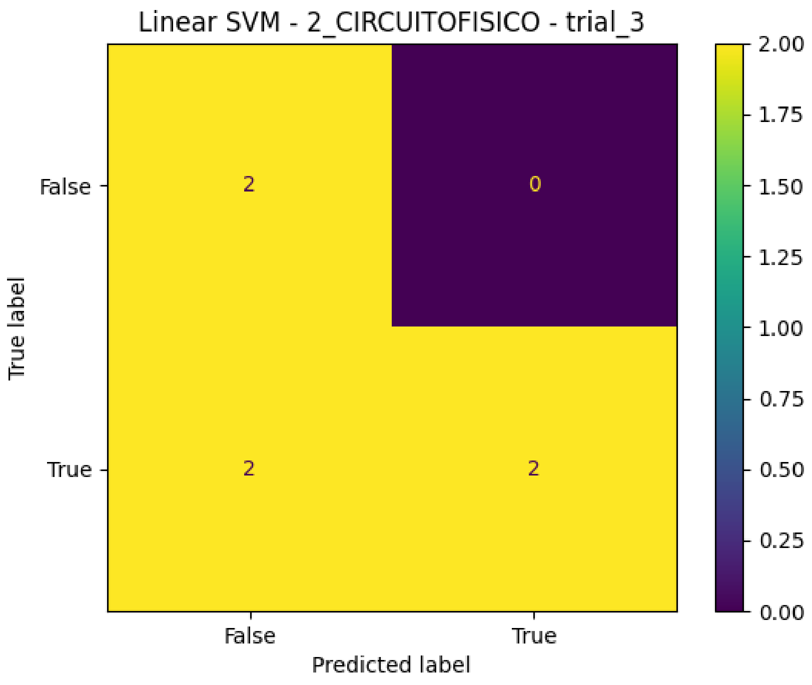

Linear SVM - ONOFF - None
True label
False
True
Predicted label
29
4
2
14
5
10
15
20
25

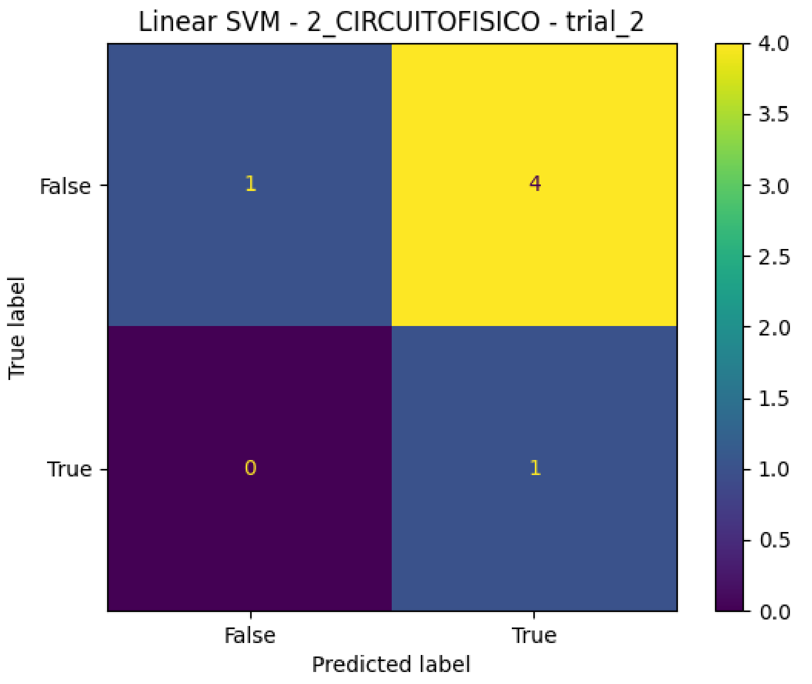

Linear SVM - 3_DUPLATAREFA - trial_3
True label
False
True
2
2
0
2
False
True
Predicted label
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

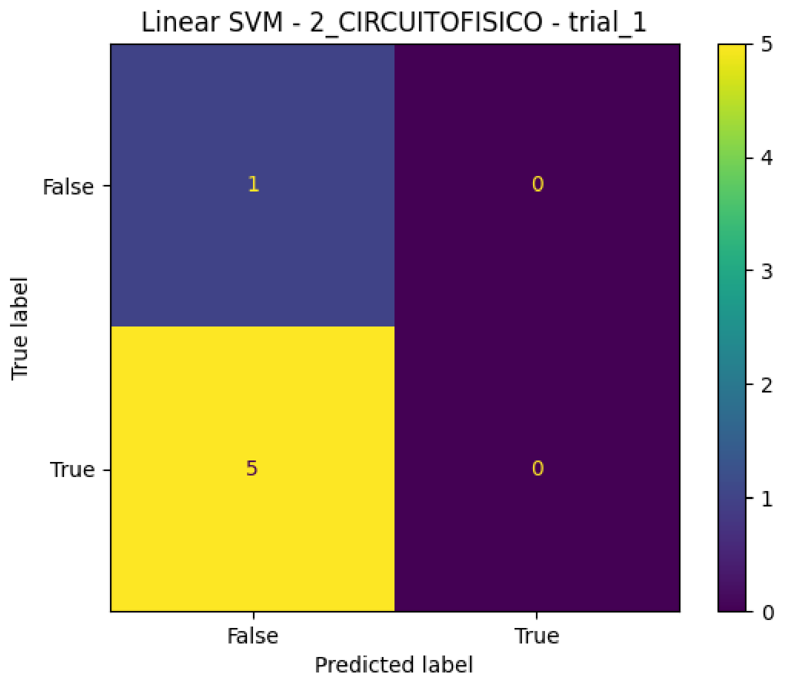

Linear SVM - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
3
1
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

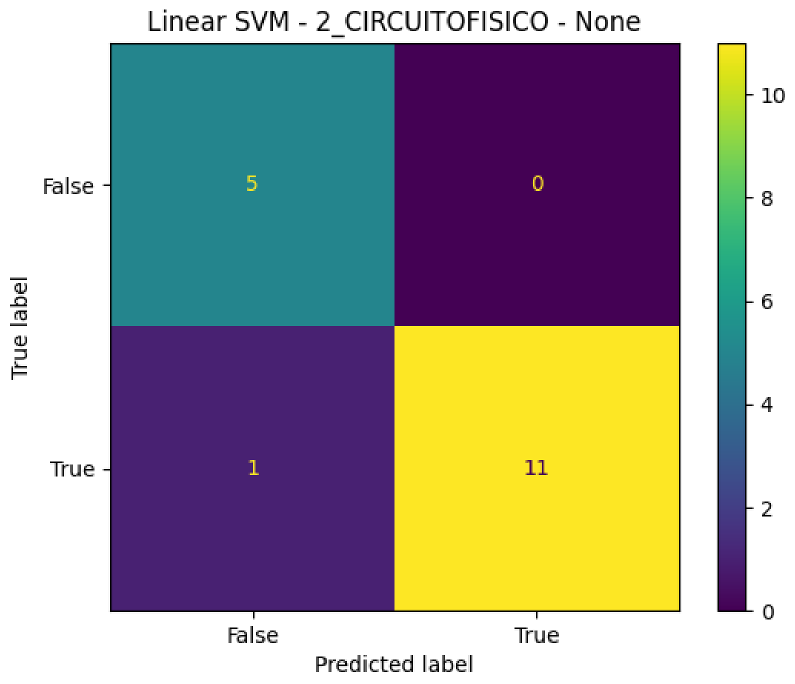

Linear SVM - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
0
3
0
3
0.0
0.5
1.0
1.5
2.0
2.5
3.0

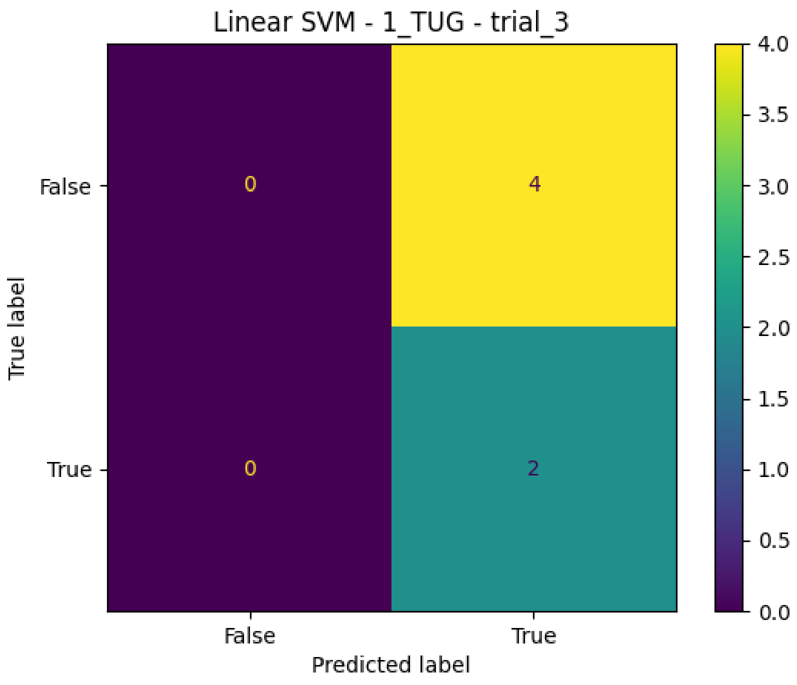

Linear SVM - 3_DUPLATAREFA - None
True label
False
True
Predicted label
8
1
3
5
1
2
3
4
5
6
7
8

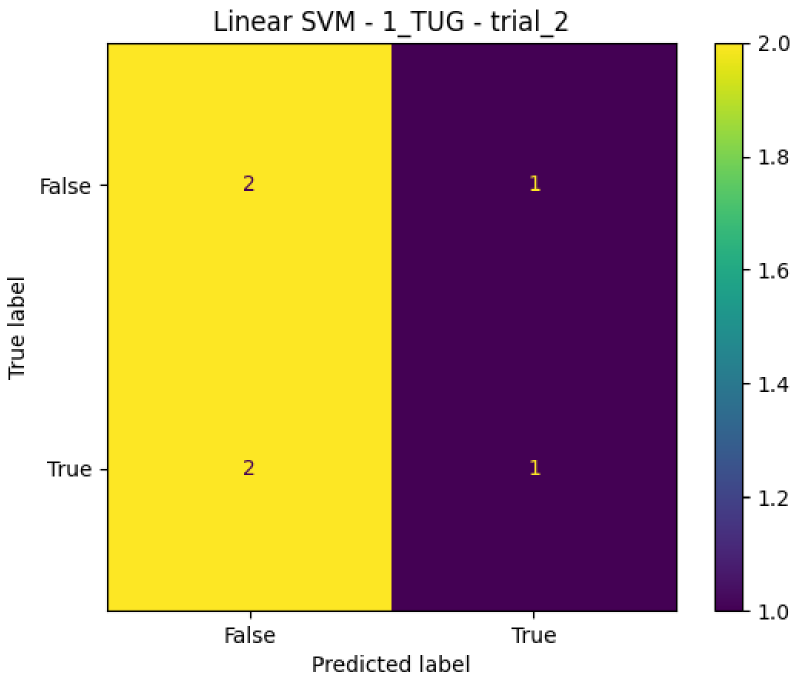

Linear SVM - 2_CIRCUITOFISICO - trial_3
True label
False
True
2
0
2
2
Predicted label
False
True
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

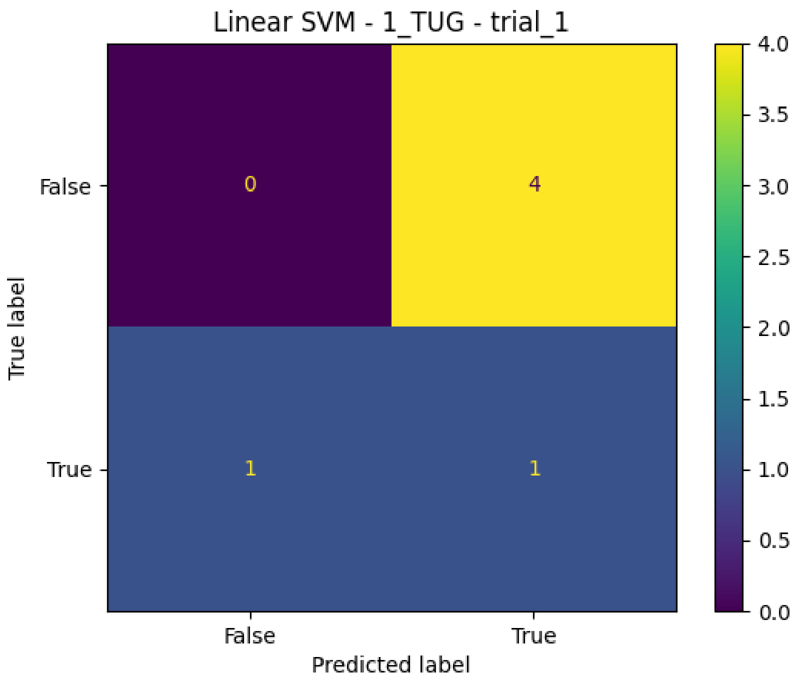

Linear SVM - 2_CIRCUITOFISICO - trial_2
True label
False
True
1
4
0
1
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

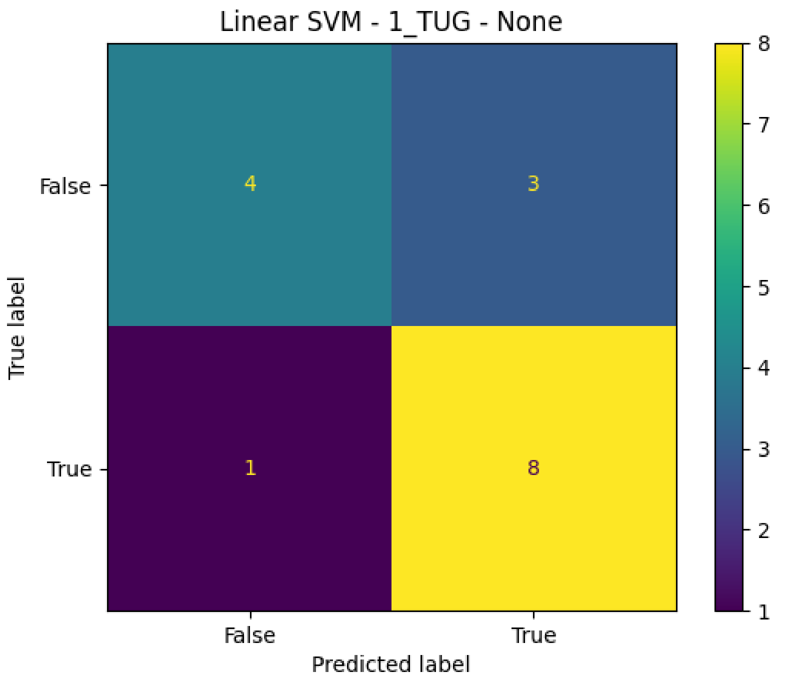

Linear SVM - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
5
0
False
True
Predicted label
5
4
3
2
1
0

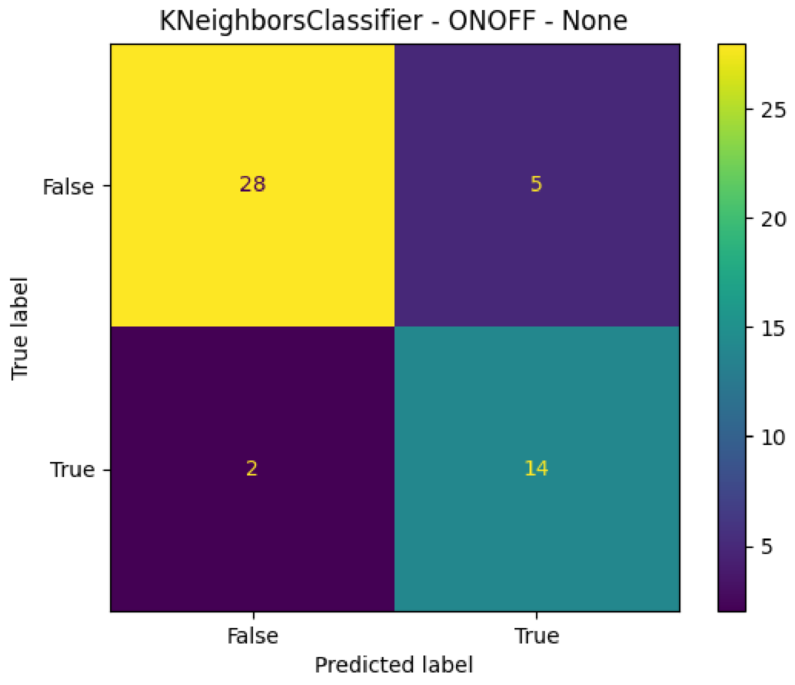

Linear SVM - 2_CIRCUITOFISICO - None
True label
False
True
5
0
1
11
False
True
Predicted label
10
8
6
4
2
0

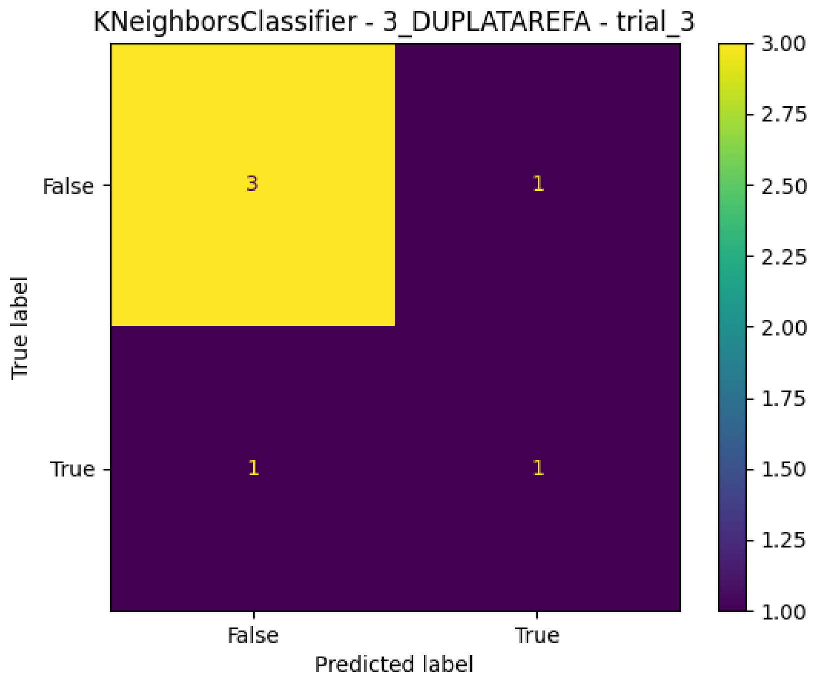

Linear SVM - 1_TUG - trial_3
True label
False
True
0
4
0
2
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

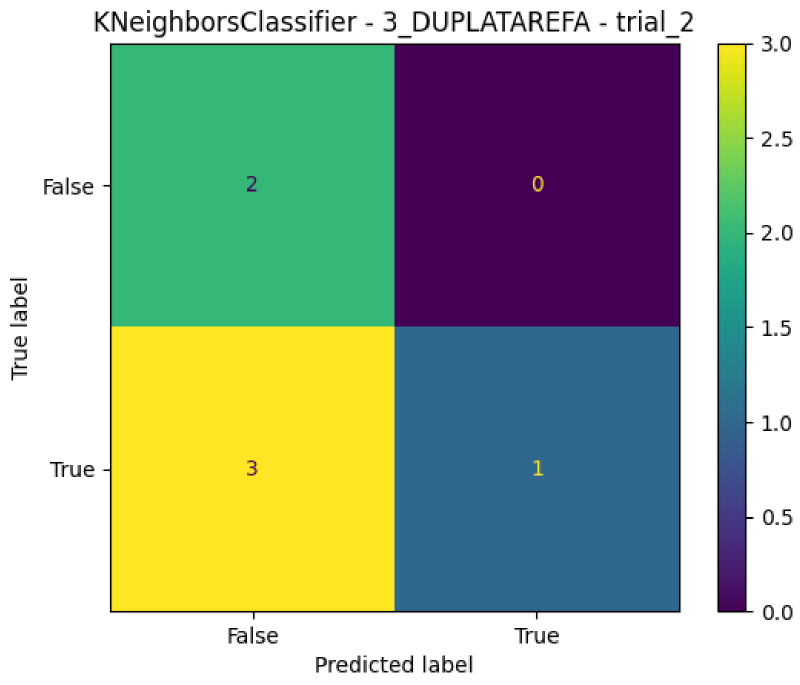

Linear SVM - 1_TUG - trial_2
True label
False
True
2
1
2
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

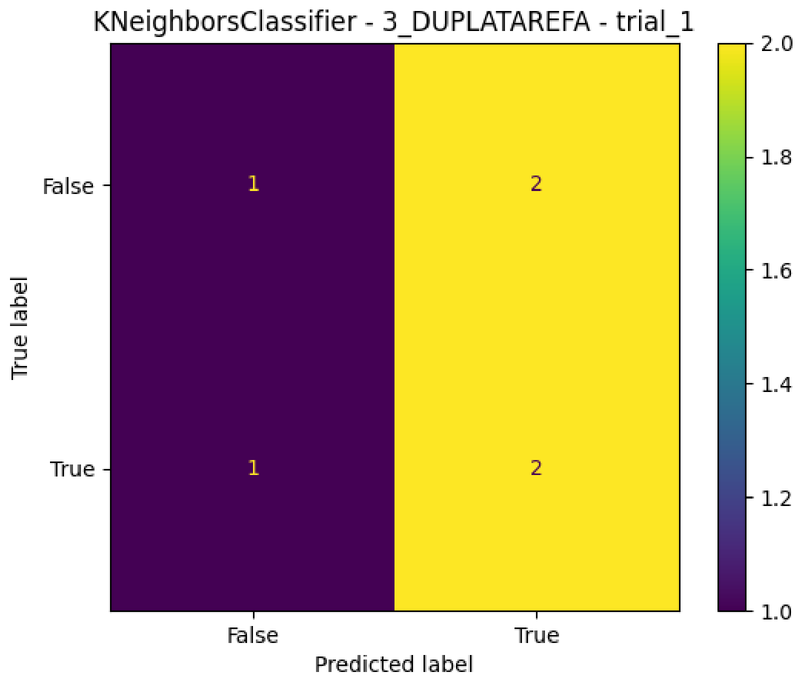

Linear SVM - 1_TUG - trial_1
True label
False
True
0
4
1
1
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

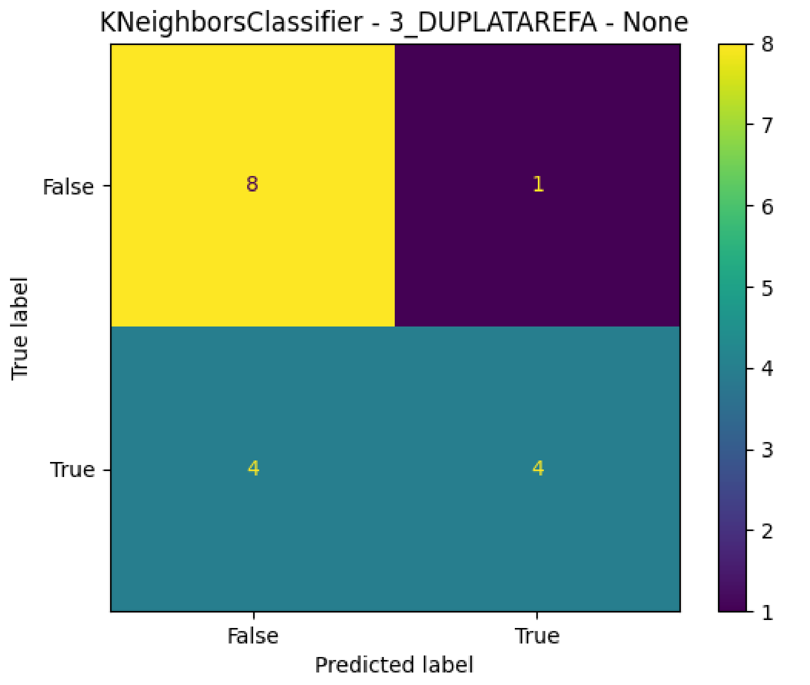

Linear SVM - 1_TUG - None
True label
False
True
4
3
1
8
Predicted label
False
True
8
7
6
5
4
3
2
1

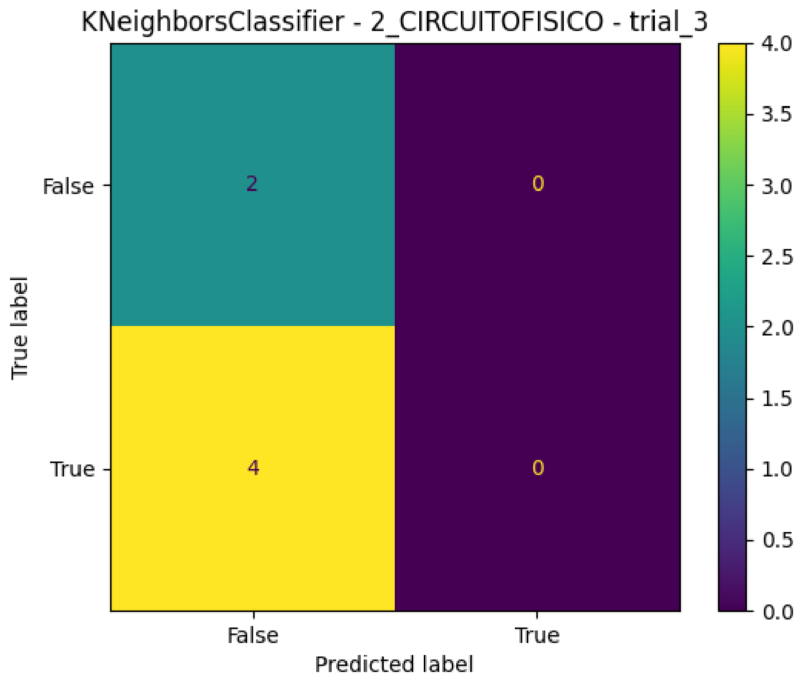

KNeighborsClassifier - ONOFF - None
True label
Predicted label
False
True
False
True
28
5
2
14
25
20
15
10
5

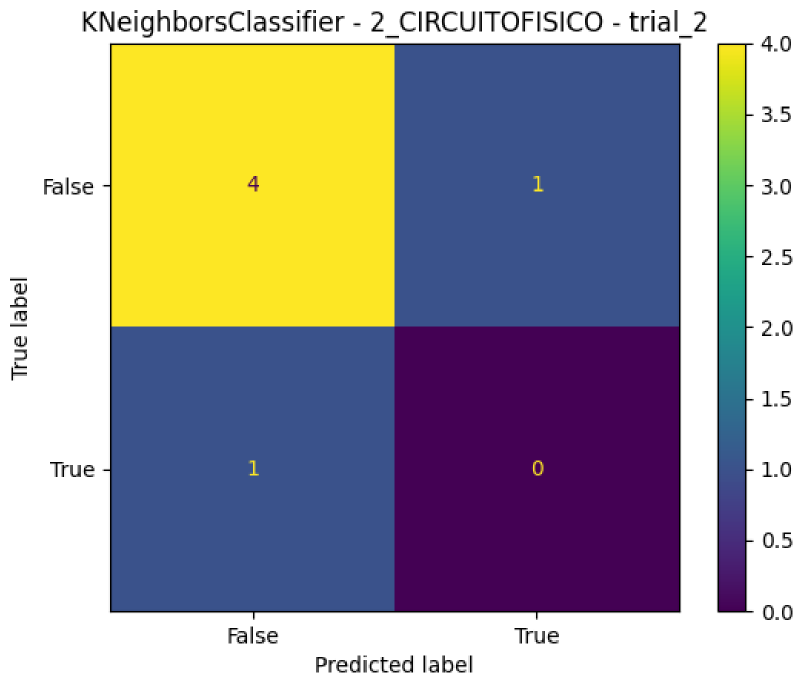

KNeighborsClassifier - 3_DUPLATAREFA - trial_3
True label
Predicted label
False
True
False
True
3
1
1
1
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

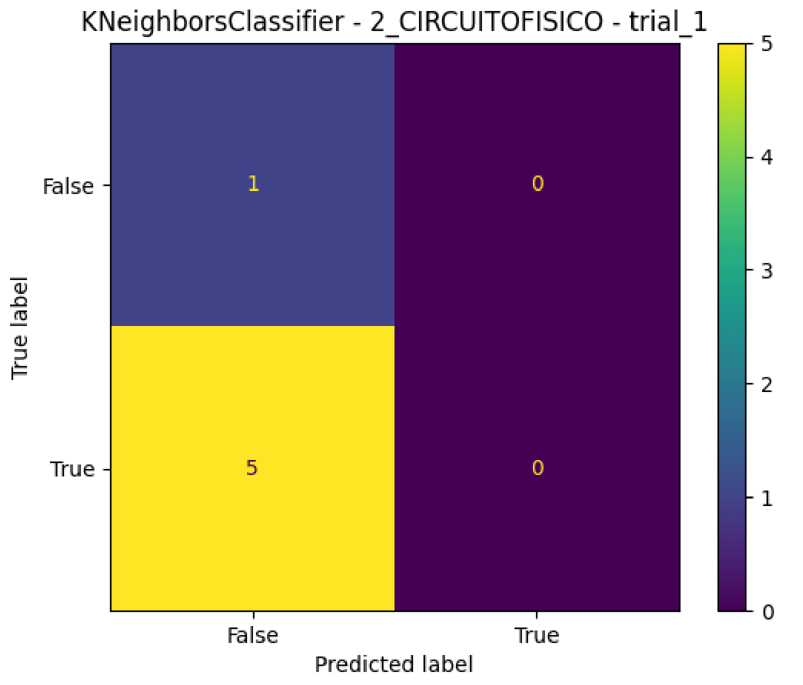

KNeighborsClassifier - 3_DUPLATAREFA - trial_2
True label
Predicted label
False
True
False
True
2
0
3
1
3.0
2.5
2.0
1.5
1.0
0.5
0.0

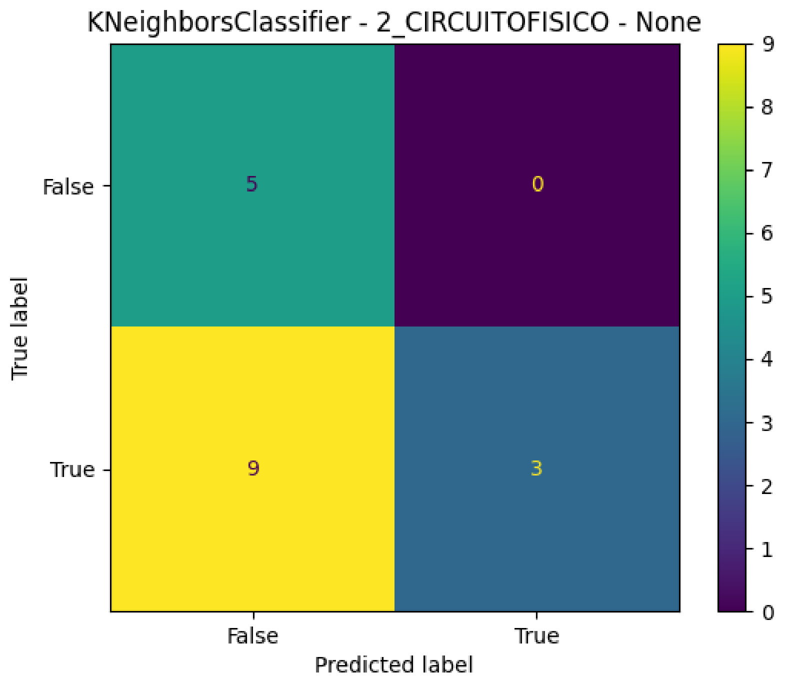

KNeighborsClassifier - 3_DUPLATAREFA - trial_1
True label
Predicted label
False
True
False
True
1
2
1
2
2.0
1.8
1.6
1.4
1.2
1.0

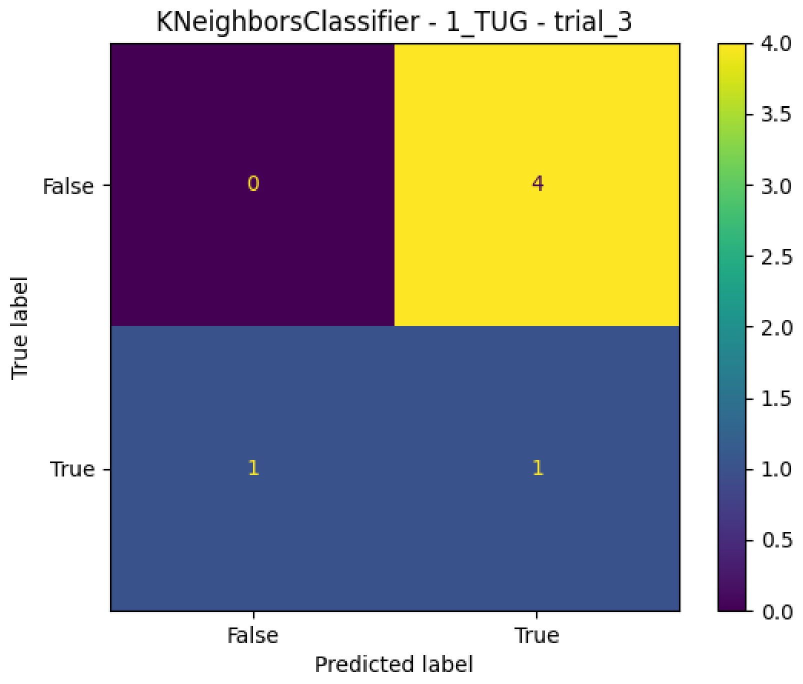

KNeighborsClassifier - 3_DUPLATAREFA - None
True label
False
True
Predicted label
False
True
8
1
4
4
1
2
3
4
5
6
7
8

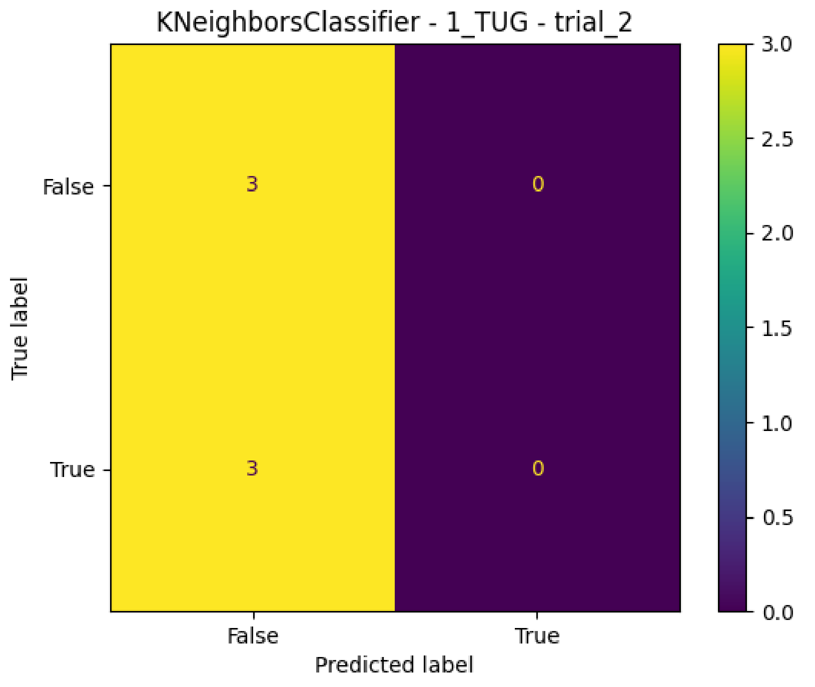

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_3
True label
False
True
Predicted label
False
True
2
0
4
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

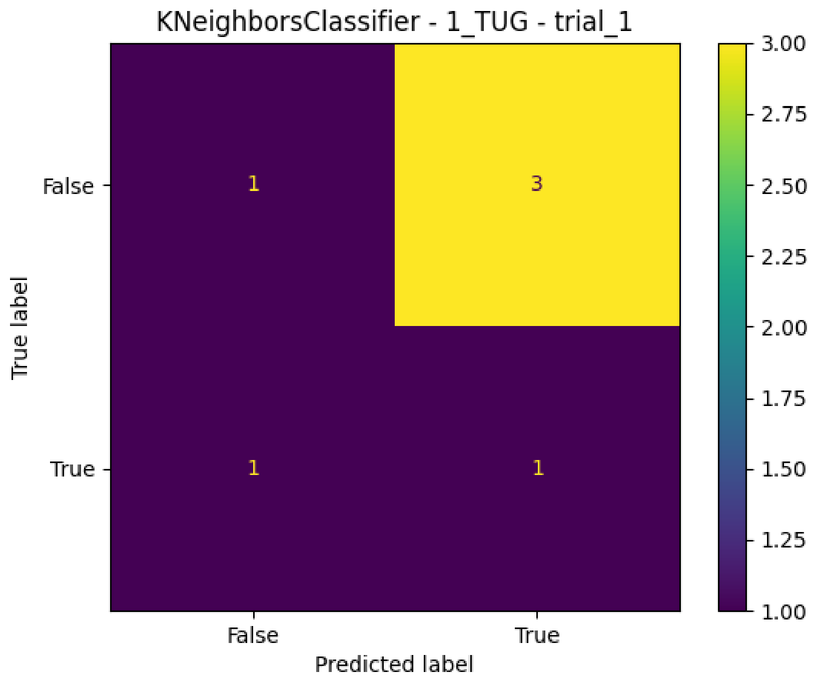

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
1
0
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

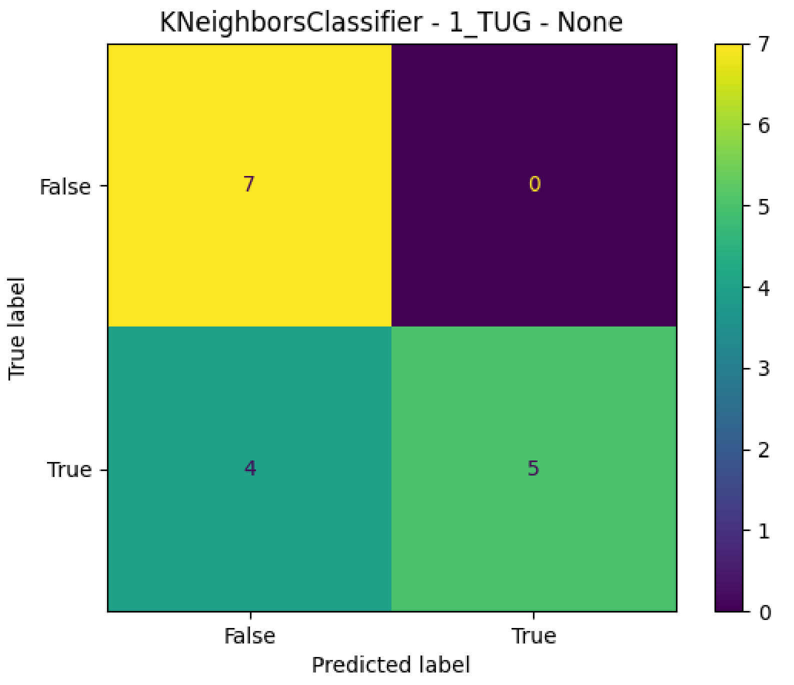

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
5
0
Predicted label
False
True
5
4
3
2
1
0

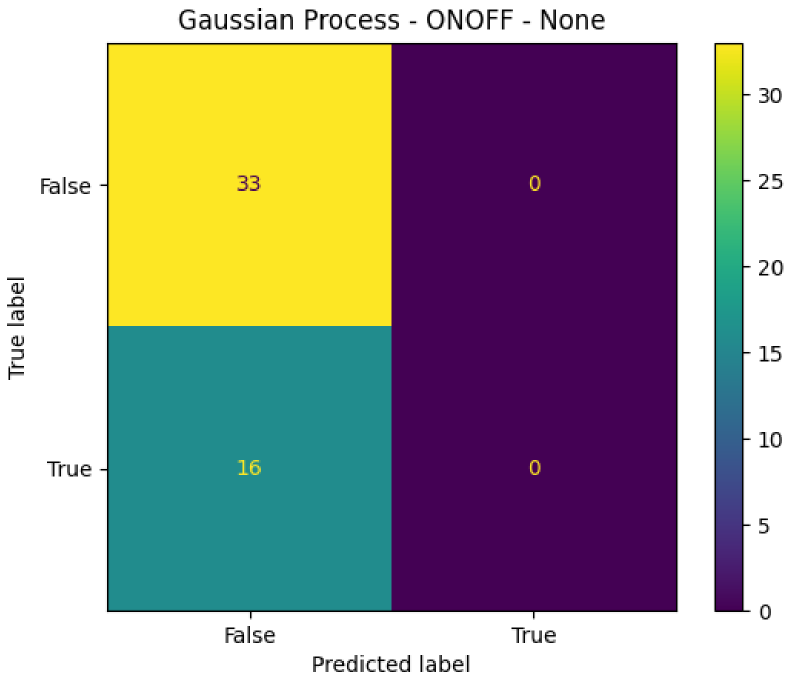

KNeighborsClassifier - 2_CIRCUITOFISICO - None
True label
False
True
Predicted label
False
True
5
0
9
3
9
8
7
6
5
4
3
2
1
0

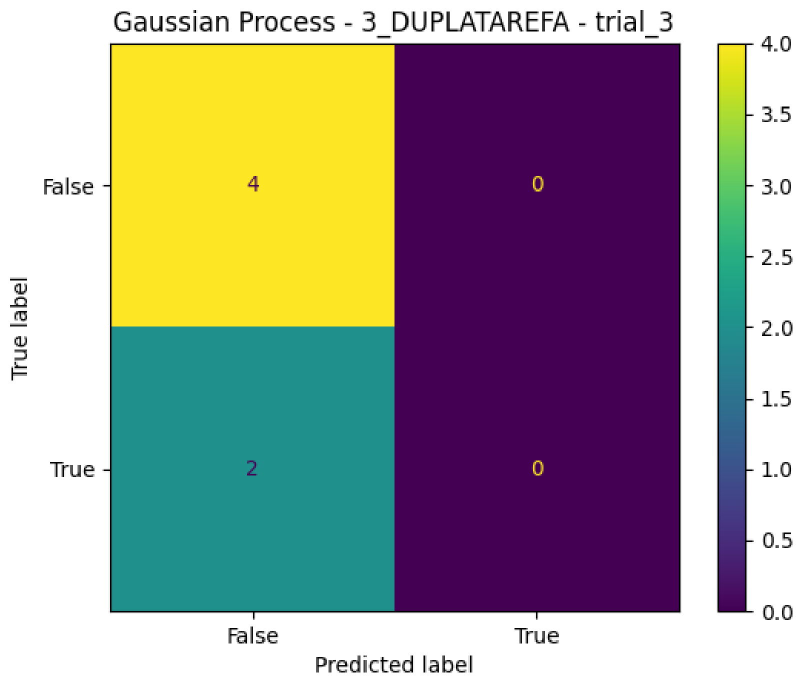

KNeighborsClassifier - 1_TUG - trial_3
True label
False
True
Predicted label
False
True
0
4
1
1
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

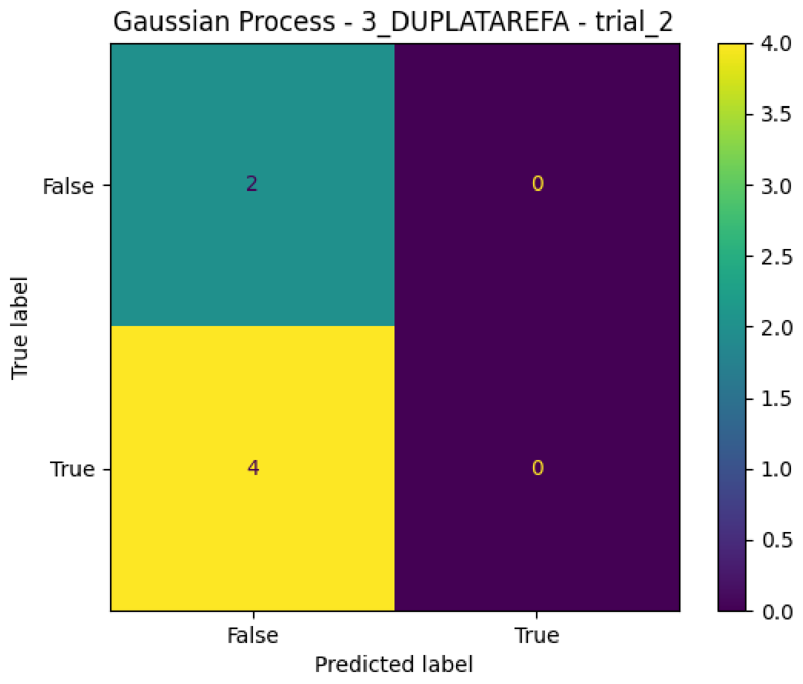

KNeighborsClassifier - 1_TUG - trial_2
True label
False
True
3
0
3
0
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

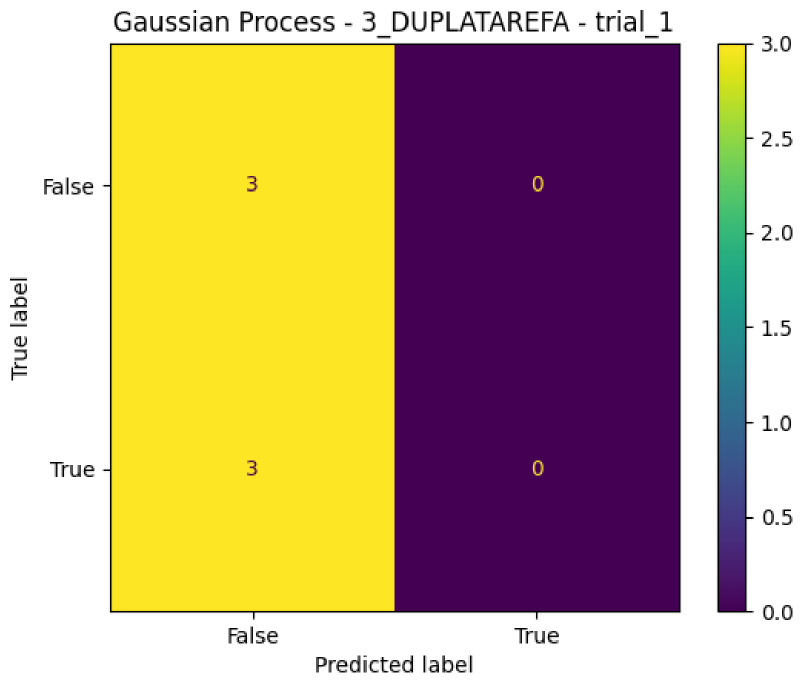

KNeighborsClassifier - 1_TUG - trial_1
True label
False
True
1
3
1
1
False
True
Predicted label
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

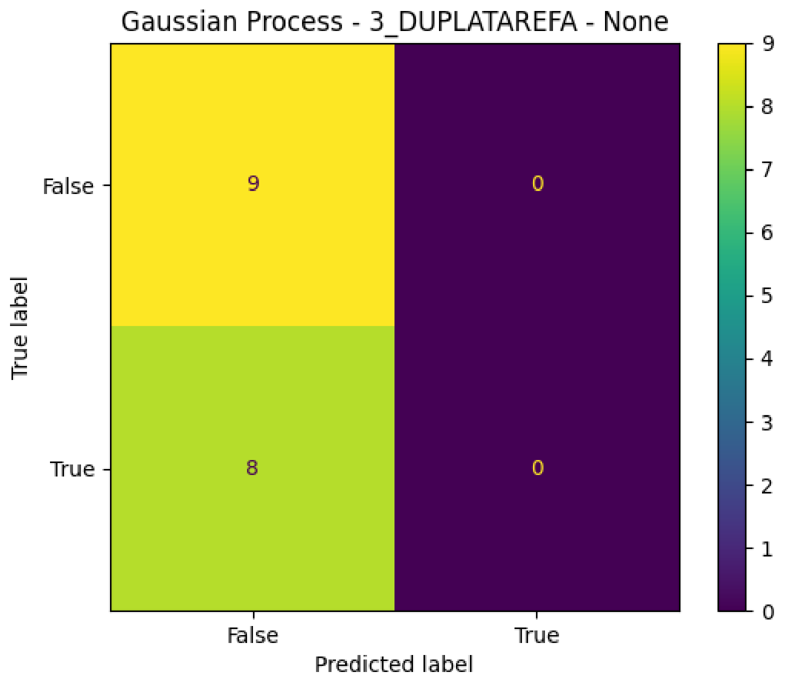

KNeighborsClassifier - 1_TUG - None
True label
False
True
7
0
4
5
Predicted label
False
True
0
1
2
3
4
5
6
7

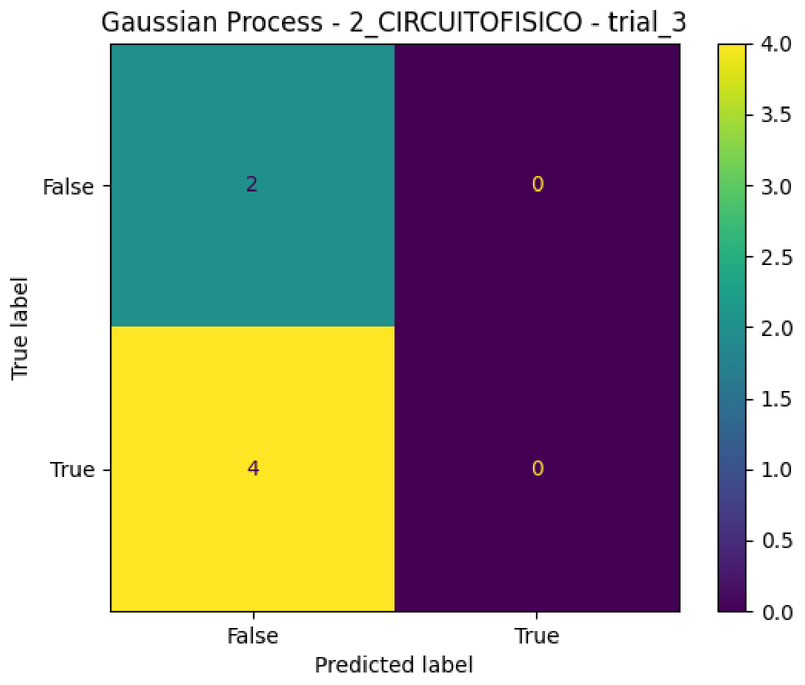

Gaussian Process - ONOFF - None
True label
False
True
33
0
16
0
Predicted label
False
True
0
5
10
15
20
25
30

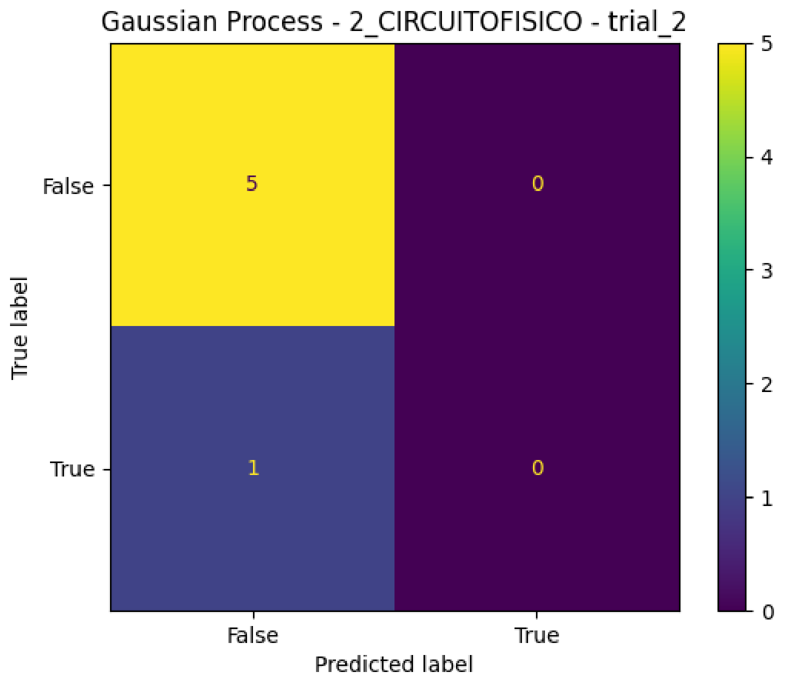

Gaussian Process - 3_DUPLATAREFA - trial_3
True label
False
True
4
0
2
0
Predicted label
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

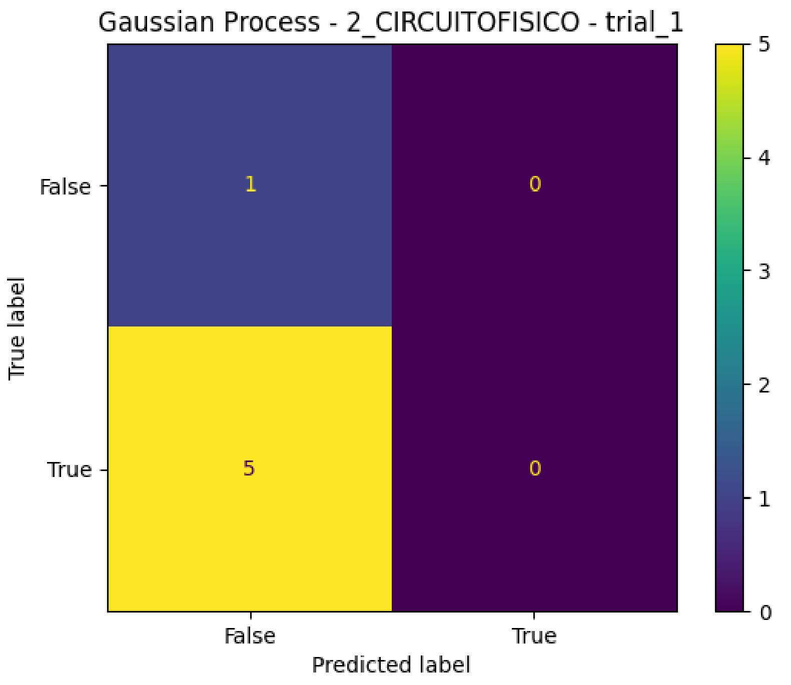

Gaussian Process - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
4
0
Predicted label
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

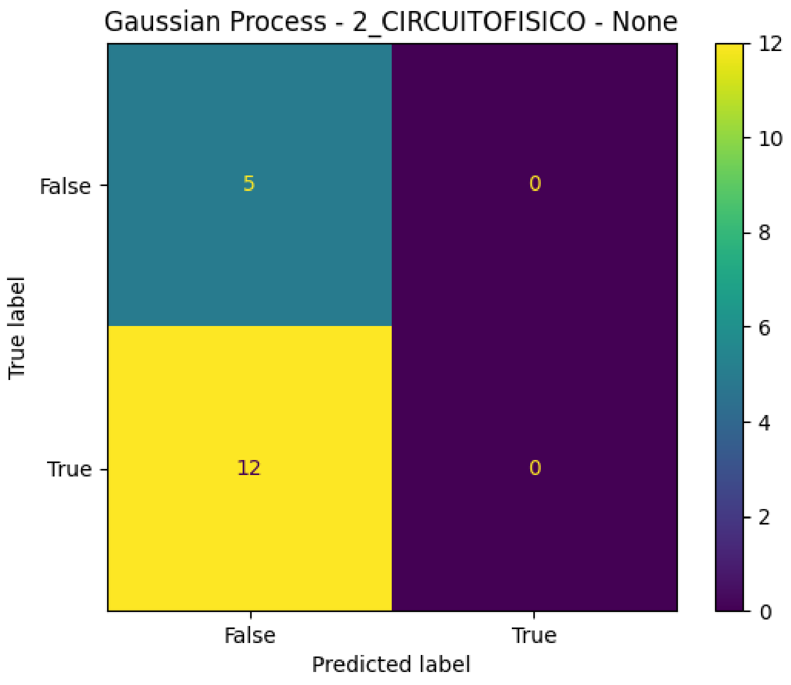

Gaussian Process - 3_DUPLATAREFA - trial_1
True label
False
True
3
0
3
0
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

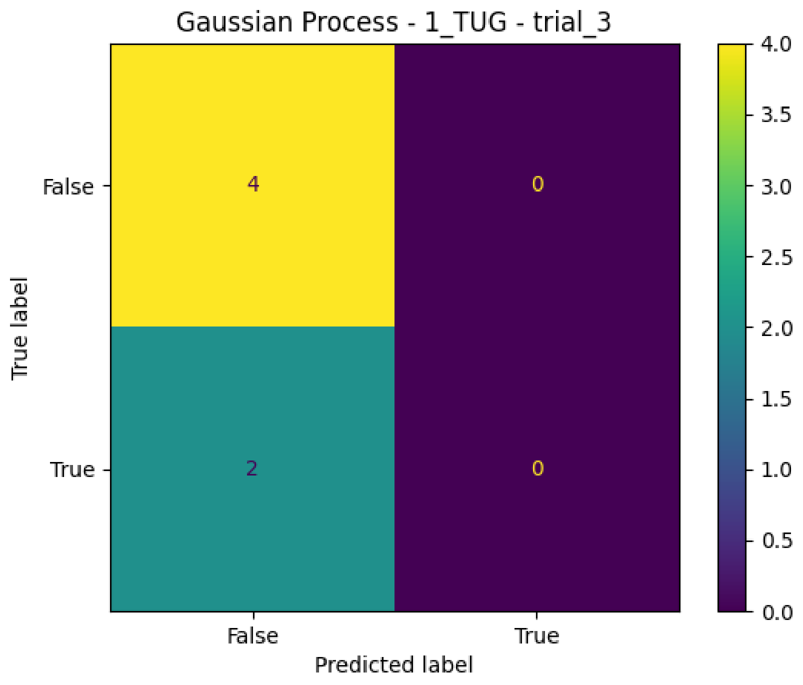

Gaussian Process - 3_DUPLATAREFA - None
True label
False
True
9
0
8
0
False
True
Predicted label
9
8
7
6
5
4
3
2
1
0

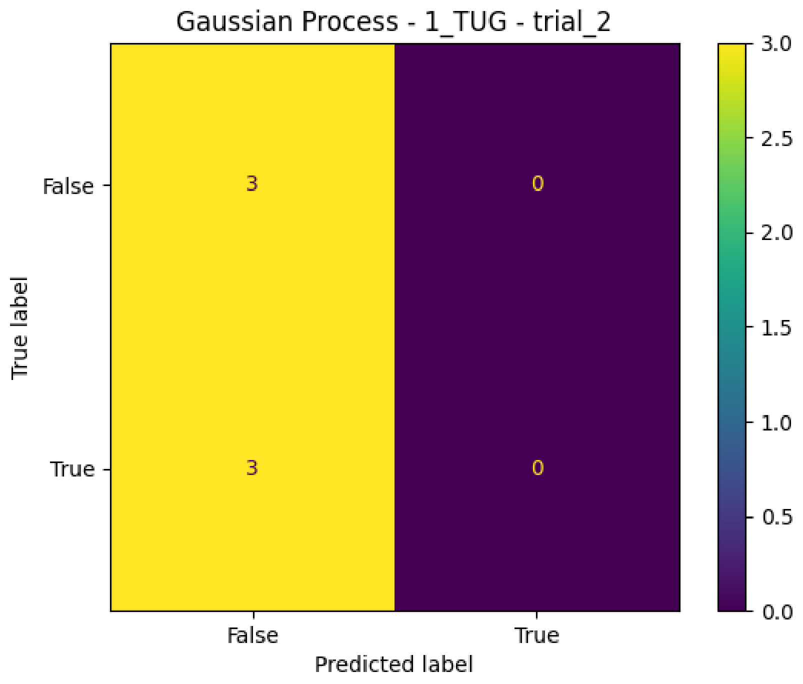

Gaussian Process - 2_CIRCUITOFISICO - trial_3
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

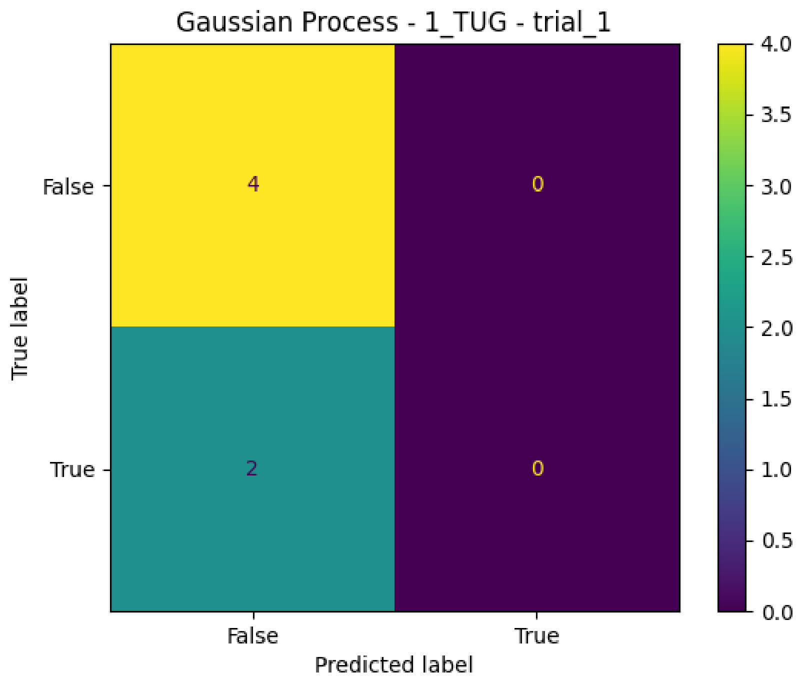

Gaussian Process - 2_CIRCUITOFISICO - trial_2
True label
False
True
5
0
1
0
False
True
Predicted label
5
4
3
2
1
0

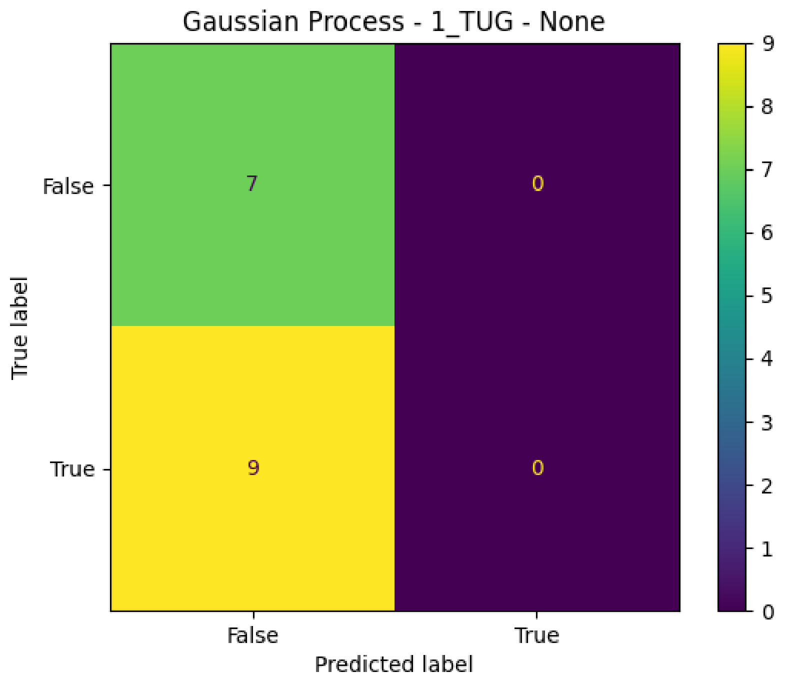

Gaussian Process - 2_CIRCUITOFISICO - trial_1
True label
Predicted label
False
True
False
True
1
0
5
0
0
1
2
3
4
5

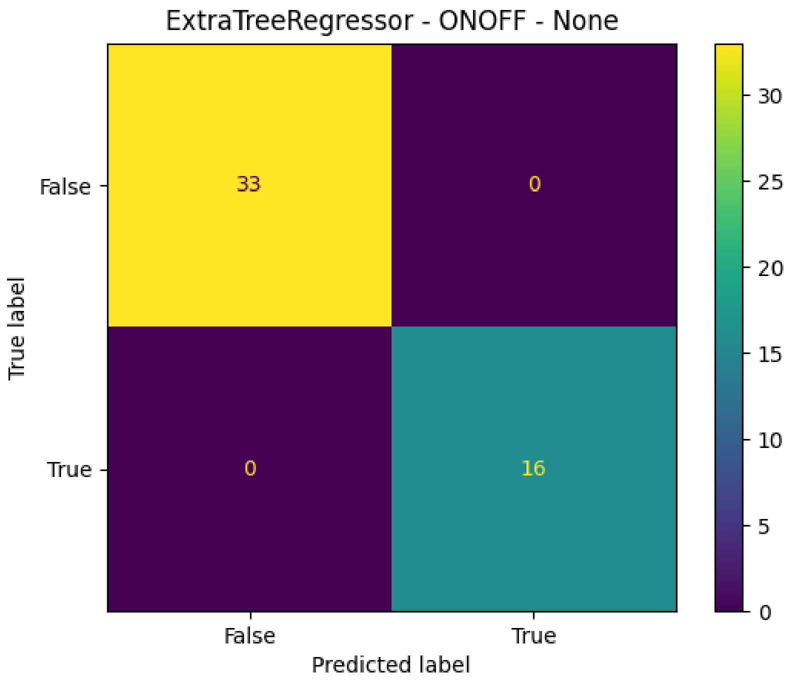

Gaussian Process - 2_CIRCUITOFISICO - None
True label
Predicted label
False
True
False
True
5
0
12
0
0
2
4
6
8
10
12

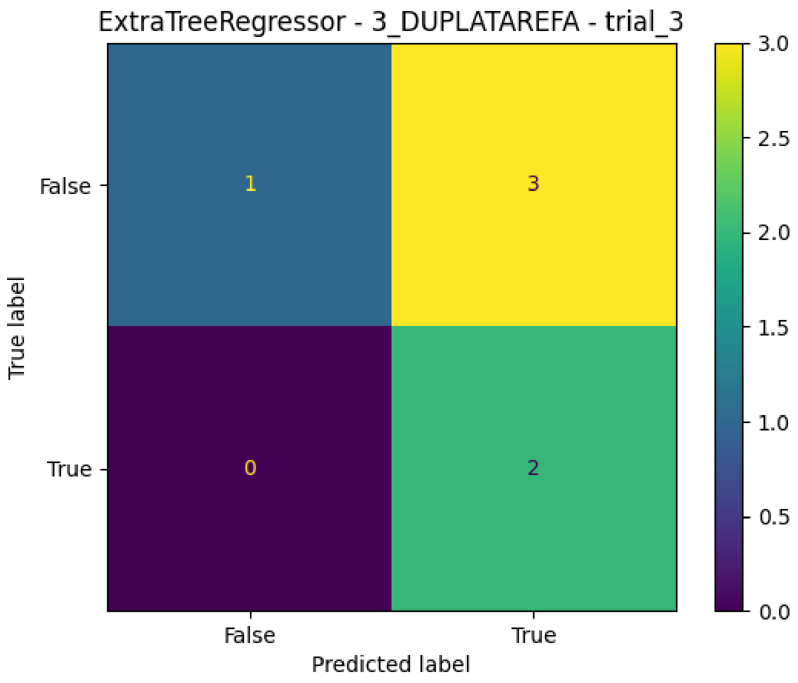

Gaussian Process - 1_TUG - trial_3
True label
False
True
4
0
2
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

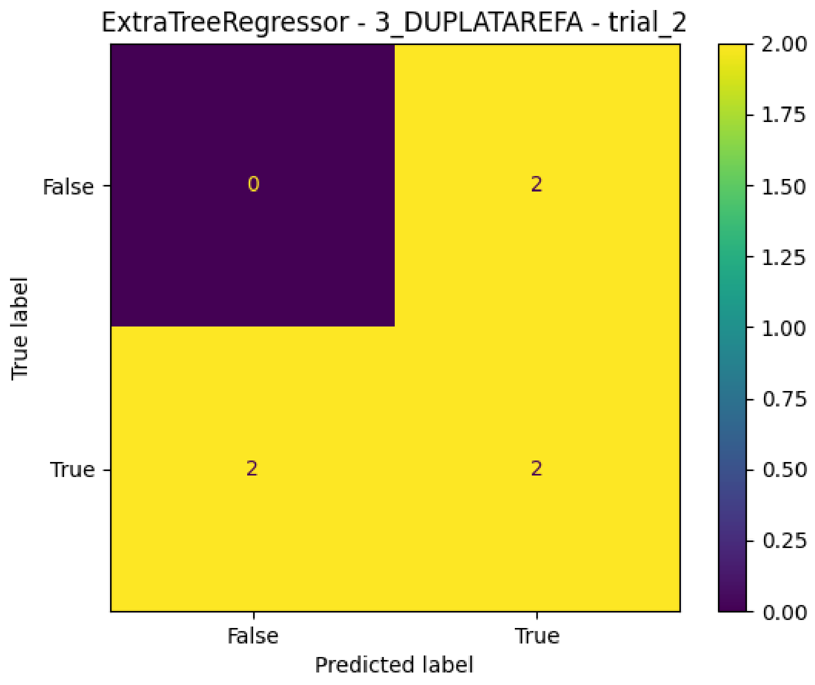

Gaussian Process - 1_TUG - trial_2
True label
False
True
3
0
3
0
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

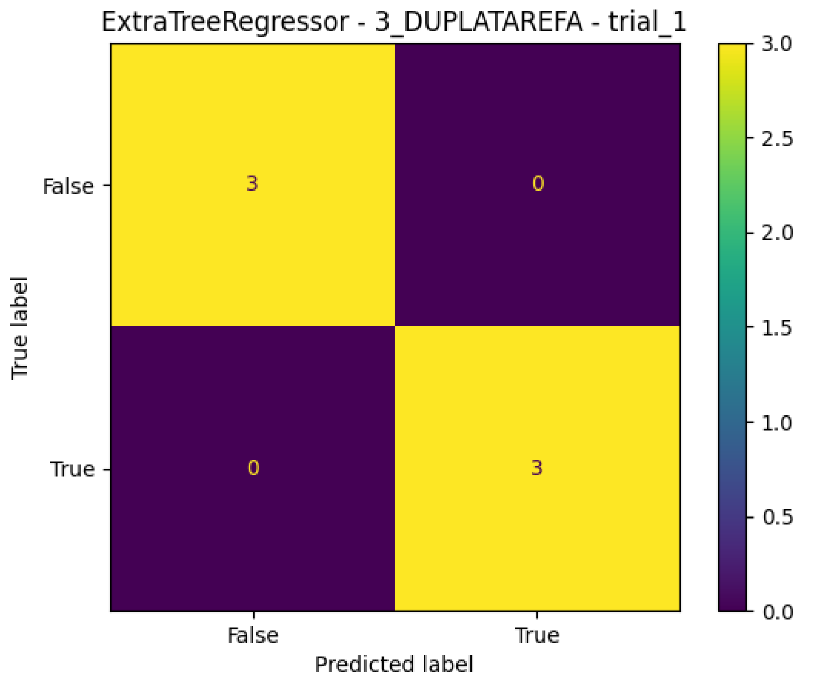

Gaussian Process - 1_TUG - trial_1
True label
False
True
Predicted label
4
0
2
0
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

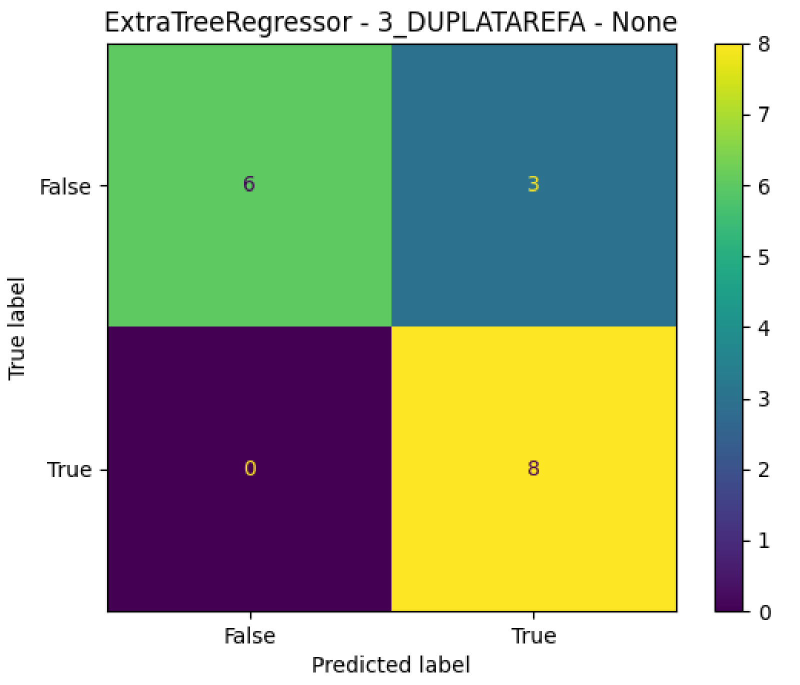

Gaussian Process - 1_TUG - None
True label
False
True
Predicted label
7
0
9
0
9
8
7
6
5
4
3
2
1
0

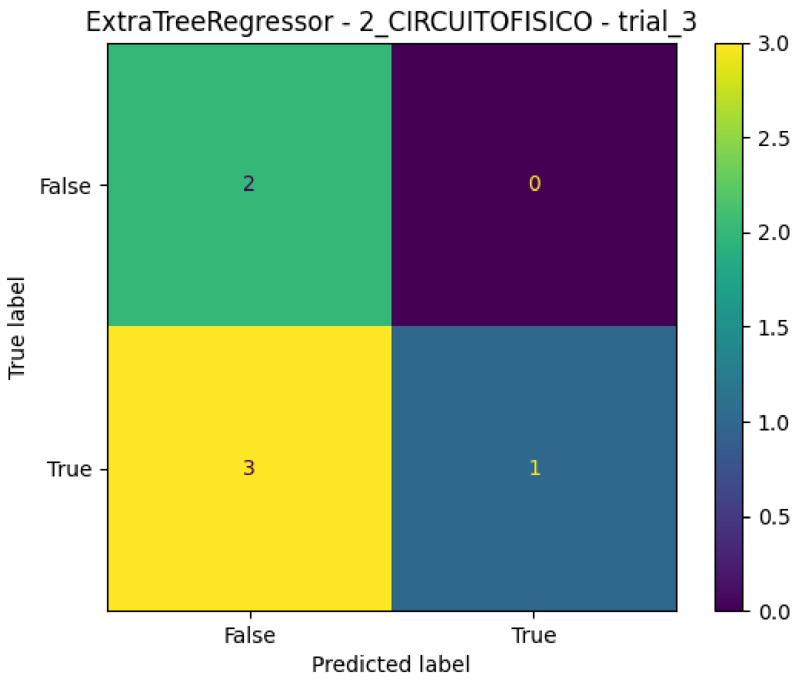

ExtraTreeRegressor - ONOFF - None
True label
False
True
33
0
0
16
Predicted label
False
True
30
25
20
15
10
5
0

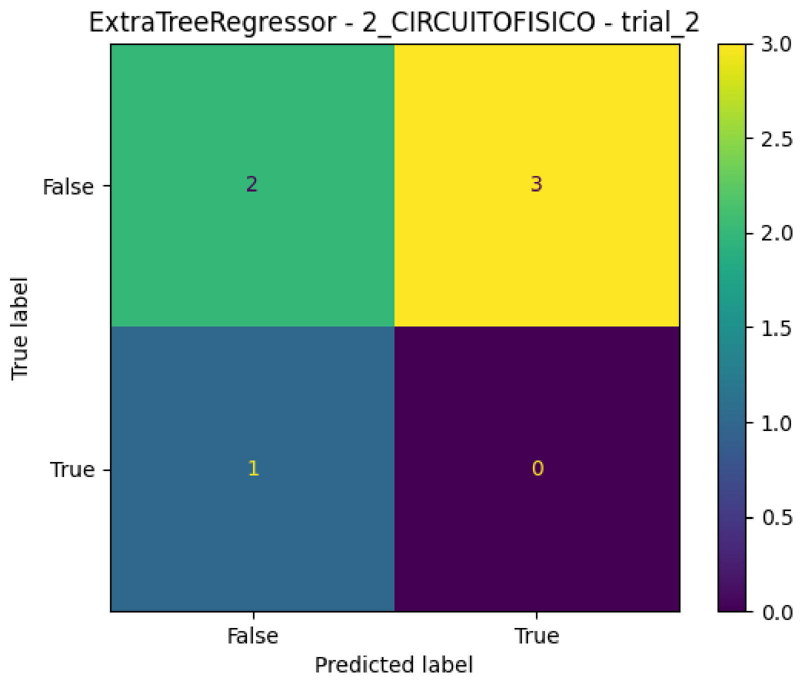

ExtraTreeRegressor - 3_DUPLATAREFA - trial_3
True label
False
True
1
3
0
2
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

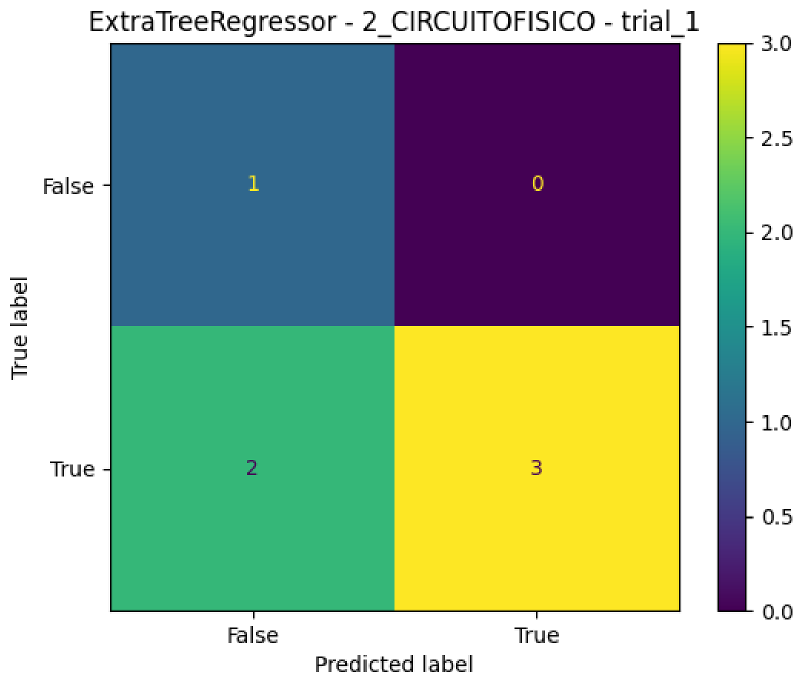

ExtraTreeRegressor - 3_DUPLATAREFA - trial_2
True label
False
True
0
2
2
2
False
True
Predicted label
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

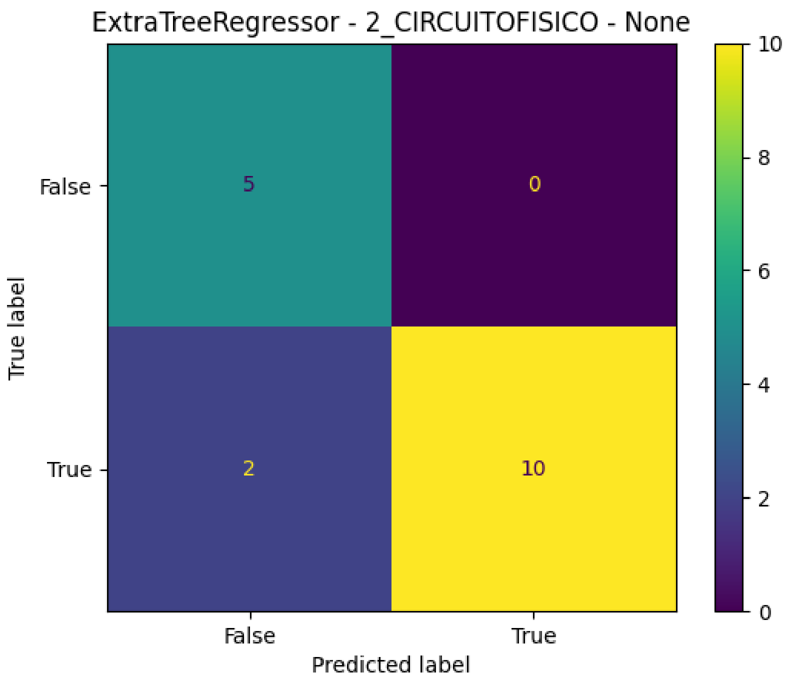

ExtraTreeRegressor - 3_DUPLATAREFA - trial_1
True label
False
True
3
0
0
3
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

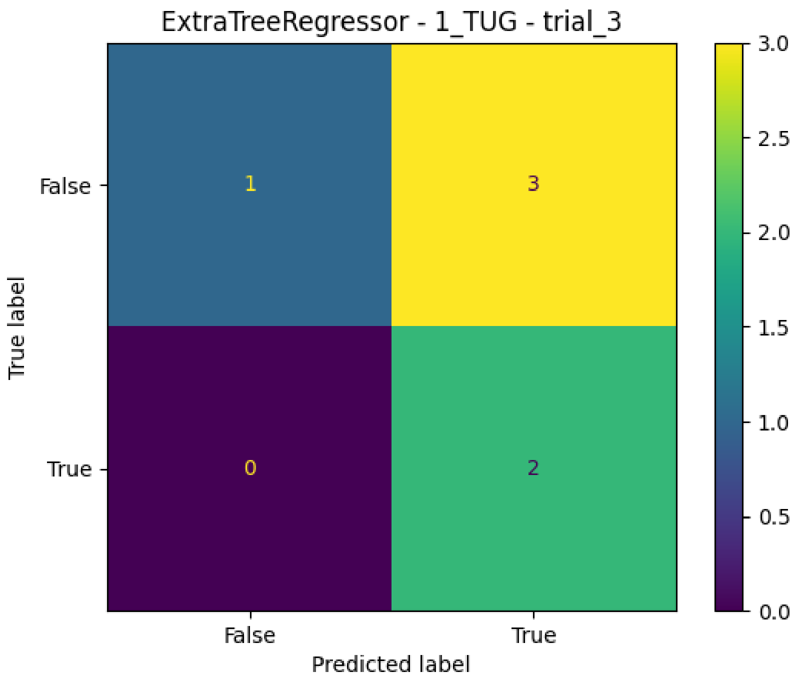

ExtraTreeRegressor - 3_DUPLATAREFA - None
True label
Predicted label
False
True
6
3
0
8
0
1
2
3
4
5
6
7
8

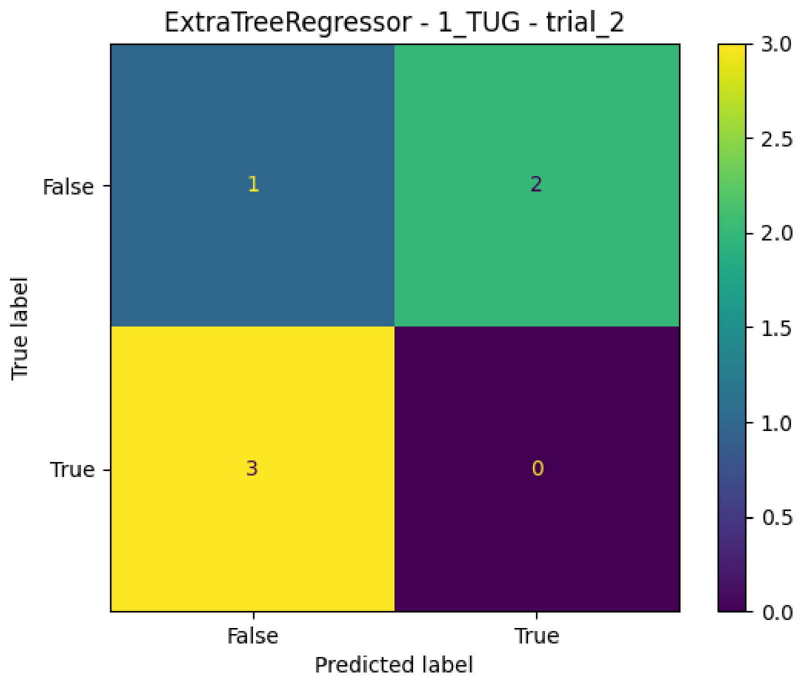

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_3
True label
Predicted label
False
True
2
0
3
1
0.0
0.5
1.0
1.5
2.0
2.5
3.0

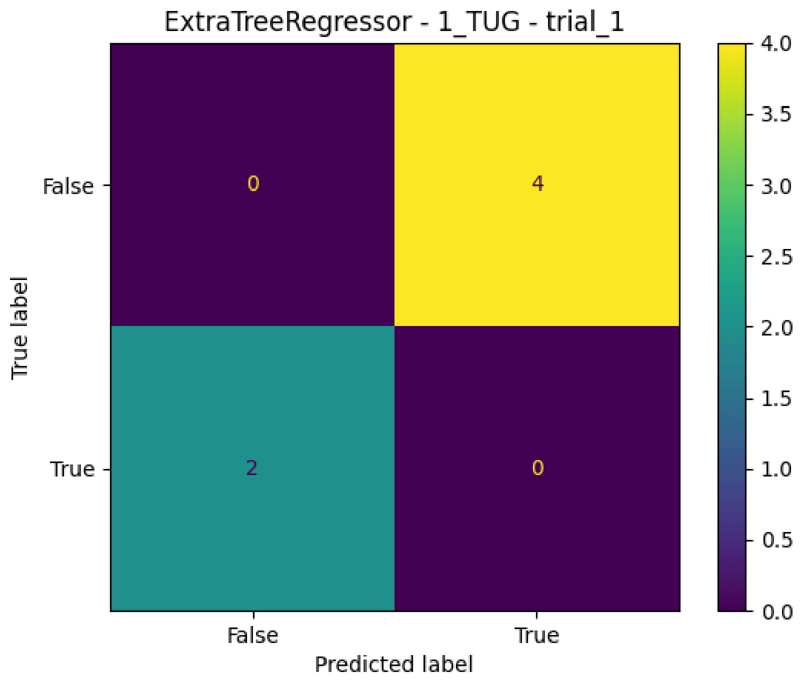

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_2
True label
False
True
2
3
1
0
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

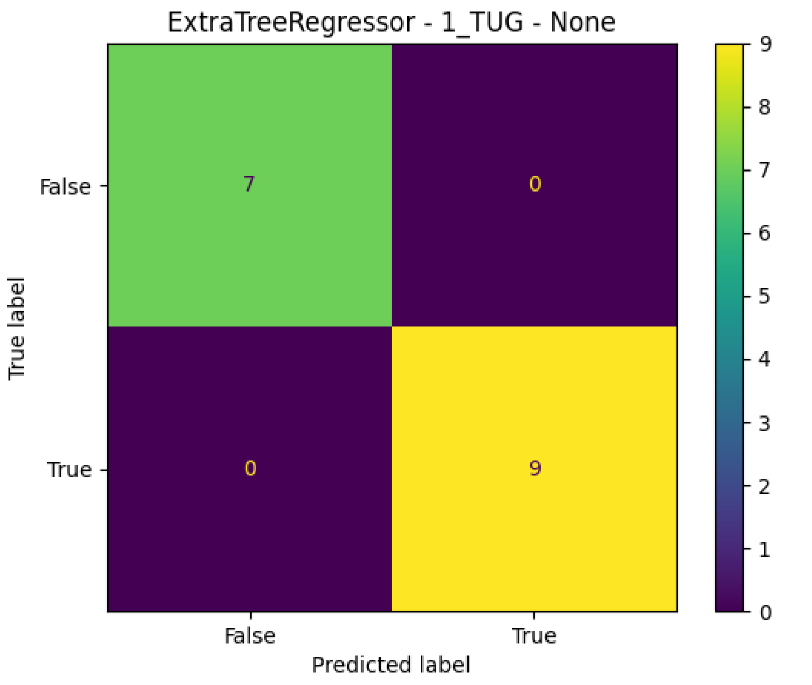

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
2
3
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

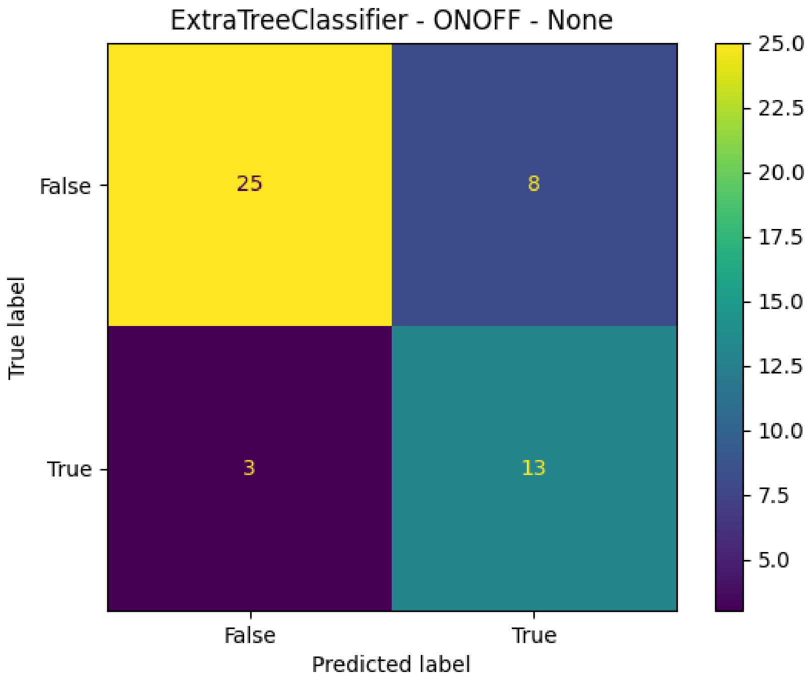

ExtraTreeRegressor - 2_CIRCUITOFISICO - None
True label
False
True
5
0
2
10
False
True
Predicted label
10
8
6
4
2
0

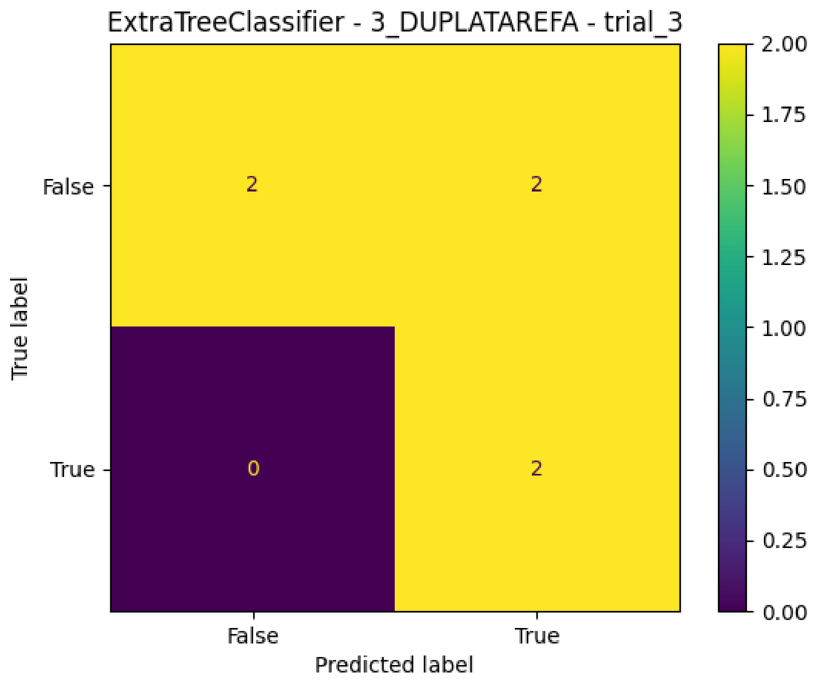

ExtraTreeRegressor - 1_TUG - trial_3
True label
False
True
1
3
0
2
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

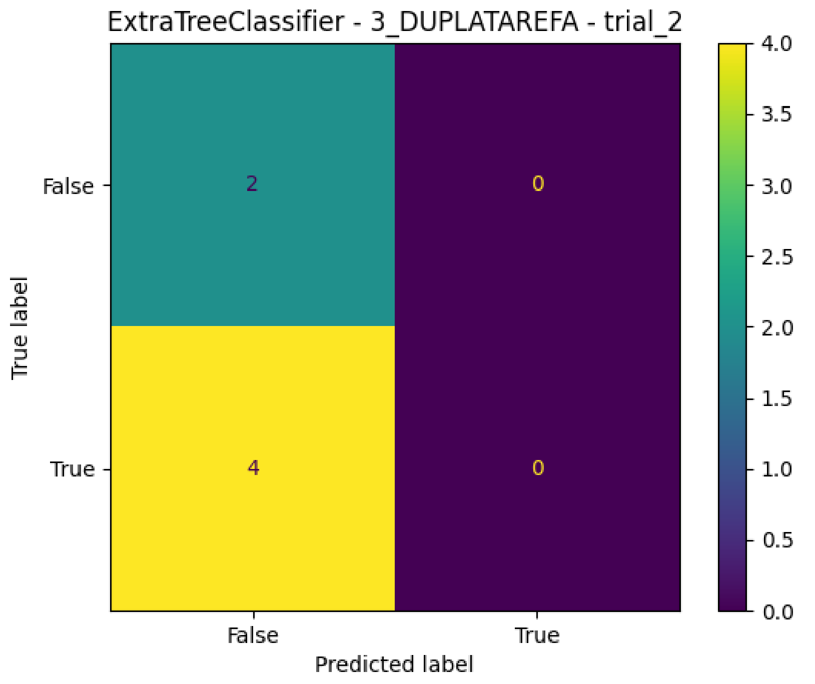

ExtraTreeRegressor - 1_TUG - trial_2
True label
False
True
1
2
3
0
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

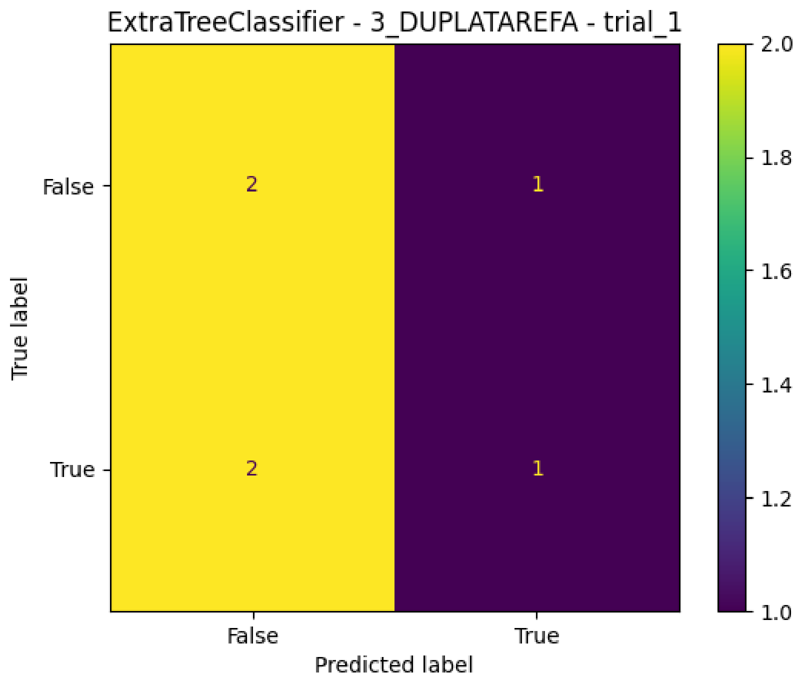

ExtraTreeRegressor - 1_TUG - trial_1
True label
False
True
0
4
2
0
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

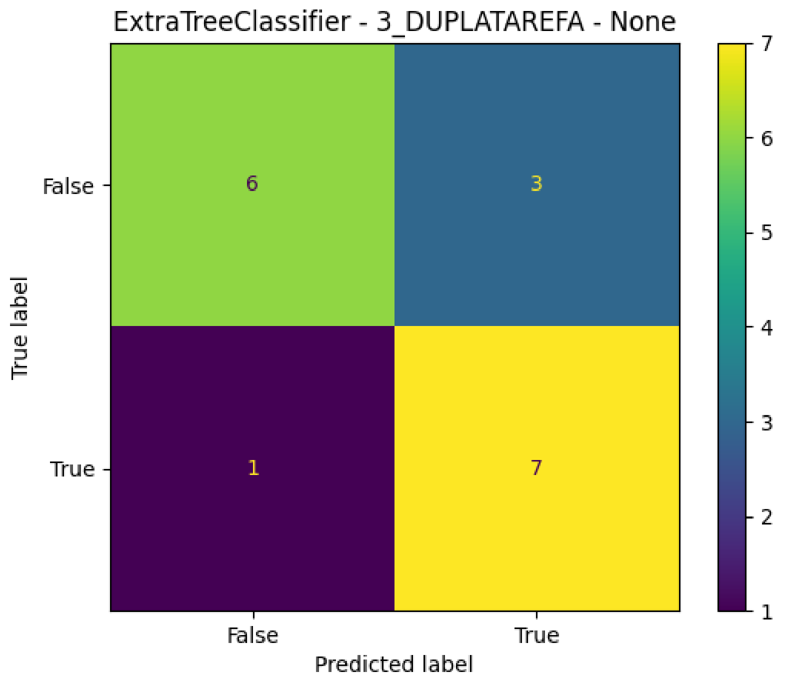

ExtraTreeRegressor - 1_TUG - None
True label
False
True
7
0
0
9
Predicted label
False
True
0
1
2
3
4
5
6
7
8
9

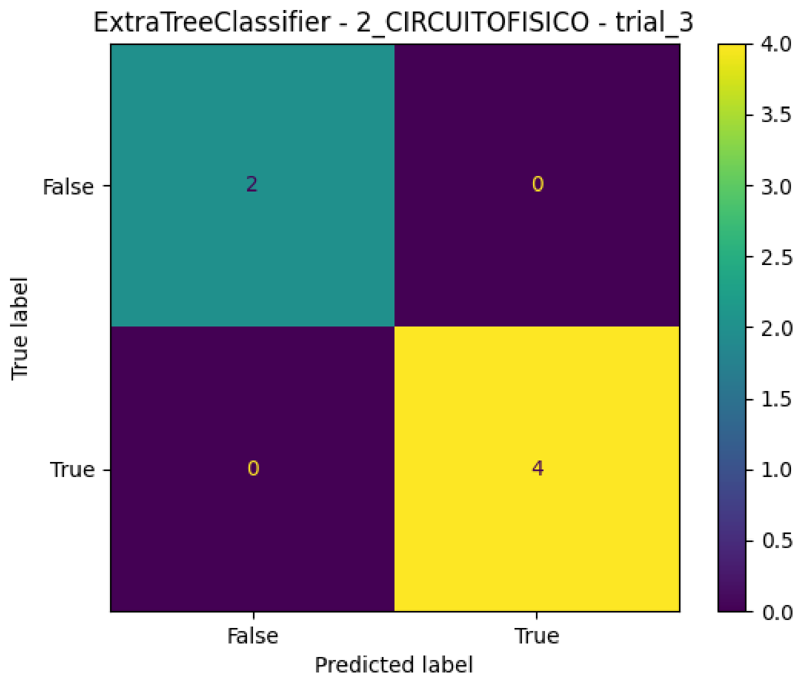

ExtraTreeClassifier - ONOFF - None
True label
False
True
25
8
3
13
Predicted label
False
True
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

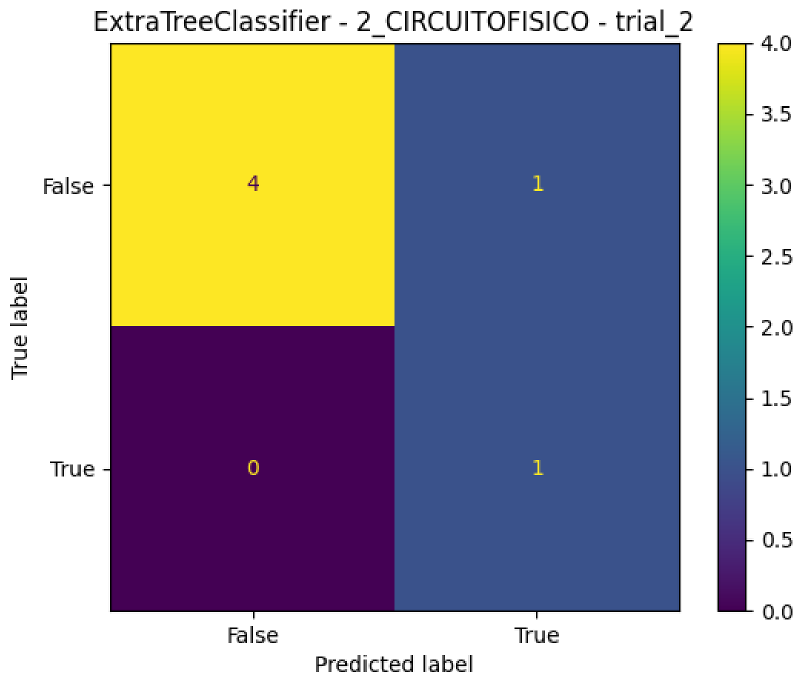

ExtraTreeClassifier - 3_DUPLATAREFA - trial_3
True label
False
True
2
2
0
2
False
True
Predicted label
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

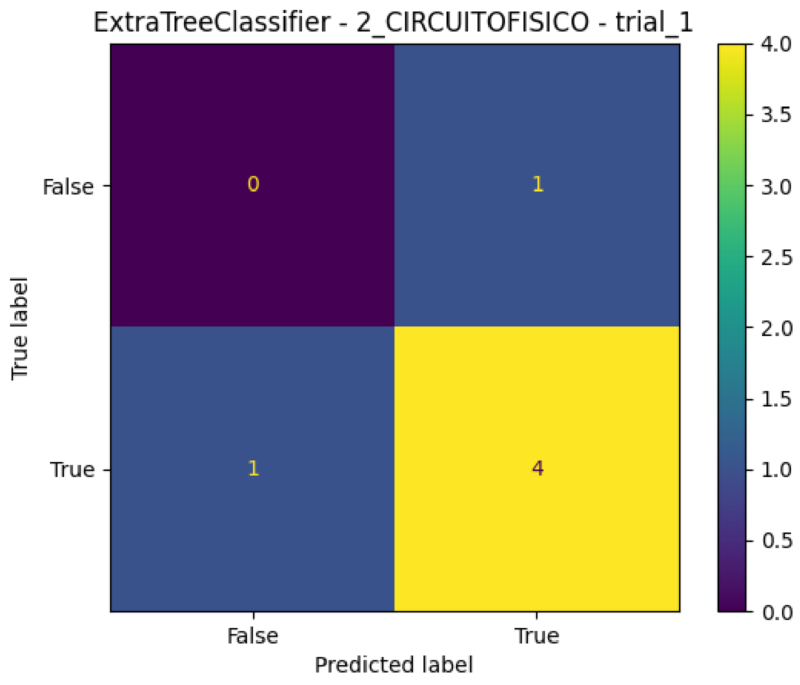

ExtraTreeClassifier - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

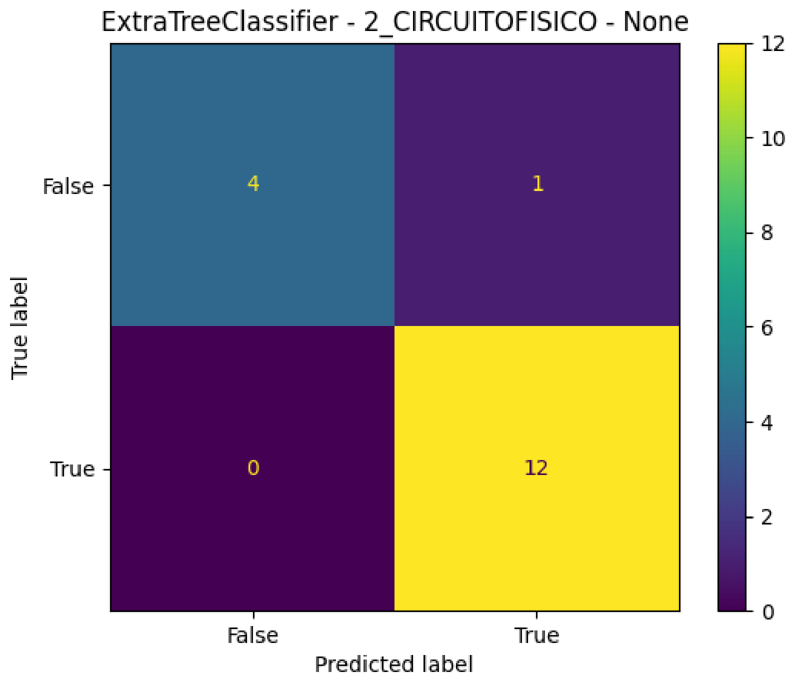

ExtraTreeClassifier - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
False
True
2
1
2
1
2.0
1.8
1.6
1.4
1.2
1.0

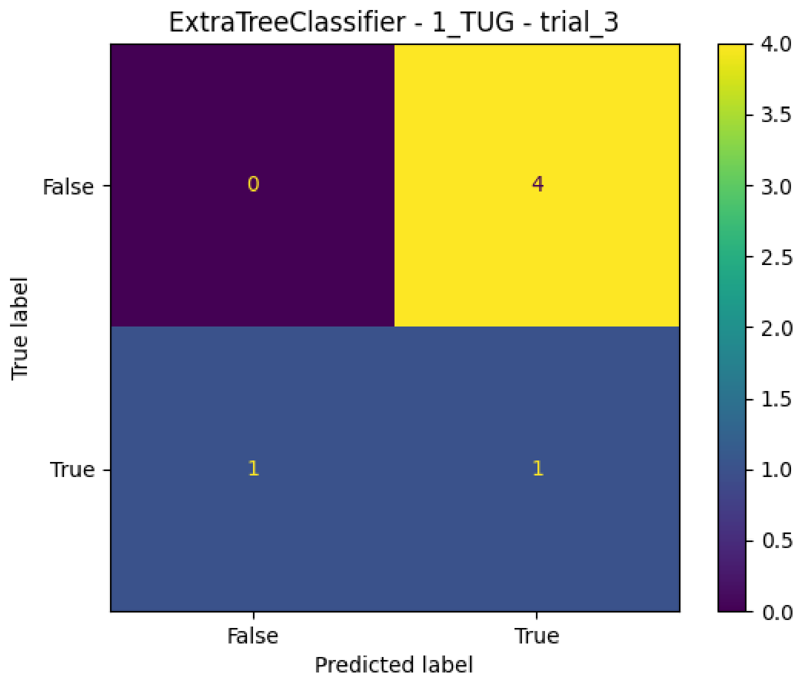

ExtraTreeClassifier - 3_DUPLATAREFA - None
True label
False
True
Predicted label
False
True
6
3
1
7
7
6
5
4
3
2
1

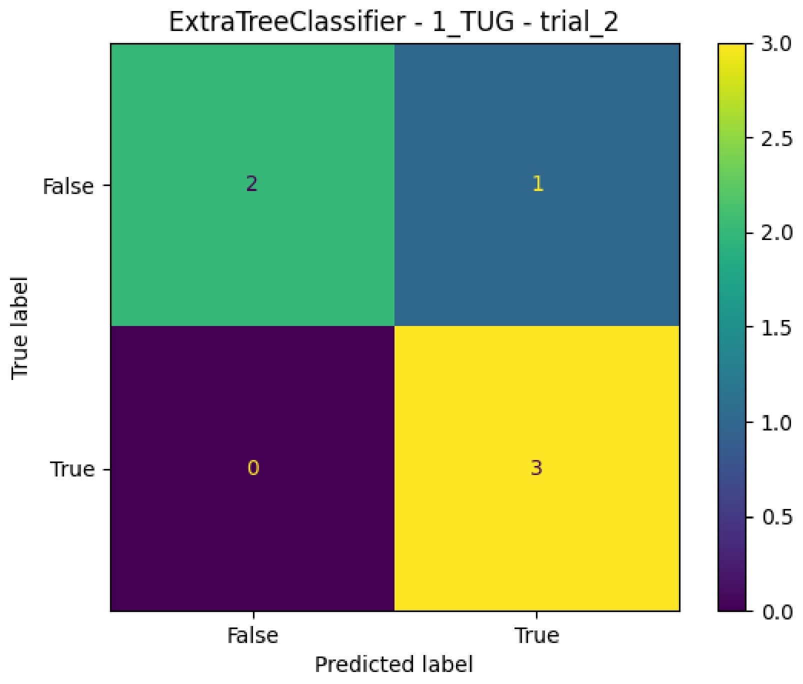

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_3
True label
False
True
2
0
0
4
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

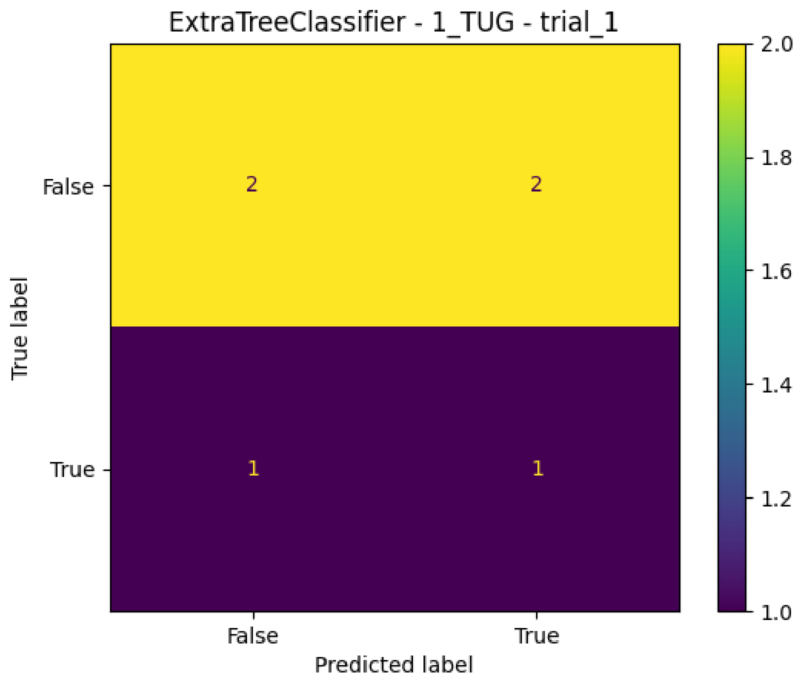

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
0
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

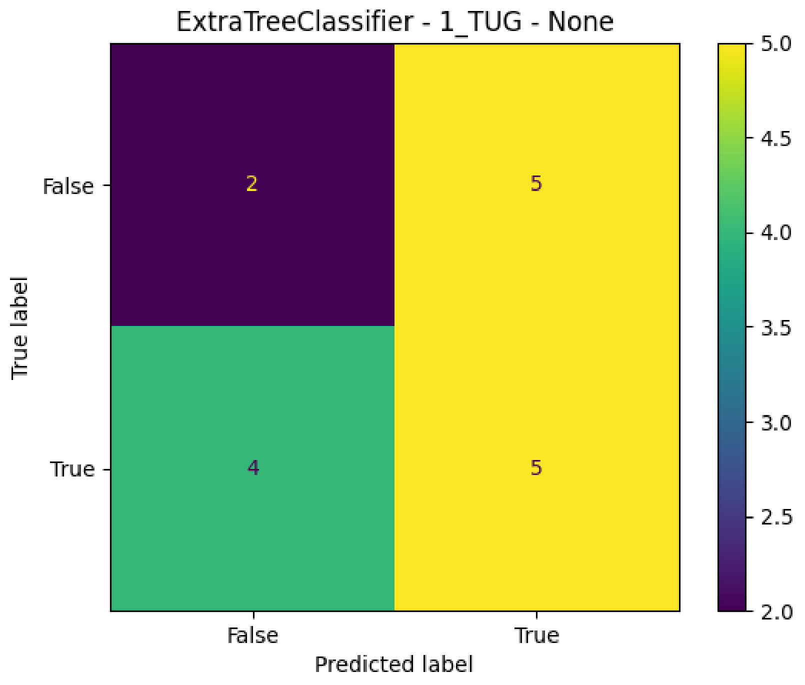

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_1
True label
False
True
0
1
1
4
False
True
Predicted label
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

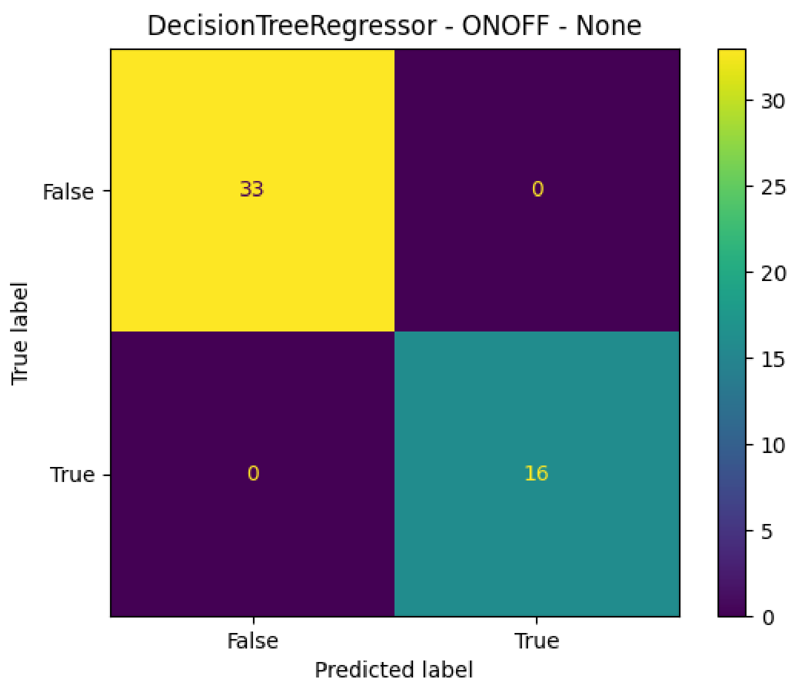

ExtraTreeClassifier - 2_CIRCUITOFISICO - None
True label
False
True
4
1
0
12
False
True
Predicted label
0
2
4
6
8
10
12

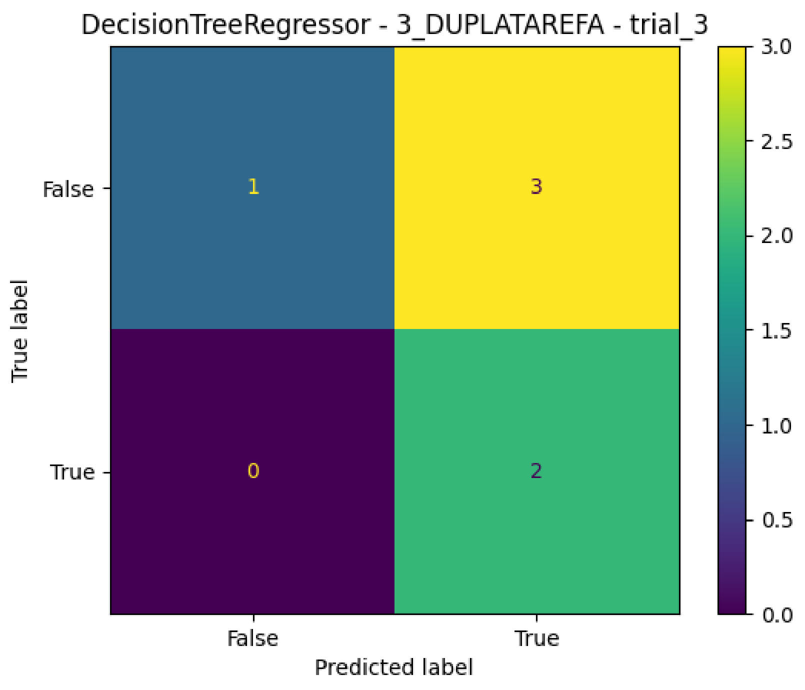

ExtraTreeClassifier - 1_TUG - trial_3
True label
Predicted label
False
True
False
True
0
4
1
1
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

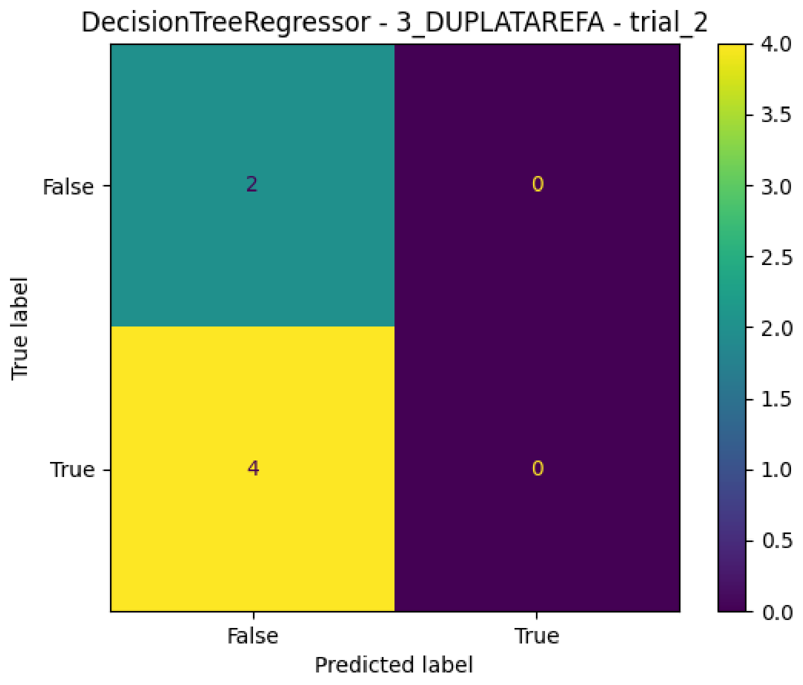

ExtraTreeClassifier - 1_TUG - trial_2
True label
Predicted label
False
True
False
True
2
1
0
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

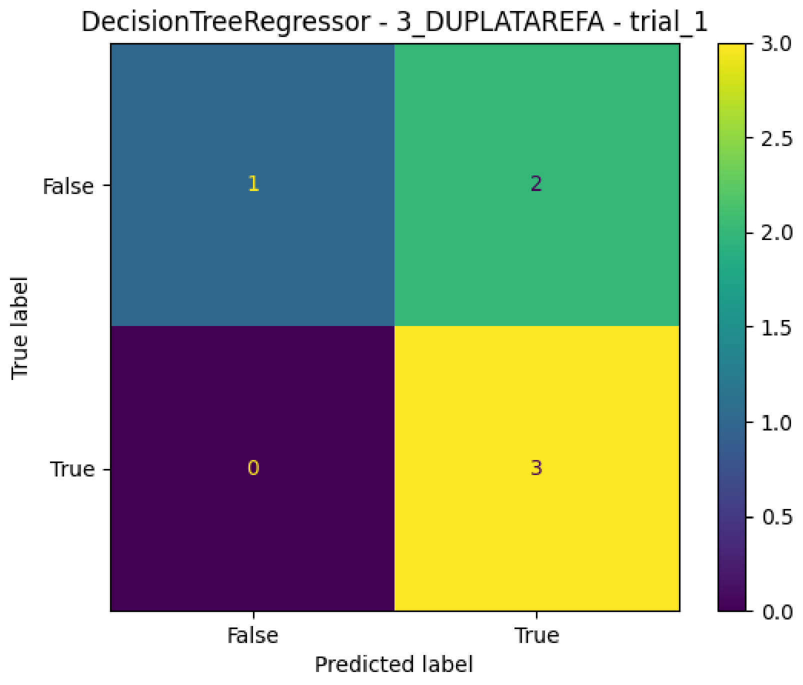

ExtraTreeClassifier - 1_TUG - trial_1
True label
False
True
2
2
1
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

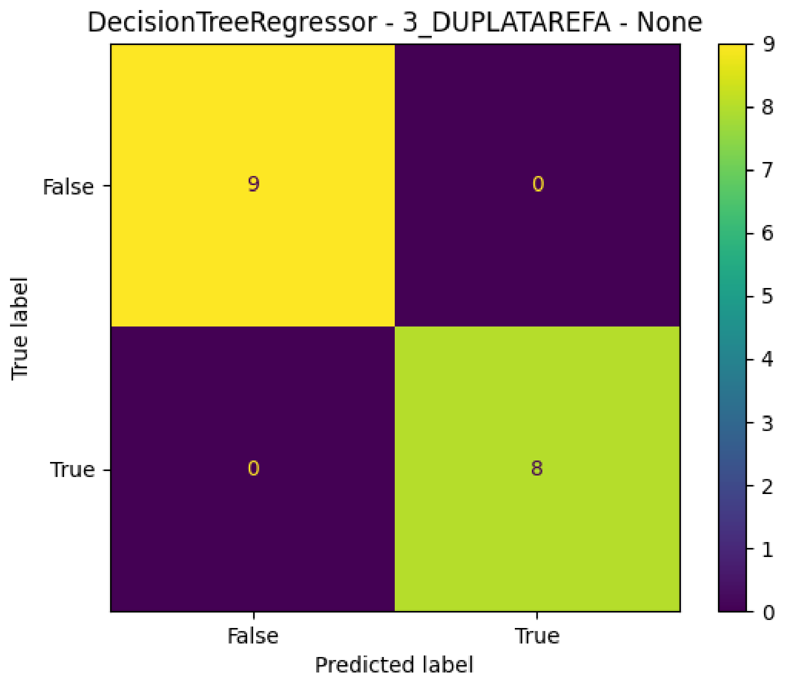

ExtraTreeClassifier - 1_TUG - None
True label
False
True
2
5
4
5
False
True
Predicted label
5.0
4.5
4.0
3.5
3.0
2.5
2.0

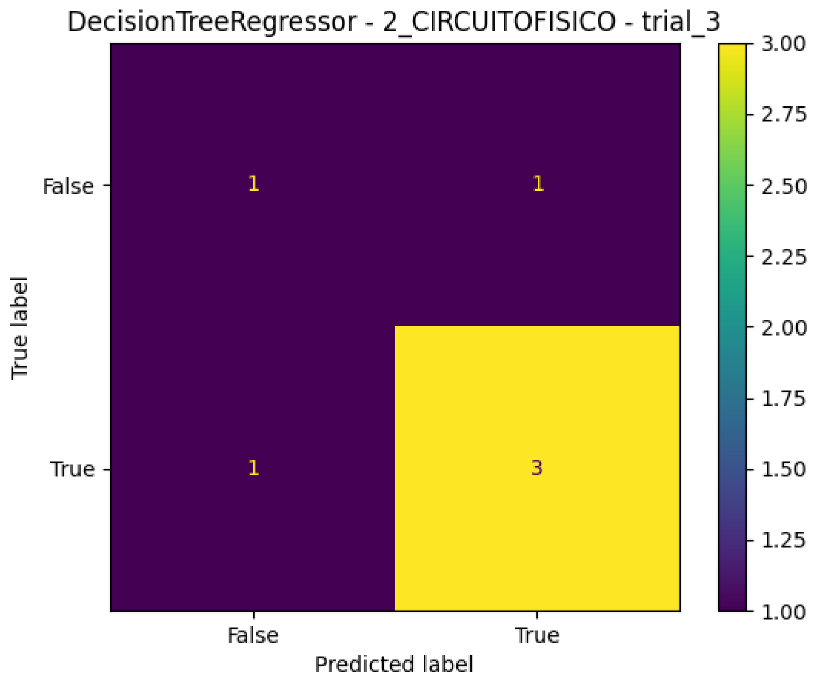

DecisionTreeRegressor - ONOFF - None
True label
False
True
33
0
0
16
Predicted label
False
True
0
5
10
15
20
25
30

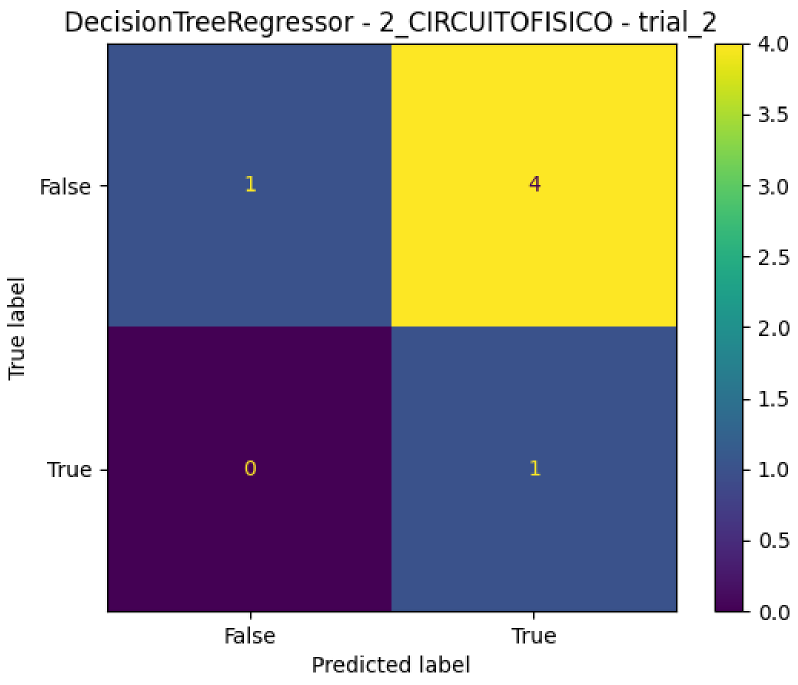

DecisionTreeRegressor - 3_DUPLATAREFA - trial_3
True label
False
True
1
3
0
2
Predicted label
False
True
0.0
0.5
1.0
1.5
2.0
2.5
3.0

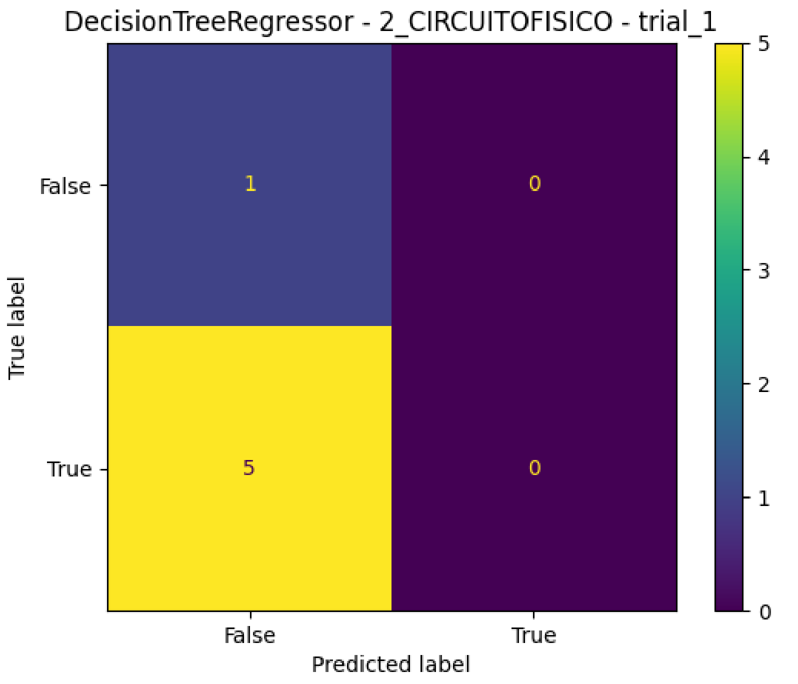

DecisionTreeRegressor - 3_DUPLATAREFA - trial_2
True label
False
True
2
0
4
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

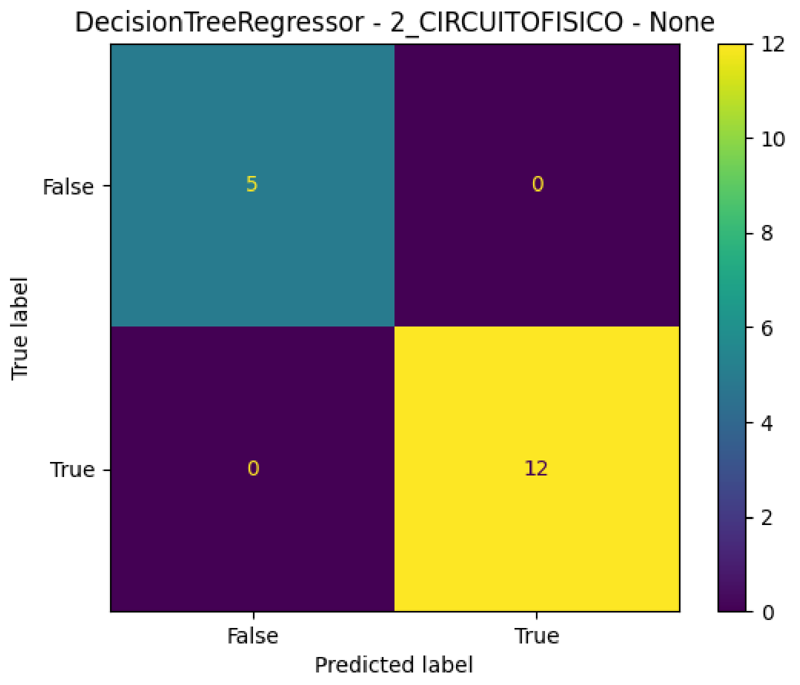

DecisionTreeRegressor - 3_DUPLATAREFA - trial_1
True label
False
True
1
2
0
3
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

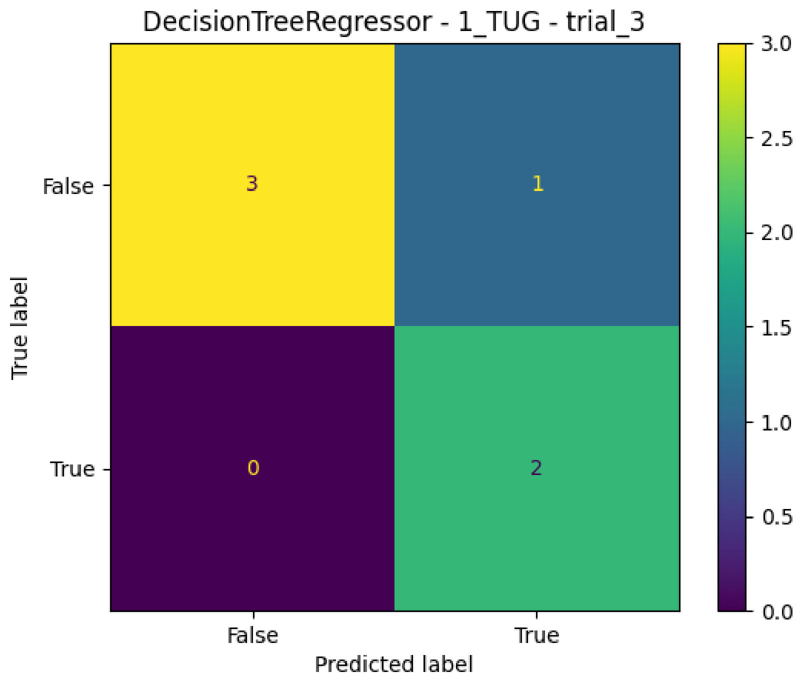

DecisionTreeRegressor - 3_DUPLATAREFA - None
True label
False
True
9
0
0
8
Predicted label
False
True
0
1
2
3
4
5
6
7
8
9

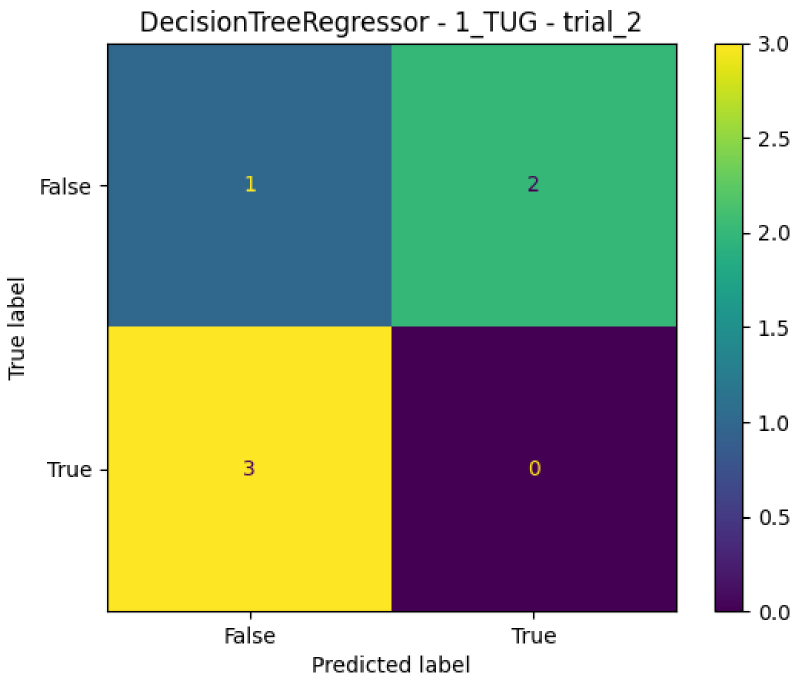

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_3
True label
False
True
1
1
1
3
Predicted label
False
True
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

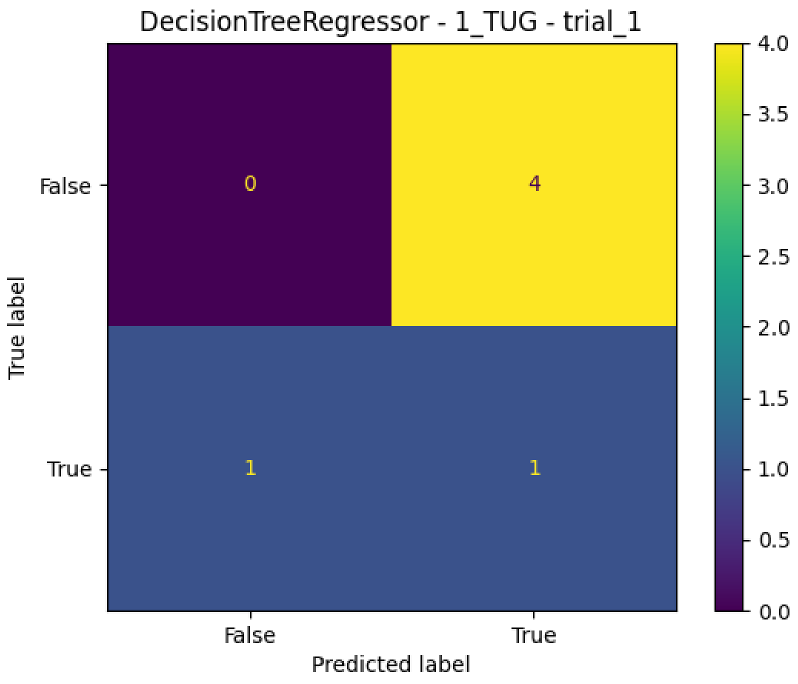

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_2
True label
False
True
1
4
0
1
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

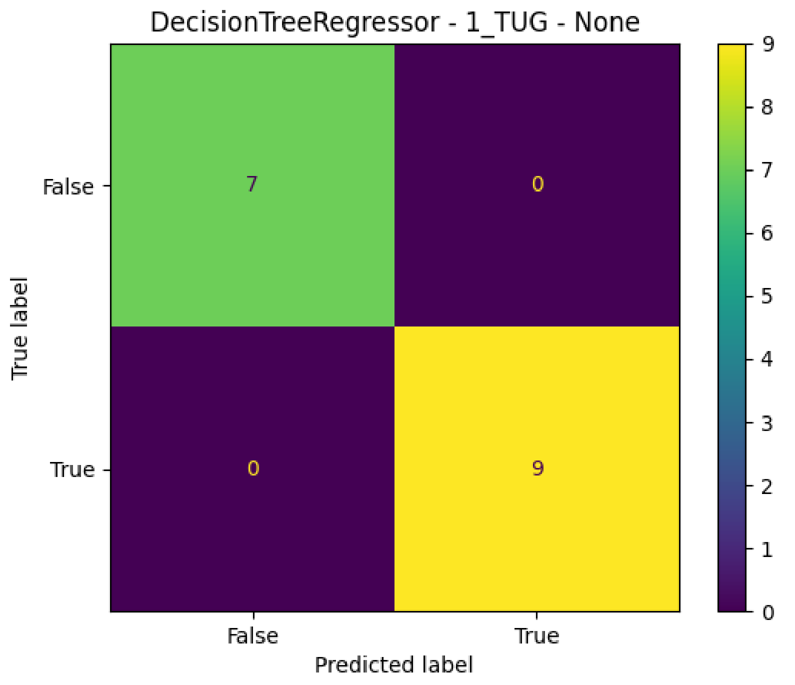

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
5
0
Predicted label
False
True
5
4
3
2
1
0

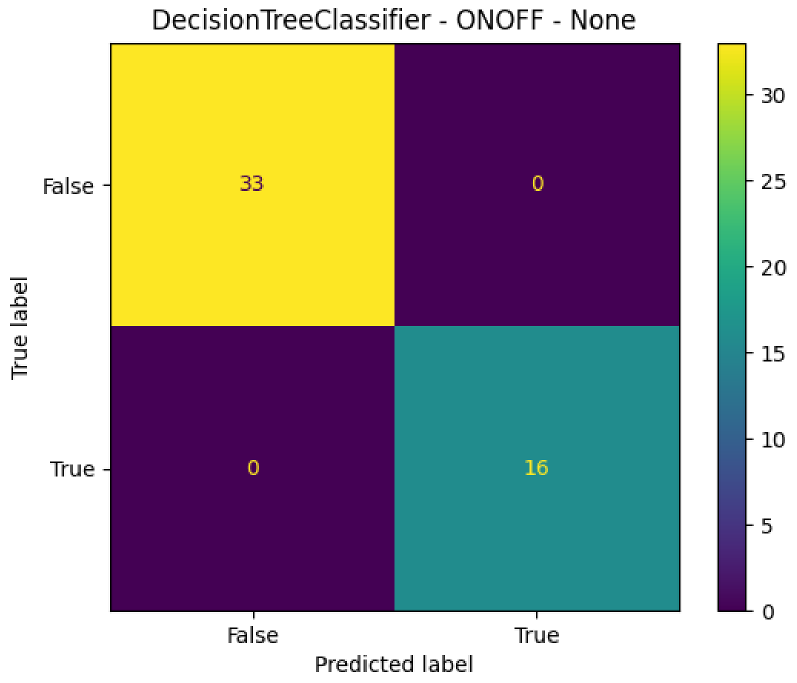

DecisionTreeRegressor - 2_CIRCUITOFISICO - None
True label
False
True
5
0
0
12
Predicted label
False
True
0
2
4
6
8
10
12

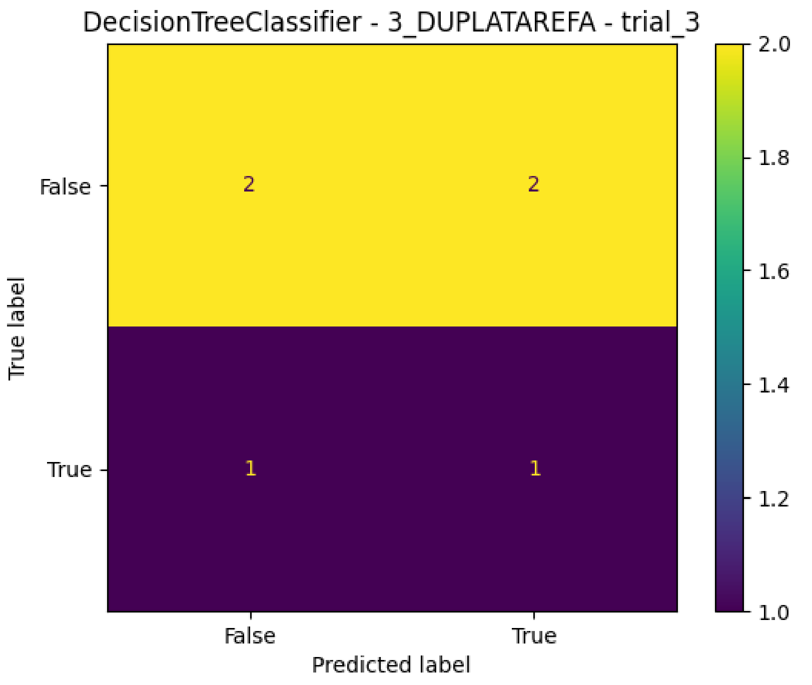

DecisionTreeRegressor - 1_TUG - trial_3
True label
False
True
3
1
0
2
Predicted label
False
True
0.0
0.5
1.0
1.5
2.0
2.5
3.0

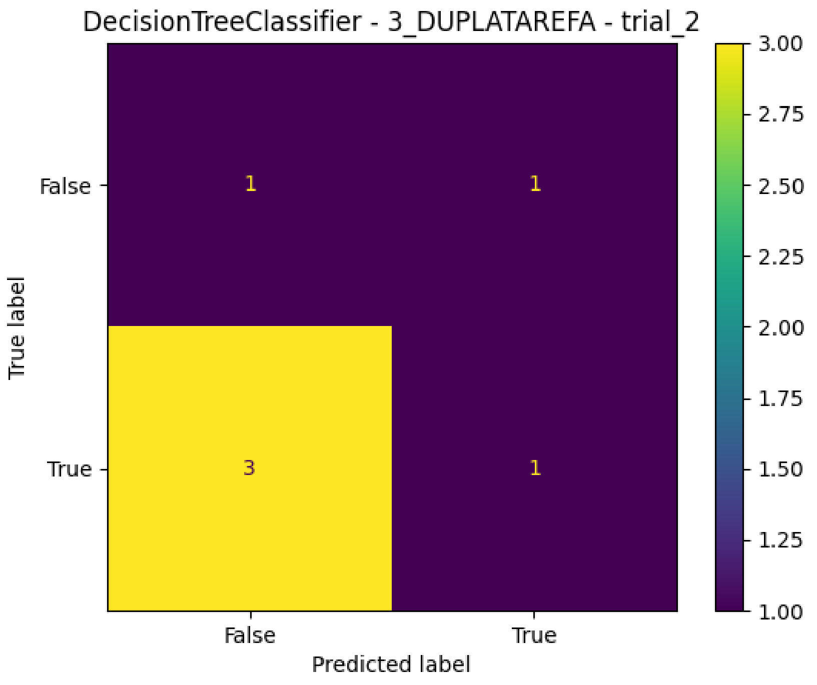

DecisionTreeRegressor - 1_TUG - trial_2
True label
False
True
1
2
3
0
Predicted label
False
True
3.0
2.5
2.0
1.5
1.0
0.5
0.0

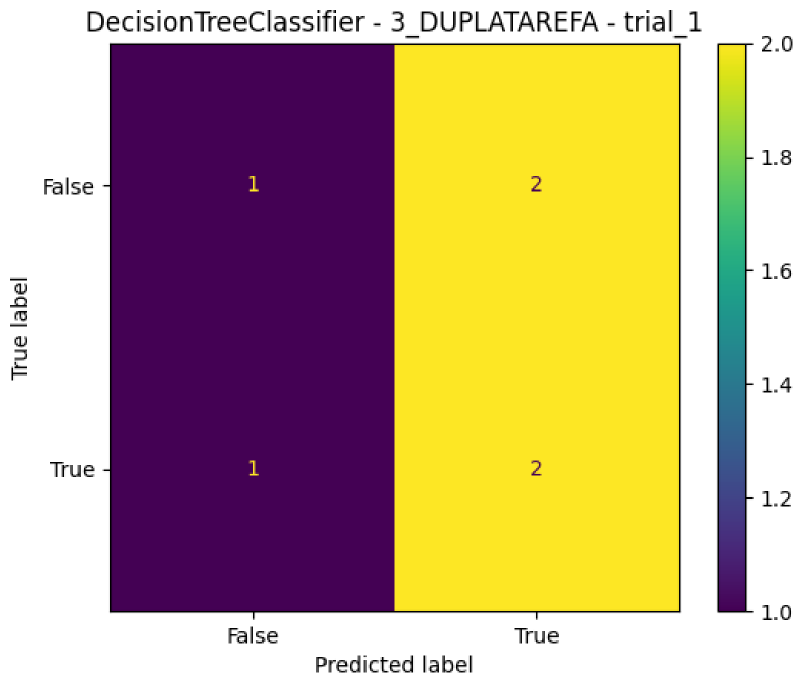

DecisionTreeRegressor - 1_TUG - trial_1
True label
False
True
0
4
1
1
Predicted label
False
True
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

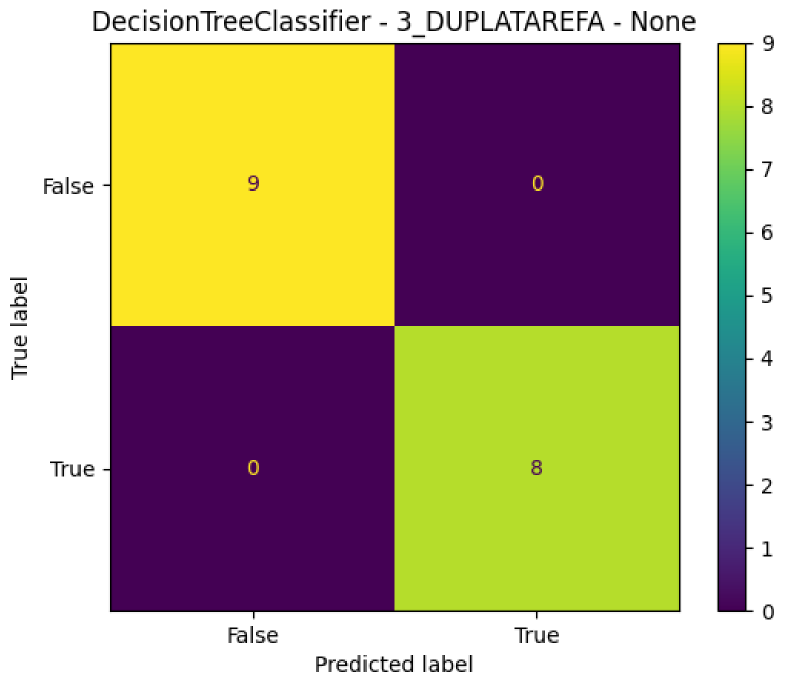

DecisionTreeRegressor - 1_TUG - None
True label
False
True
7
0
0
9
Predicted label
False
True
9
8
7
6
5
4
3
2
1
0

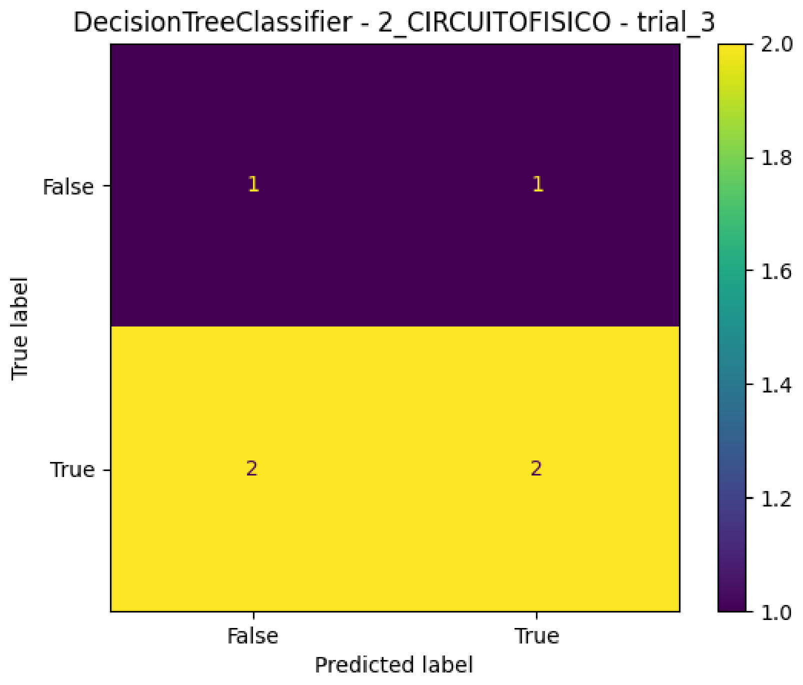

DecisionTreeClassifier - ONOFF - None
True label
False
True
33
0
0
16
Predicted label
False
True
30
25
20
15
10
5
0

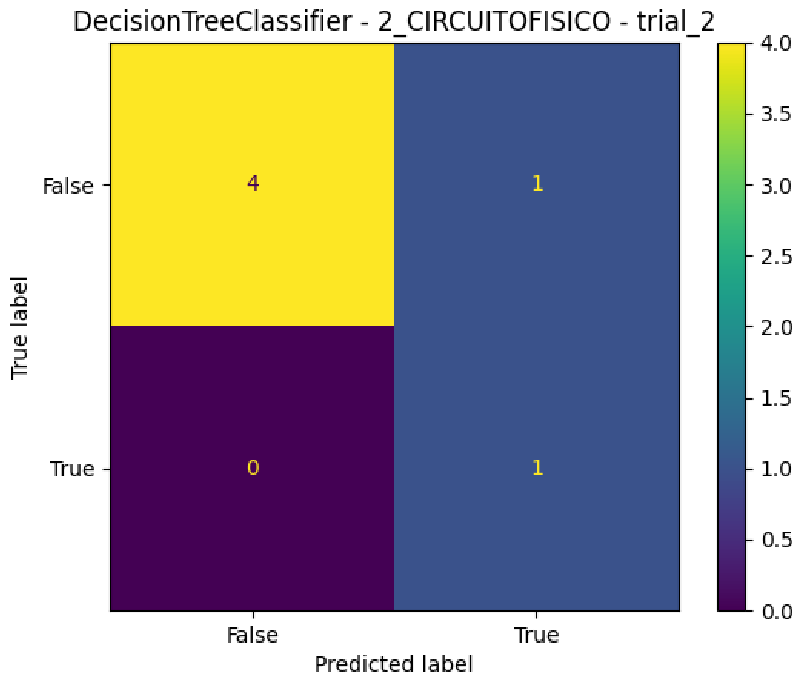

DecisionTreeClassifier - 3_DUPLATAREFA - trial_3
True label
False
True
2
2
1
1
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

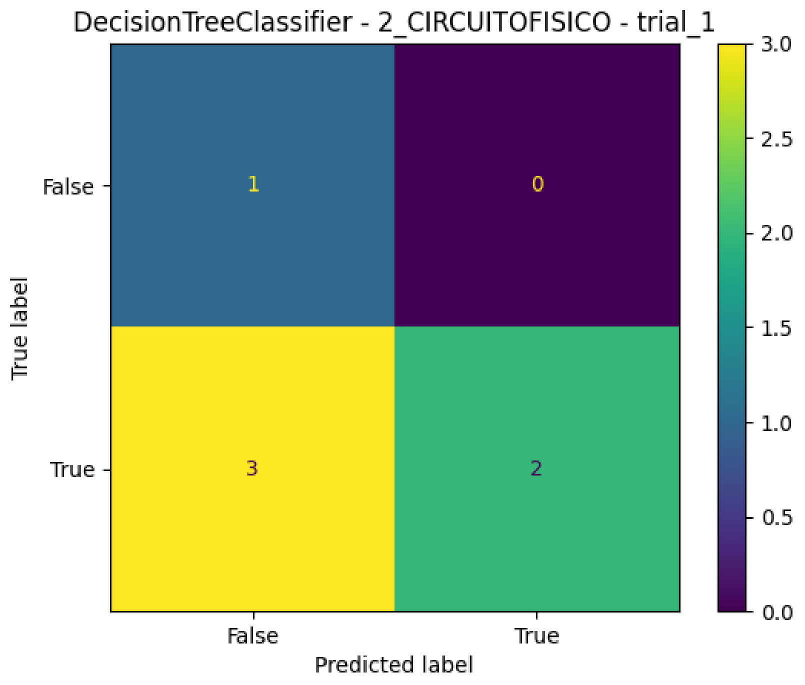

DecisionTreeClassifier - 3_DUPLATAREFA - trial_2
True label
False
True
1
1
3
1
False
True
Predicted label
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

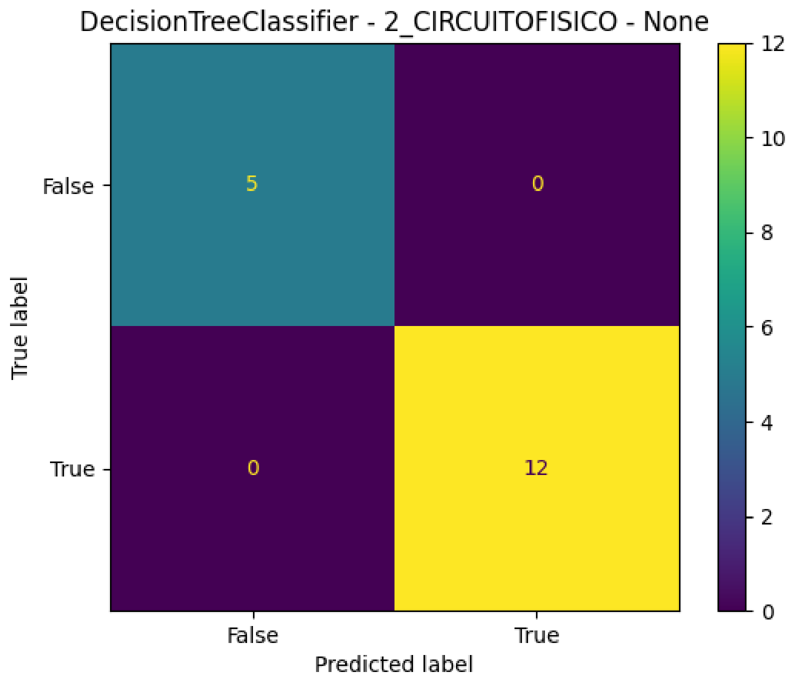

DecisionTreeClassifier - 3_DUPLATAREFA - trial_1
True label
Predicted label
False
True
1
2
1
2
2.0
1.8
1.6
1.4
1.2
1.0

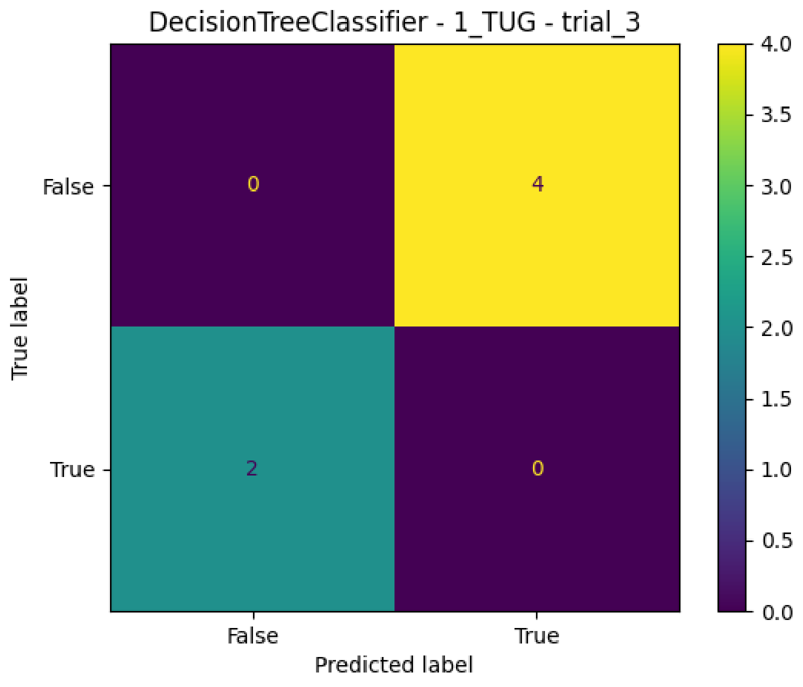

DecisionTreeClassifier - 3_DUPLATAREFA - None
True label
Predicted label
False
True
9
0
0
8
9
8
7
6
5
4
3
2
1
0

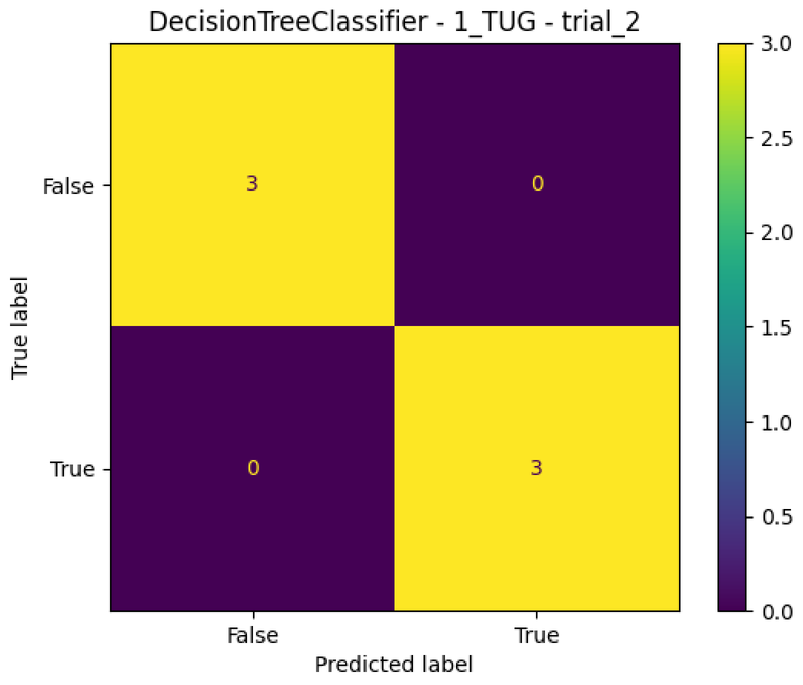

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_3
True label
False
True
1
1
2
2
False
True
Predicted label
2.0
1.8
1.6
1.4
1.2
1.0

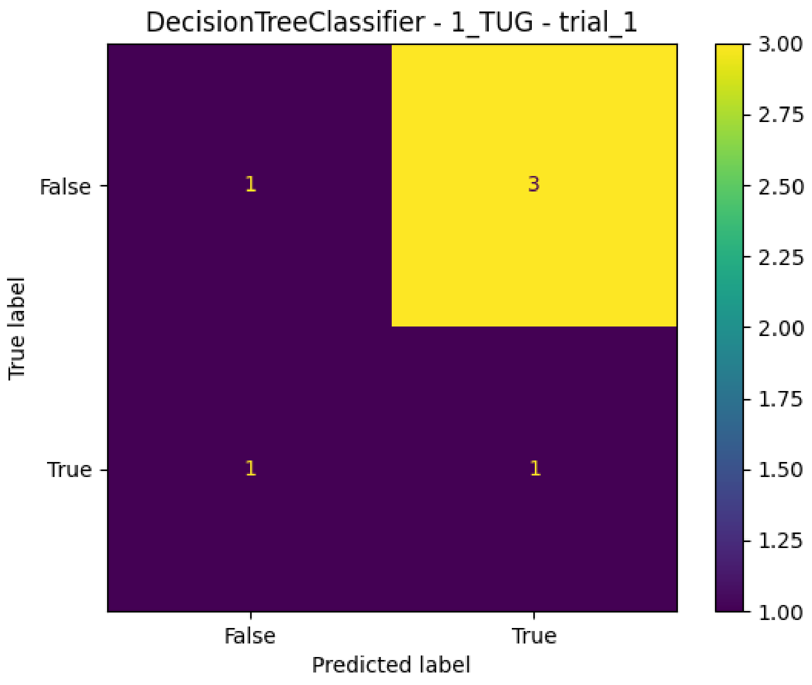

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
0
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

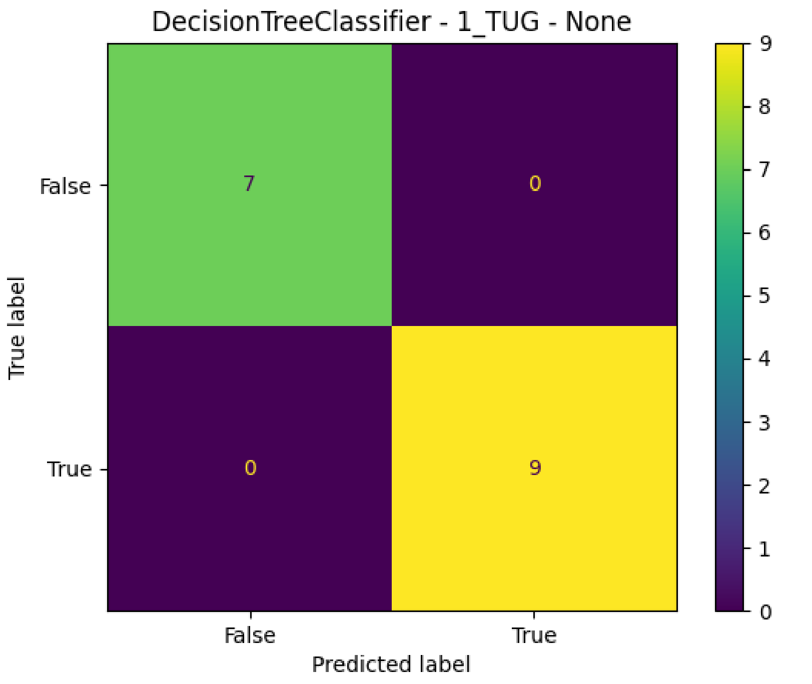

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
3
2
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

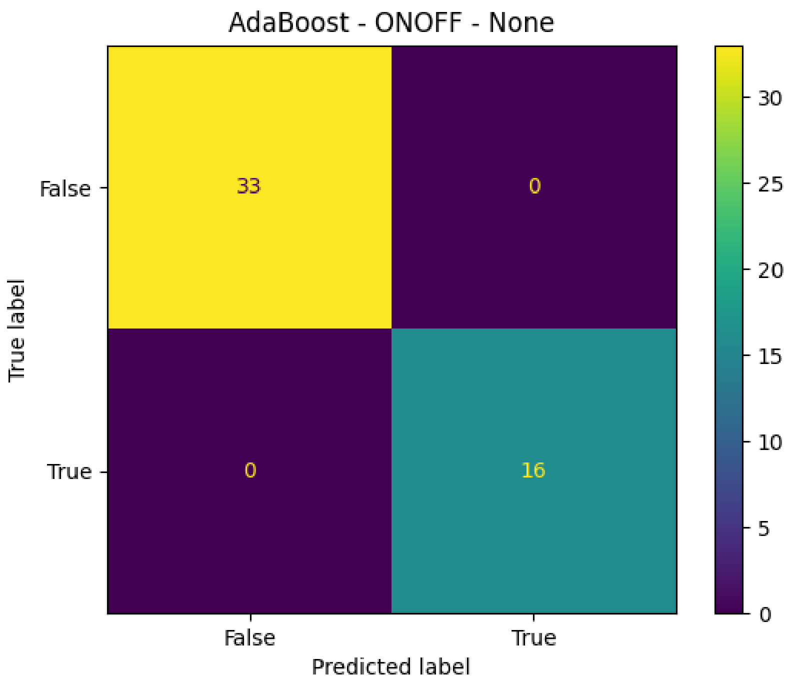

DecisionTreeClassifier - 2_CIRCUITOFISICO - None
True label
False
True
5
0
0
12
False
True
Predicted label
12
10
8
6
4
2
0

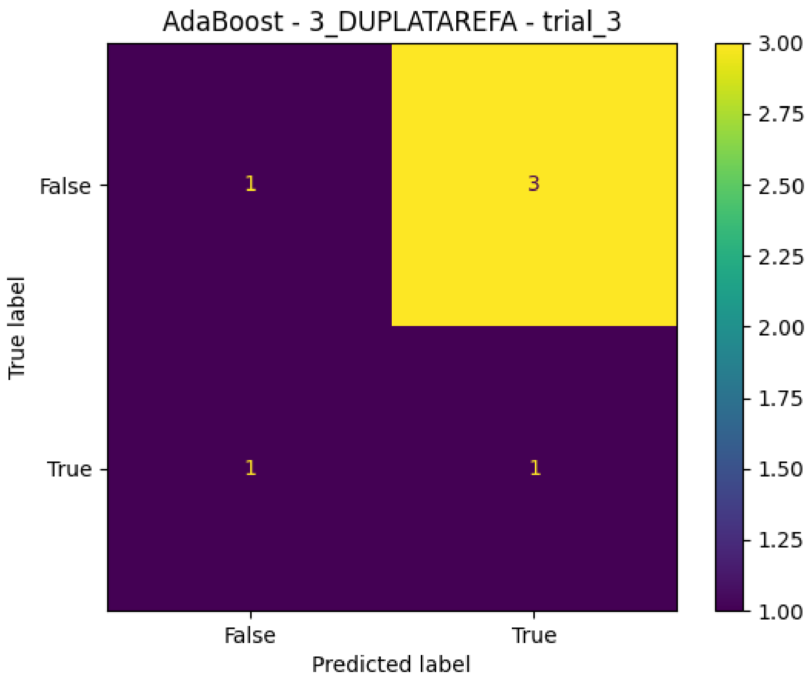

DecisionTreeClassifier - 1_TUG - trial_3
True label
False
True
0
4
2
0
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

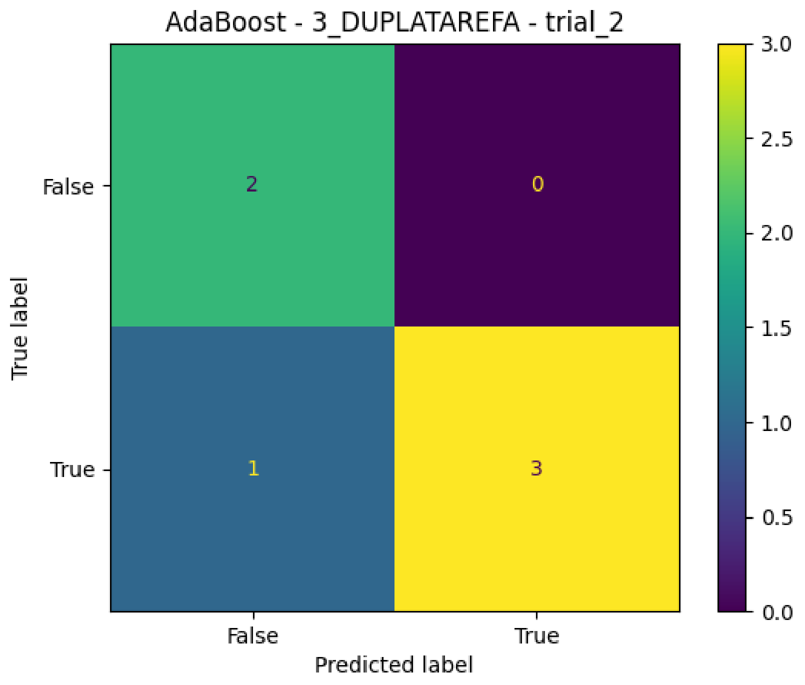

DecisionTreeClassifier - 1_TUG - trial_2
True label
False
True
3
0
0
3
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

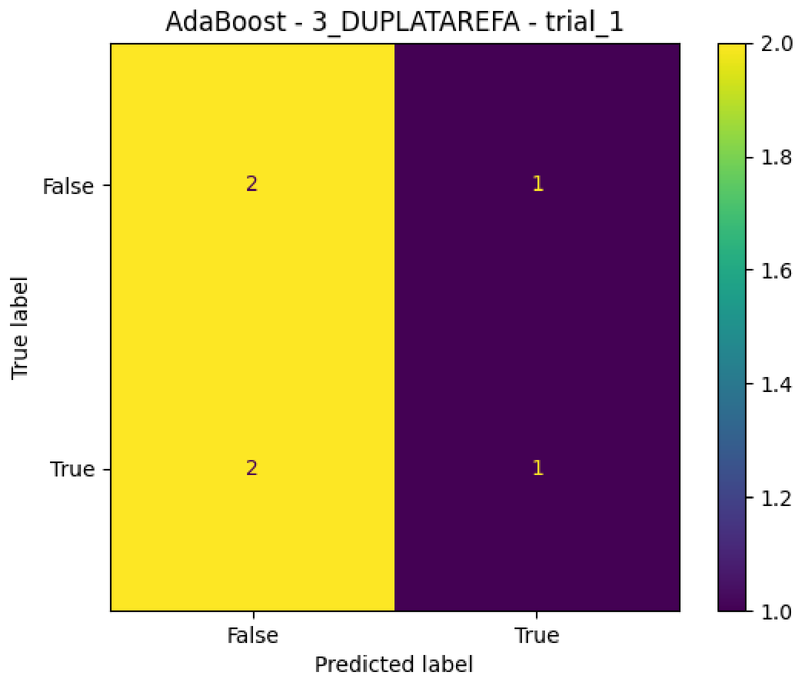

DecisionTreeClassifier - 1_TUG - trial_1
True label
False
True
Predicted label
1
3
1
1
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

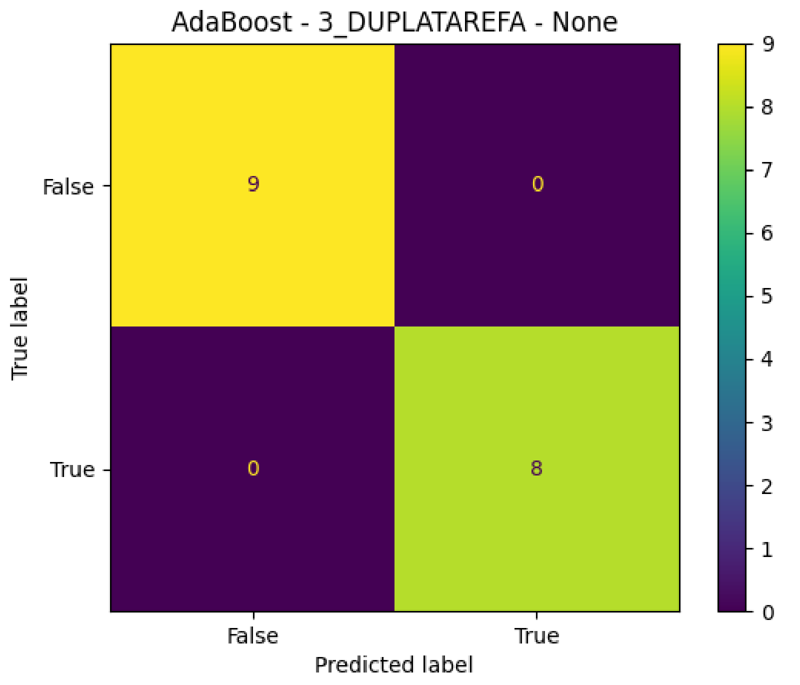

DecisionTreeClassifier - 1_TUG - None
True label
False
True
Predicted label
7
0
0
9
9
8
7
6
5
4
3
2
1
0

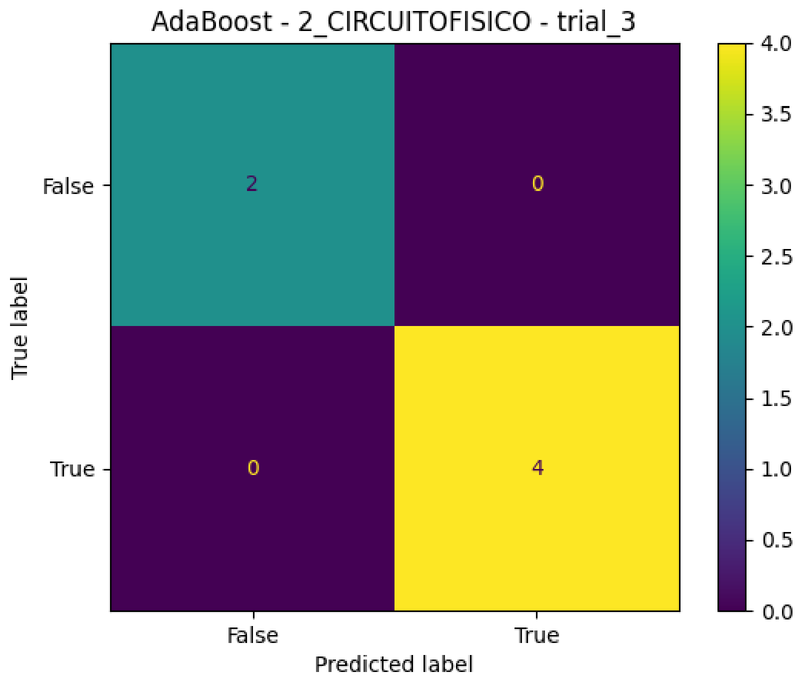

AdaBoost - ONOFF - None
True label
False
True
Predicted label
33
0
0
16
30
25
20
15
10
5
0

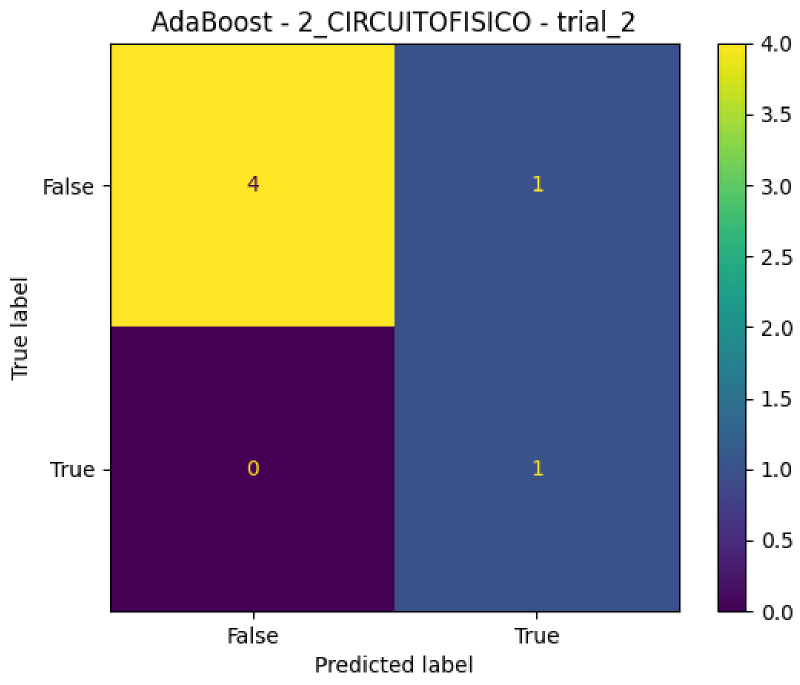

AdaBoost - 3_DUPLATAREFA - trial_3
True label
False
True
Predicted label
1
3
1
1
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

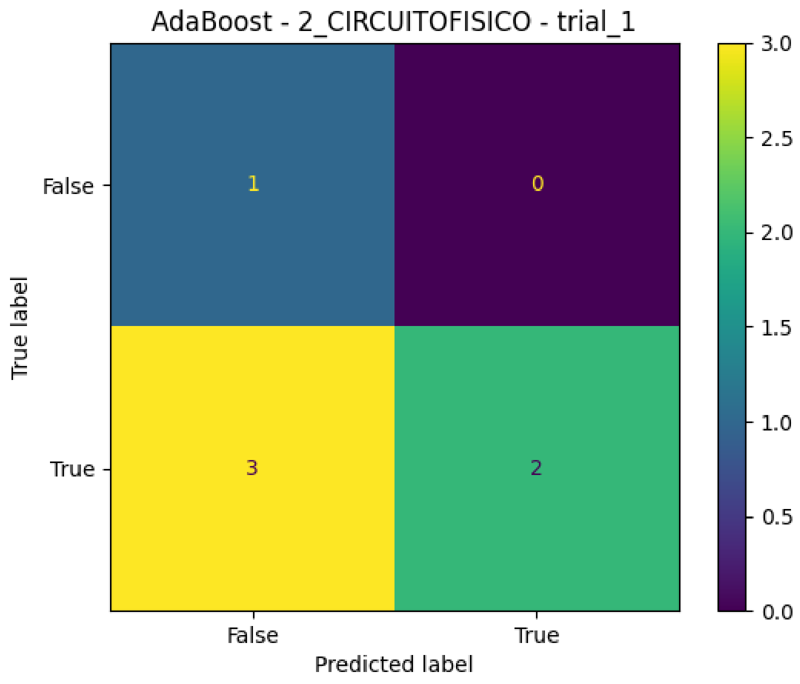

AdaBoost - 3_DUPLATAREFA - trial_2
True label
False
True
Predicted label
2
0
1
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0

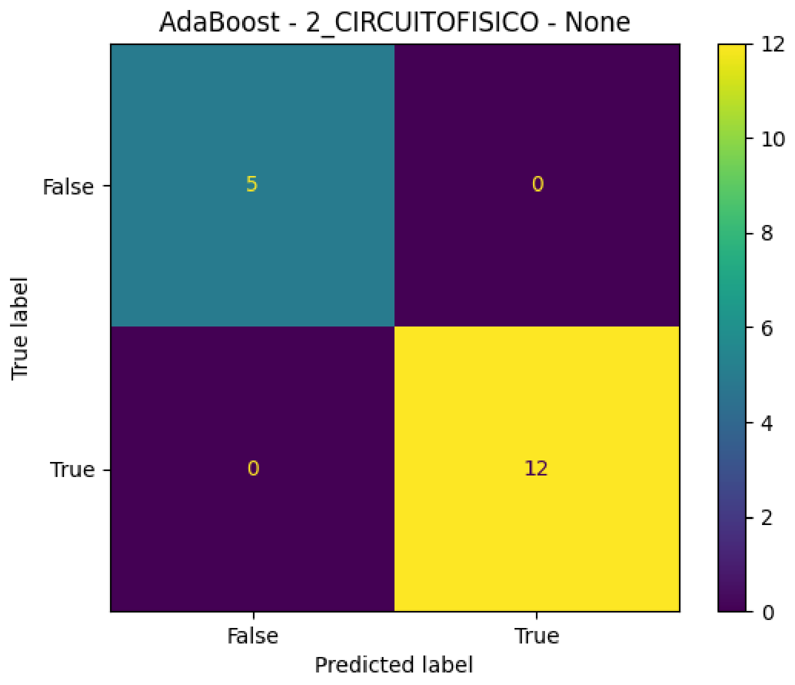

AdaBoost - 3_DUPLATAREFA - trial_1
True label
False
True
Predicted label
2
1
2
1
2.0
1.8
1.6
1.4
1.2
1.0

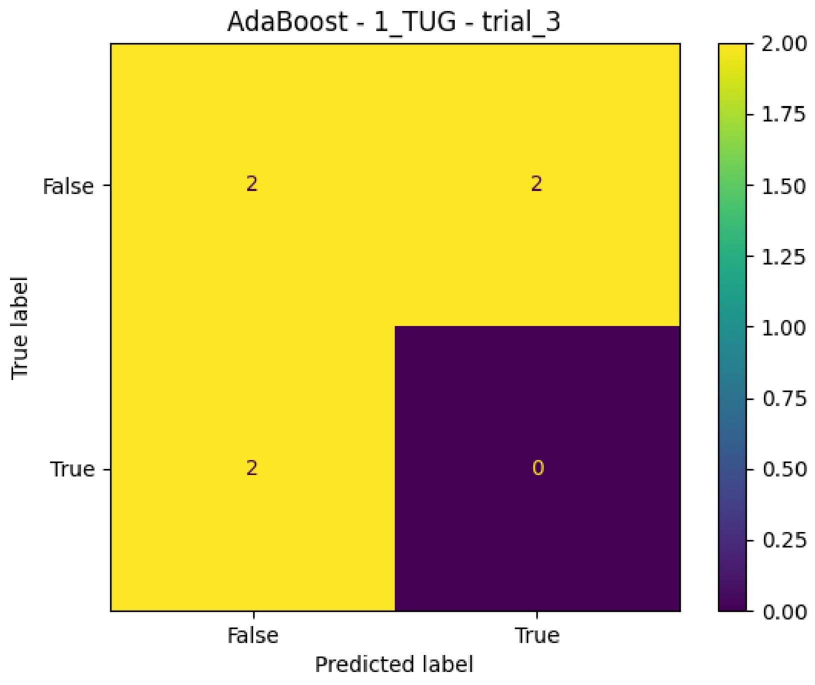

AdaBoost - 3_DUPLATAREFA - None
True label
False
True
Predicted label
9
0
0
8
0
1
2
3
4
5
6
7
8
9

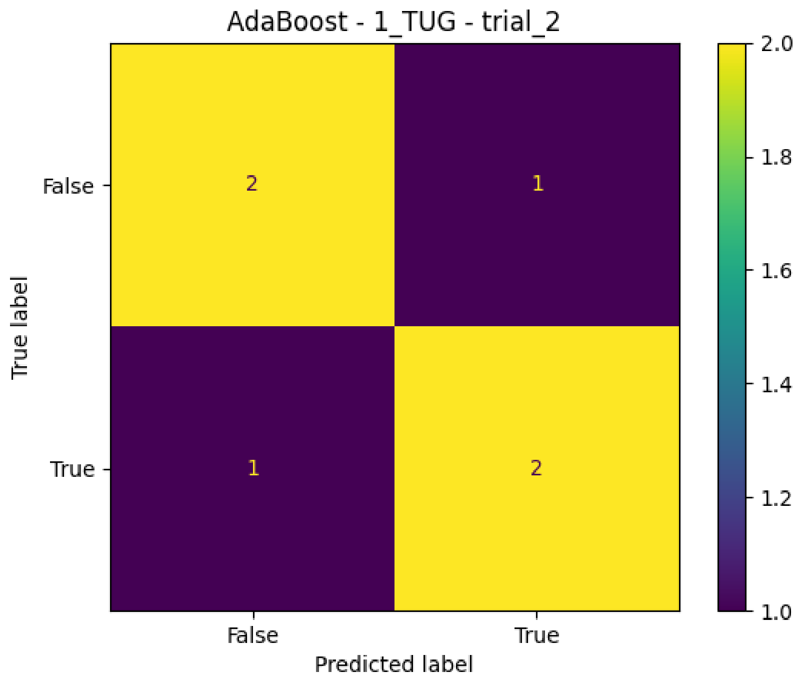

AdaBoost - 2_CIRCUITOFISICO - trial_3
True label
False
True
Predicted label
2
0
0
4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


AdaBoost - 2_CIRCUITOFISICO - trial_2
True label
False
True
4
1
0
1
False
True
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

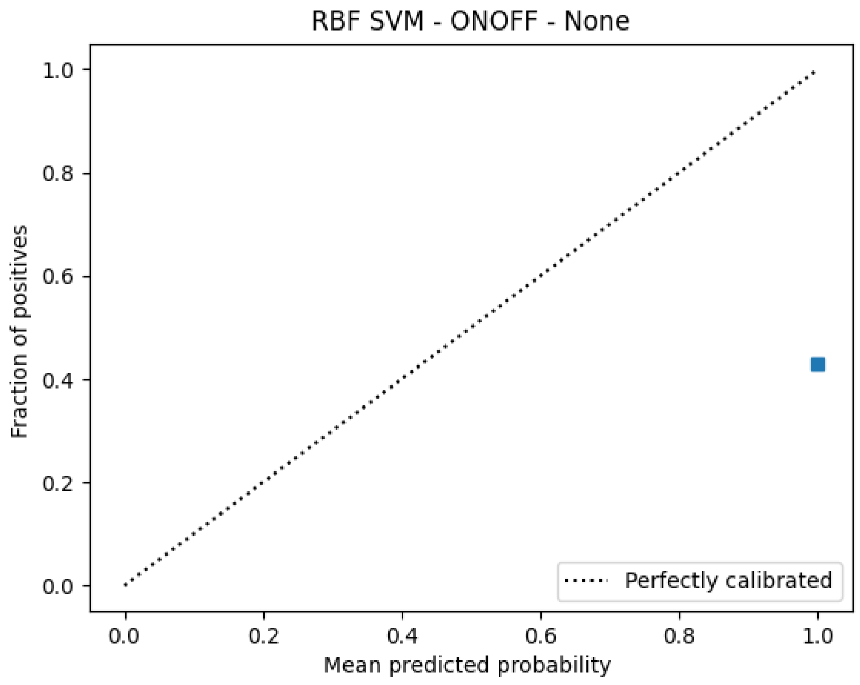

AdaBoost - 2_CIRCUITOFISICO - trial_1
True label
False
True
1
0
3
2
False
True
Predicted label
3.0
2.5
2.0
1.5
1.0
0.5
0.0

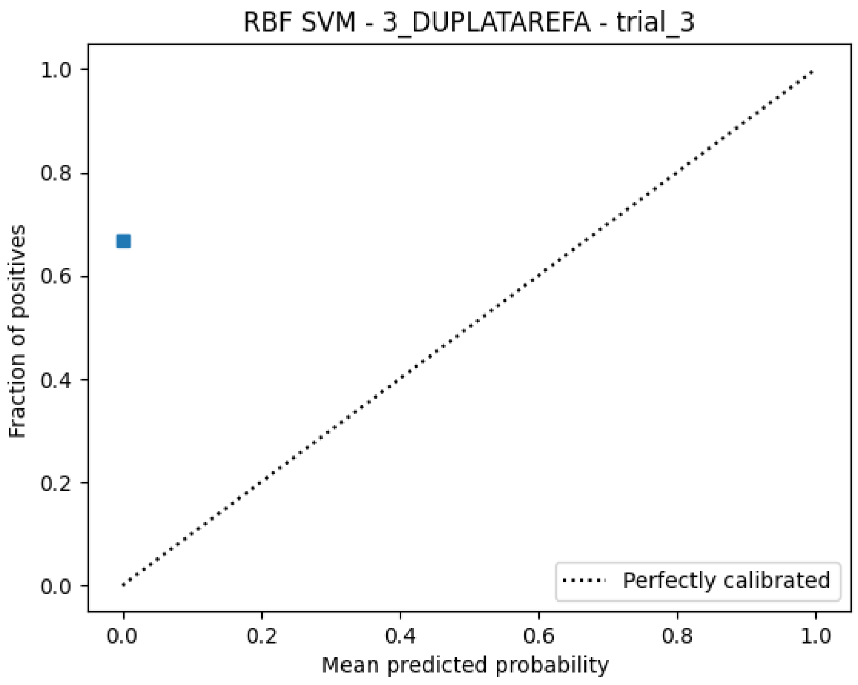

AdaBoost - 2_CIRCUITOFISICO - None
True label
False
True
5
0
0
12
Predicted label
False
True
0
2
4
6
8
10
12

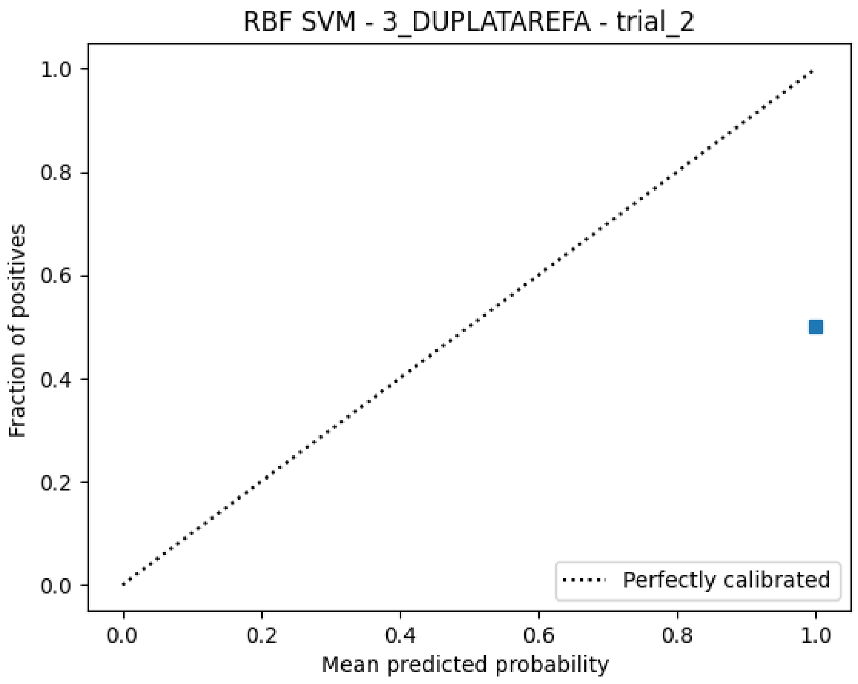

AdaBoost - 1_TUG - trial_3
True label
False
True
2
2
2
0
Predicted label
False
True
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

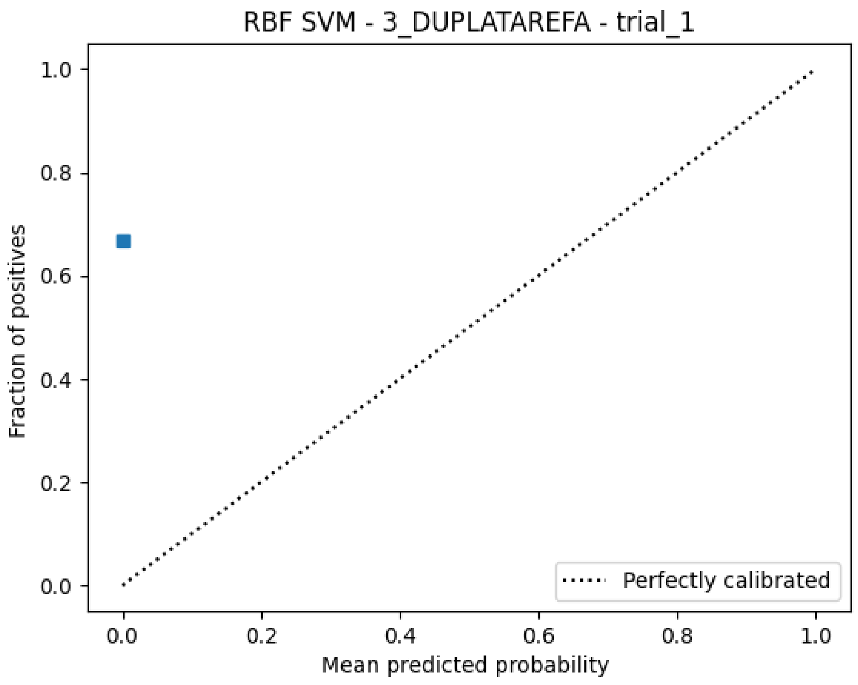


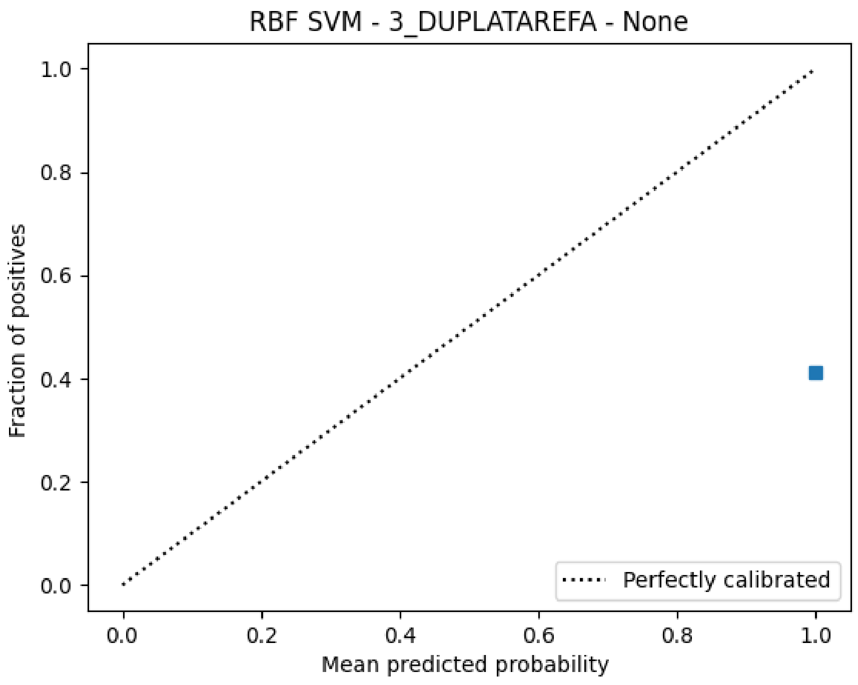
A.2 Calibration Curve Analysis

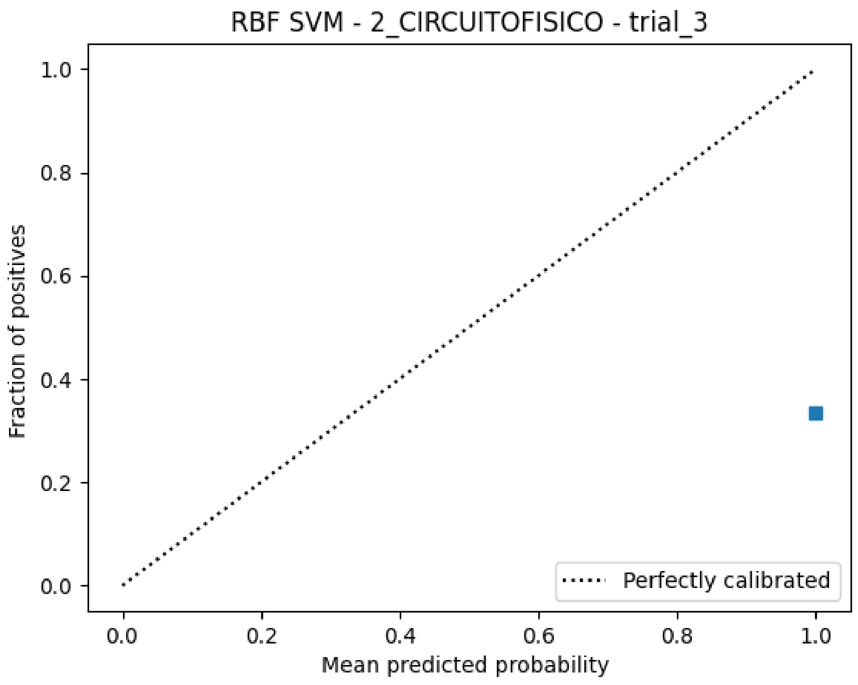


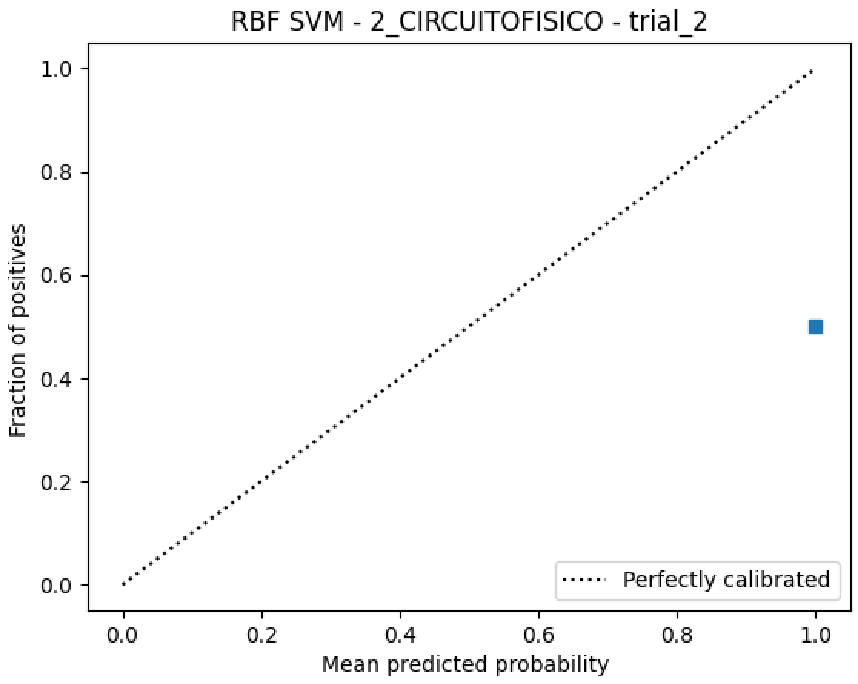

RBF SVM - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

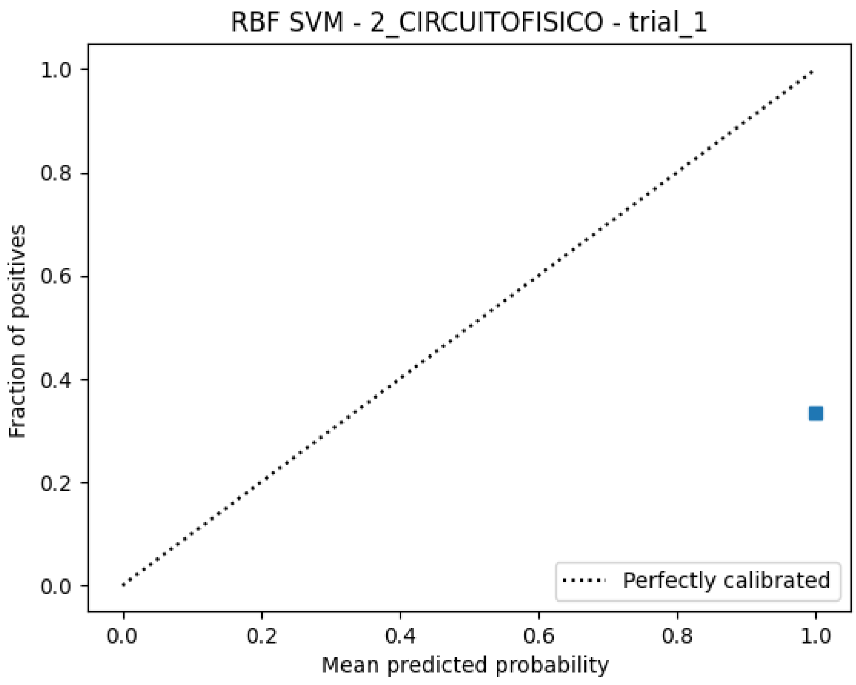

RBF SVM - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

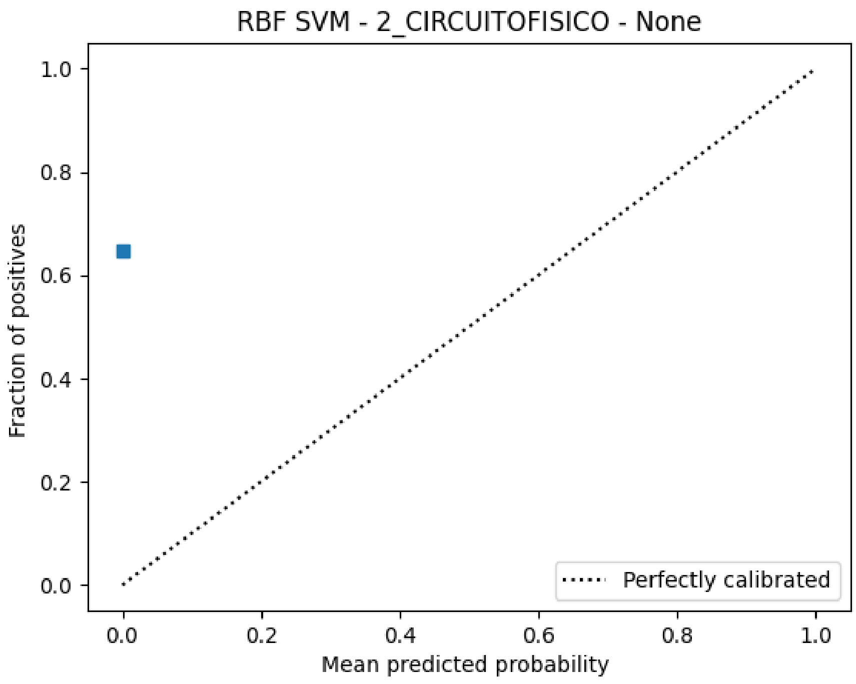

RBF SVM - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

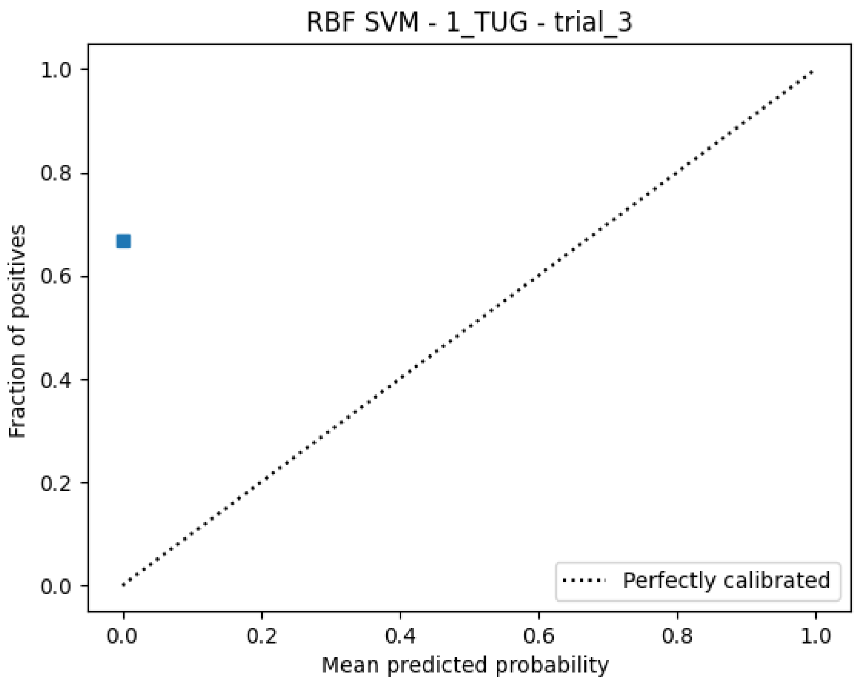

RBF SVM - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

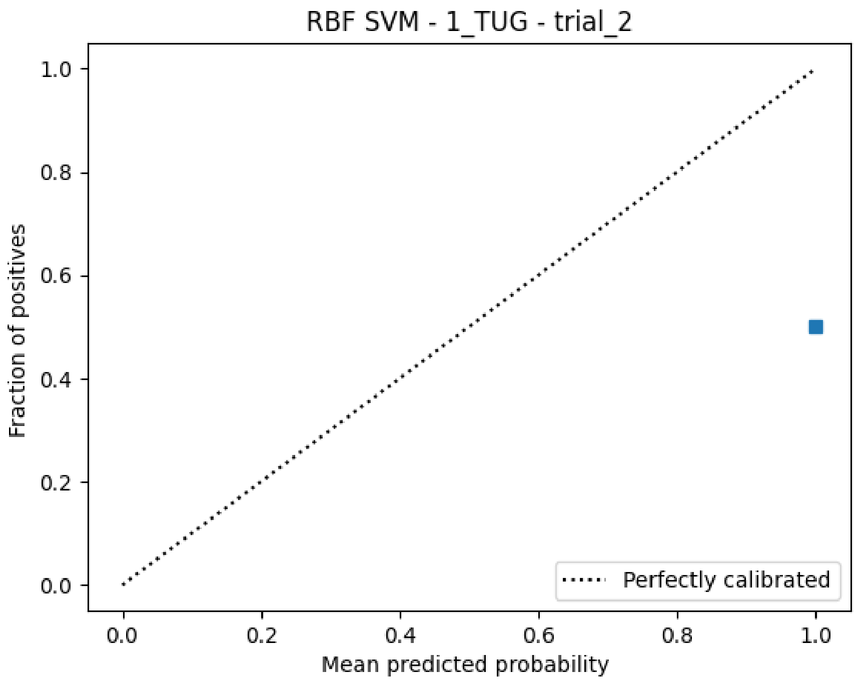

RBF SVM - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
1.0
0.8
0.6
0.4
0.2
0.0

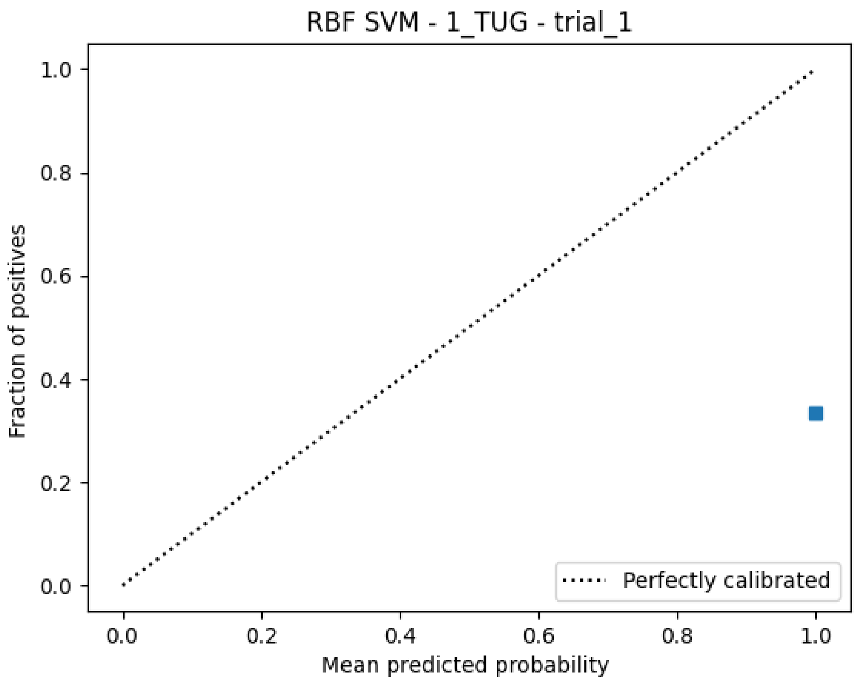

RBF SVM - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
1.0
0.8
0.6
0.4
0.2
0.0

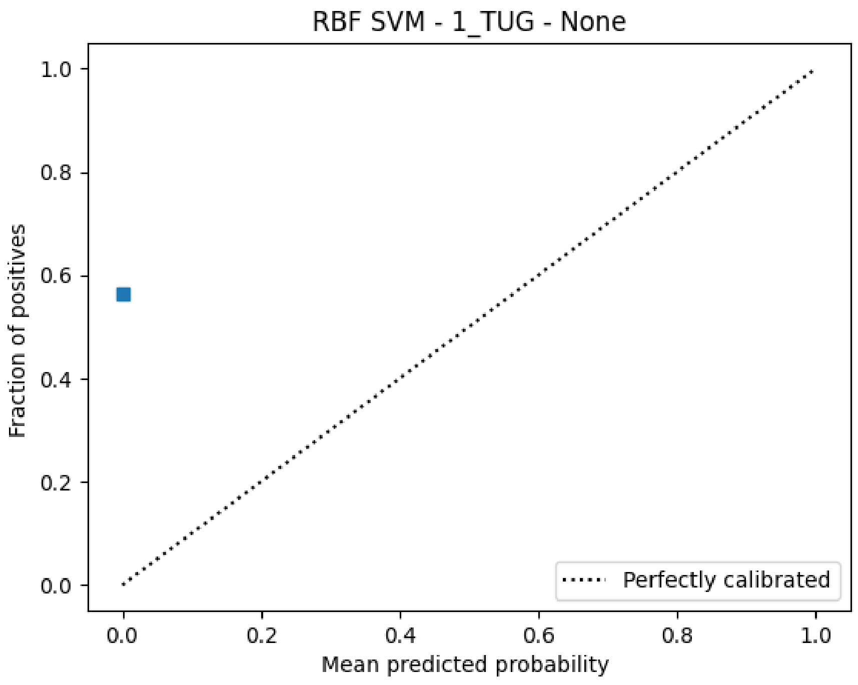

RBF SVM - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

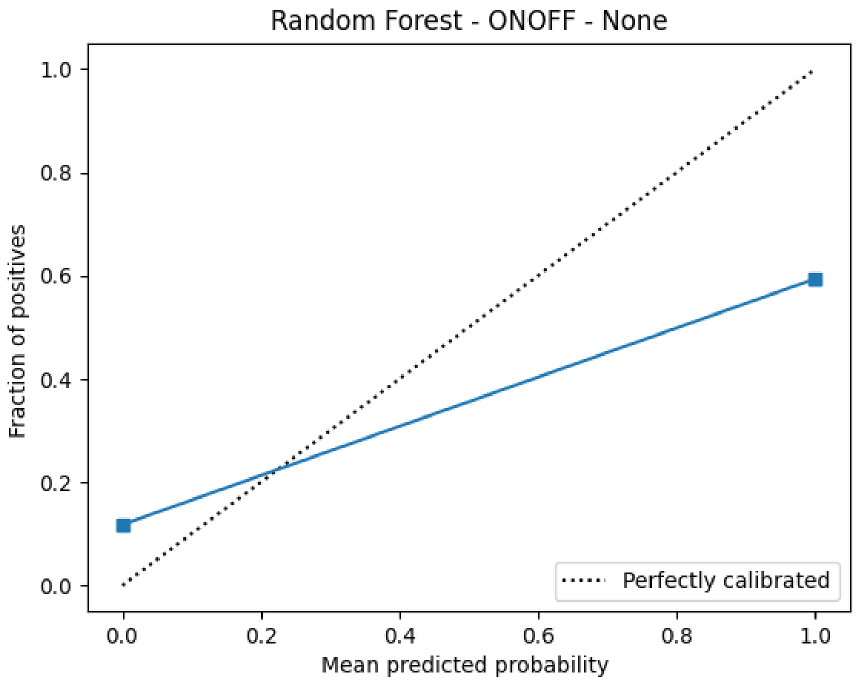

RBF SVM - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

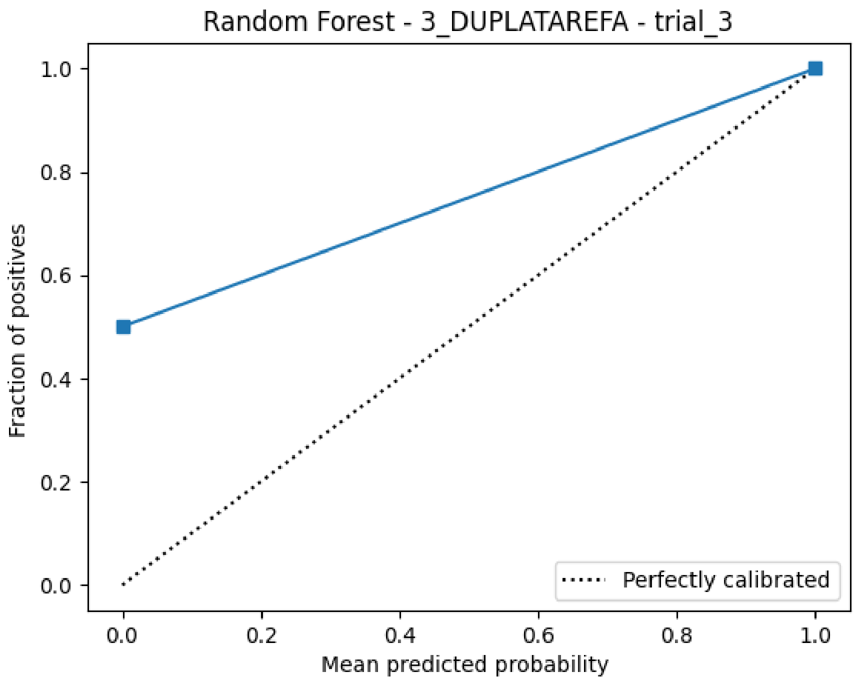

RBF SVM - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

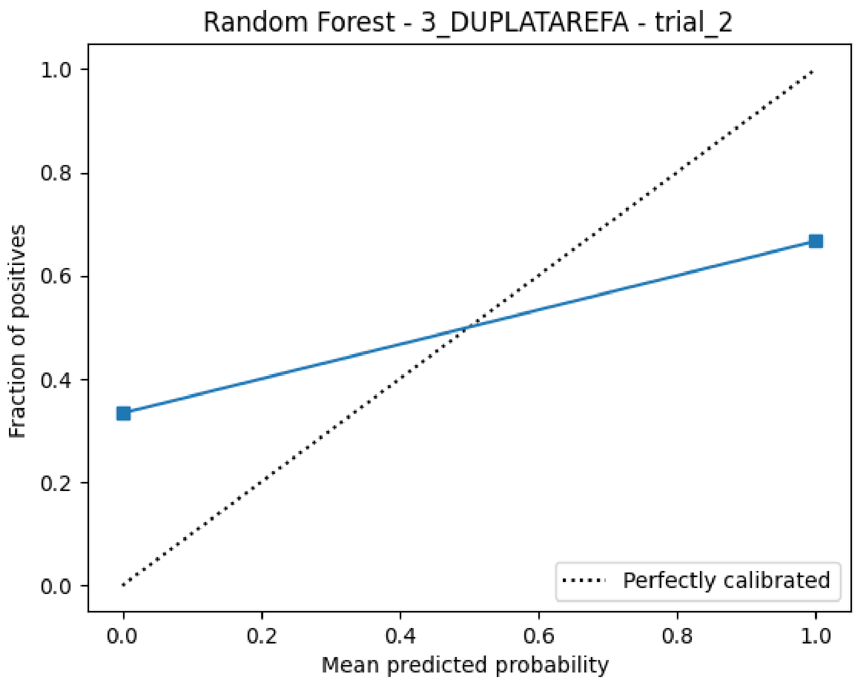

RBF SVM - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

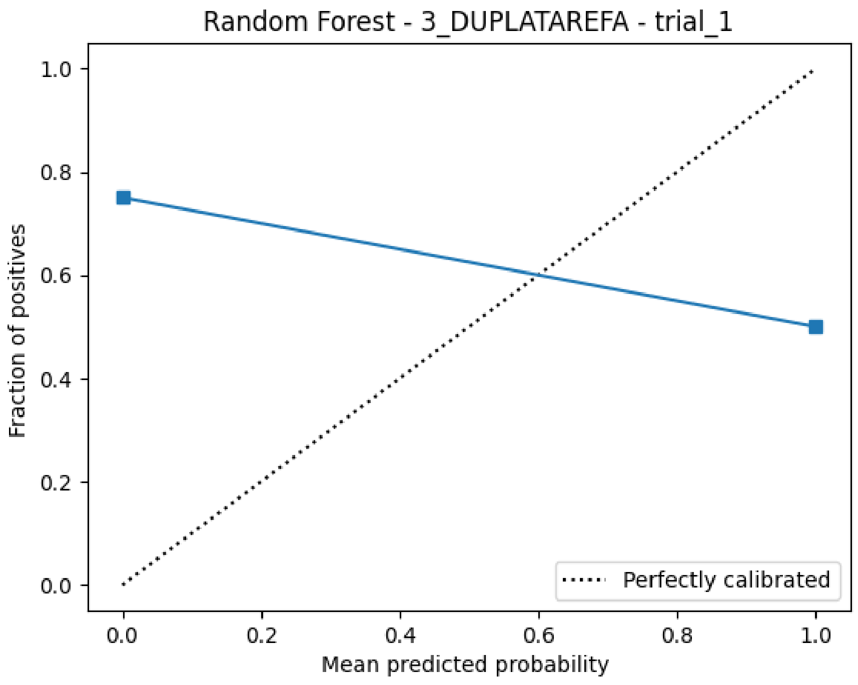

RBF SVM - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

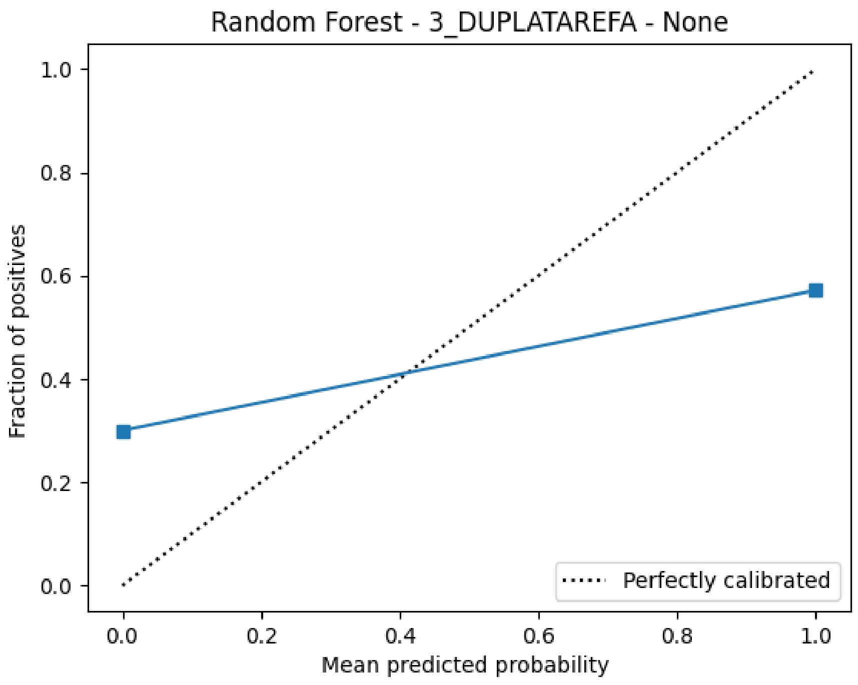

RBF SVM - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

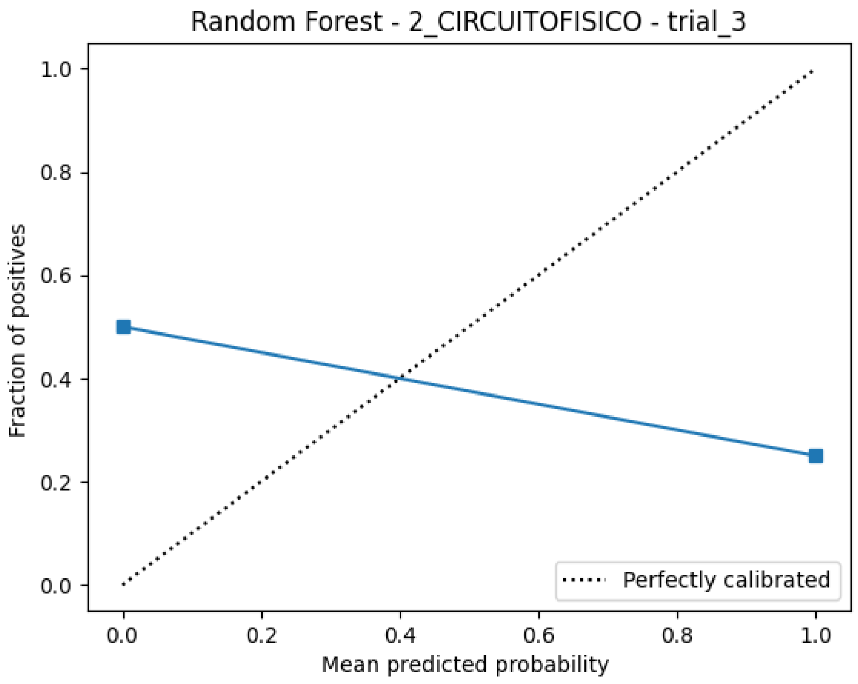

Random Forest - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

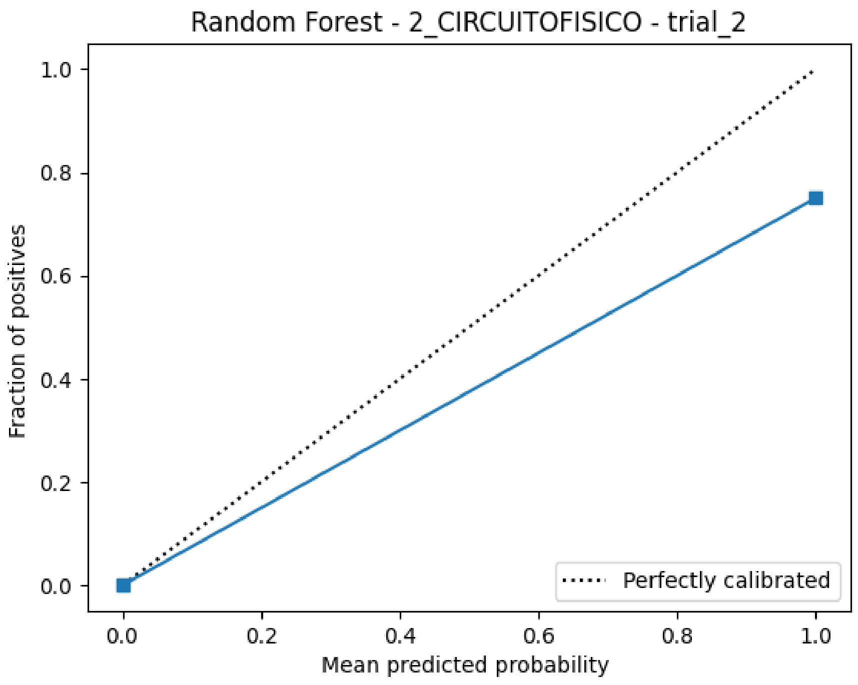

Random Forest - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

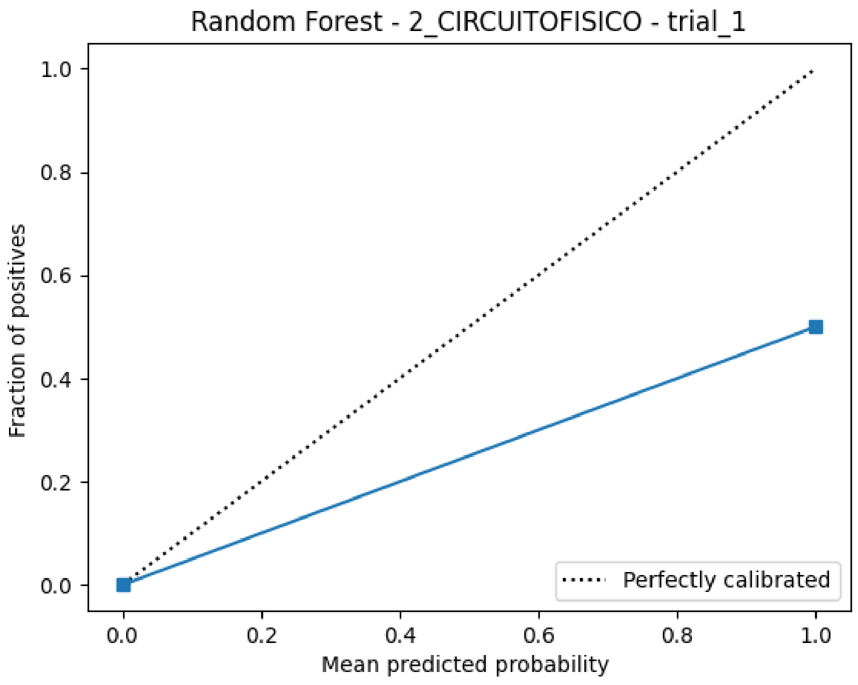

Random Forest - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

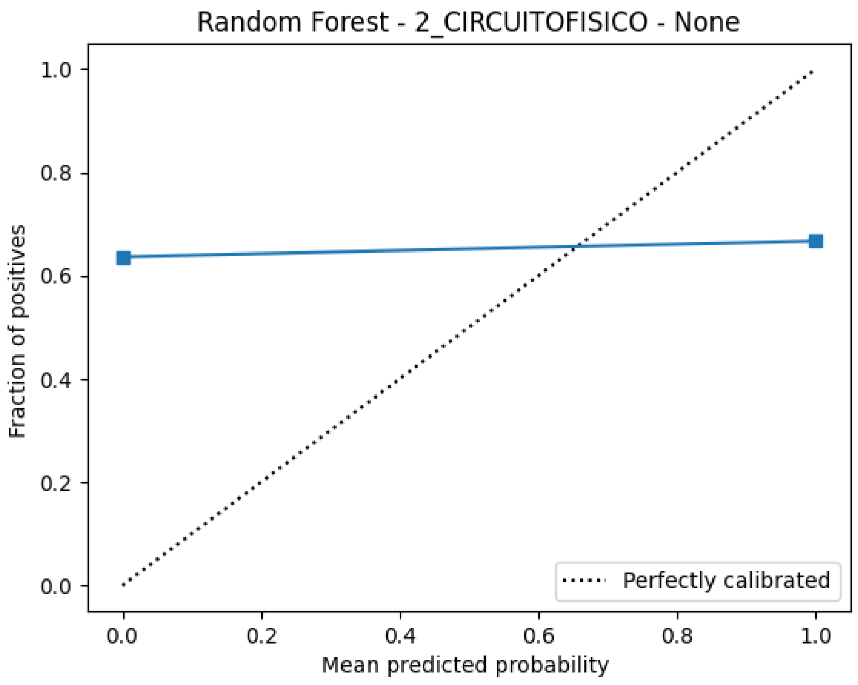

Random Forest - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

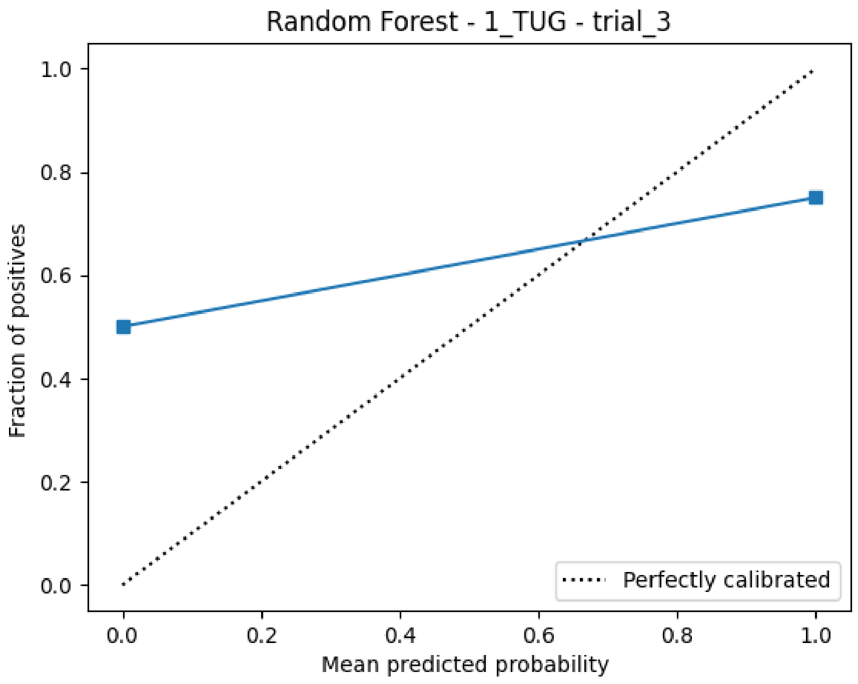

Random Forest - 3_DUPLATAREFA - None
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

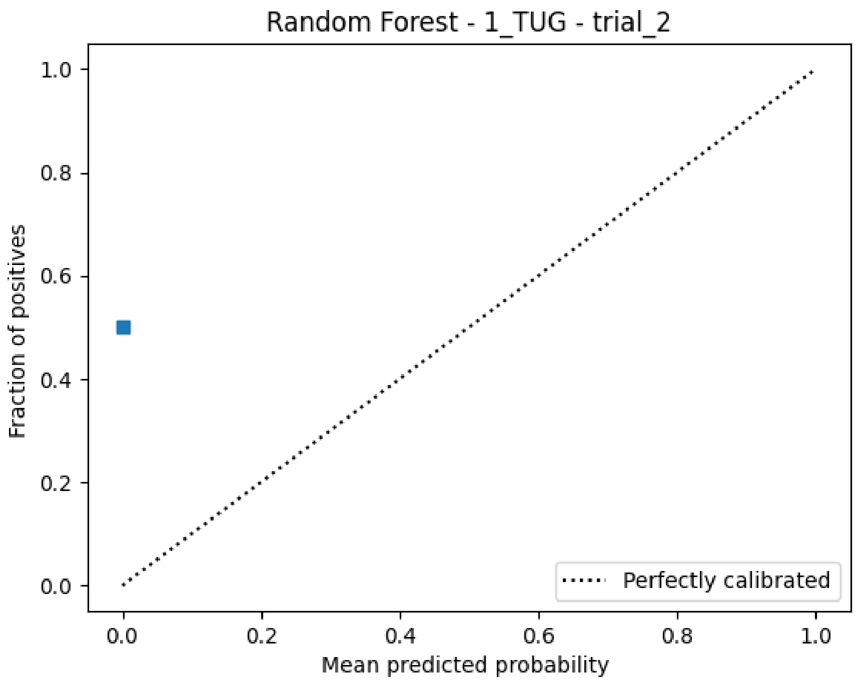

Random Forest - 2_CIRCUITOFISICO - trial_3
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

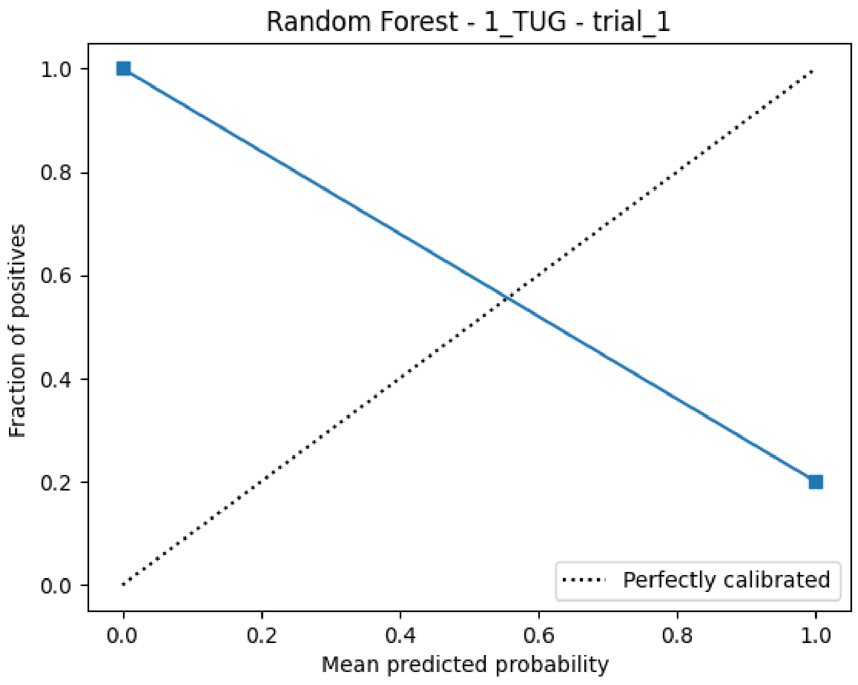

Random Forest - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

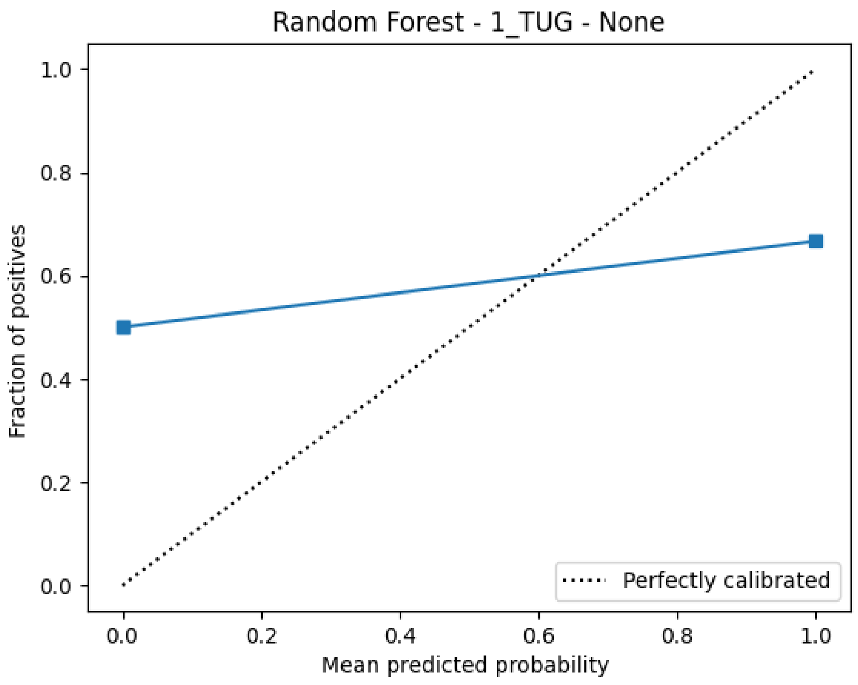

Random Forest - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

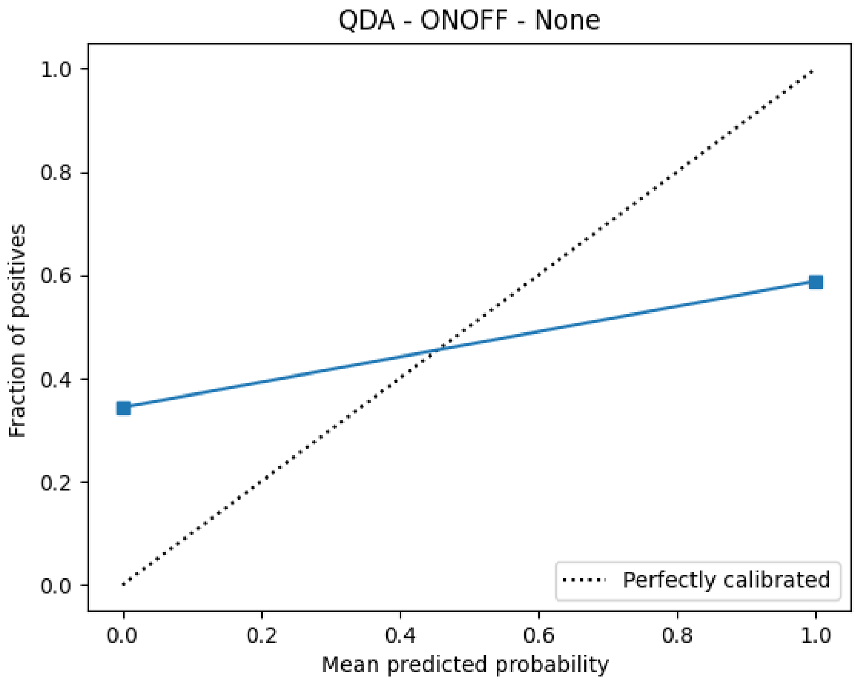

Random Forest - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

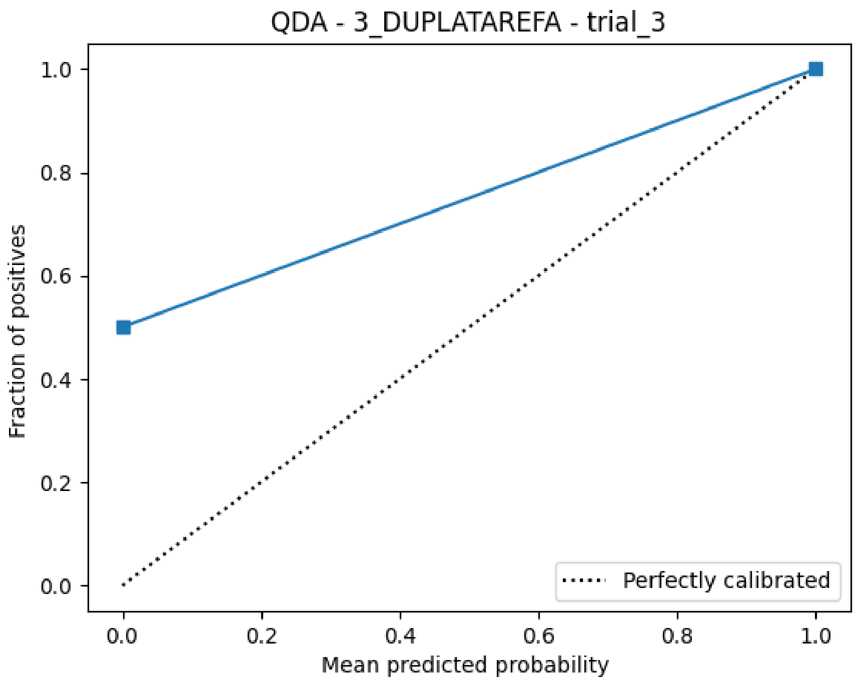

Random Forest - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

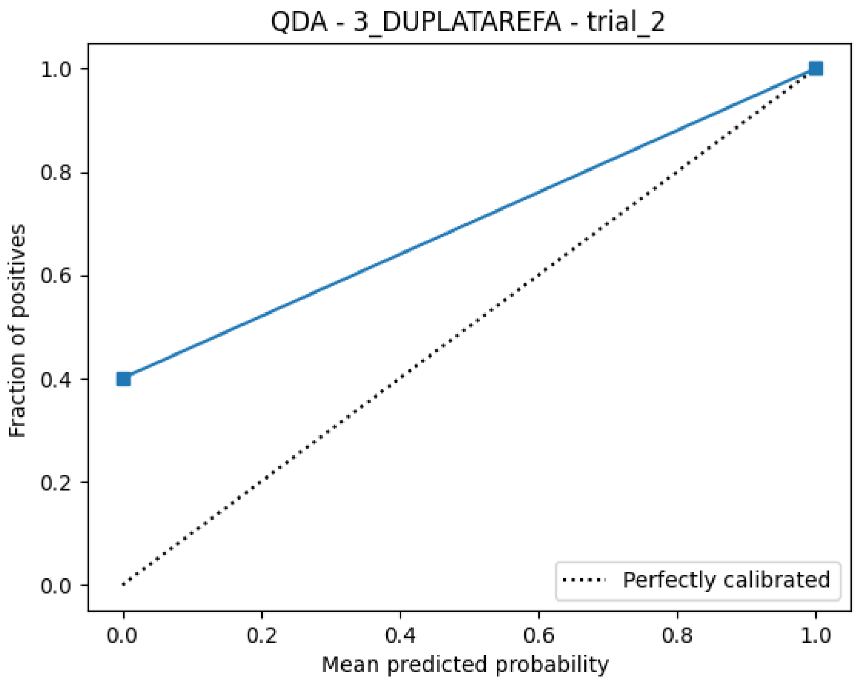

Random Forest - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

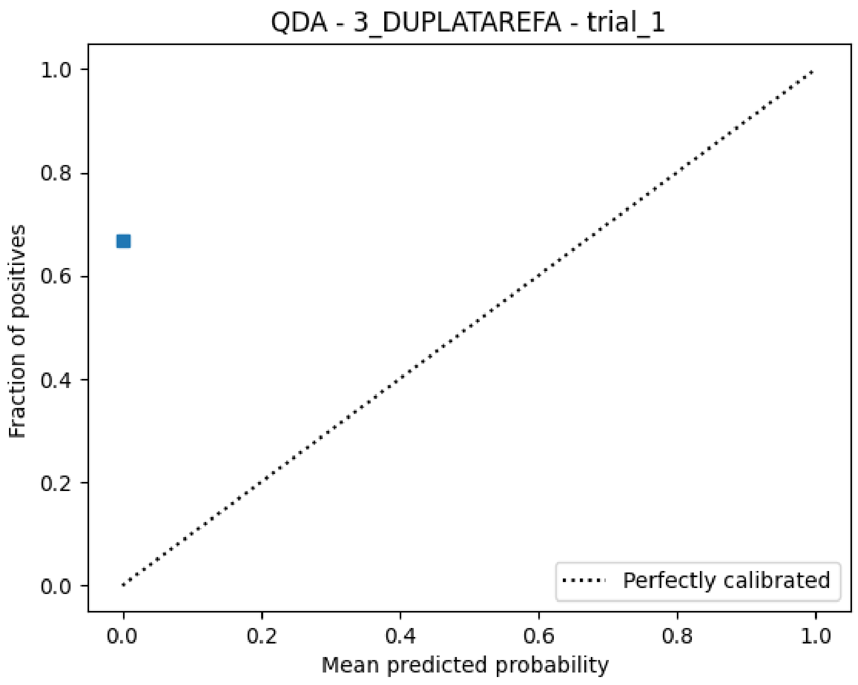

Random Forest - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

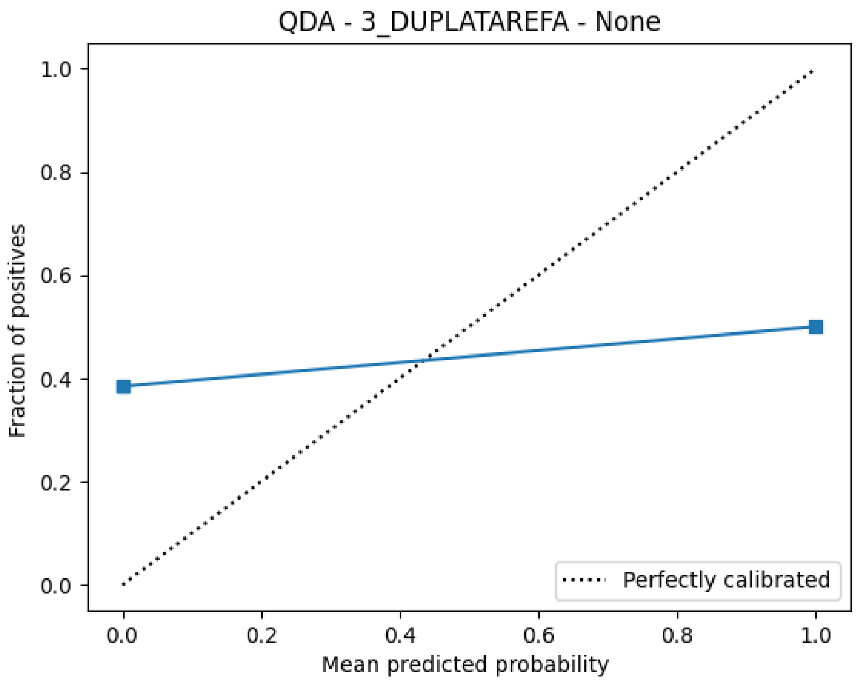

Random Forest - 1_TUG - None
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

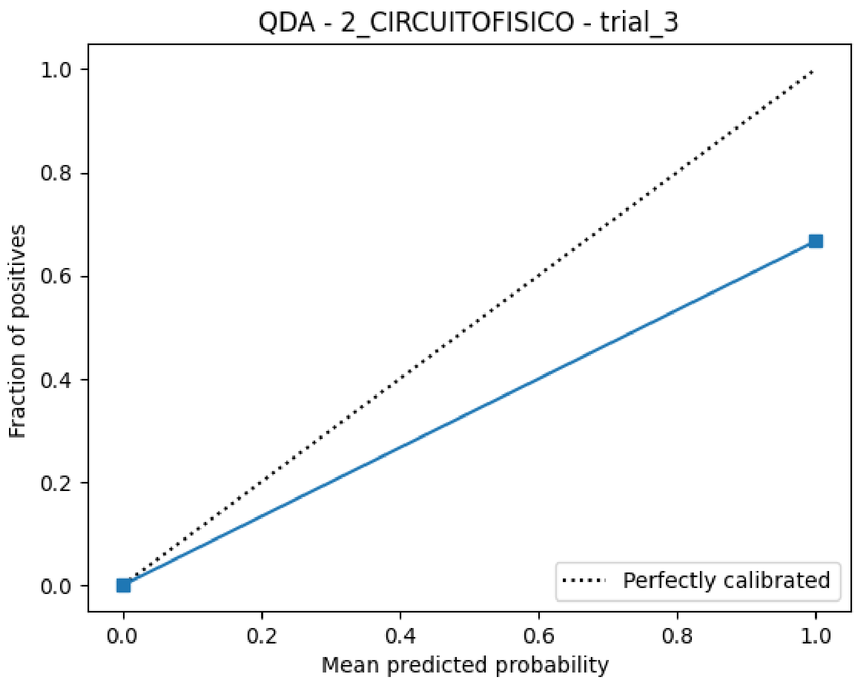

QDA - ONOFF - None
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

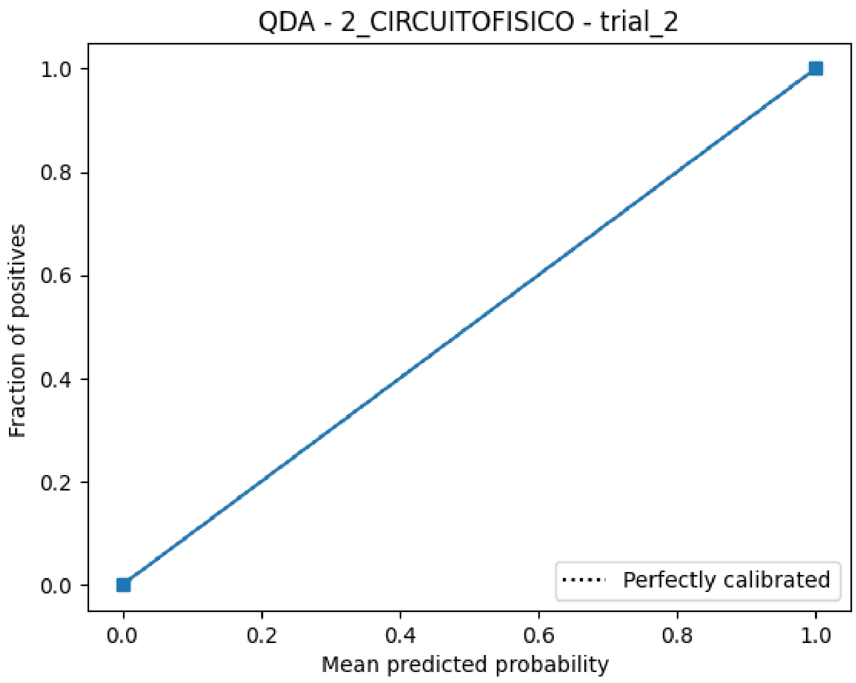

QDA - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

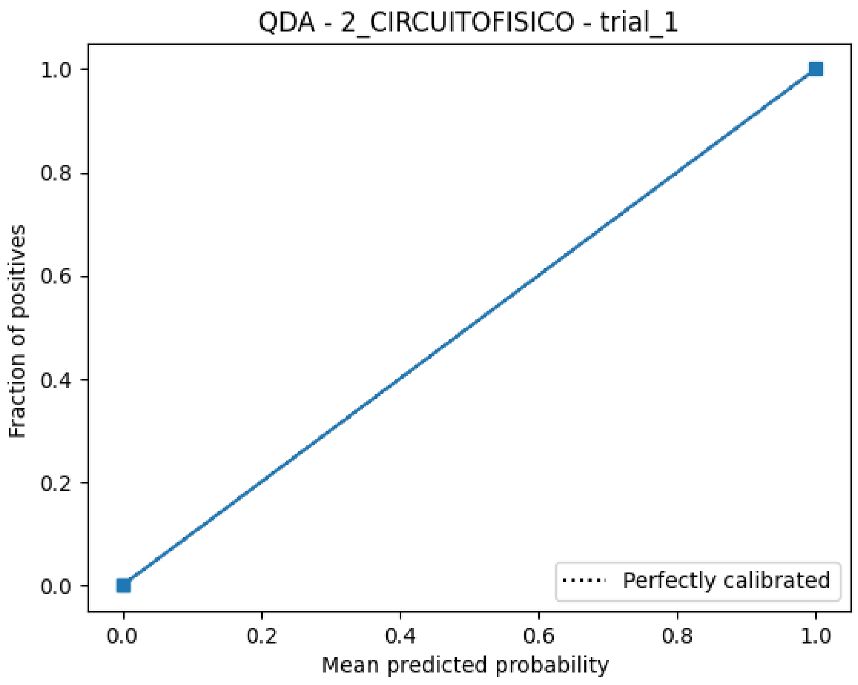

QDA - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

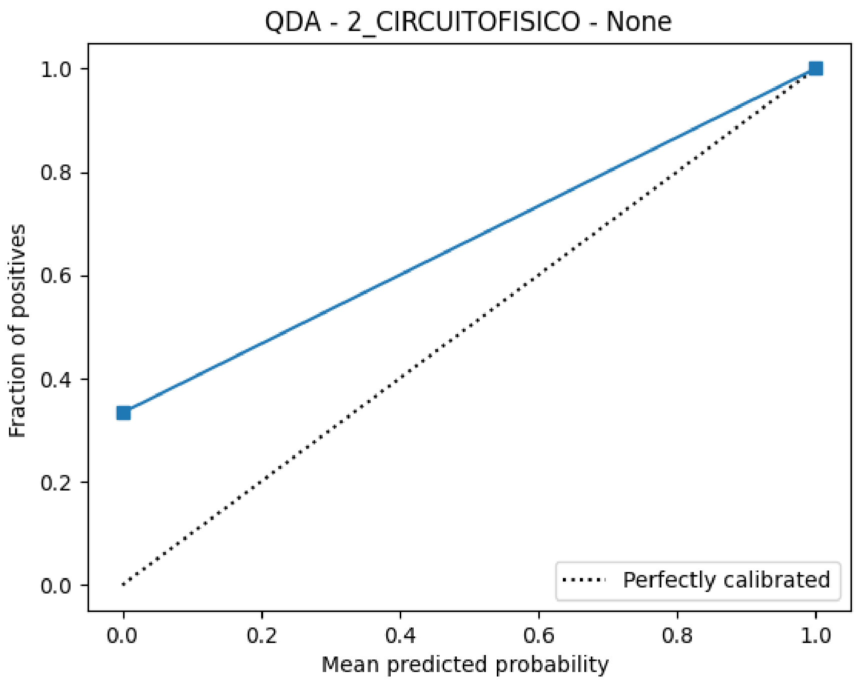

QDA - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

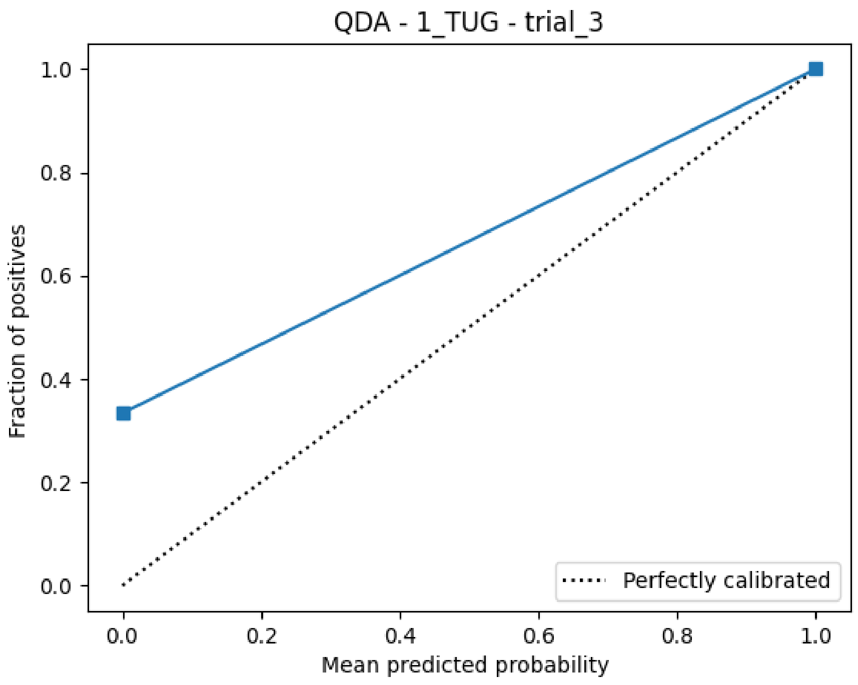

QDA - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

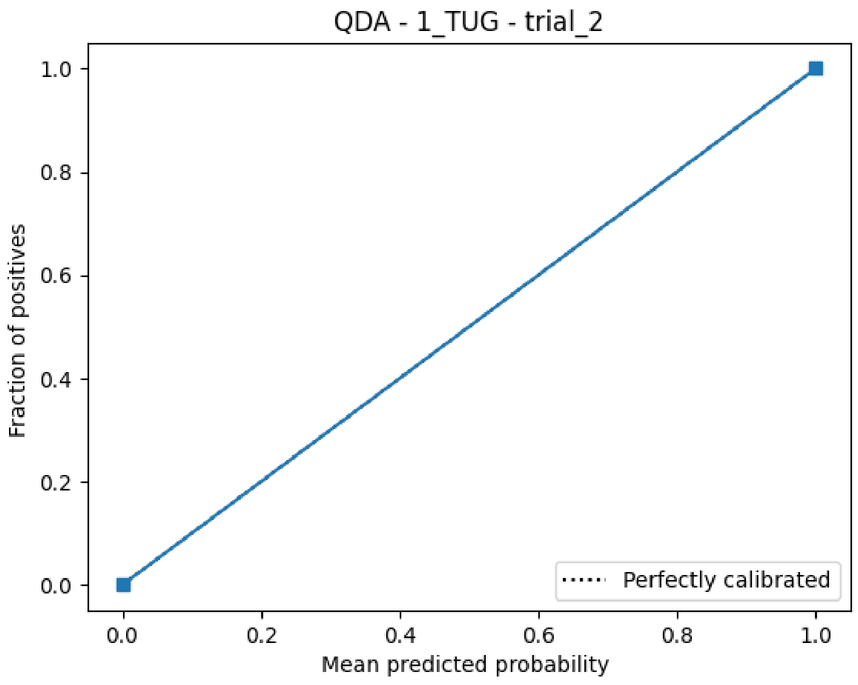

QDA - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

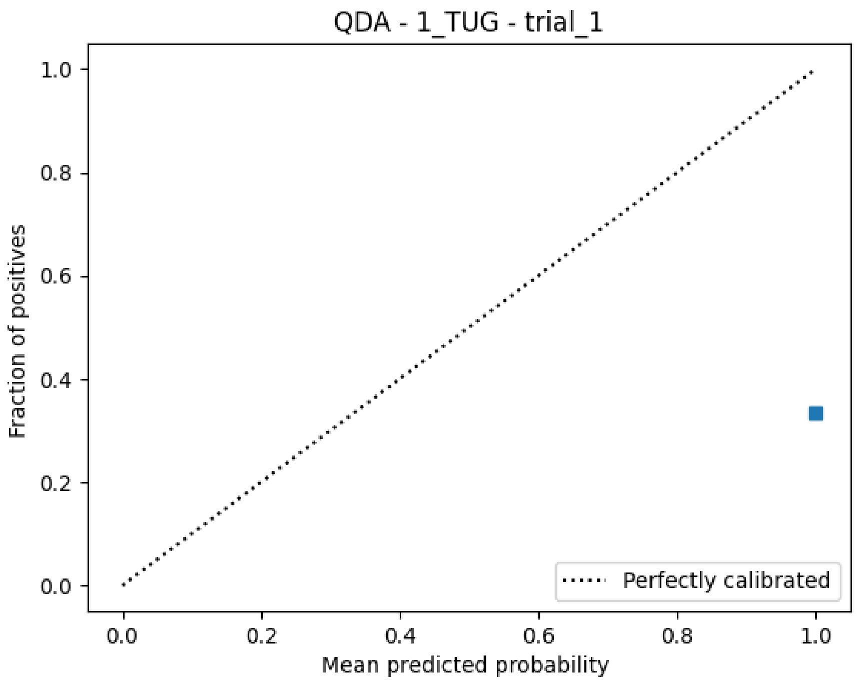

QDA - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

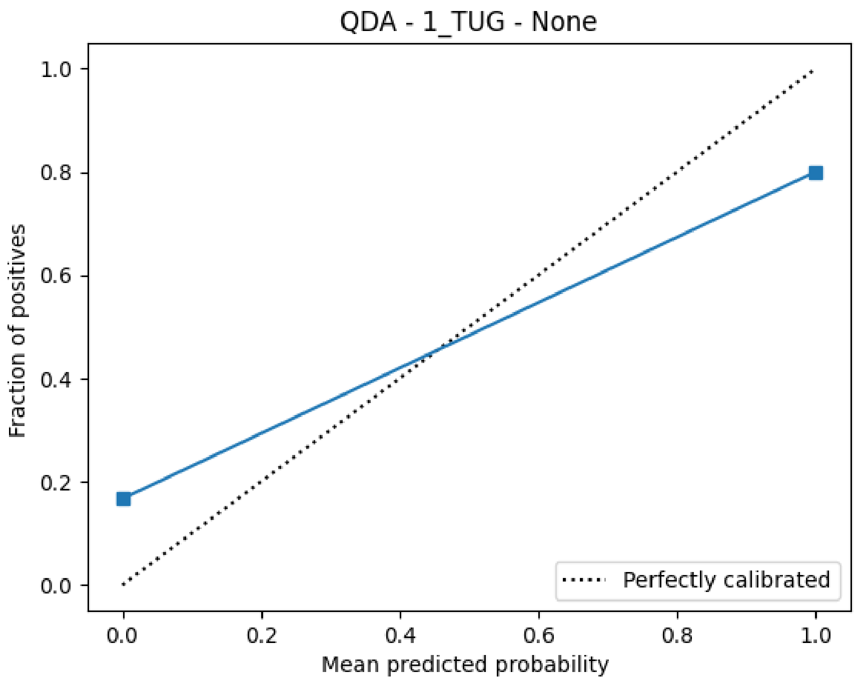

QDA - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

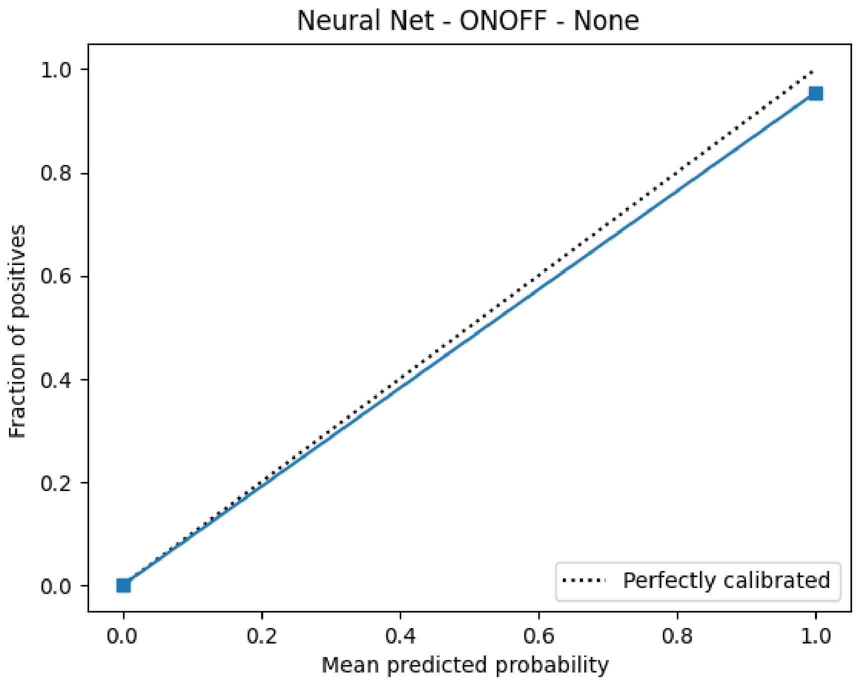

QDA - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

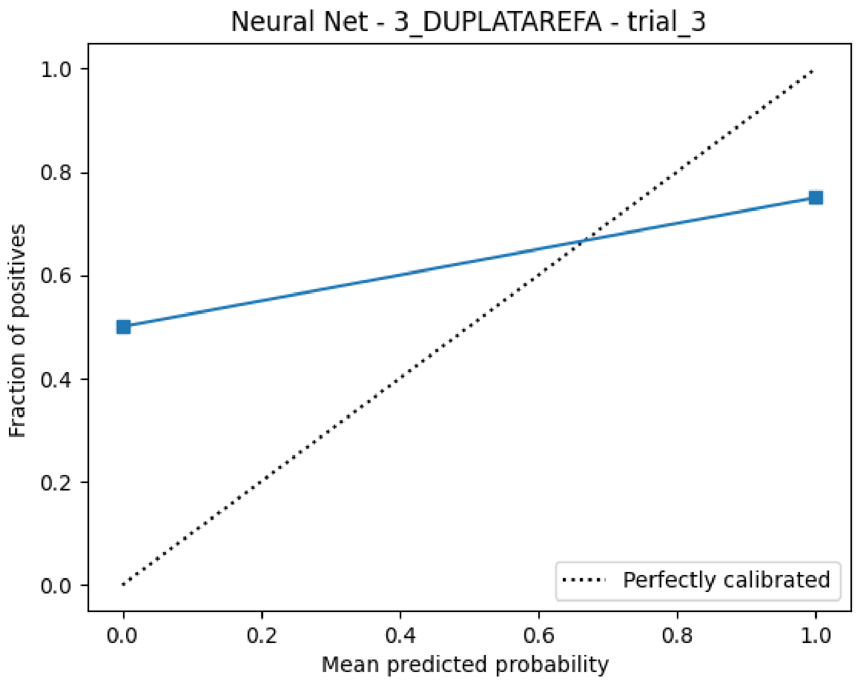

QDA - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

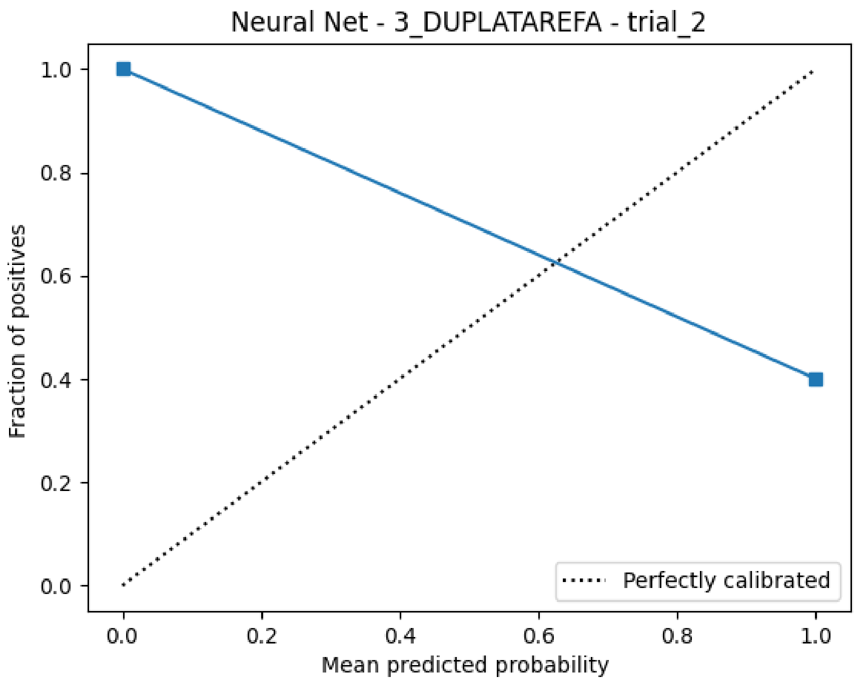

QDA - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

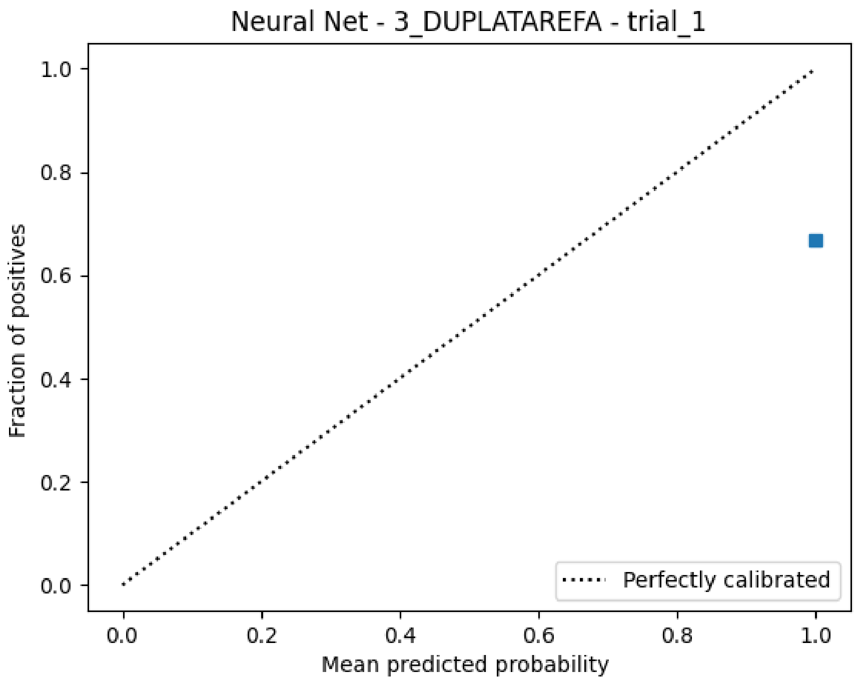

QDA - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

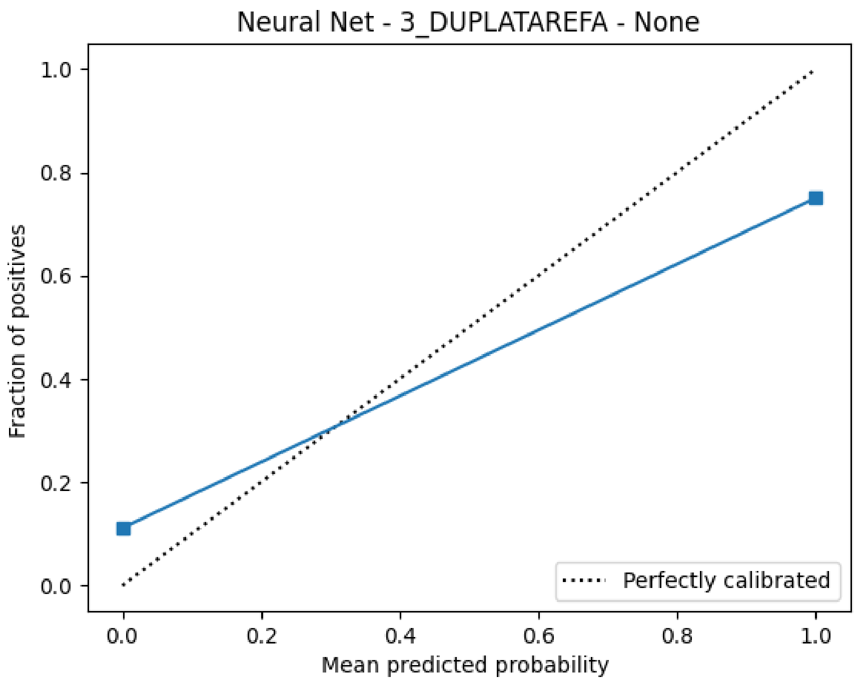

QDA - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

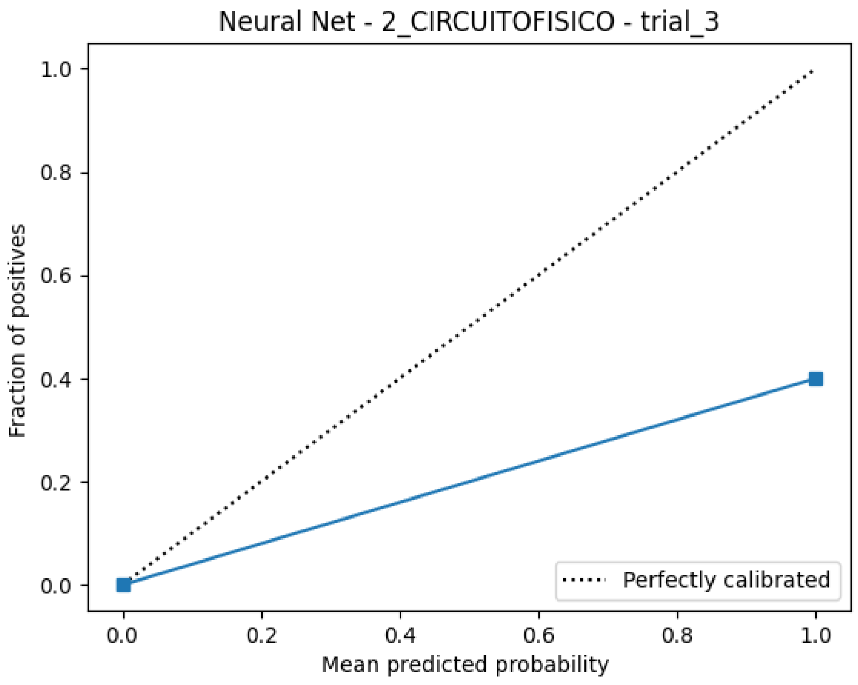

Neural Net - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

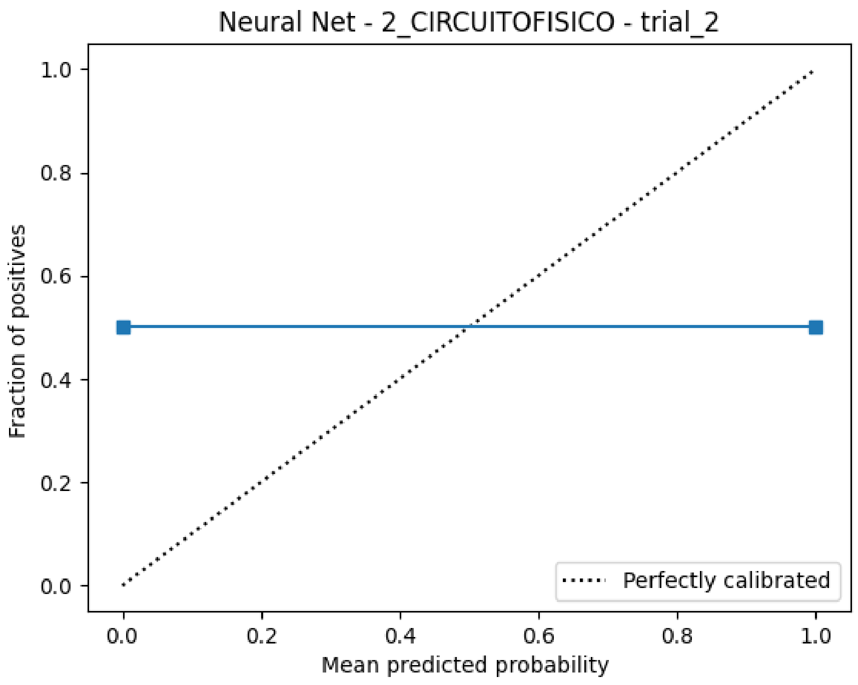

Neural Net - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

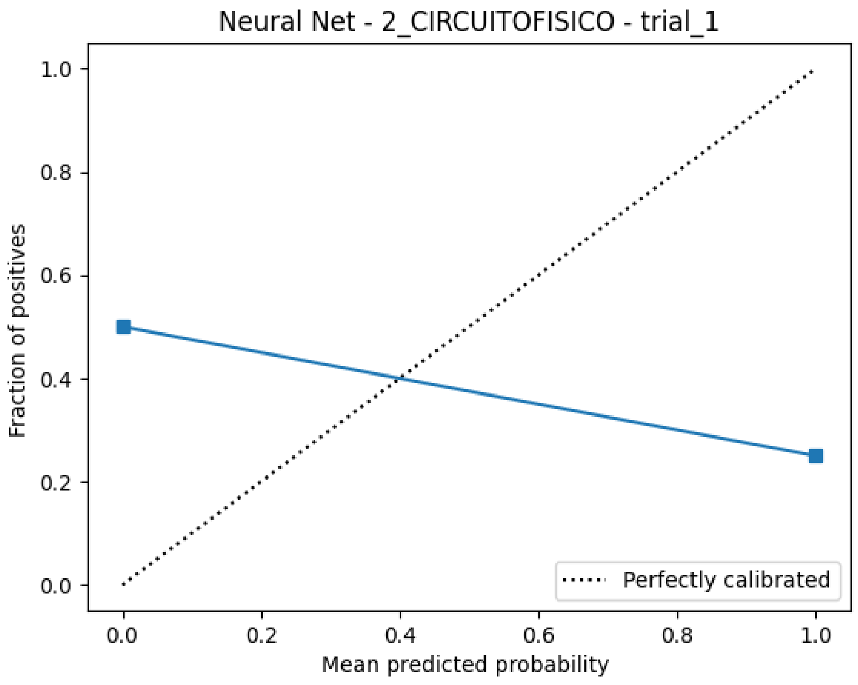

Neural Net - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

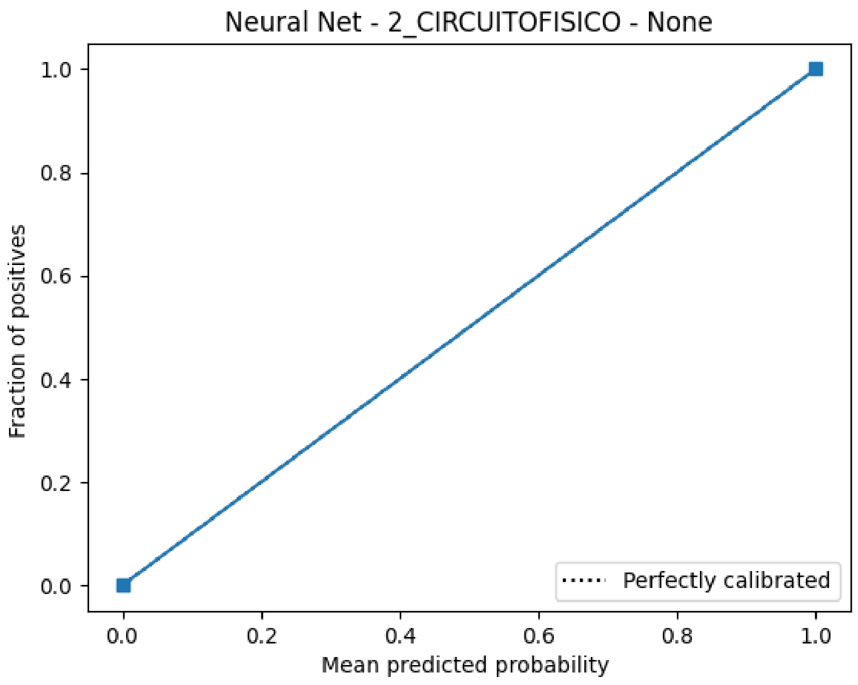

Neural Net - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

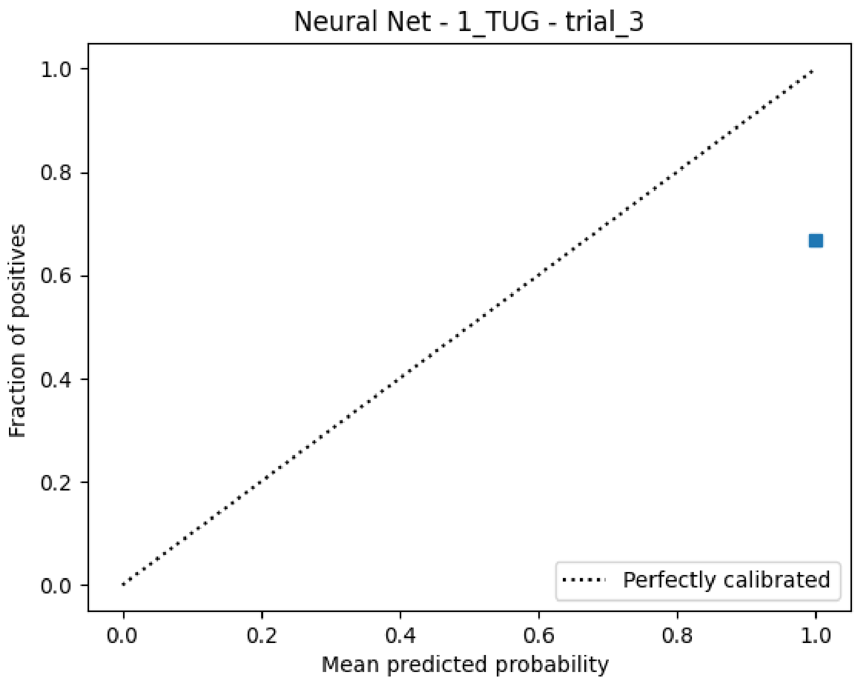

Neural Net - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

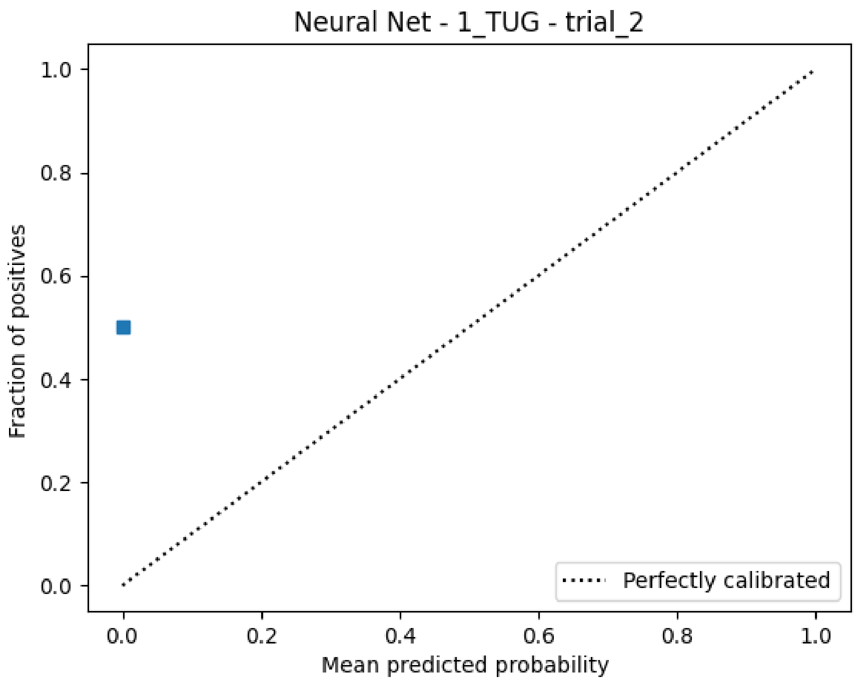

Neural Net - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

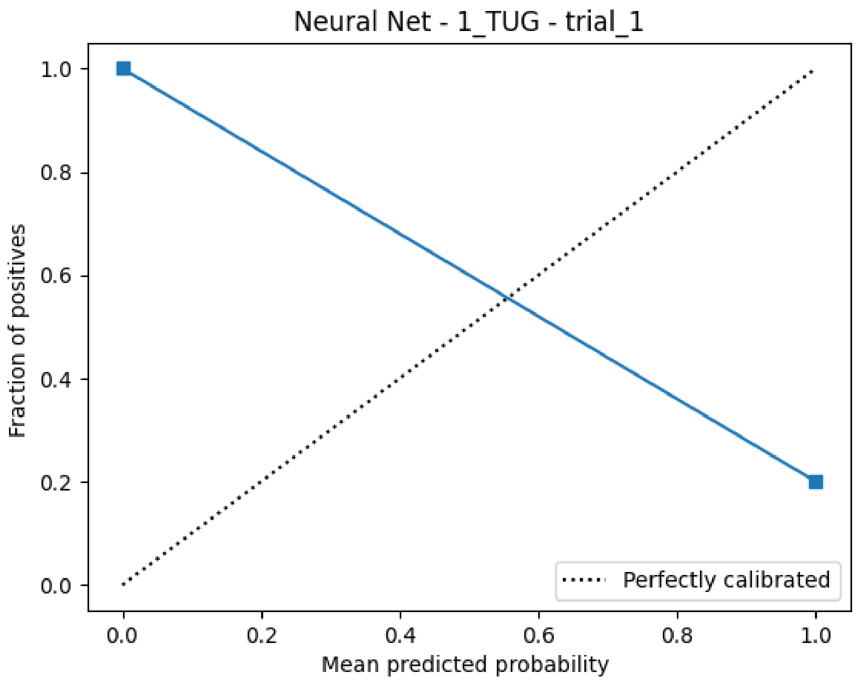

Neural Net - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

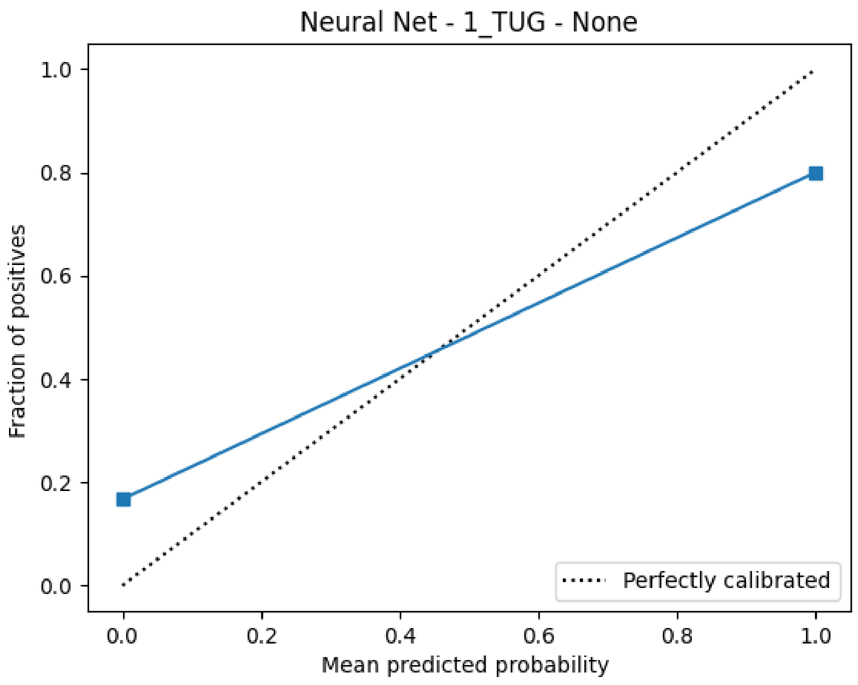

Neural Net - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

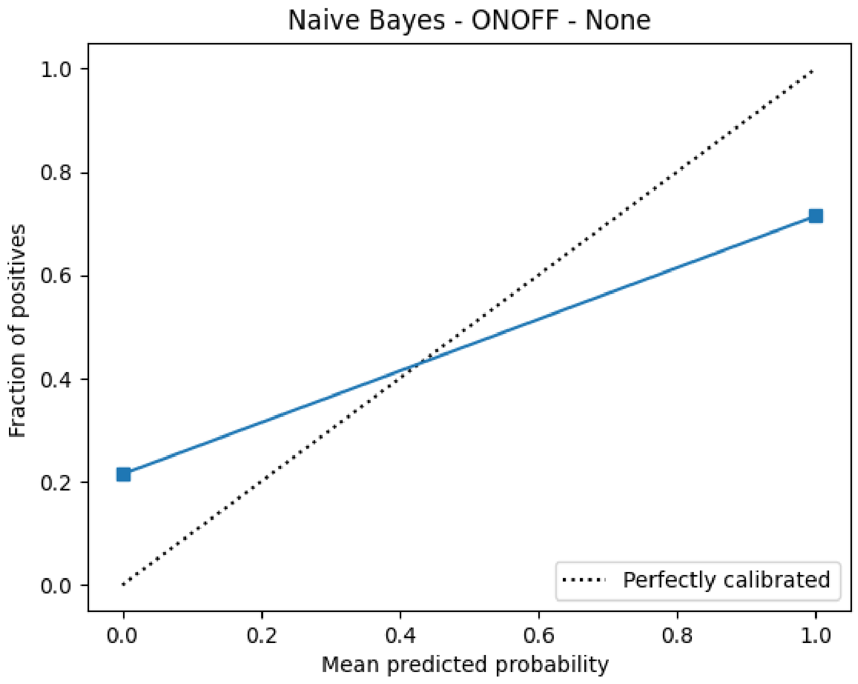

Neural Net - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

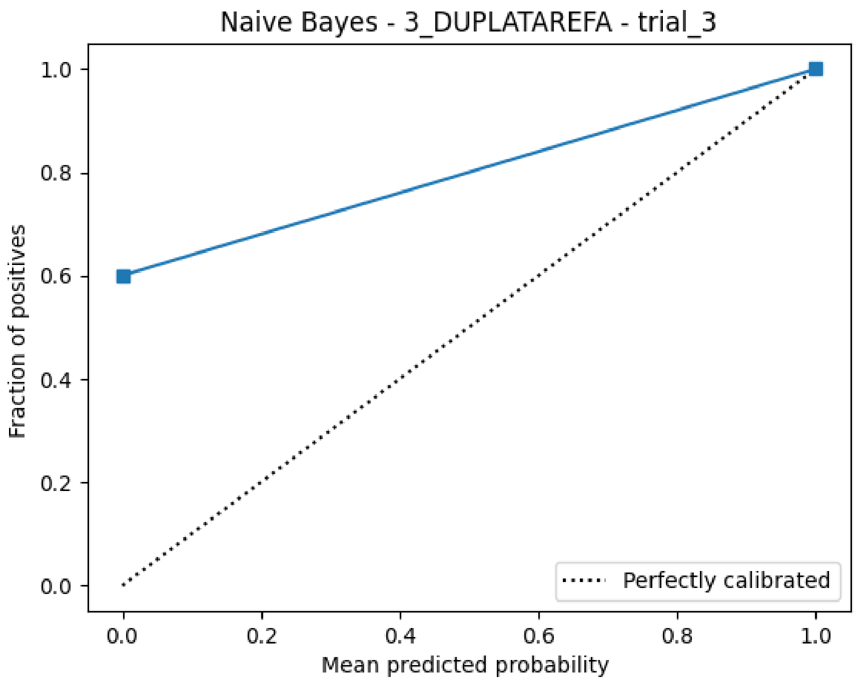

Neural Net - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

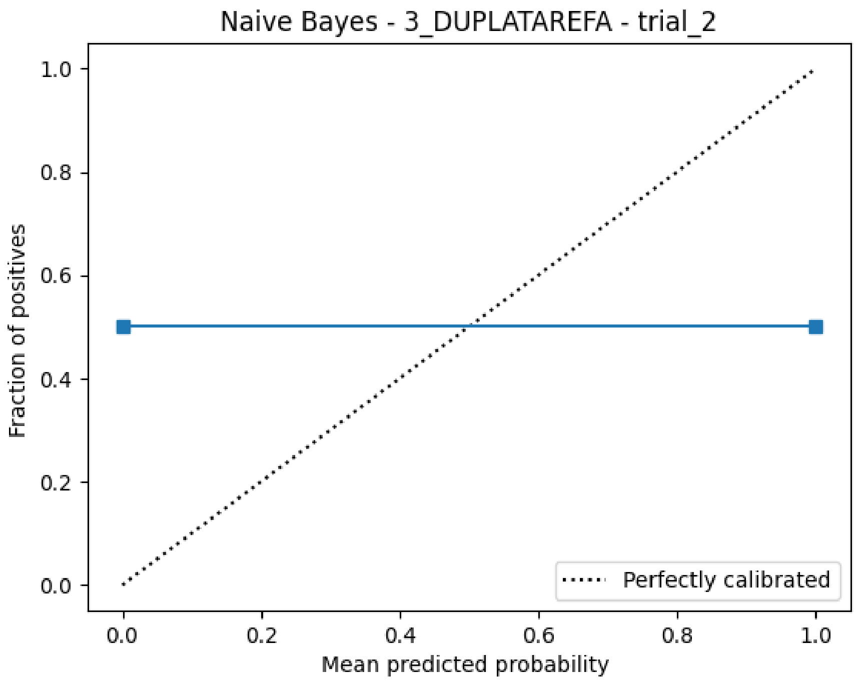

Neural Net - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

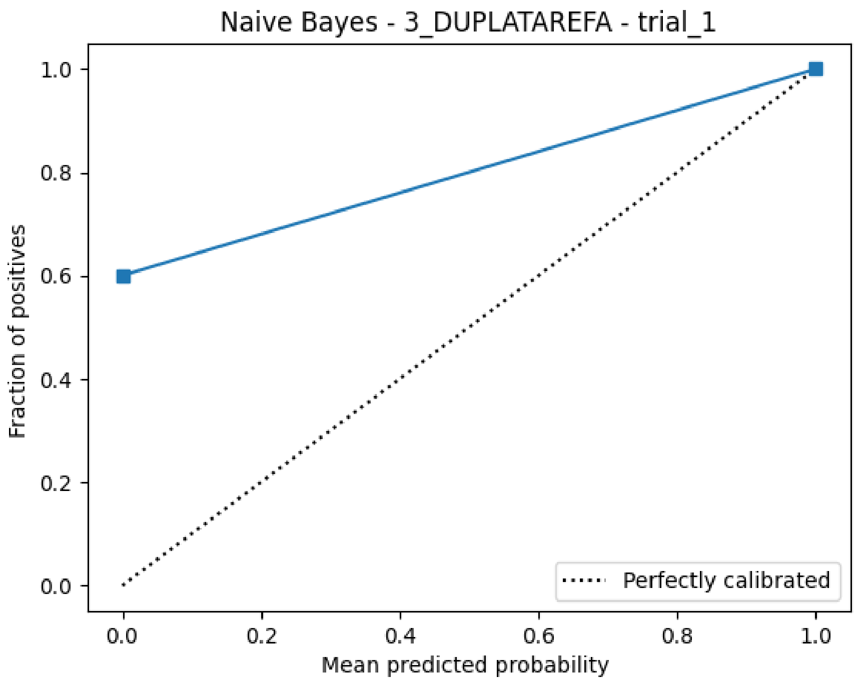

Neural Net - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

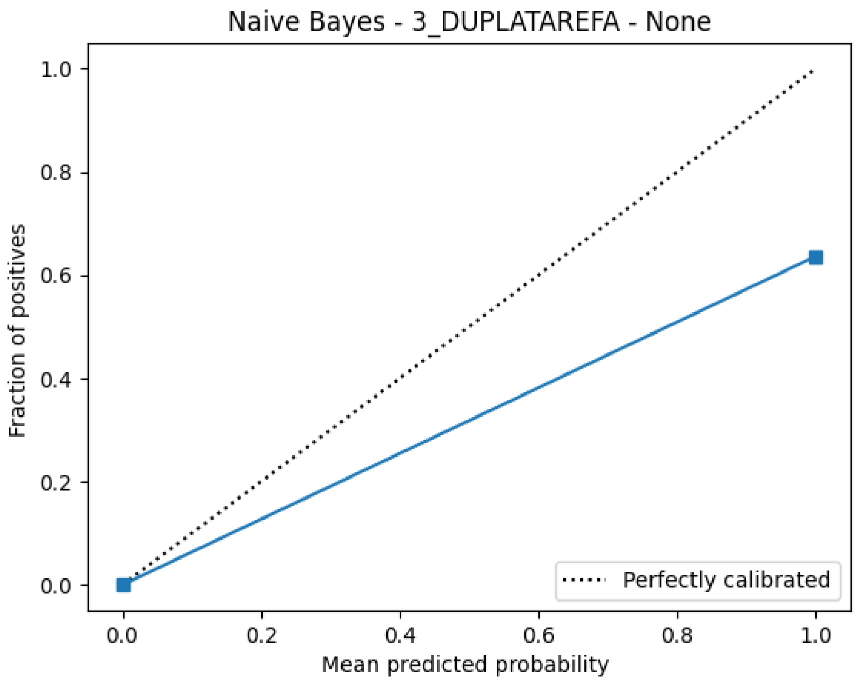

Neural Net - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

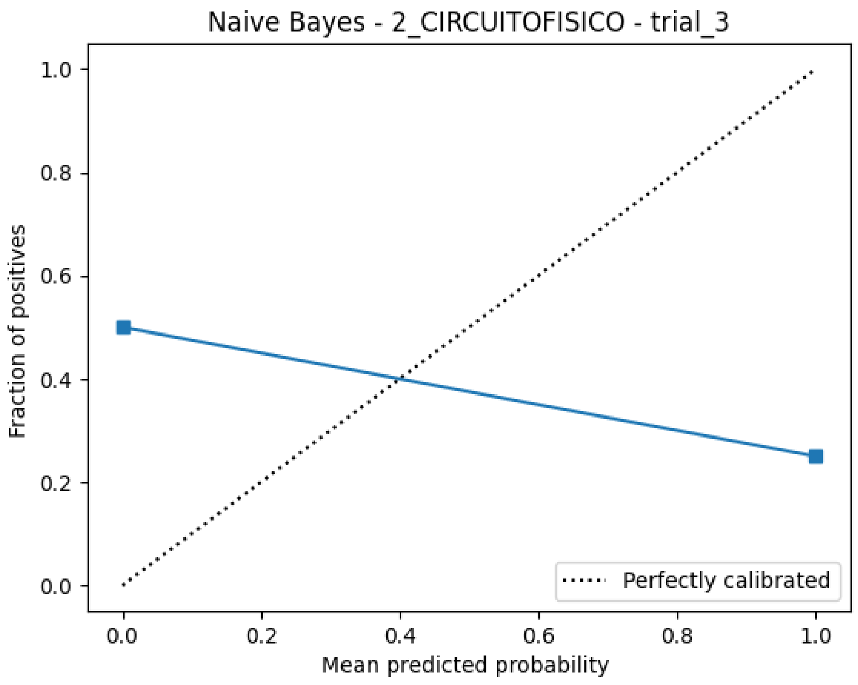

Naive Bayes - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

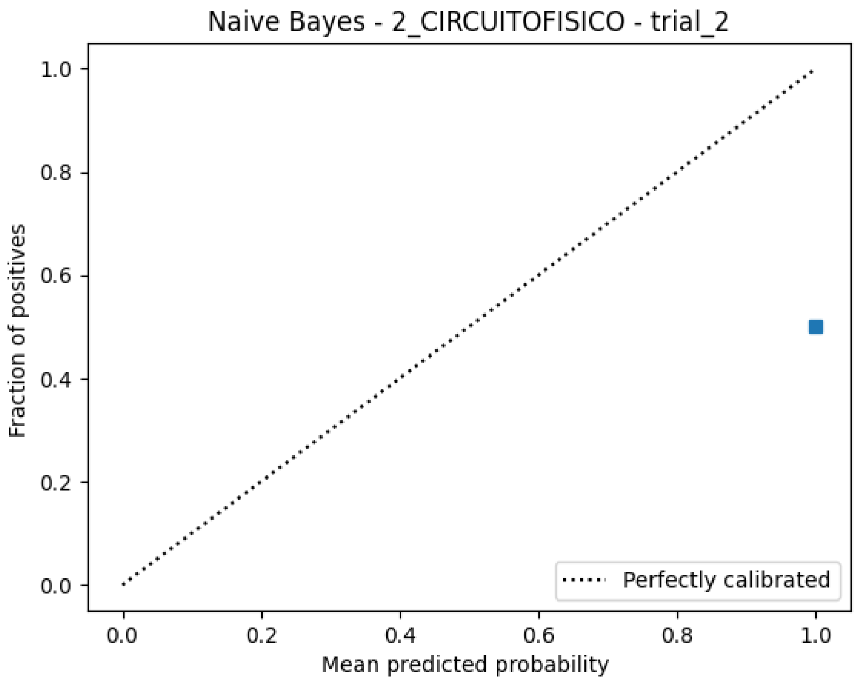

Naive Bayes - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

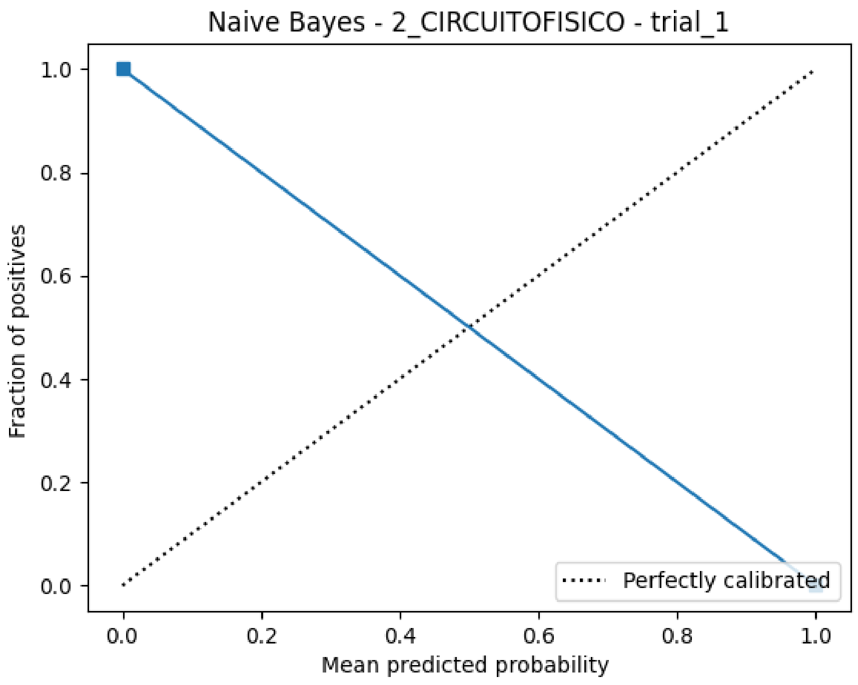

Naive Bayes - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

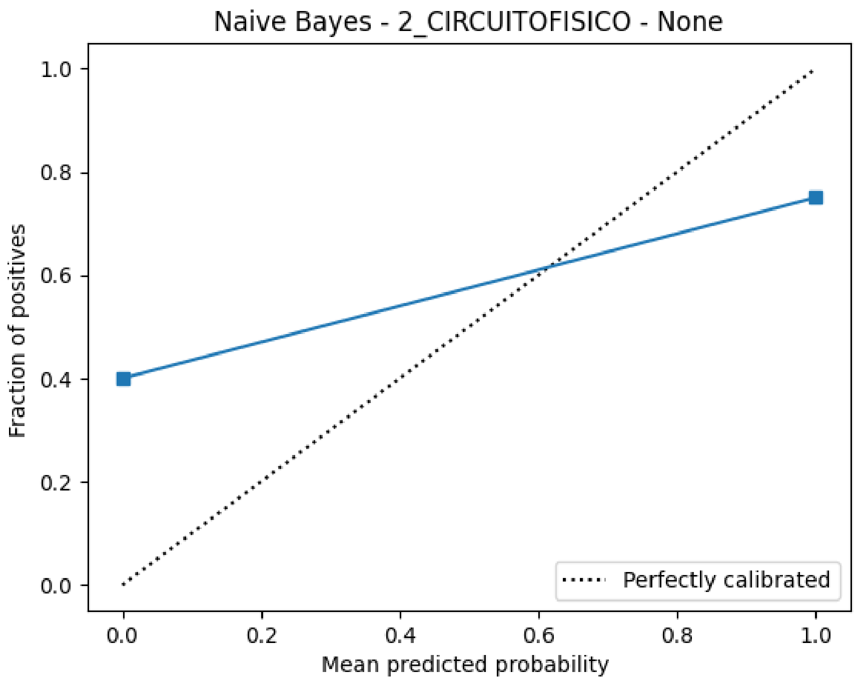

Naive Bayes - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

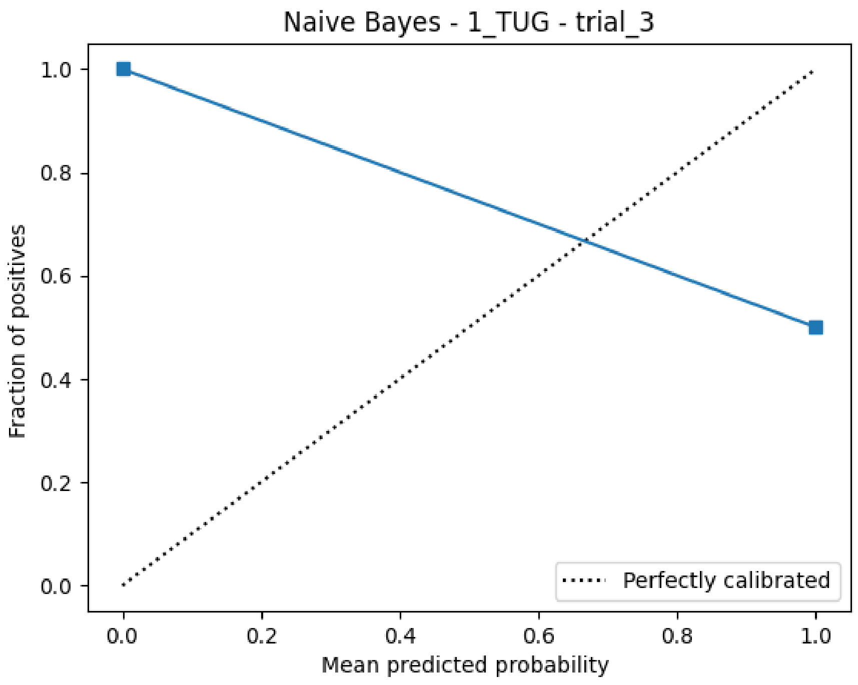

Naive Bayes - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

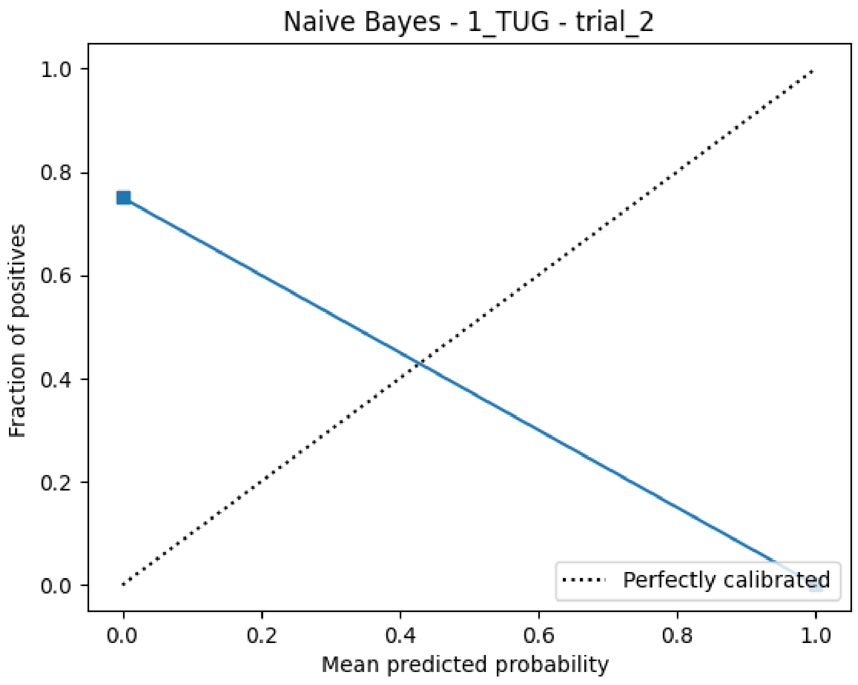

Naive Bayes - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

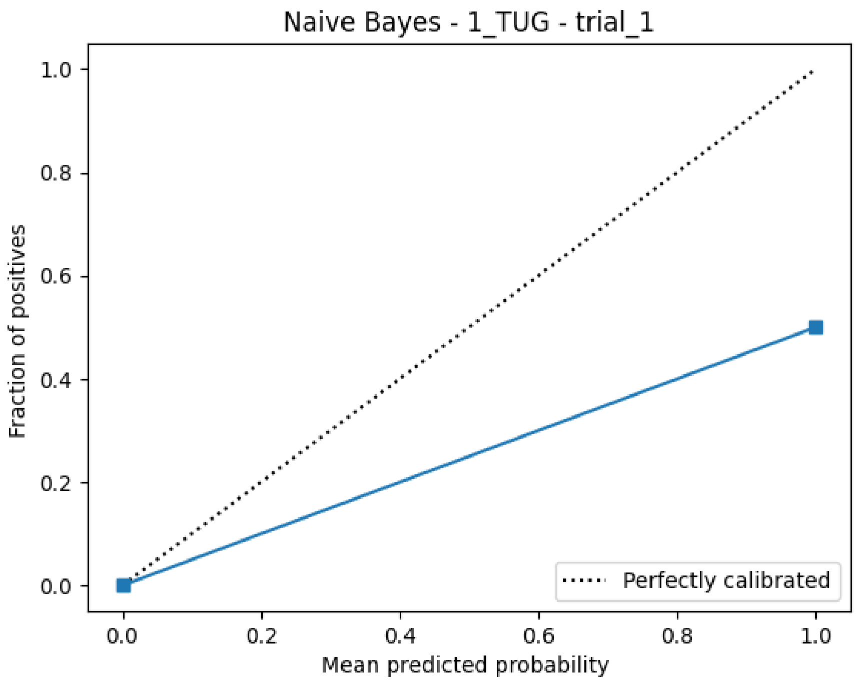

Naive Bayes - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

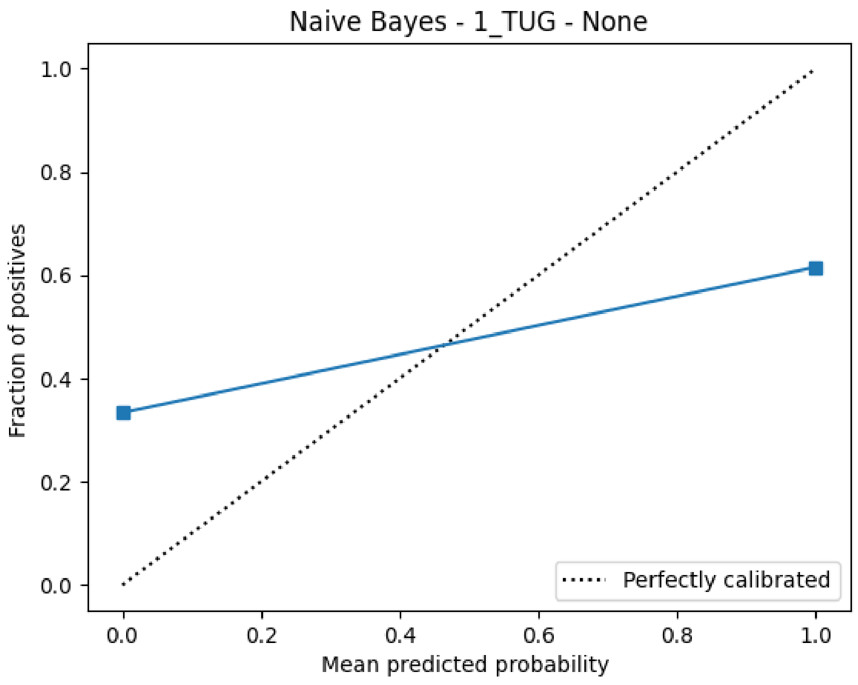

Naive Bayes - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

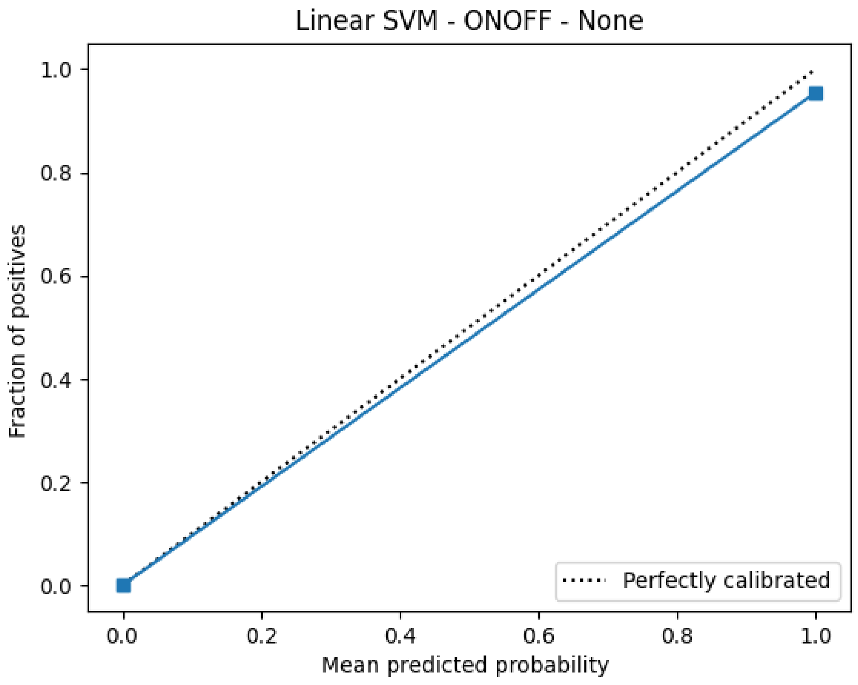

Naive Bayes - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

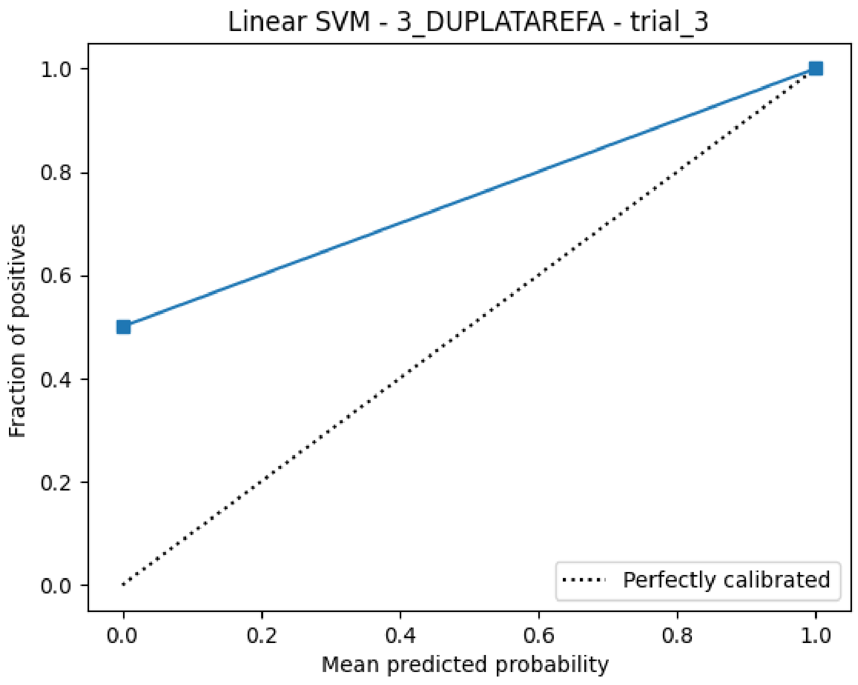

Naive Bayes - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

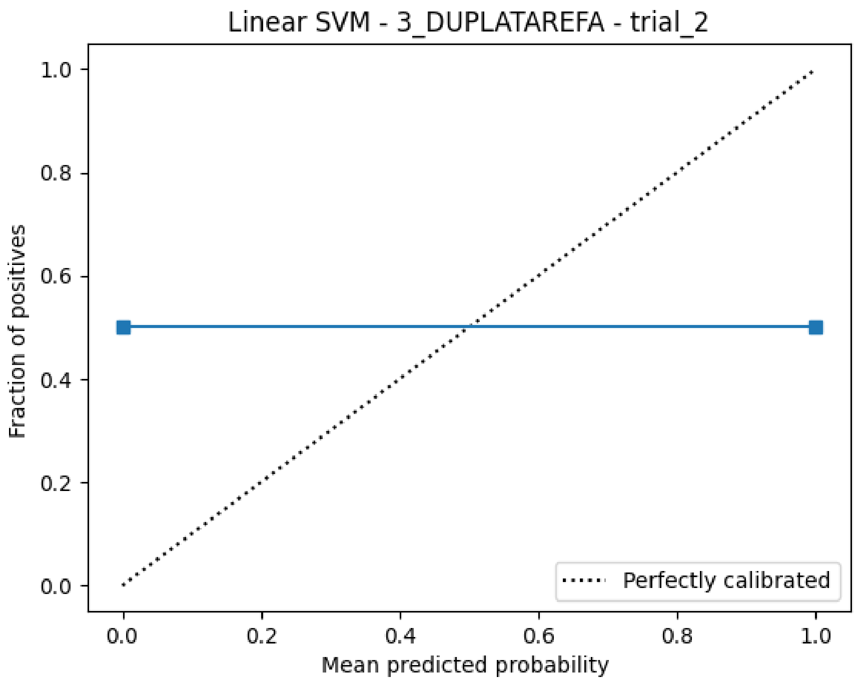

Naive Bayes - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

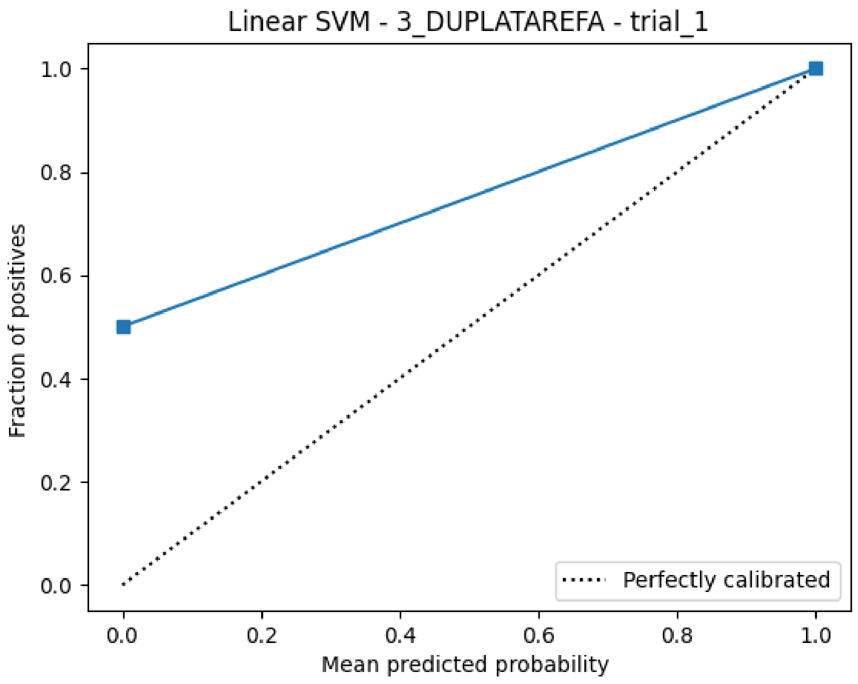

Naive Bayes - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

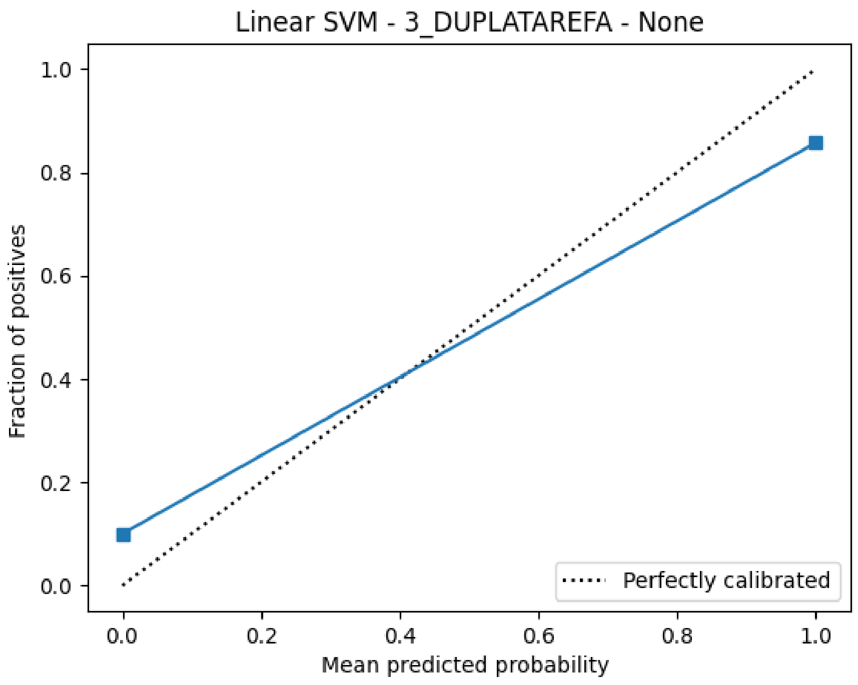

Naive Bayes - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

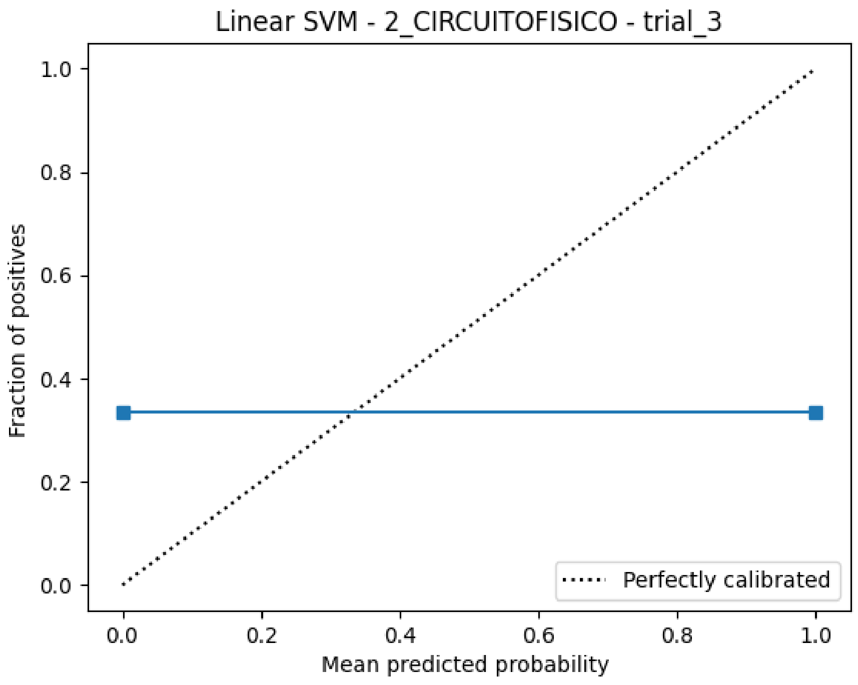

Linear SVM - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

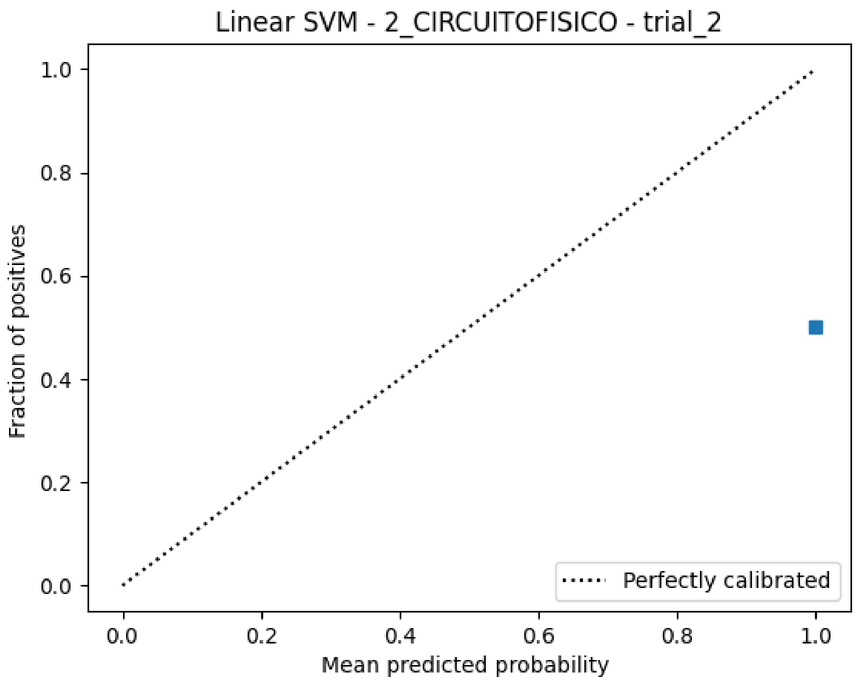

Linear SVM - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

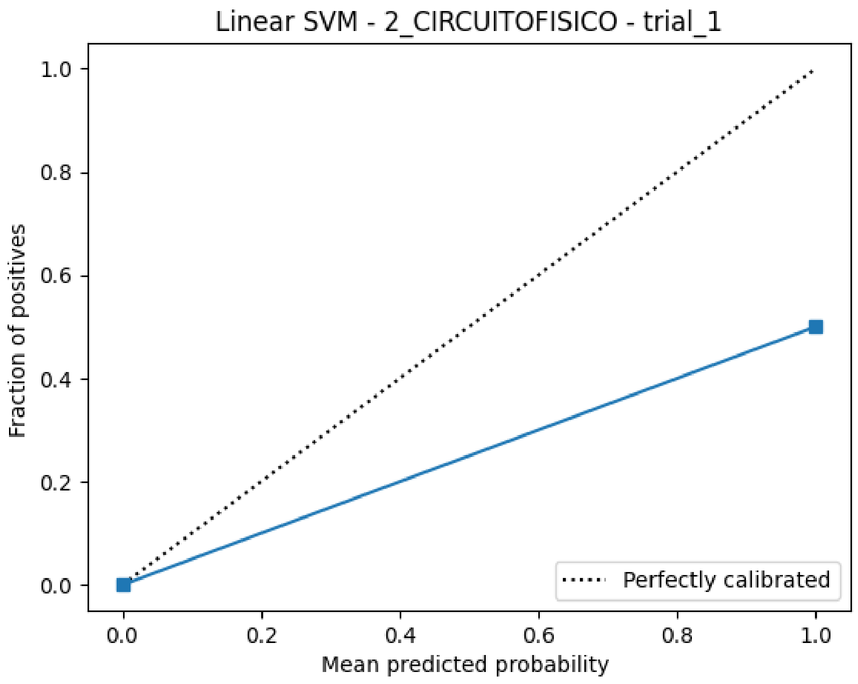

Linear SVM - 3_DUPLATAREFA - trial_2
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

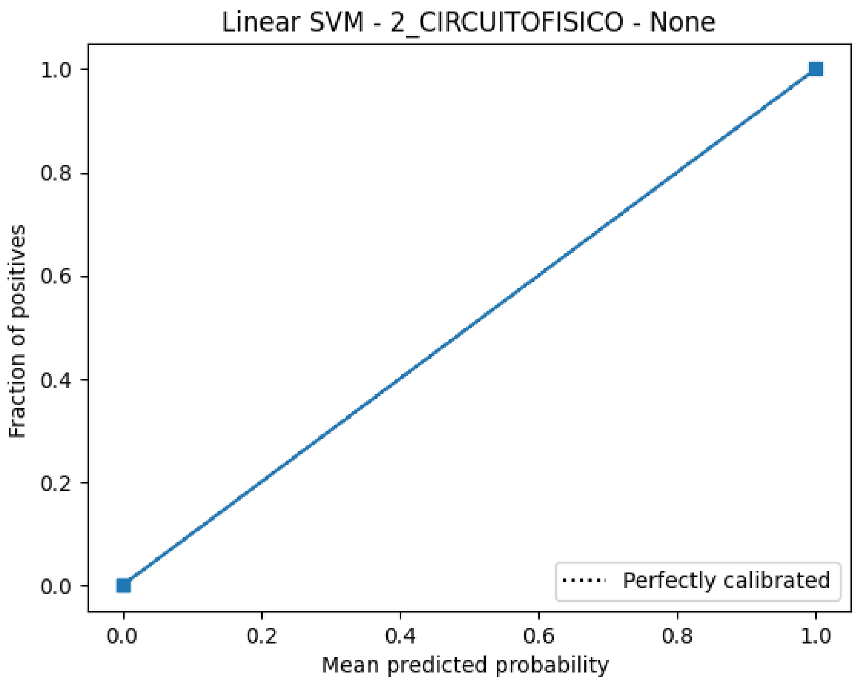

Linear SVM - 3_DUPLATAREFA - trial_1
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

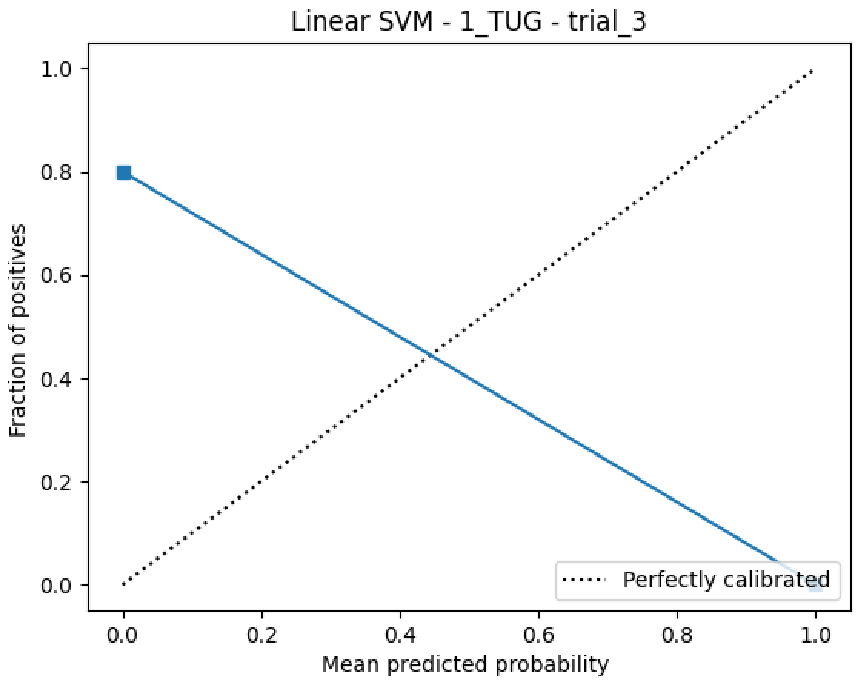

Linear SVM - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

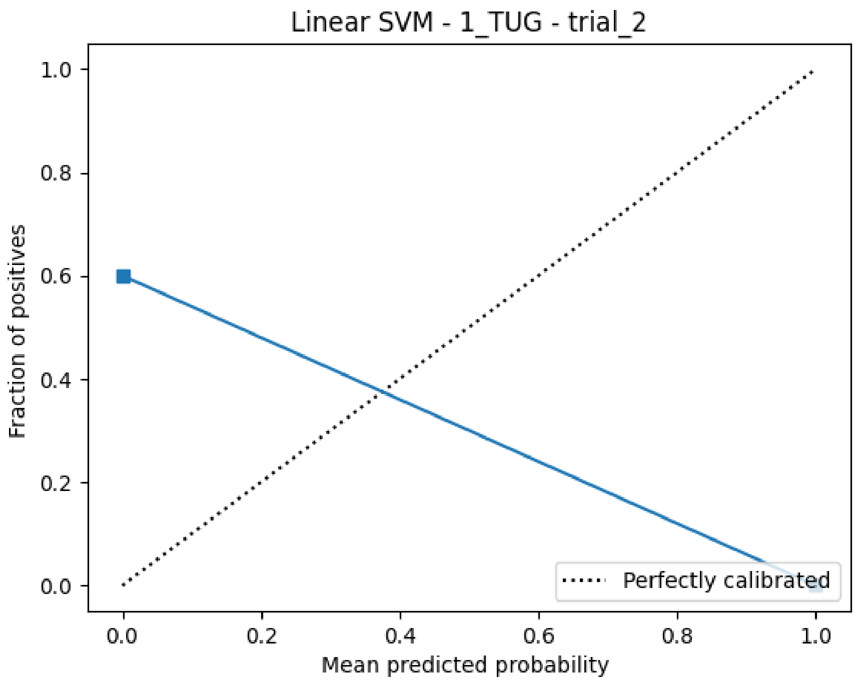

Linear SVM - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

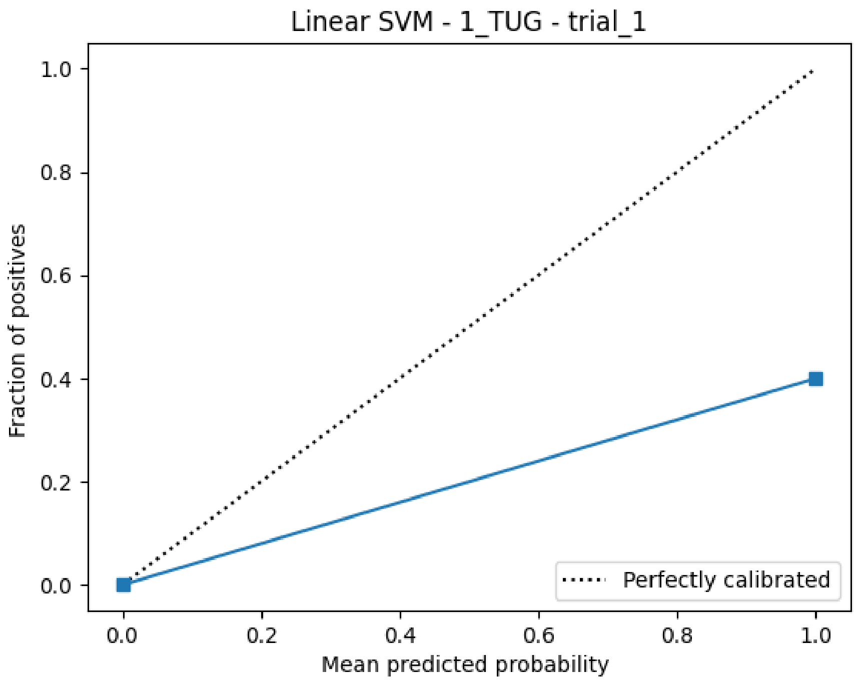

Linear SVM - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

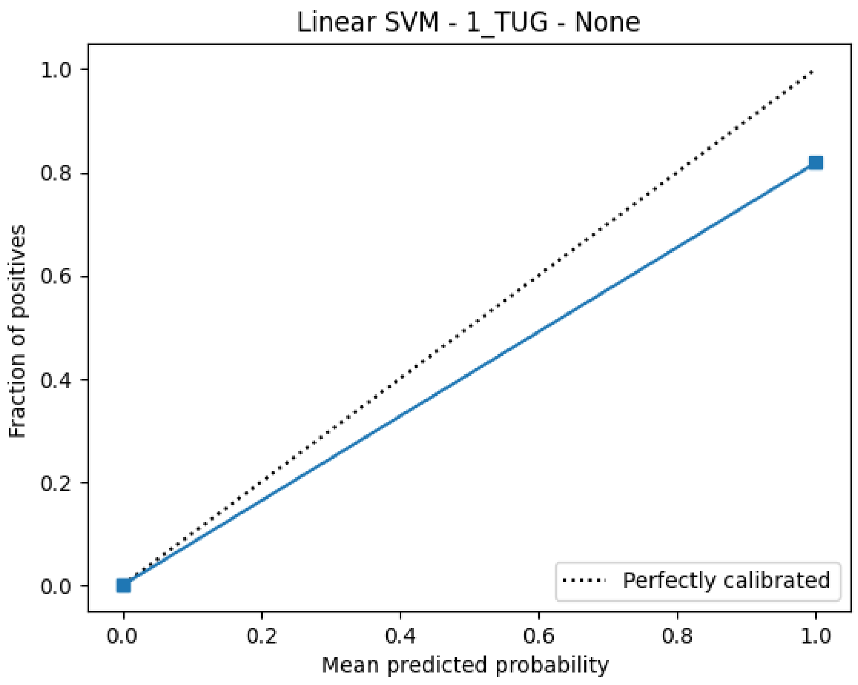

Linear SVM - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

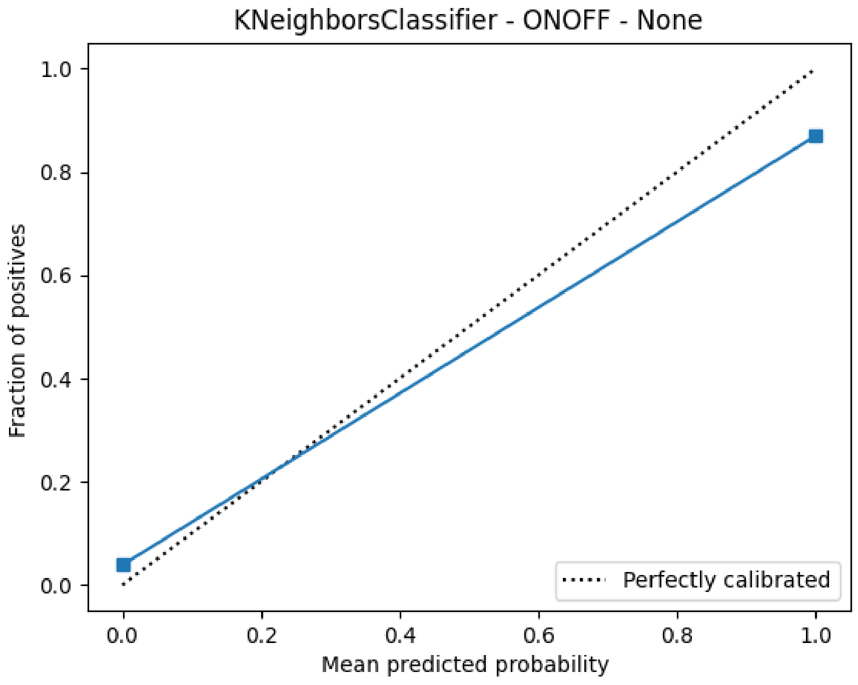

Linear SVM - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

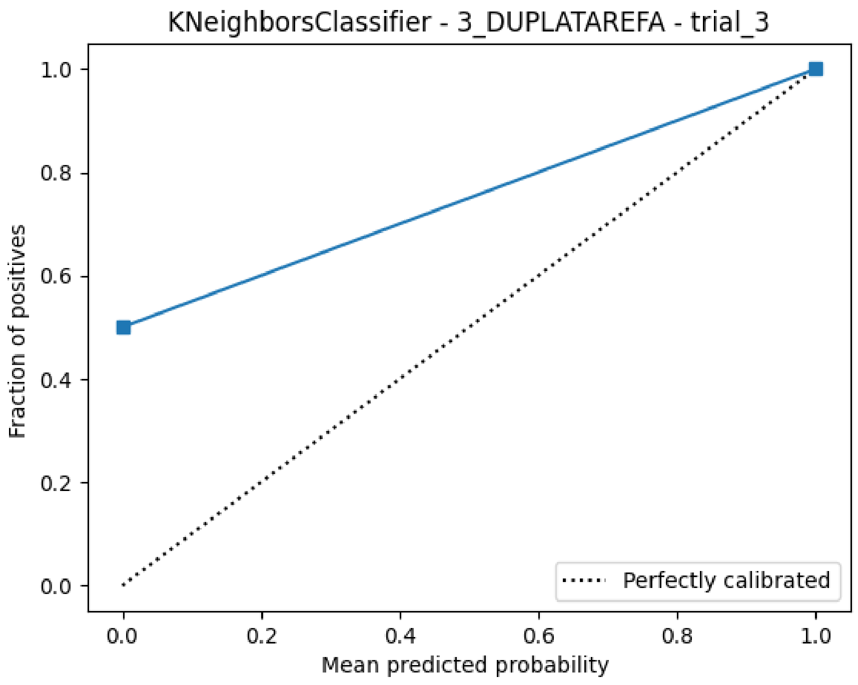

Linear SVM - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

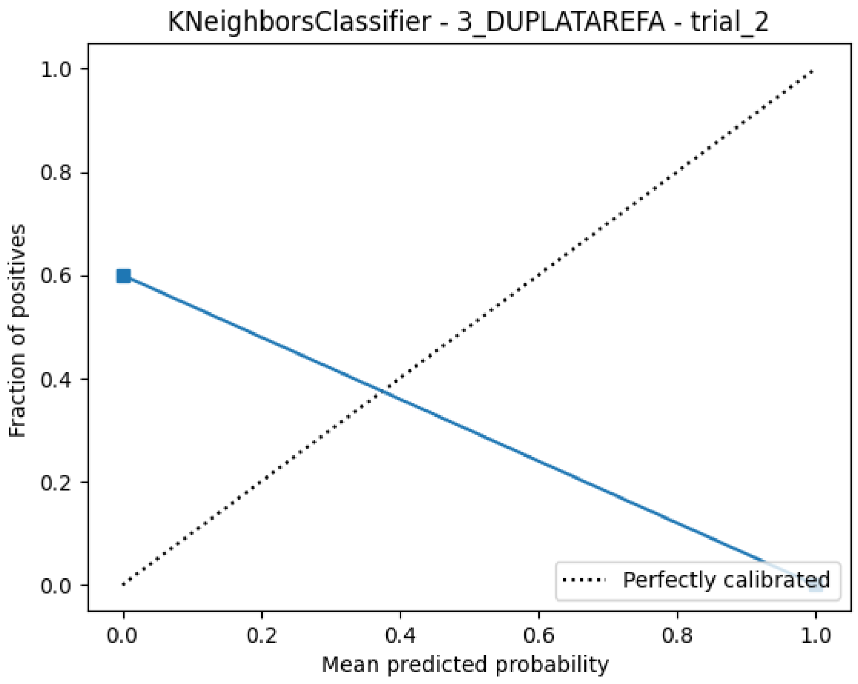

Linear SVM - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

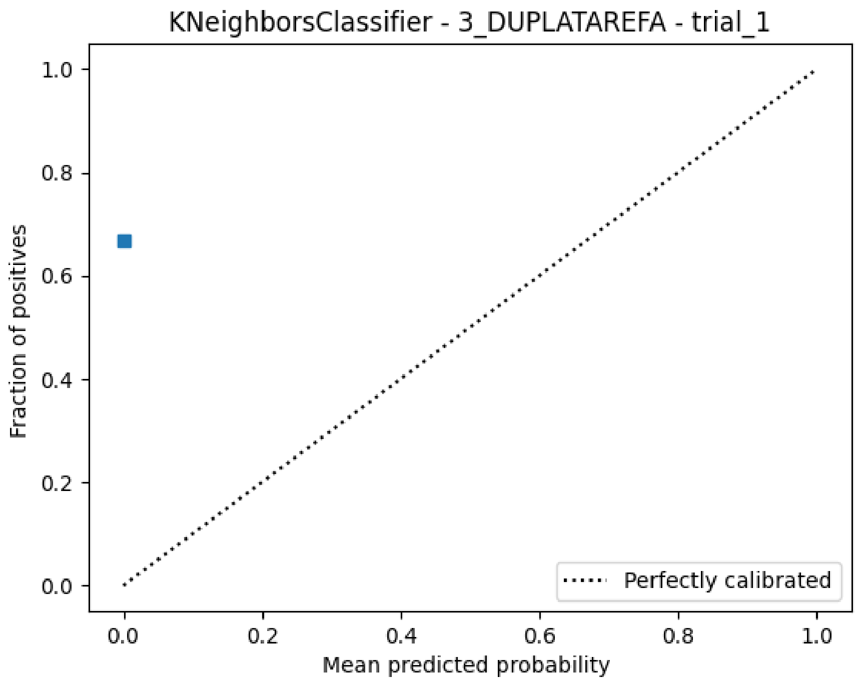

Linear SVM - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

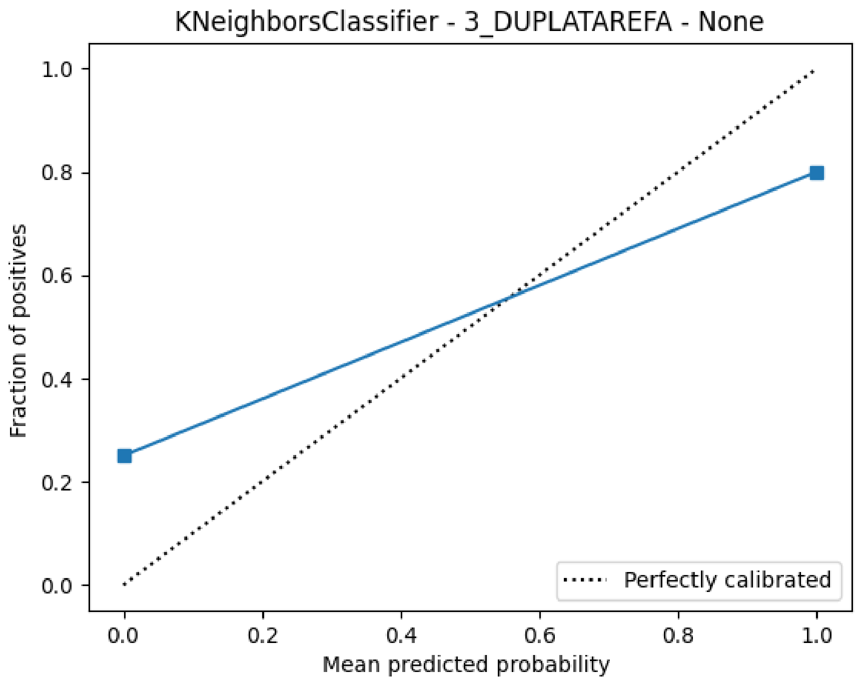

Linear SVM - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

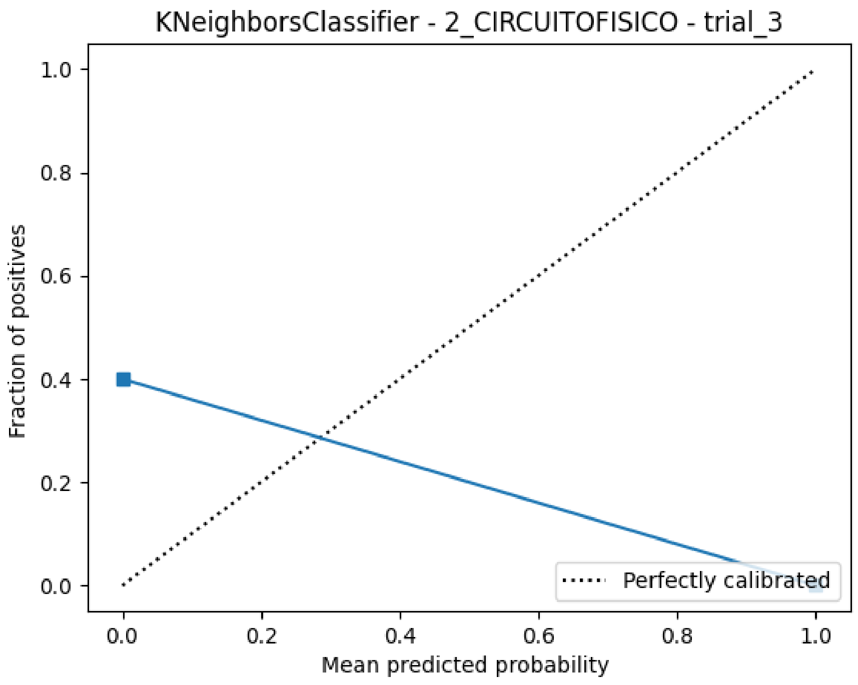

KNeighborsClassifier - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

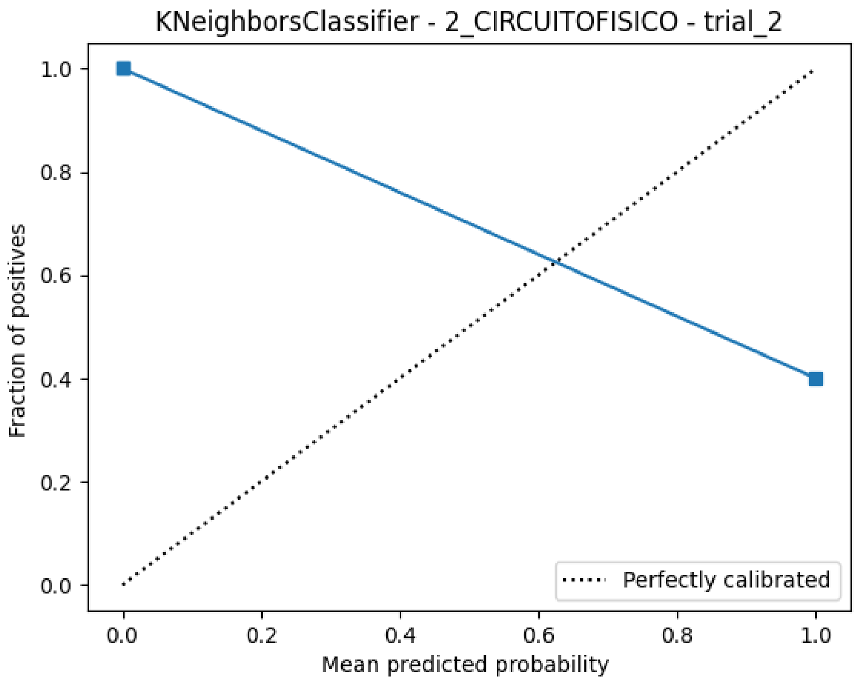

KNeighborsClassifier - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

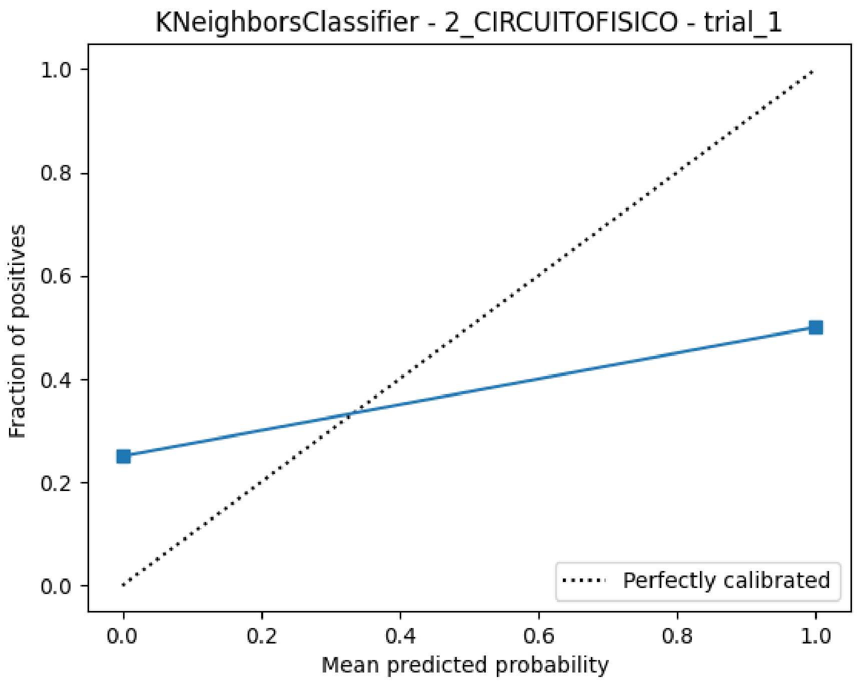

KNeighborsClassifier - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

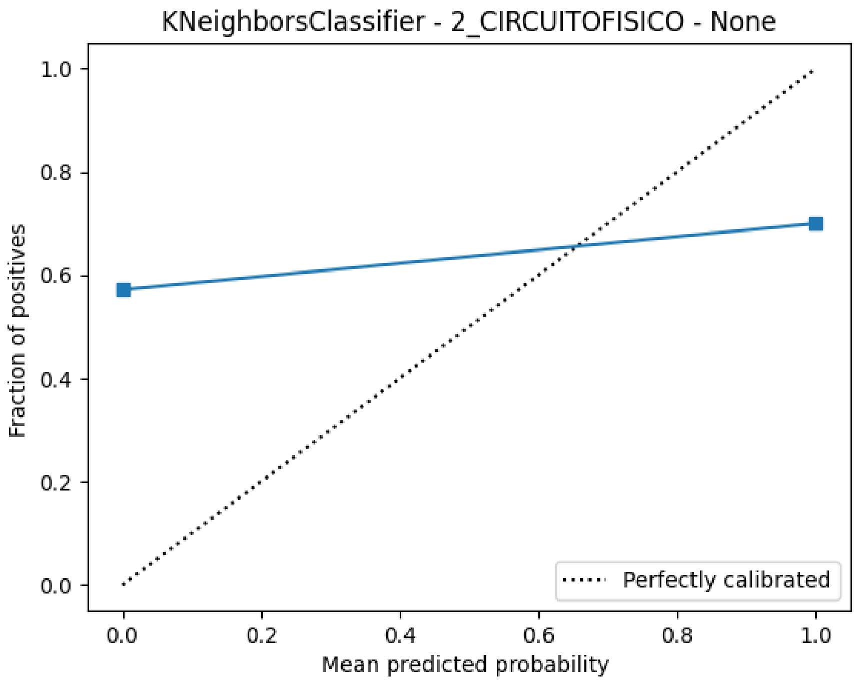

KNeighborsClassifier - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

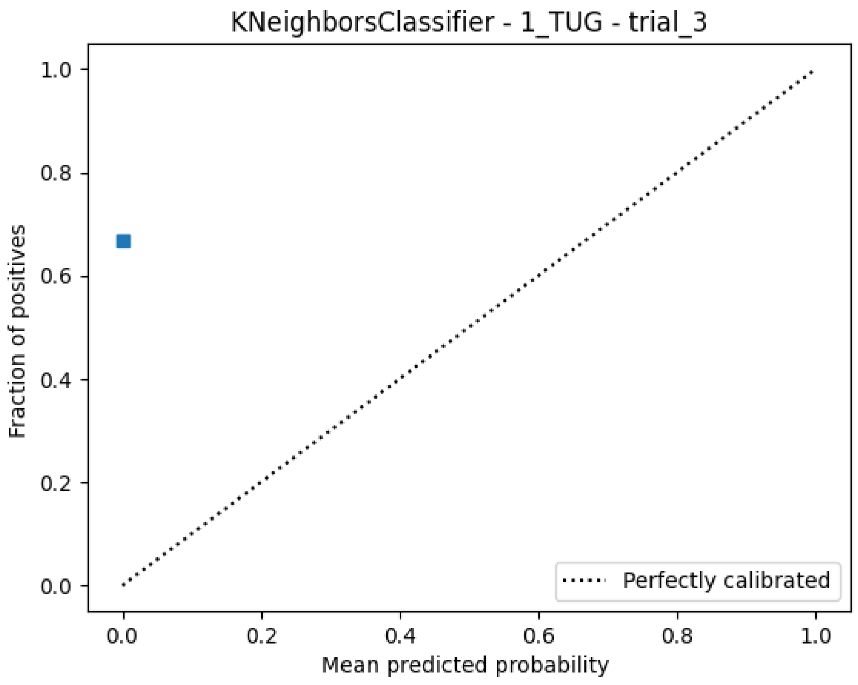

KNeighborsClassifier - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

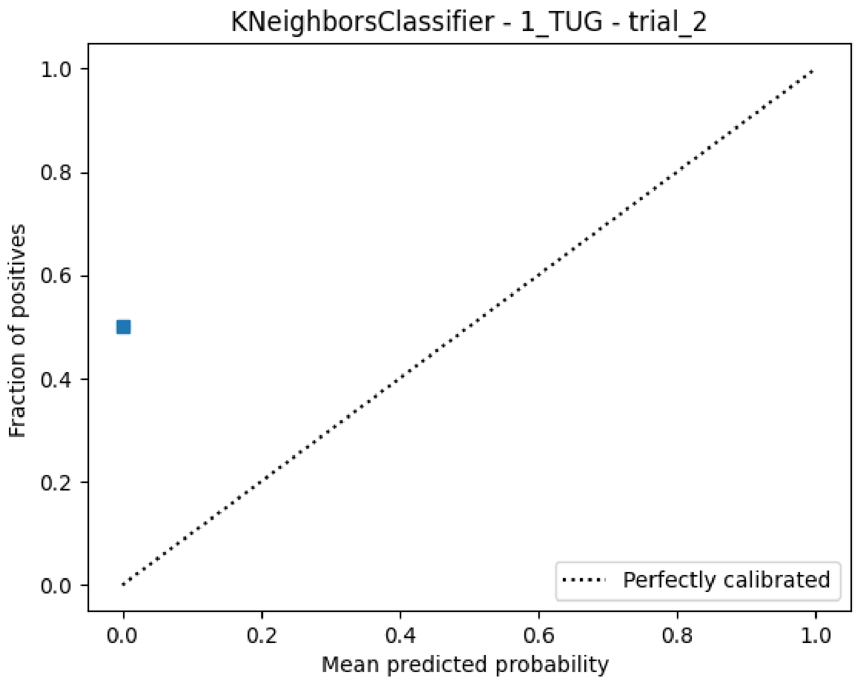

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

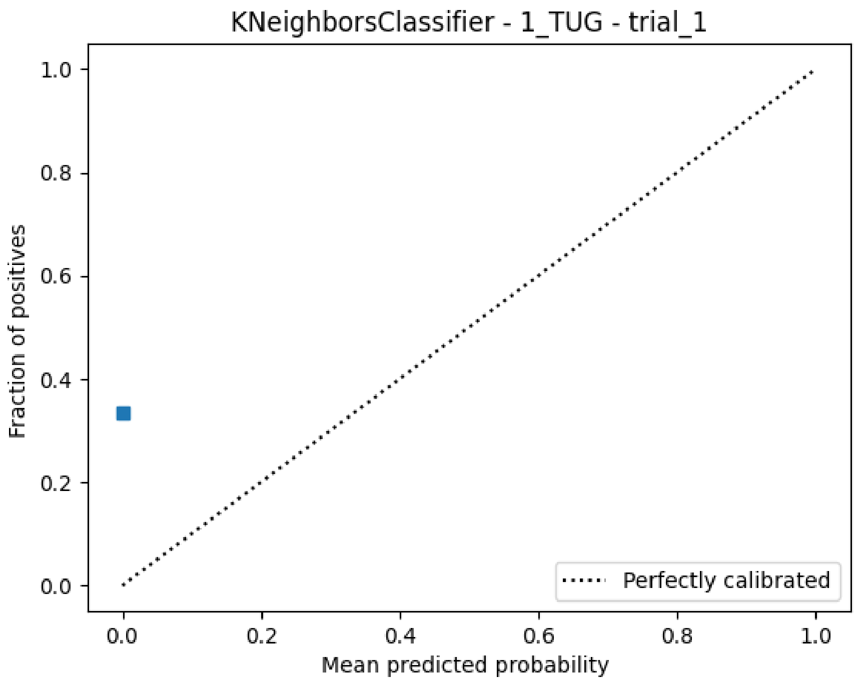

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

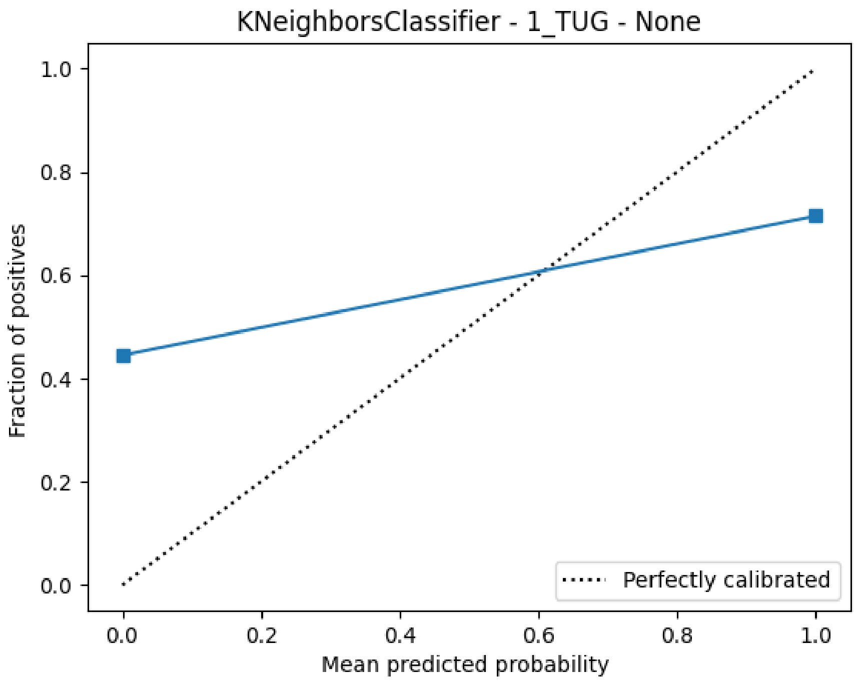

KNeighborsClassifier - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

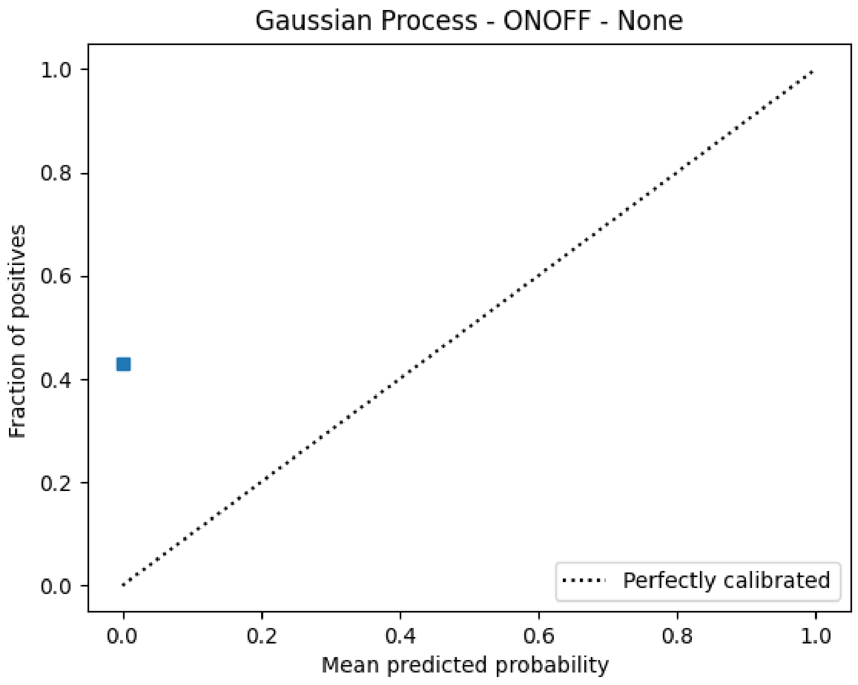

KNeighborsClassifier - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

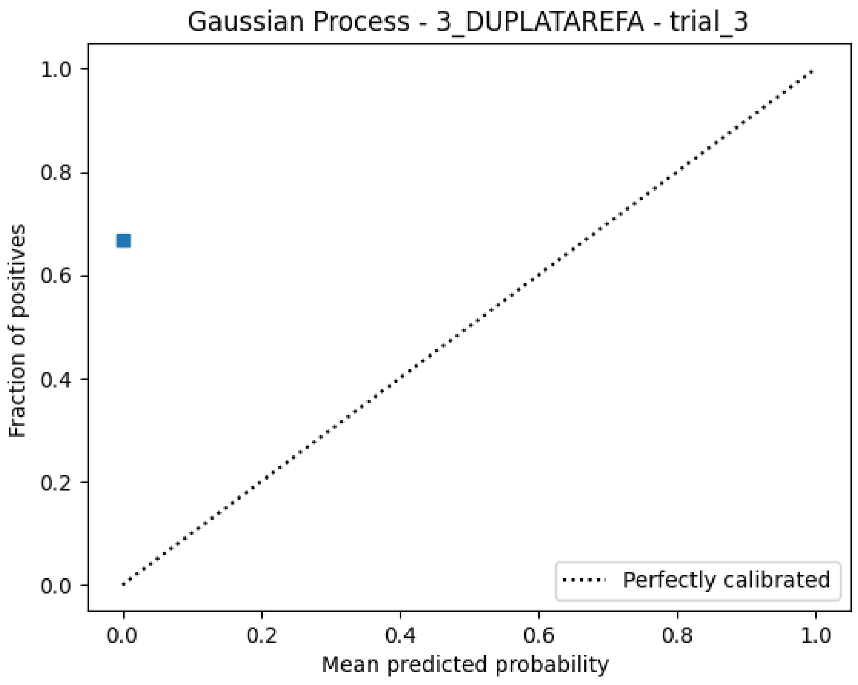

KNeighborsClassifier - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

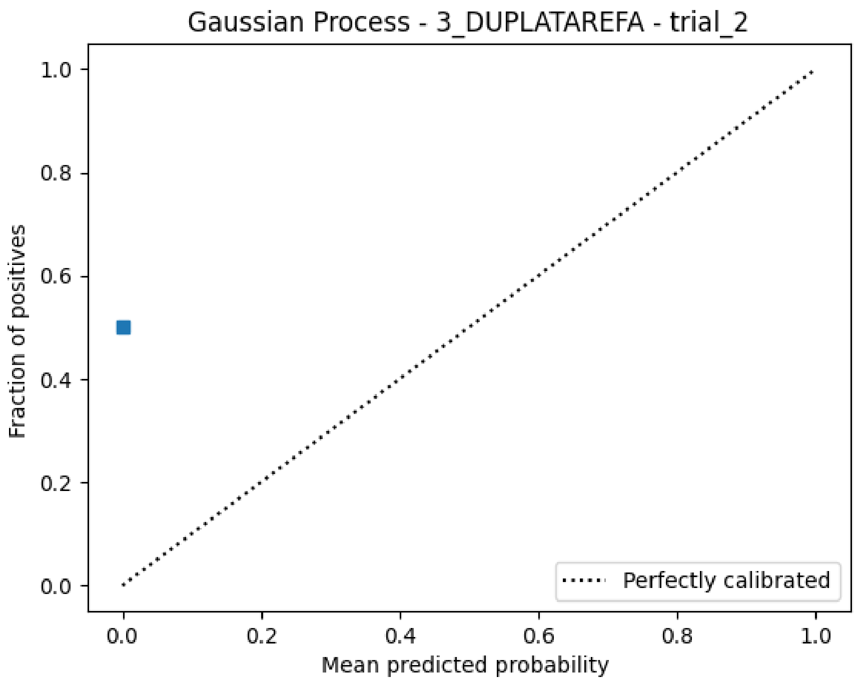

KNeighborsClassifier - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

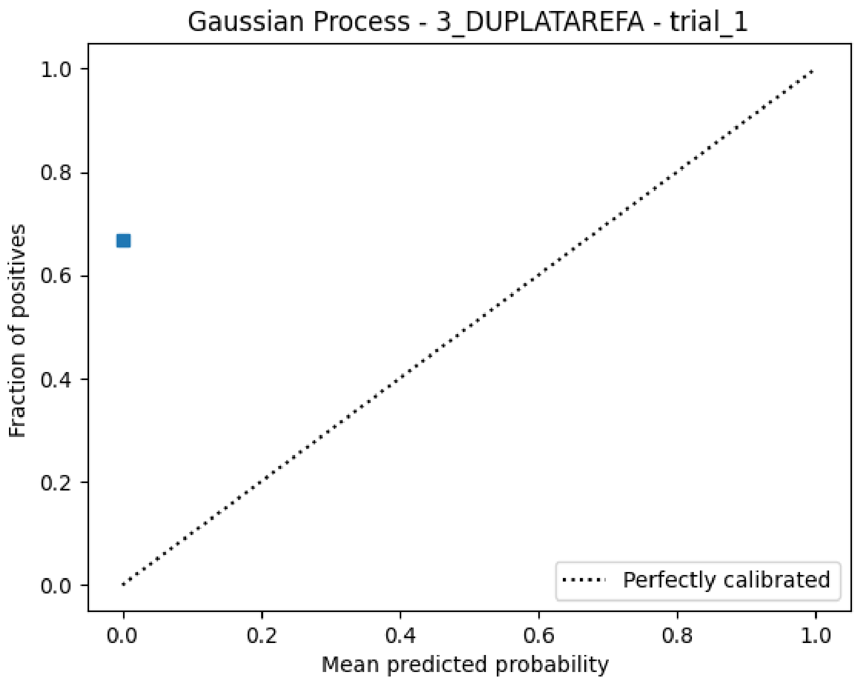

KNeighborsClassifier - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

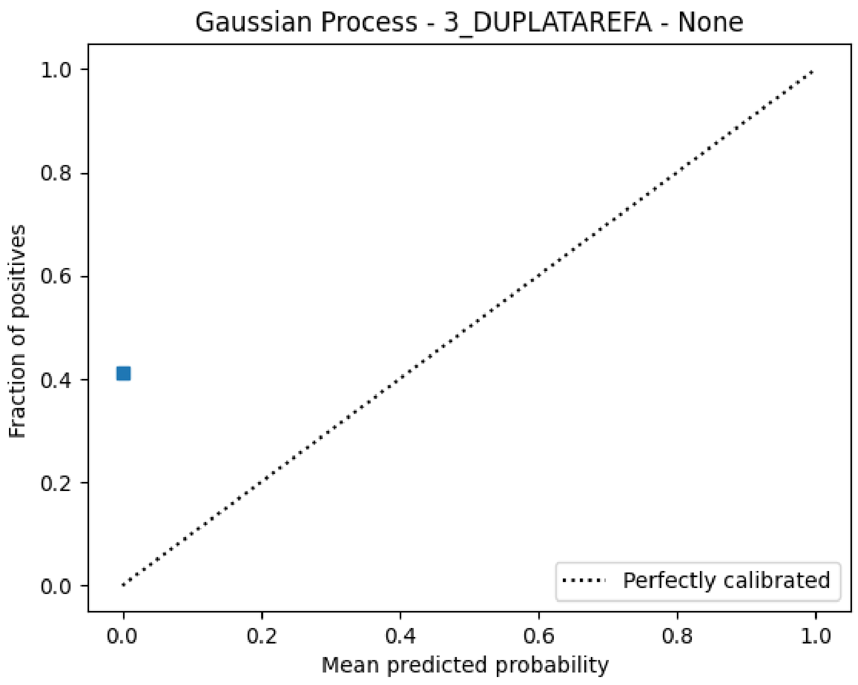

KNeighborsClassifier - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

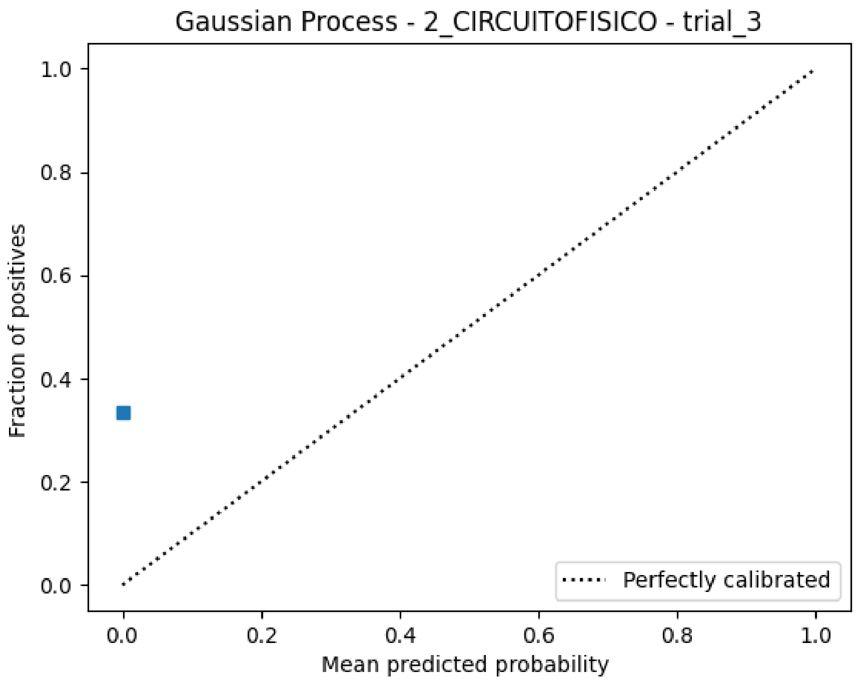

Gaussian Process - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

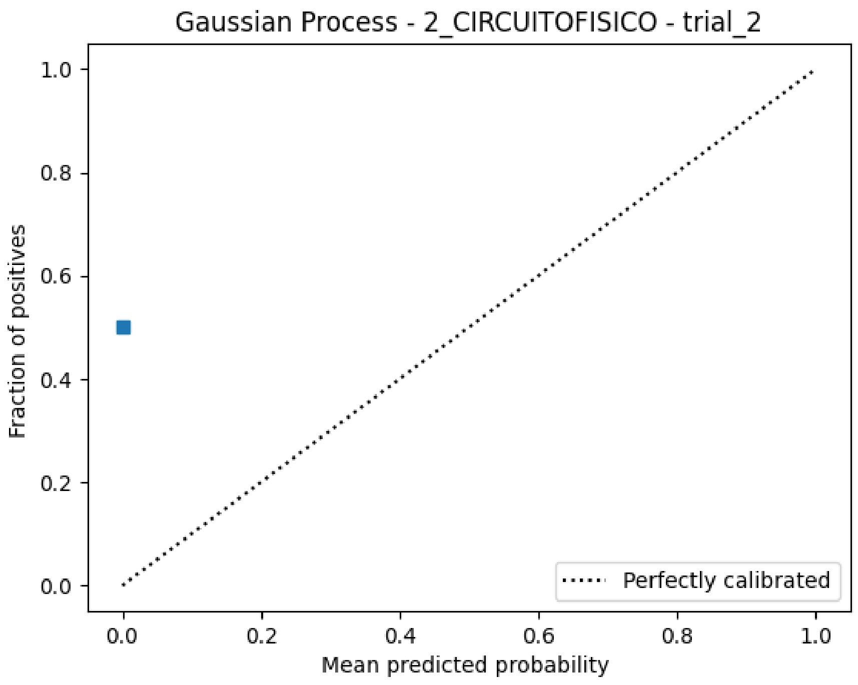

Gaussian Process - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

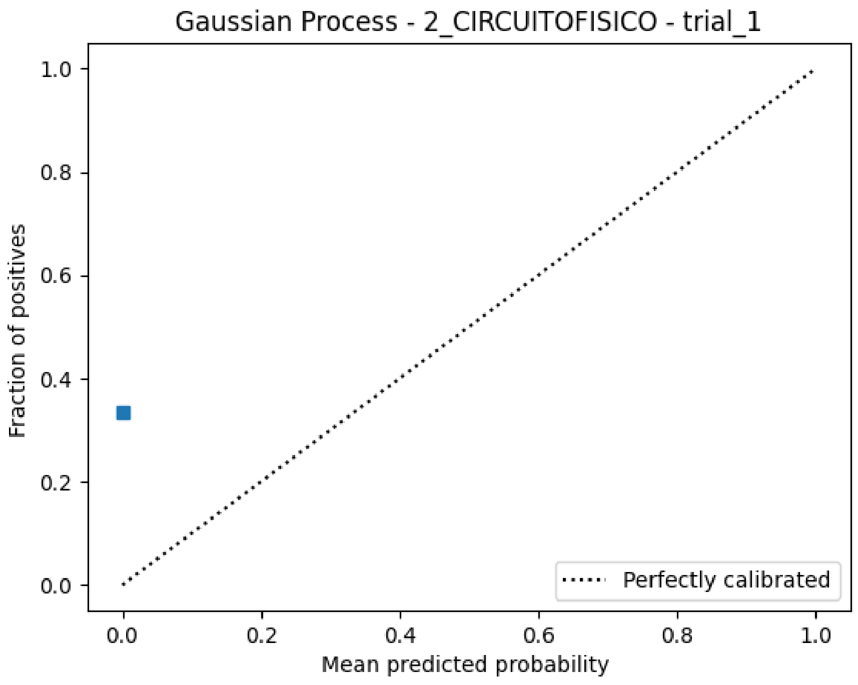

Gaussian Process - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

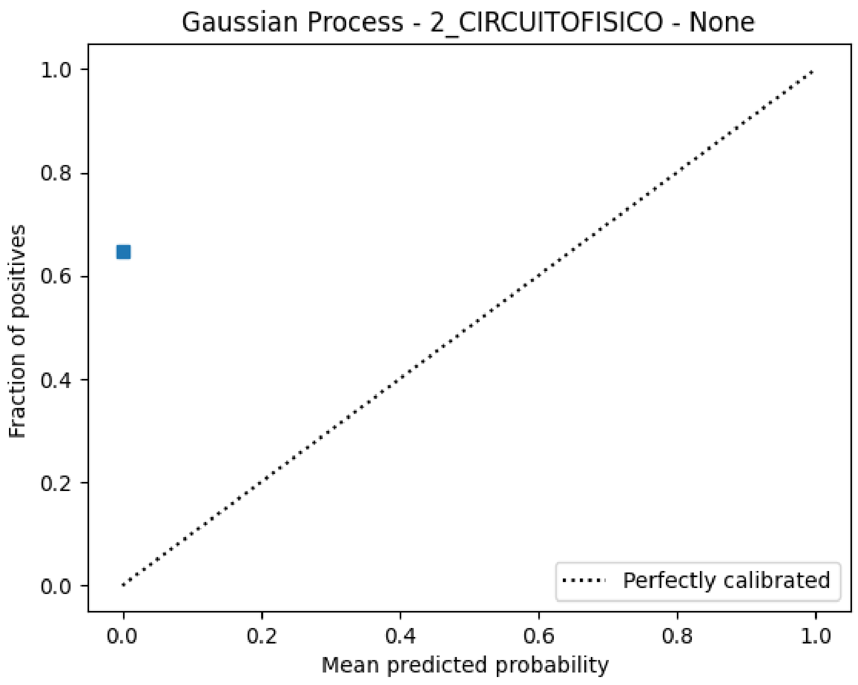

Gaussian Process - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

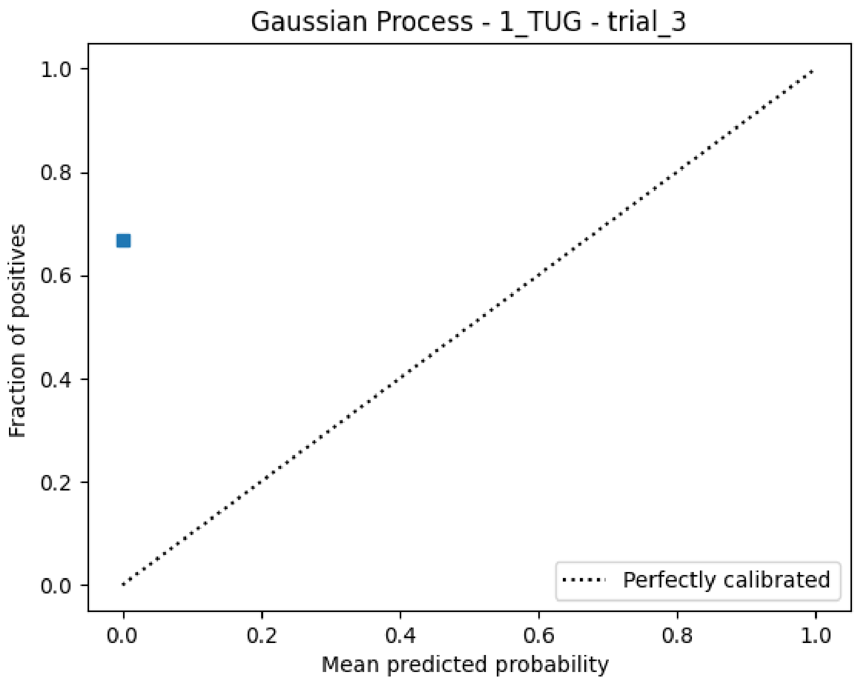

Gaussian Process - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

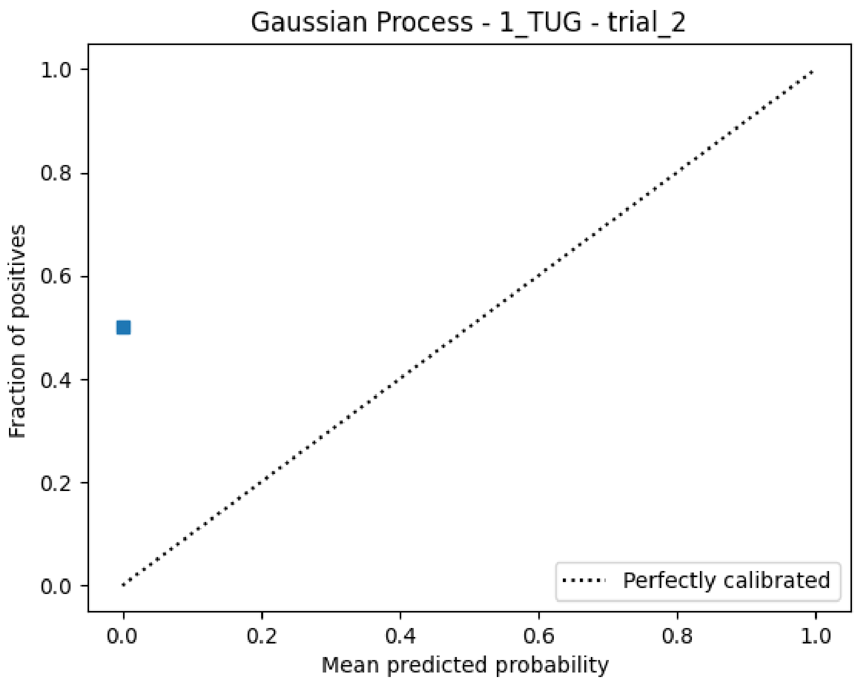

Gaussian Process - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

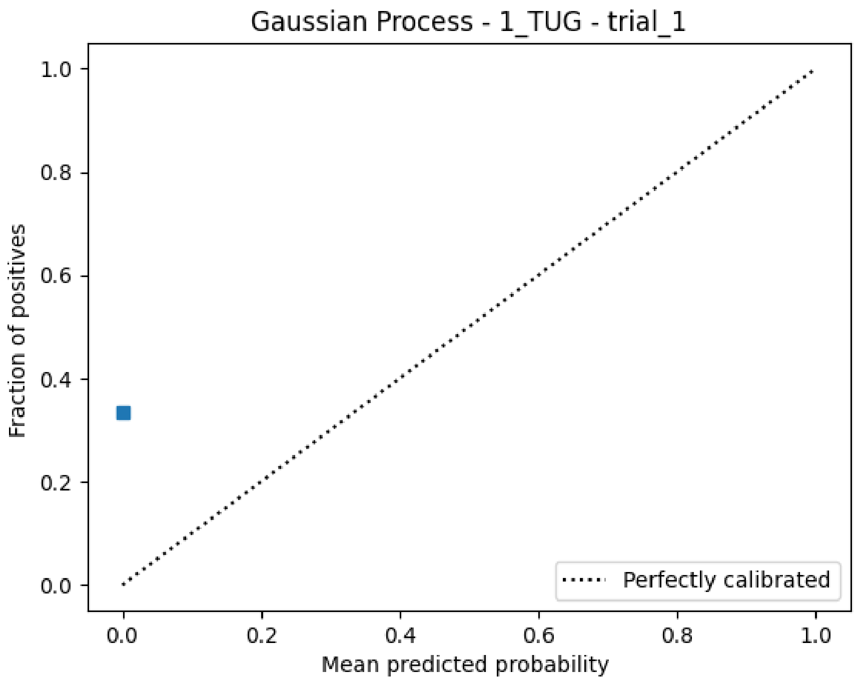

Gaussian Process - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

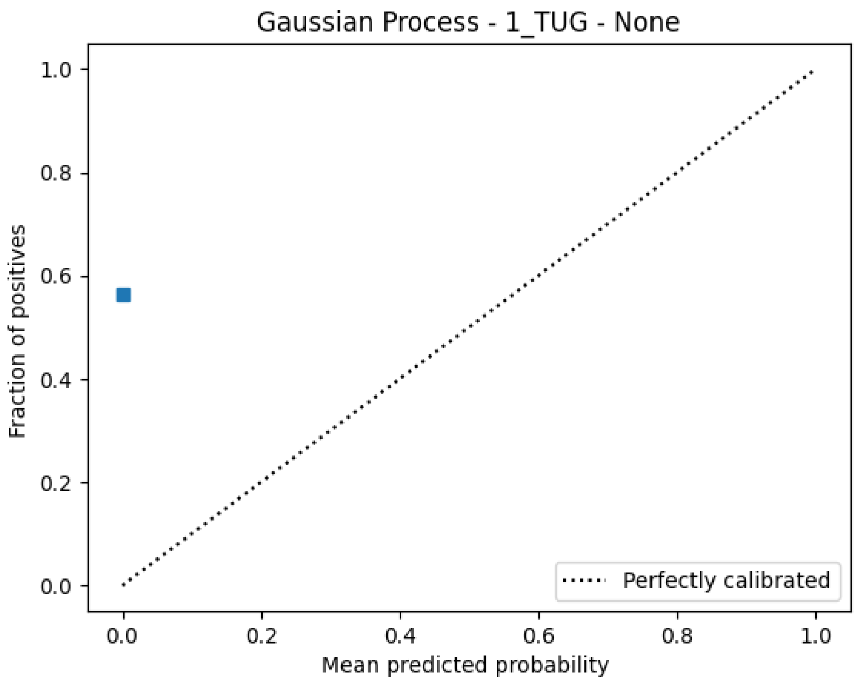

Gaussian Process - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

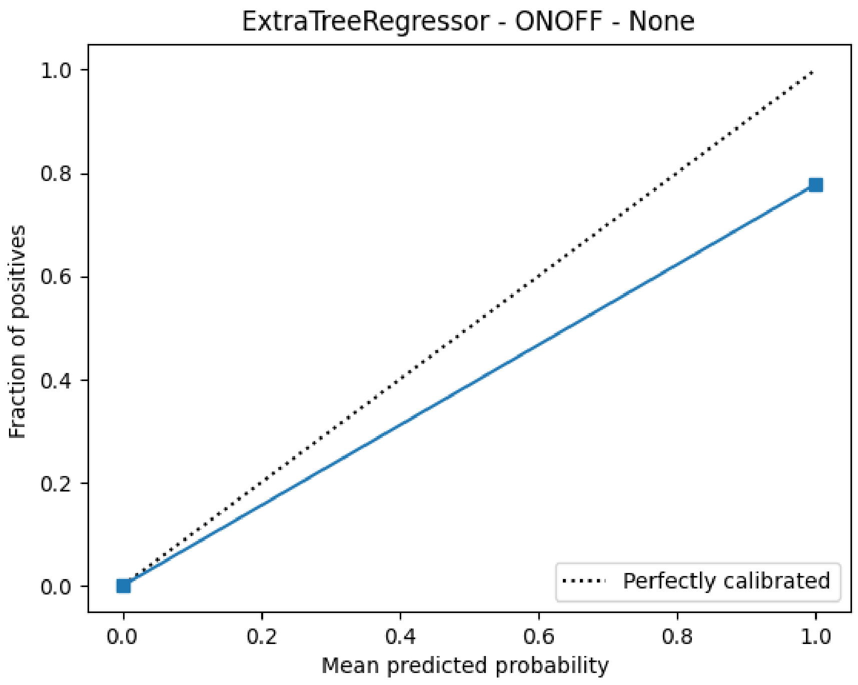

Gaussian Process - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

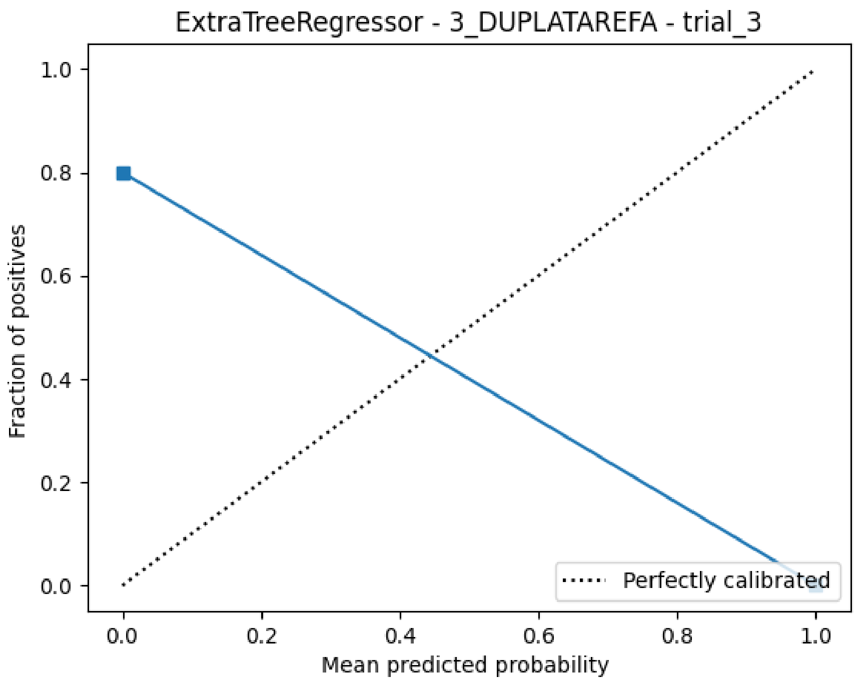

Gaussian Process - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

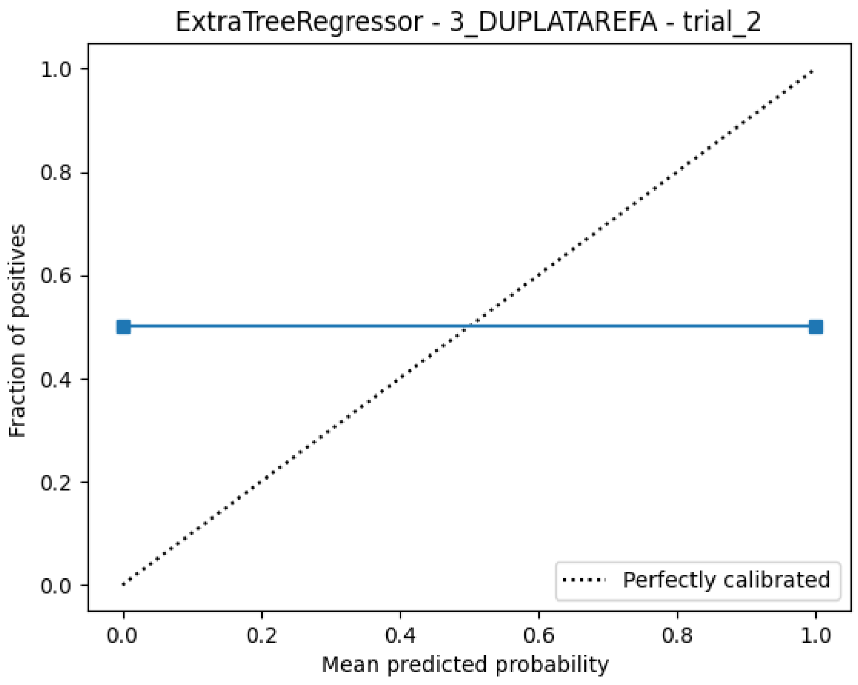

Gaussian Process - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

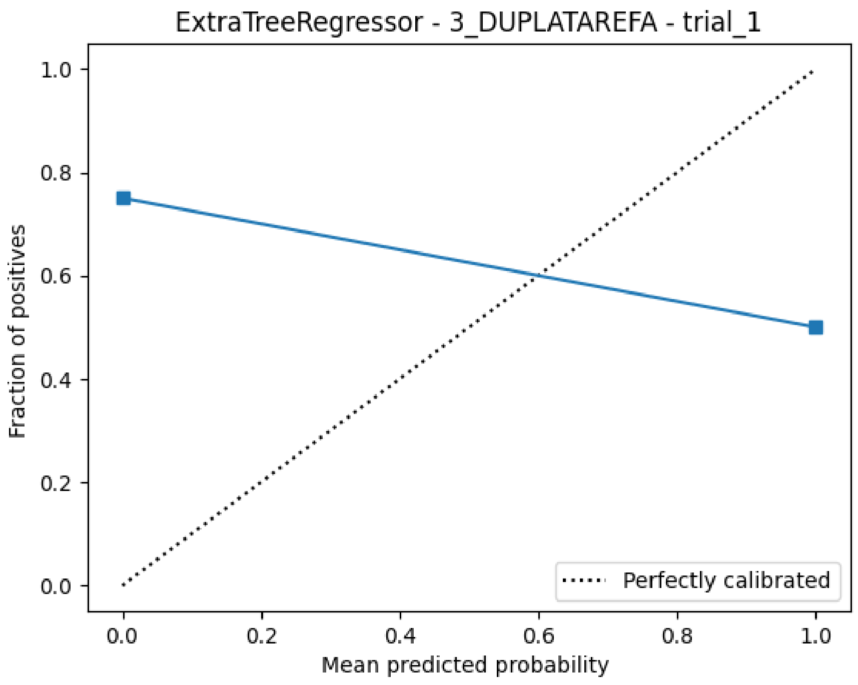

Gaussian Process - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

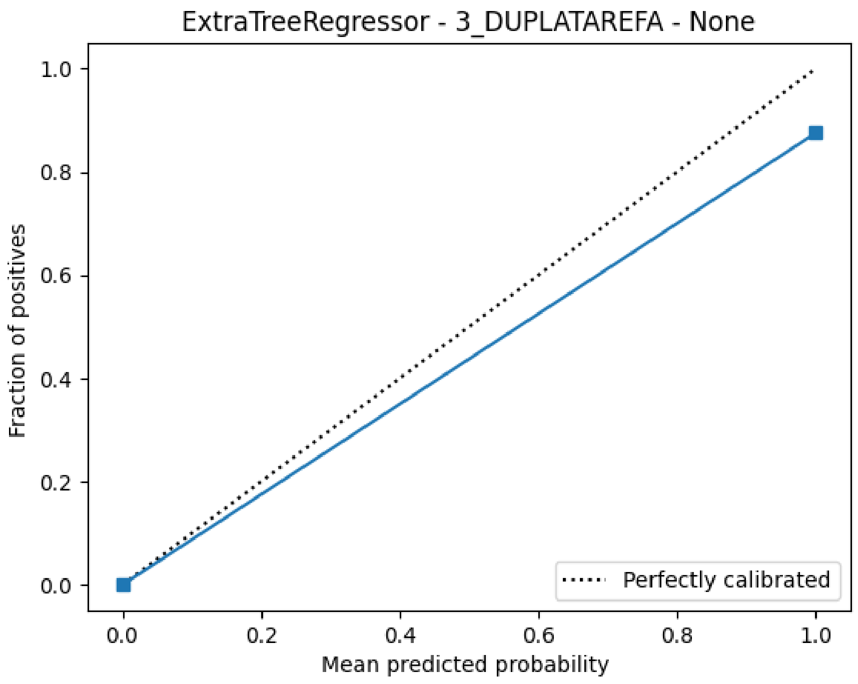

Gaussian Process - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

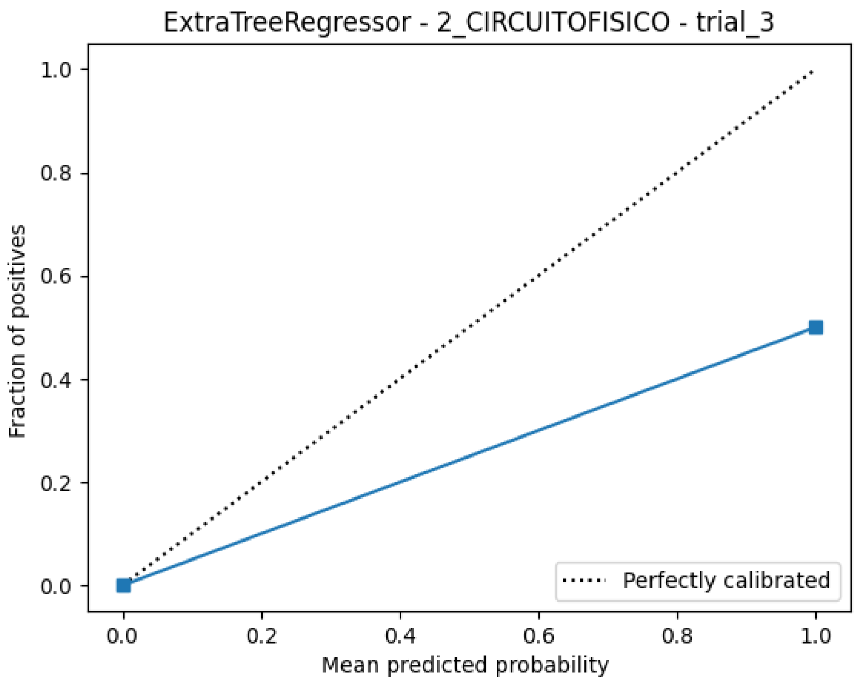

ExtraTreeRegressor - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

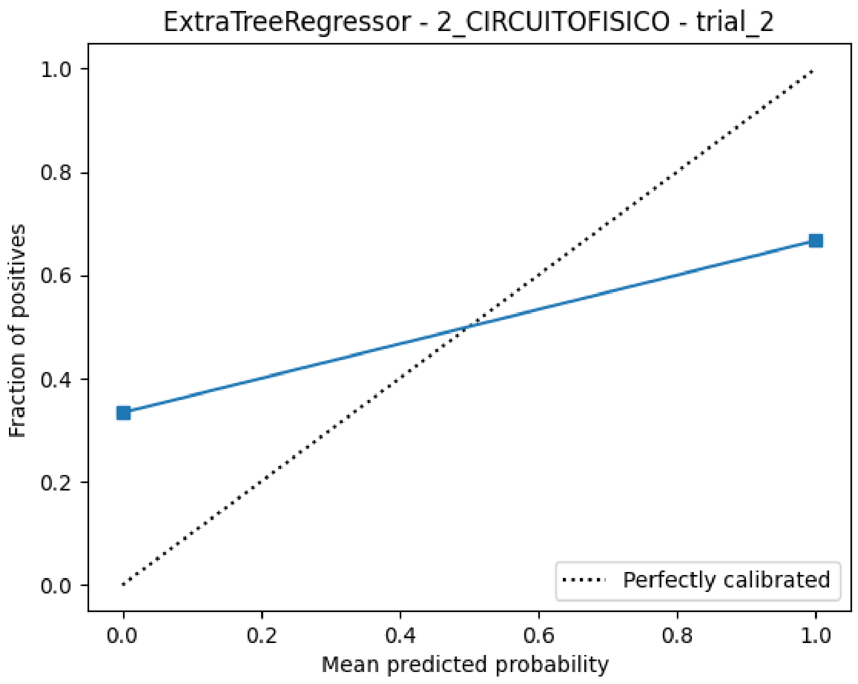

ExtraTreeRegressor - 3_DUPLATAREFA - trial_3
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
Mean predicted probability
Perfectly calibrated

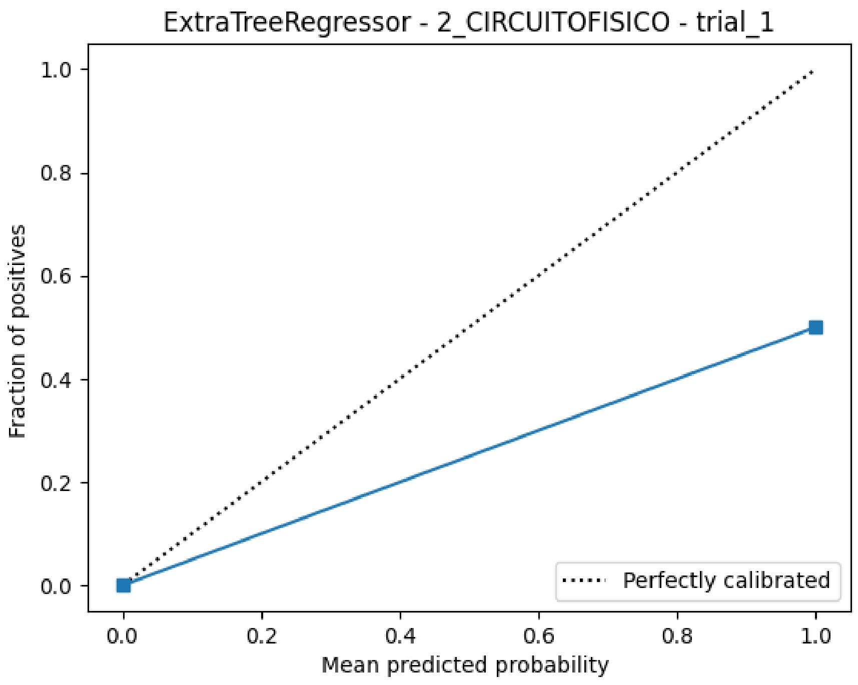

ExtraTreeRegressor - 3_DUPLATAREFA - trial_2
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
Mean predicted probability
Perfectly calibrated

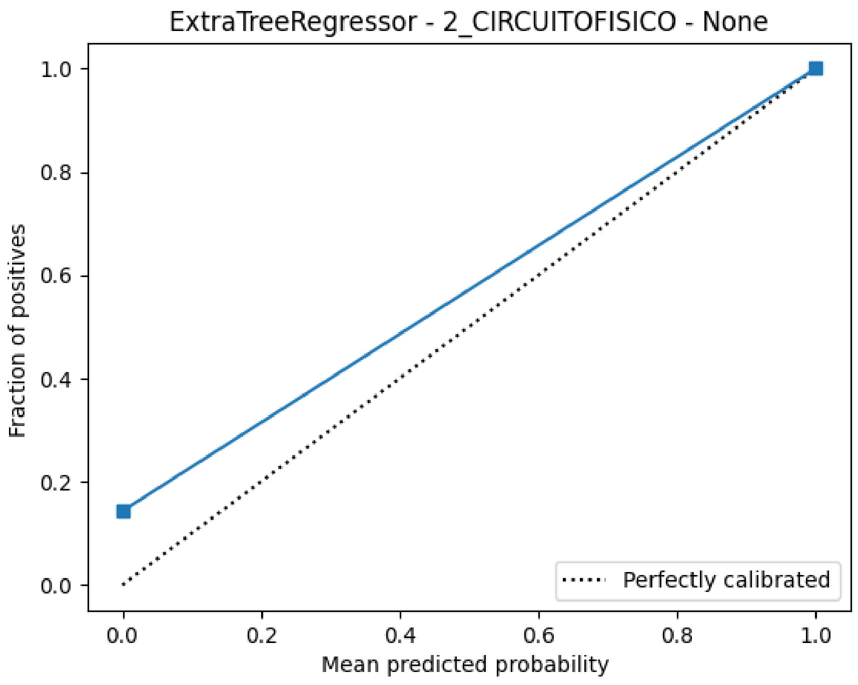

ExtraTreeRegressor - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

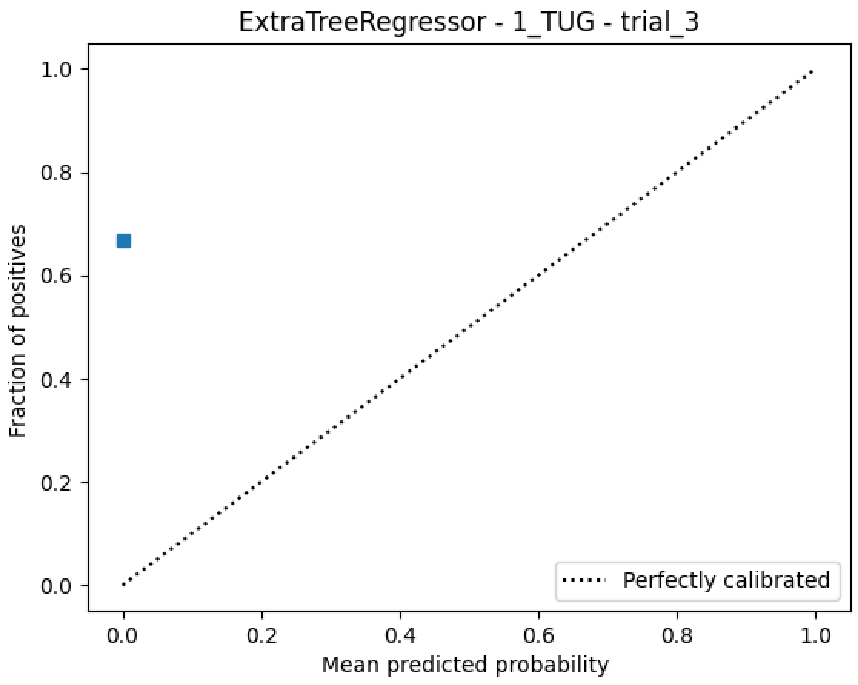

ExtraTreeRegressor - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

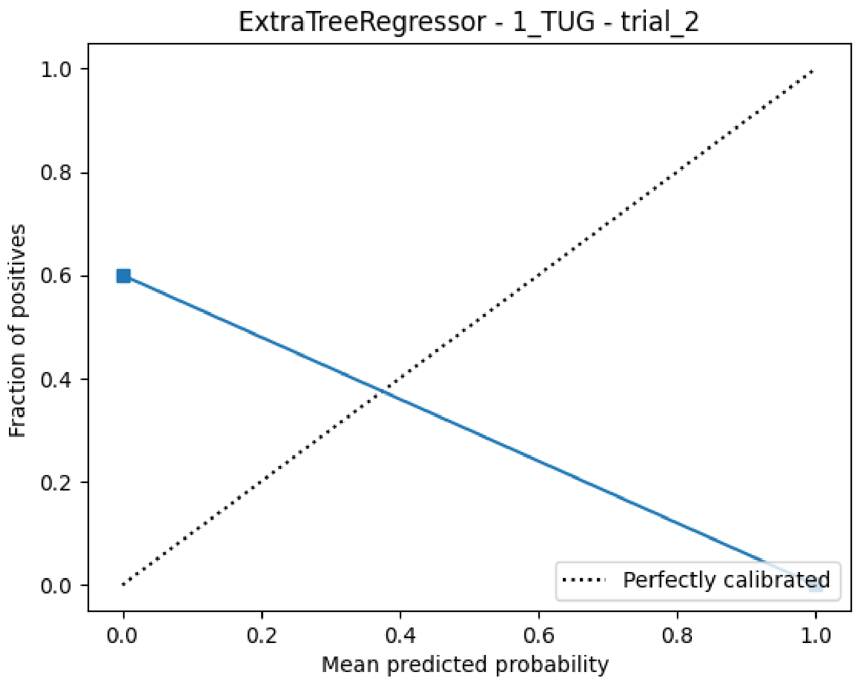

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_3
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability
Perfectly calibrated

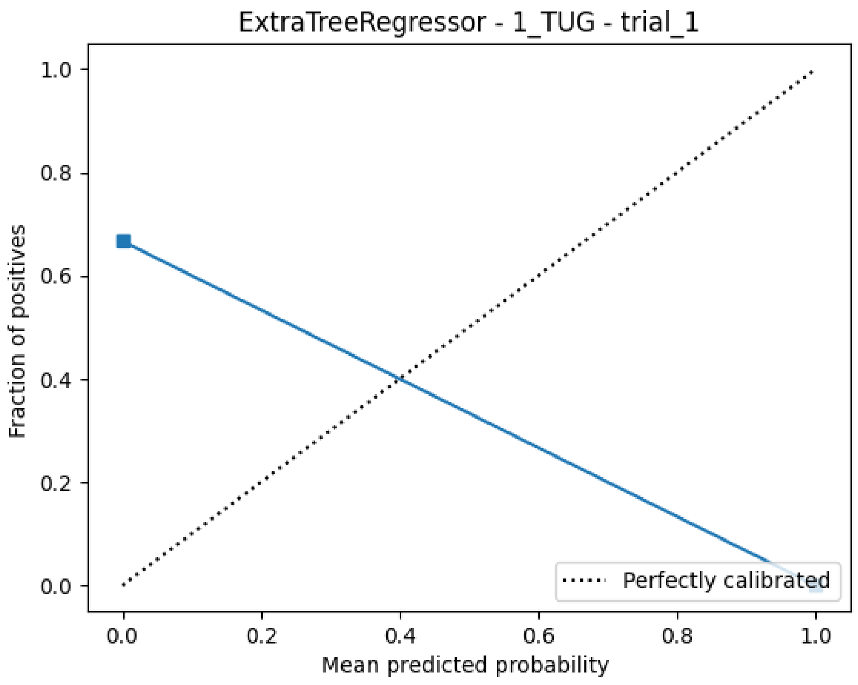

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_2
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability
Perfectly calibrated

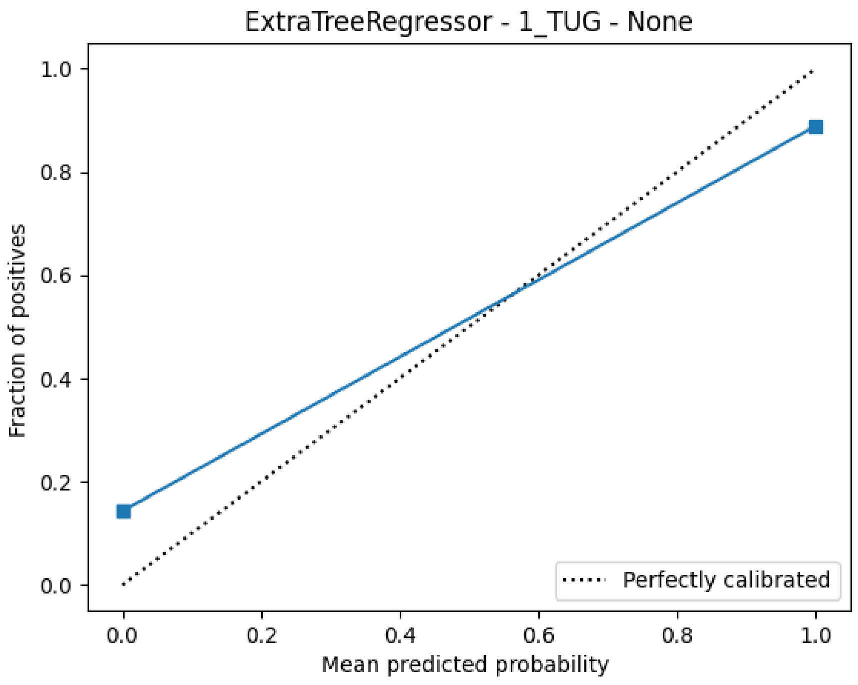

ExtraTreeRegressor - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

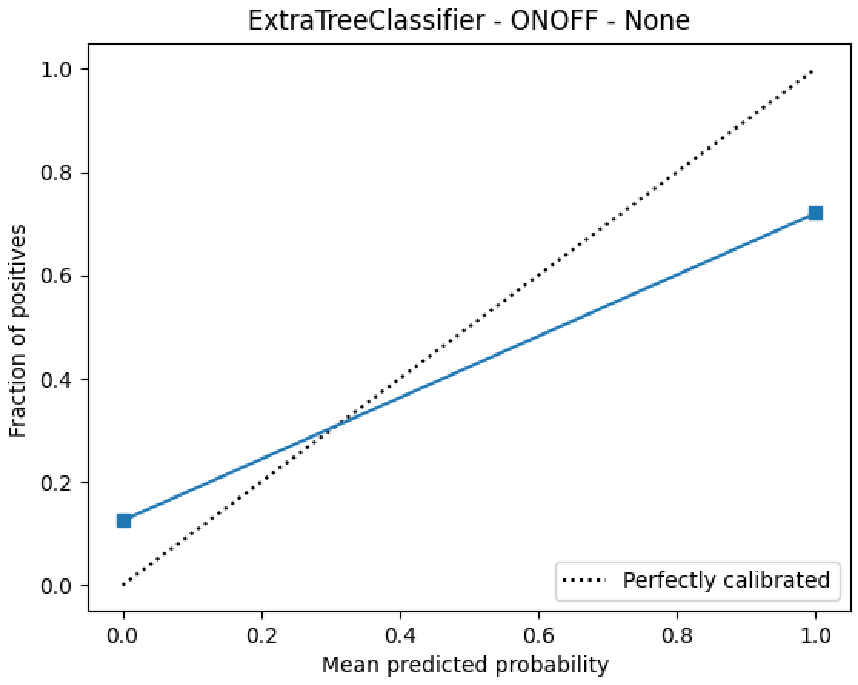

ExtraTreeRegressor - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

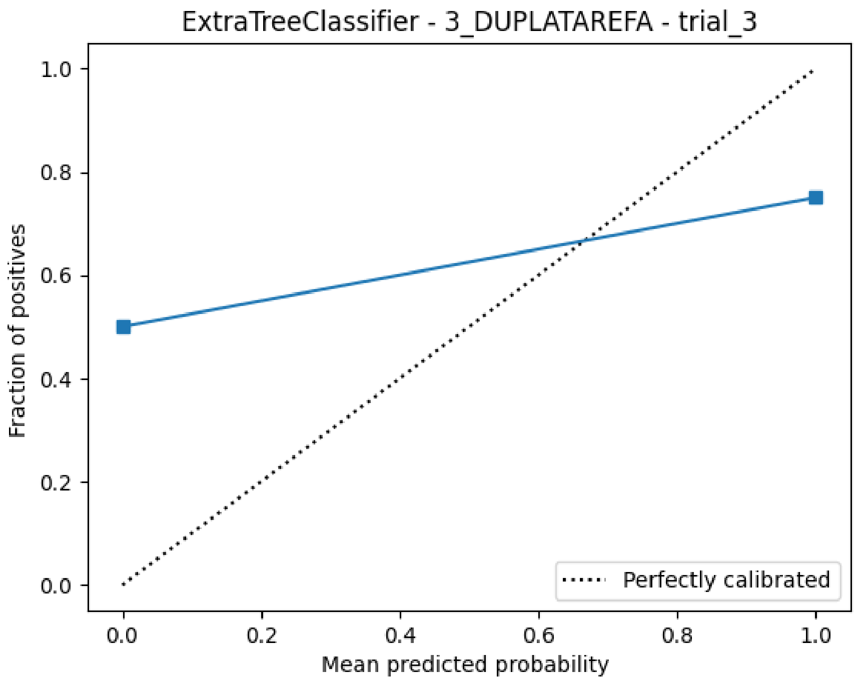

ExtraTreeRegressor - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

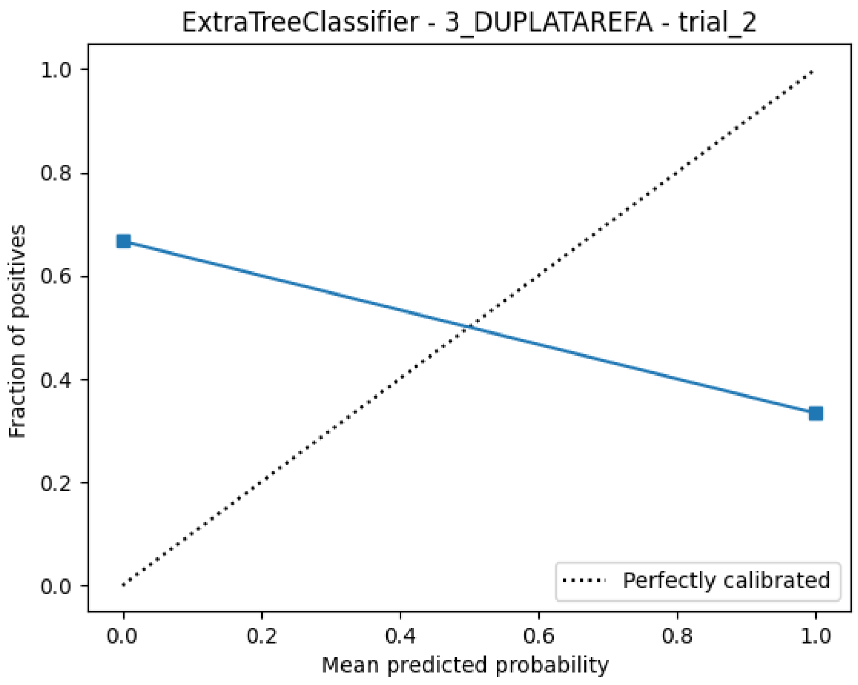

ExtraTreeRegressor - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

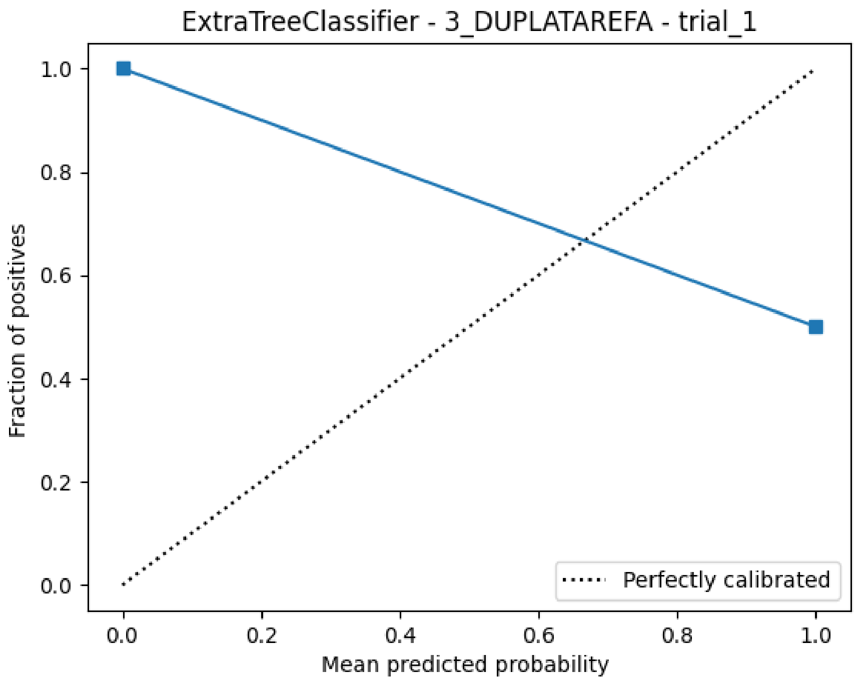

ExtraTreeRegressor - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

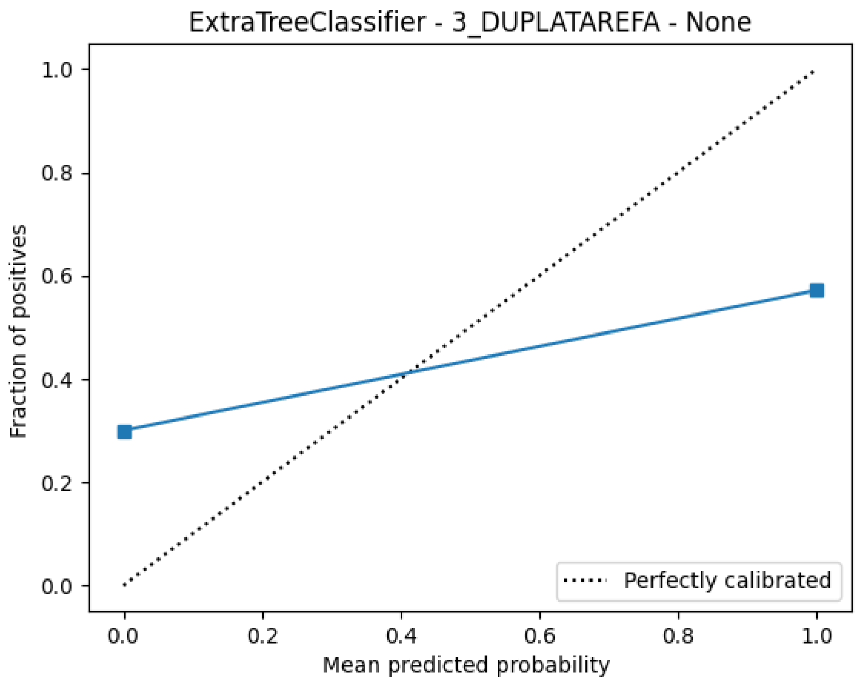

ExtraTreeRegressor - 1_TUG - None
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

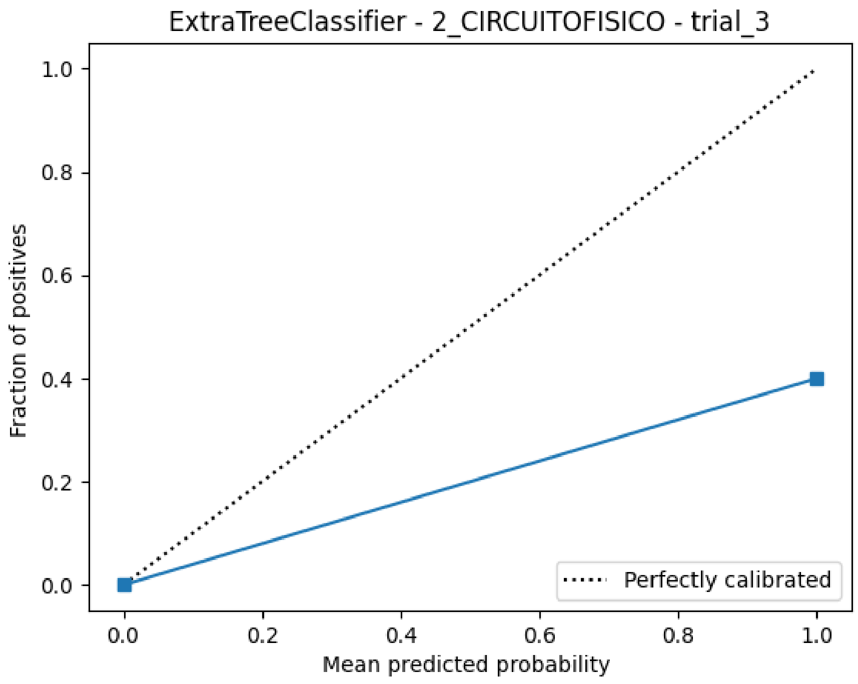

ExtraTreeClassifier - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

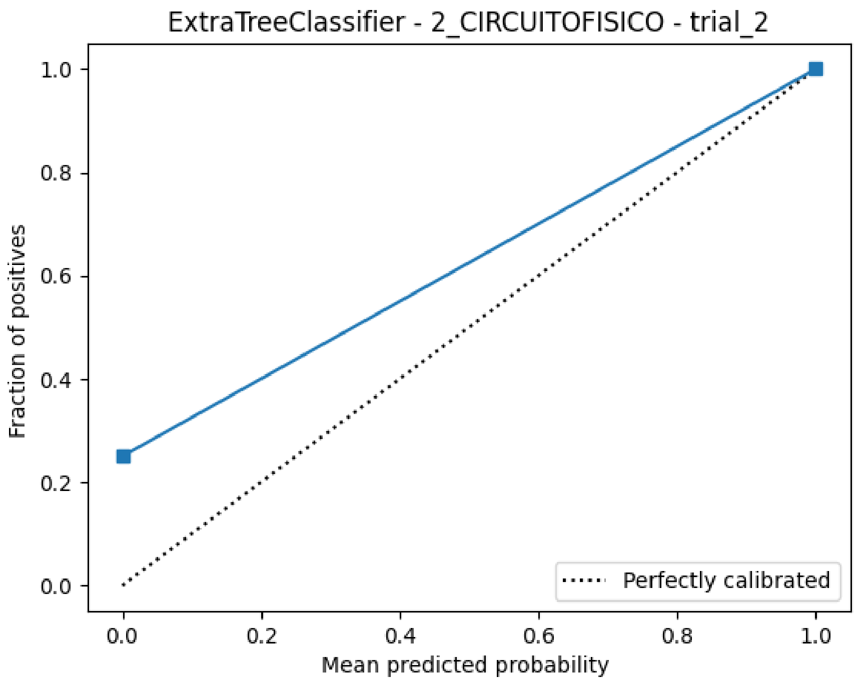

ExtraTreeClassifier - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

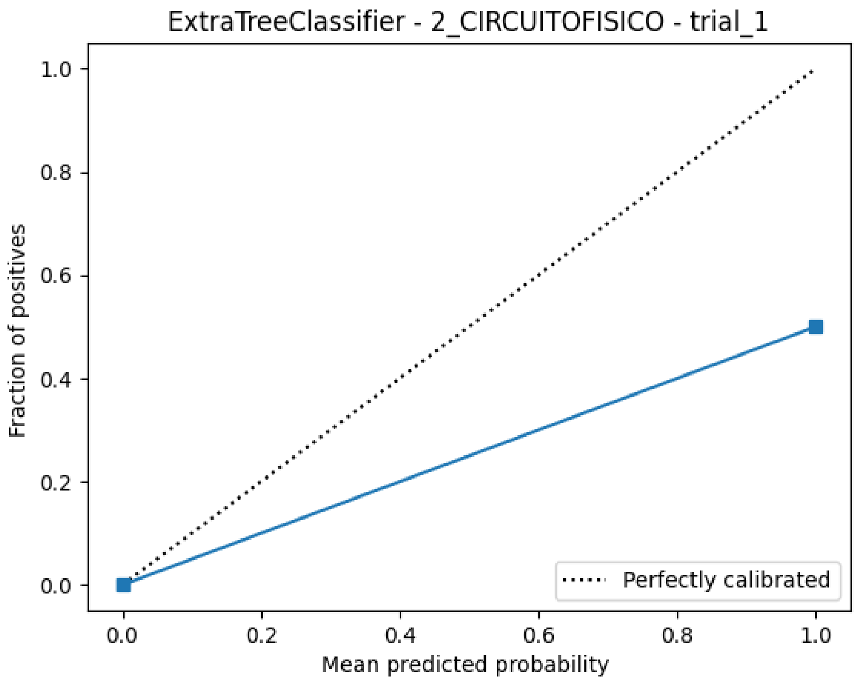

ExtraTreeClassifier - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

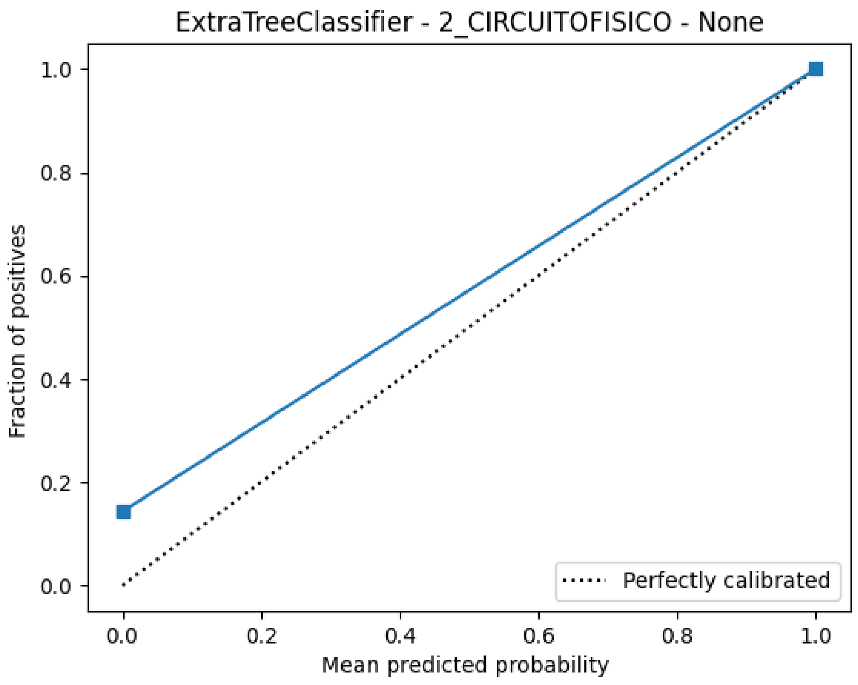

ExtraTreeClassifier - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

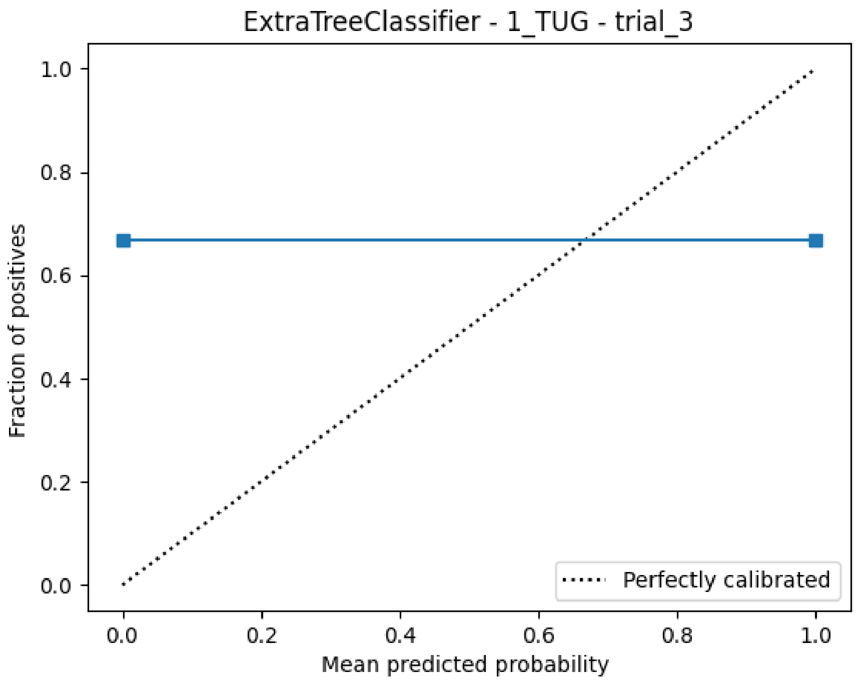

ExtraTreeClassifier - 3_DUPLATAREFA - None
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

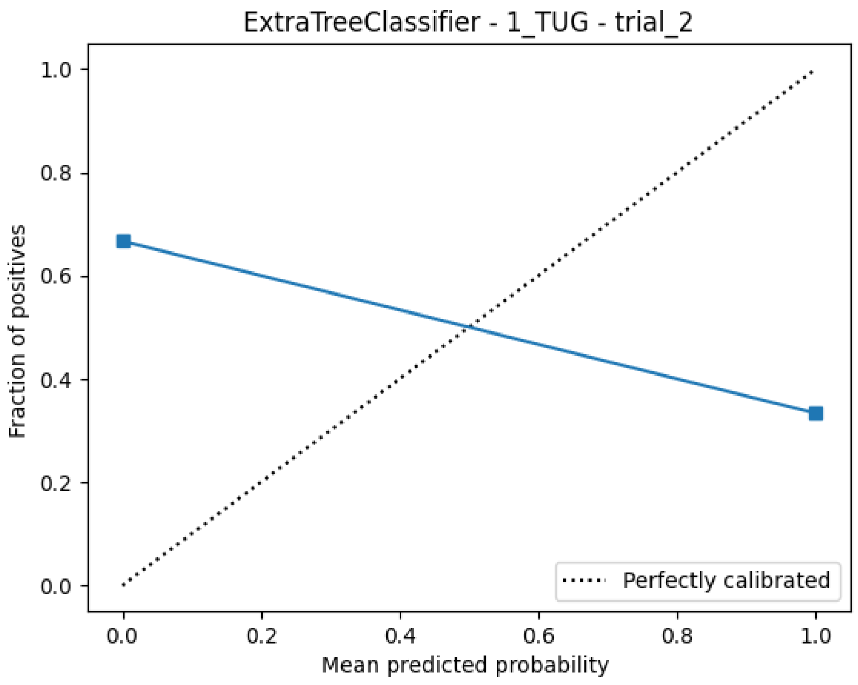

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_3
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

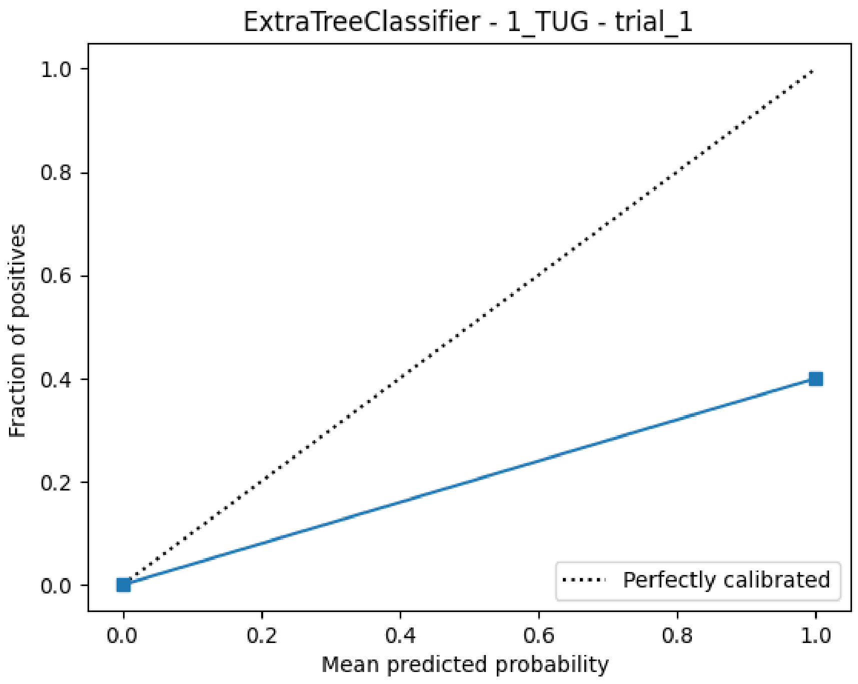

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

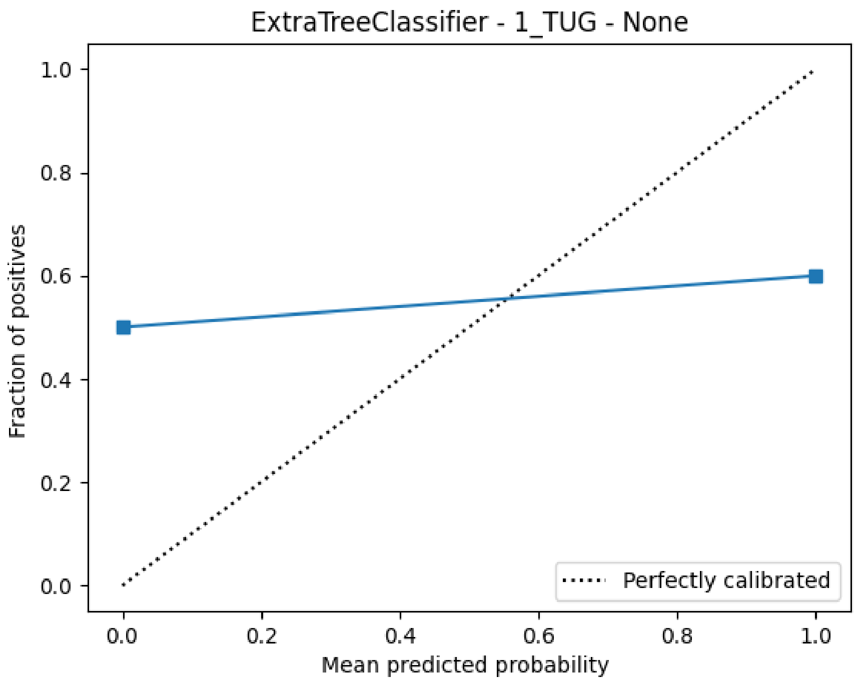

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

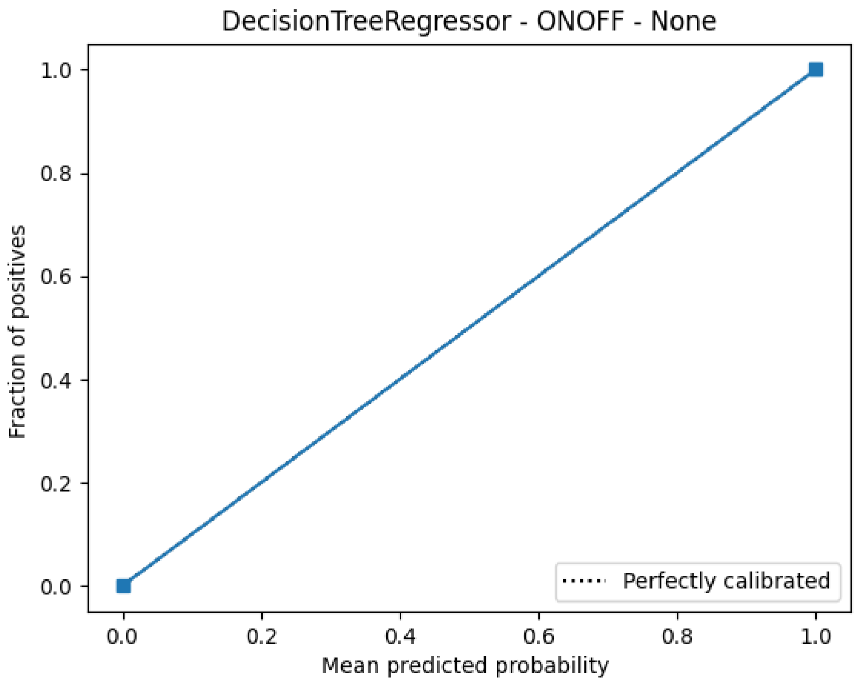

ExtraTreeClassifier - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

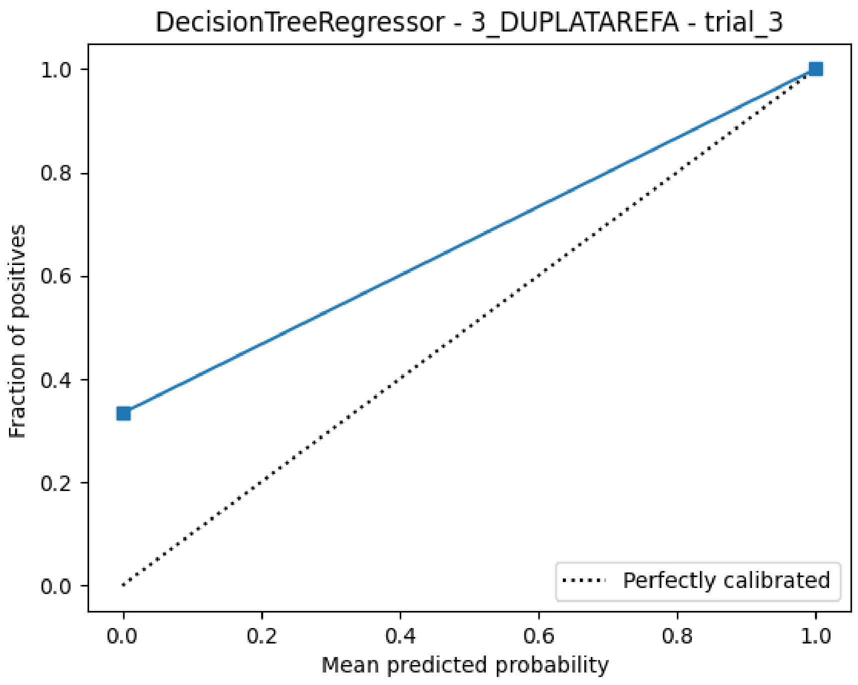

ExtraTreeClassifier - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

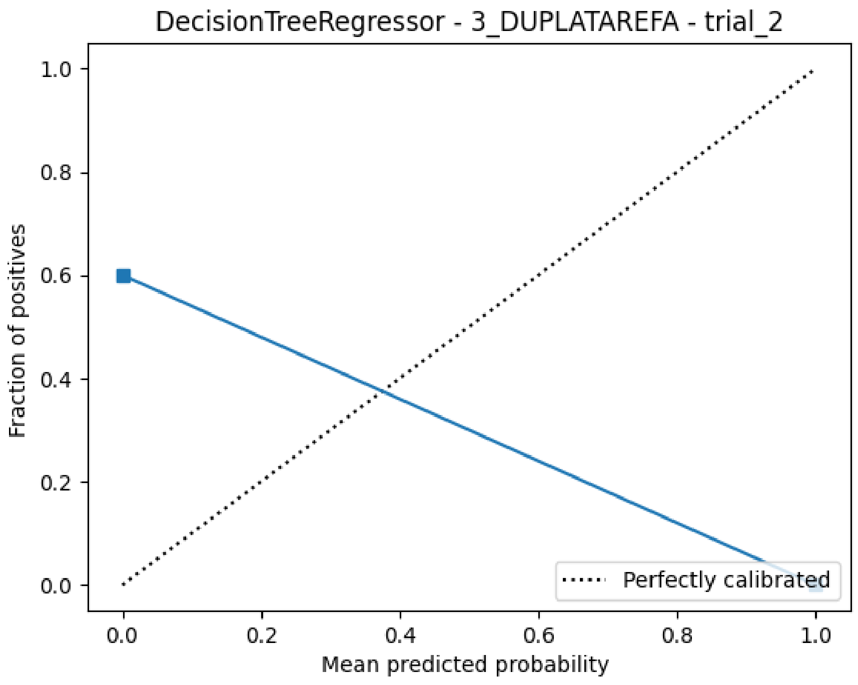

ExtraTreeClassifier - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

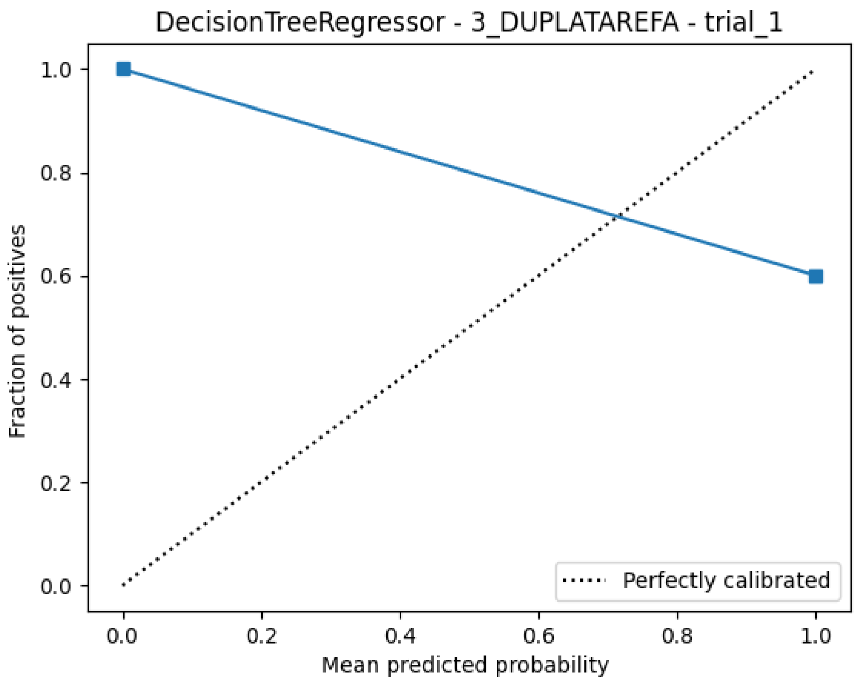

ExtraTreeClassifier - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

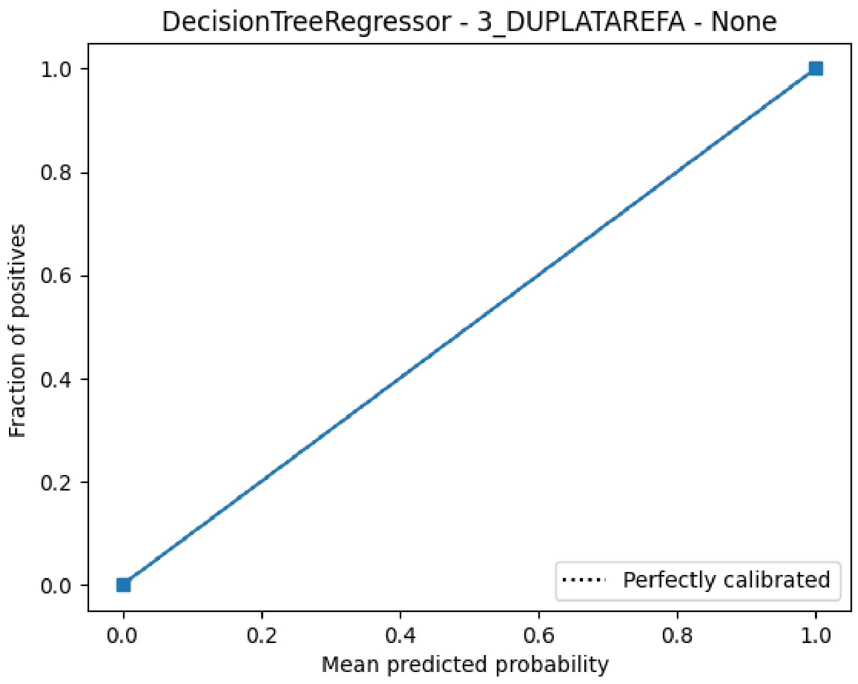

ExtraTreeClassifier - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

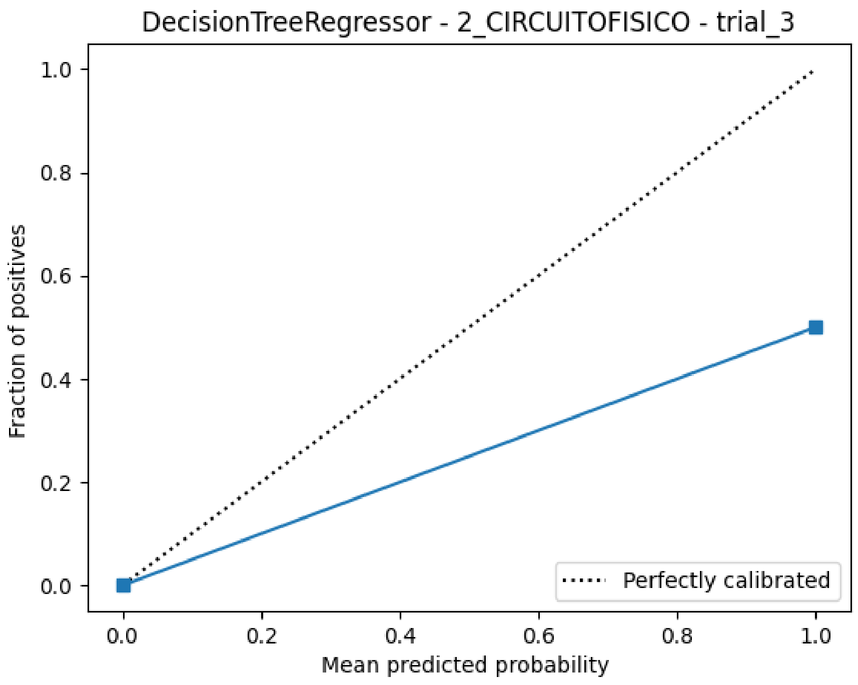

DecisionTreeRegressor - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

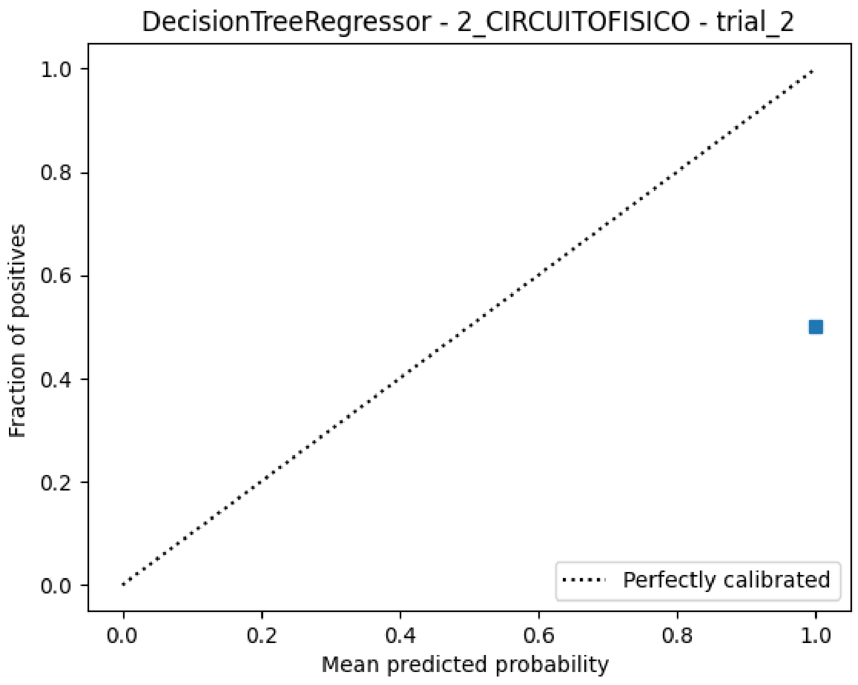

DecisionTreeRegressor - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

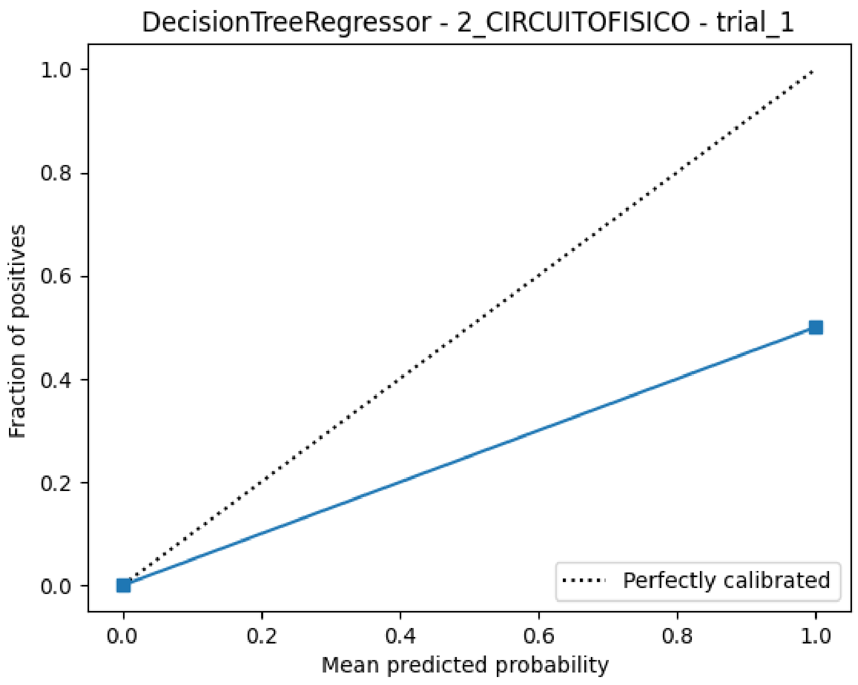

DecisionTreeRegressor - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

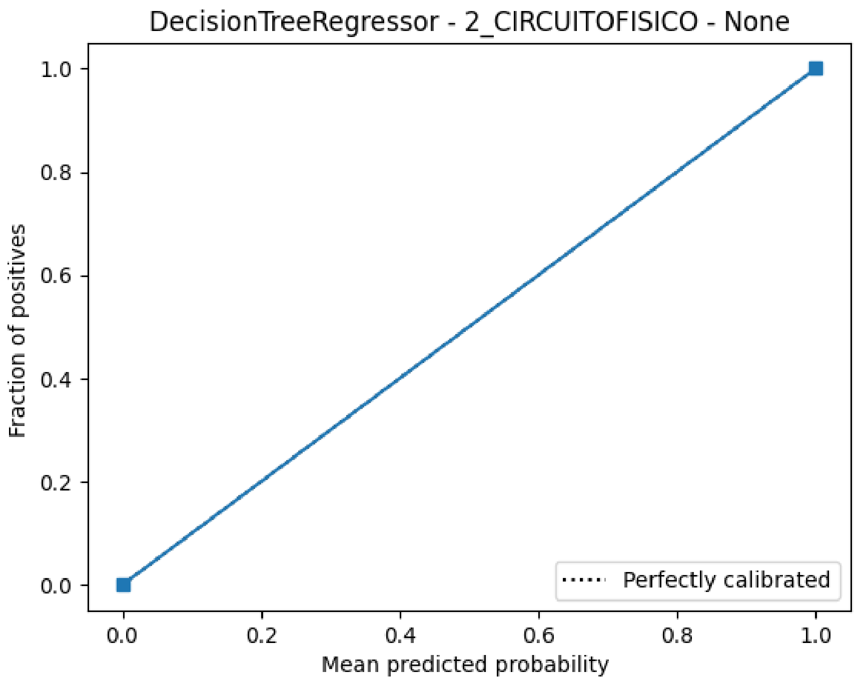

DecisionTreeRegressor - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

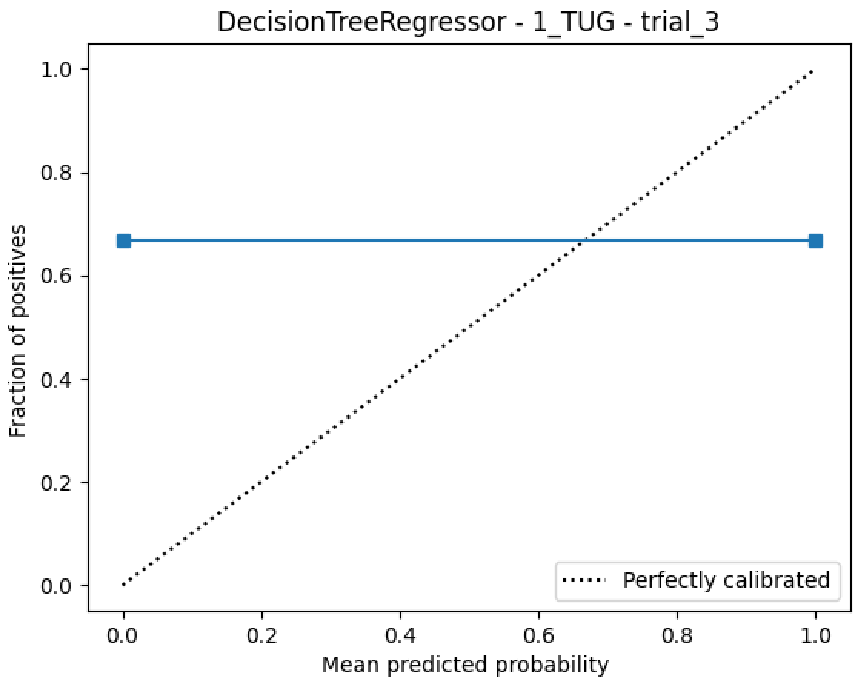

DecisionTreeRegressor - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

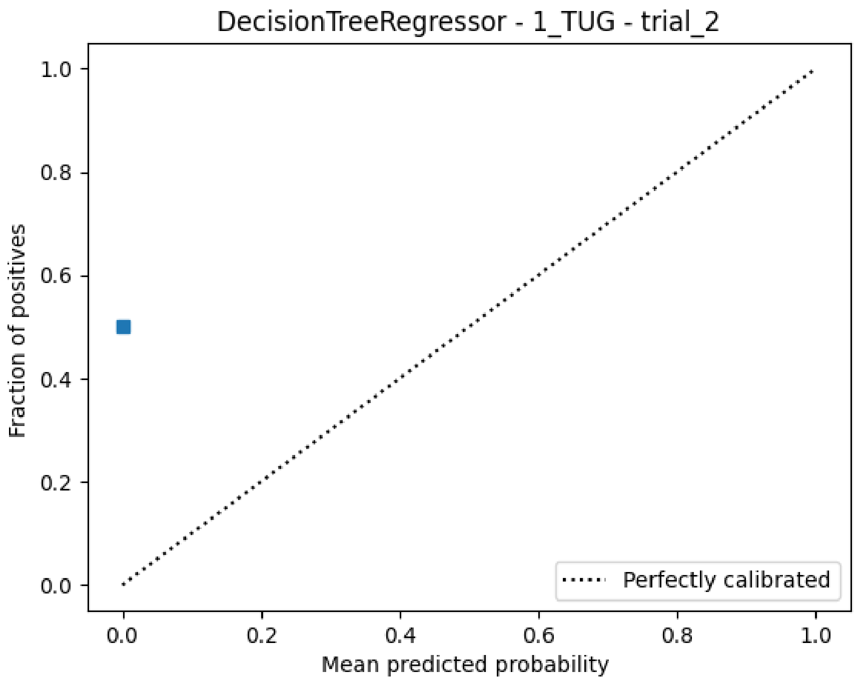

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

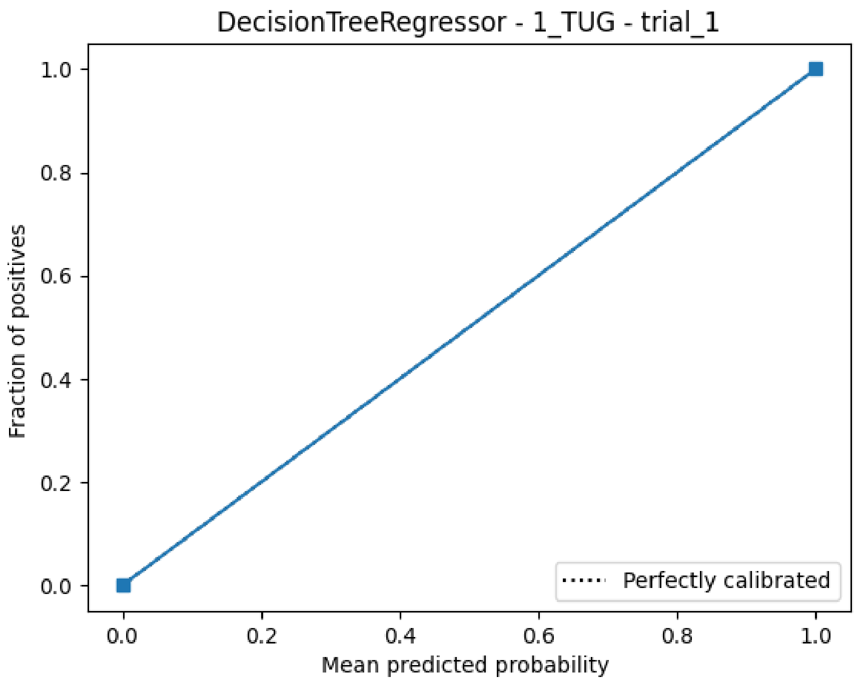

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

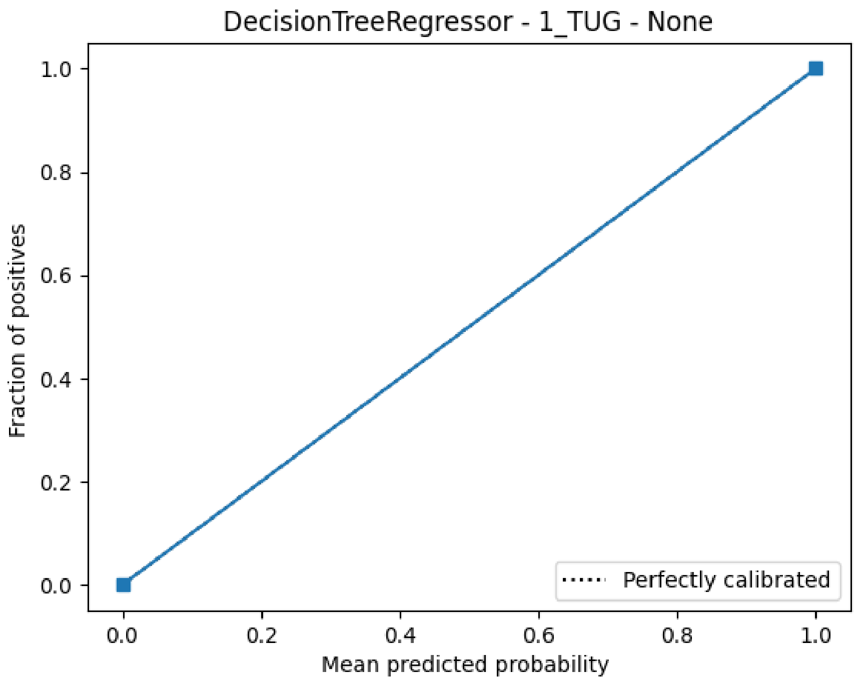

DecisionTreeRegressor - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

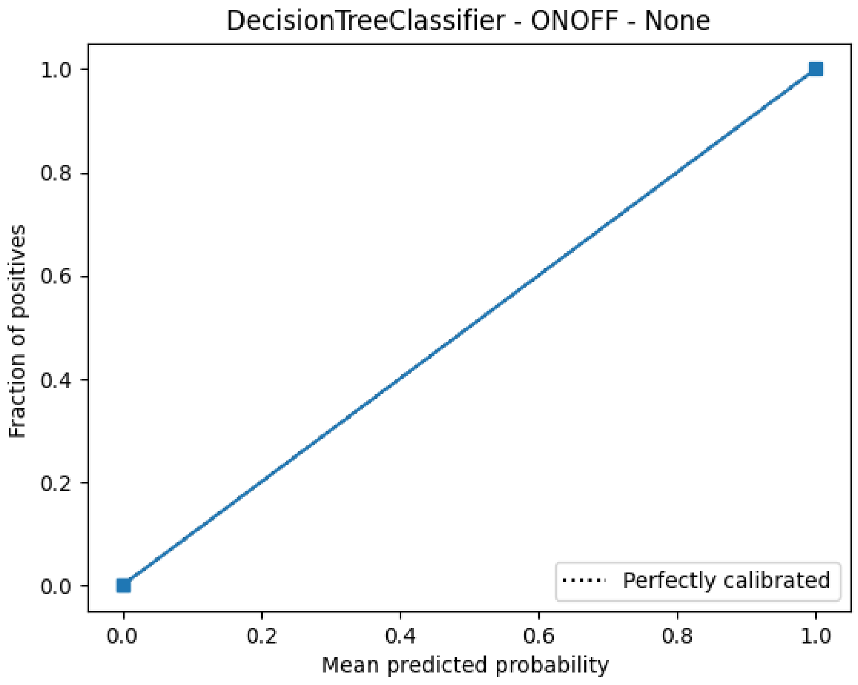

DecisionTreeRegressor - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

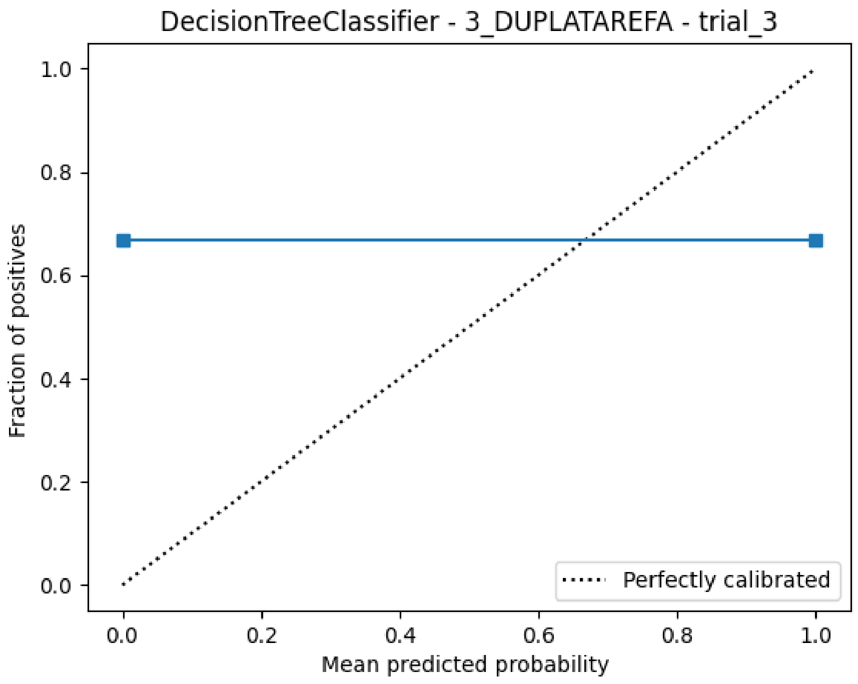

DecisionTreeRegressor - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

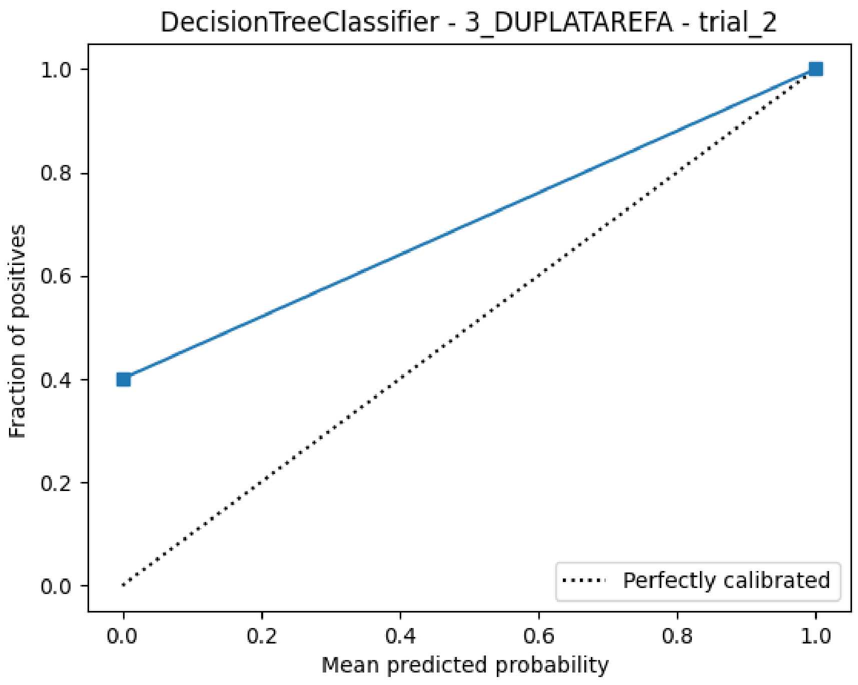

DecisionTreeRegressor - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

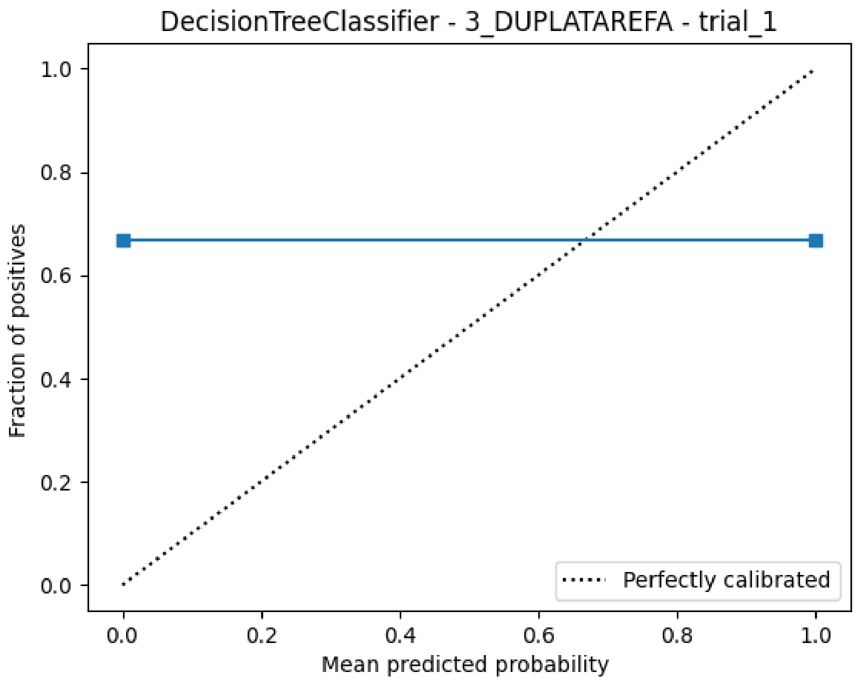

DecisionTreeRegressor - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

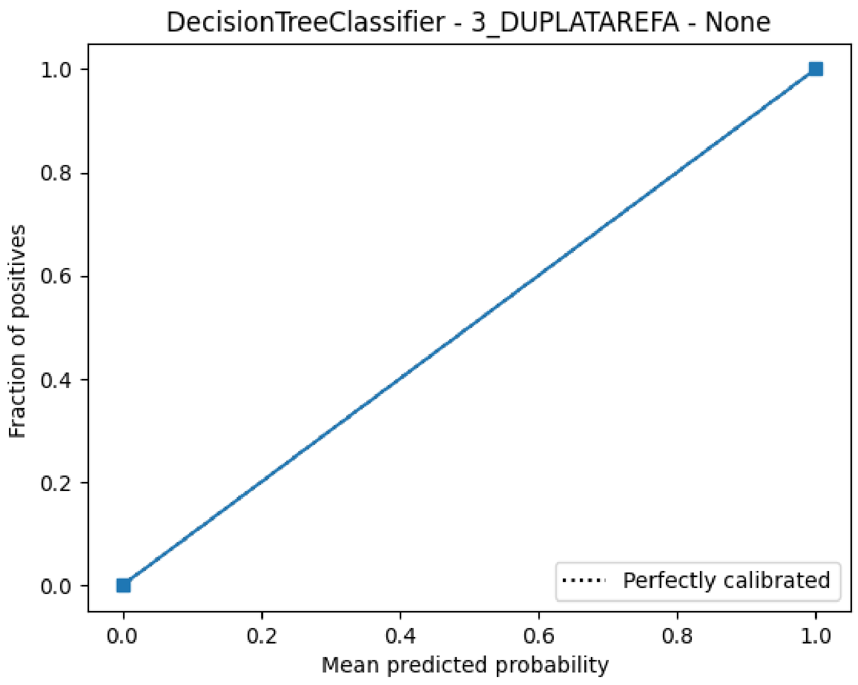

DecisionTreeRegressor - 1_TUG - None
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated

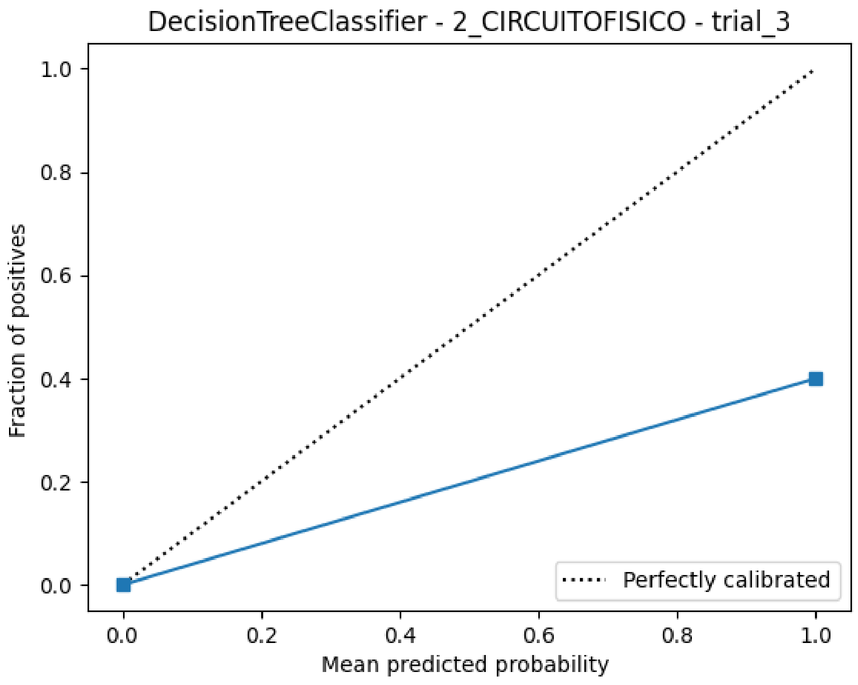

DecisionTreeClassifier - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

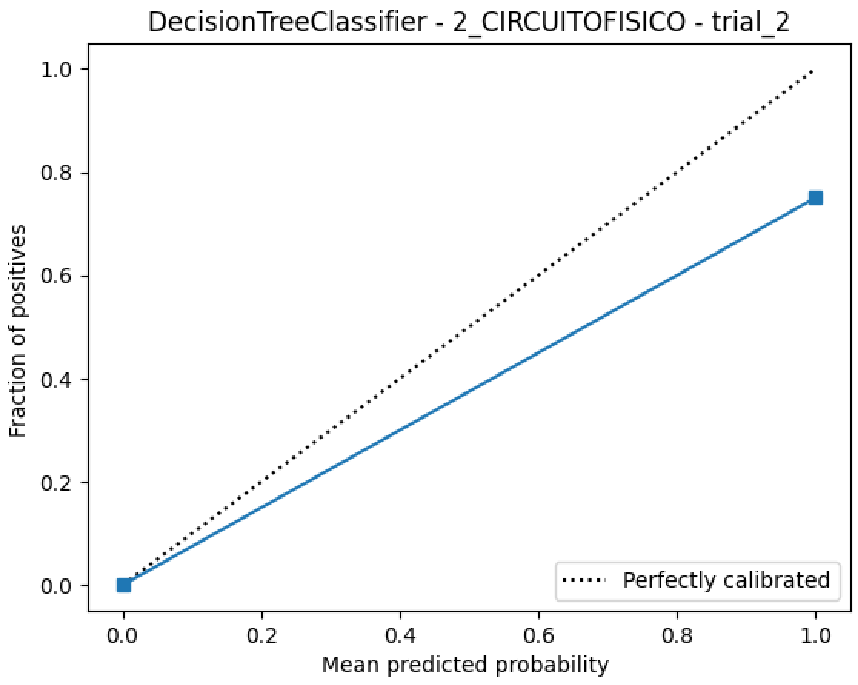

DecisionTreeClassifier - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

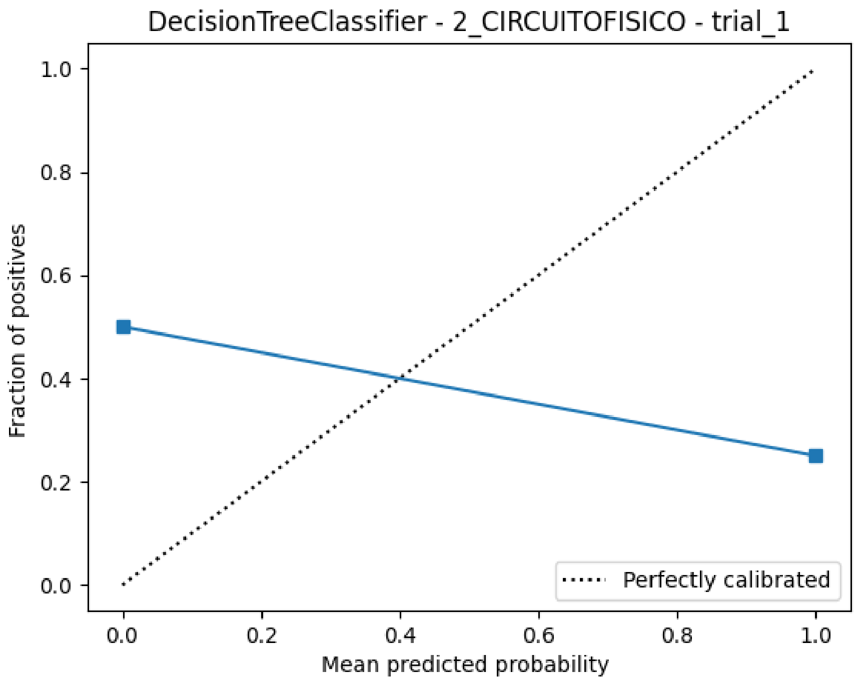

DecisionTreeClassifier - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

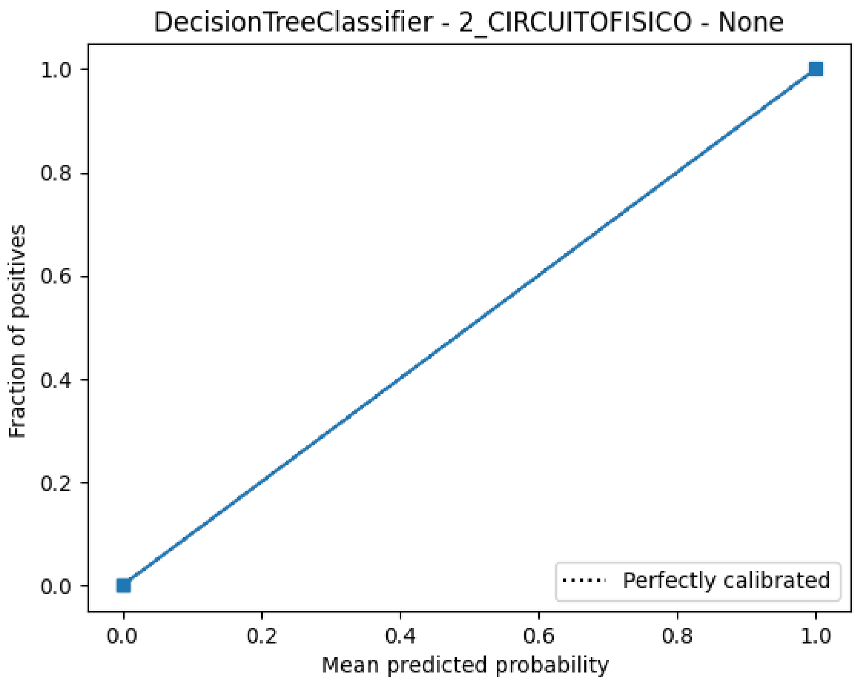

DecisionTreeClassifier - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

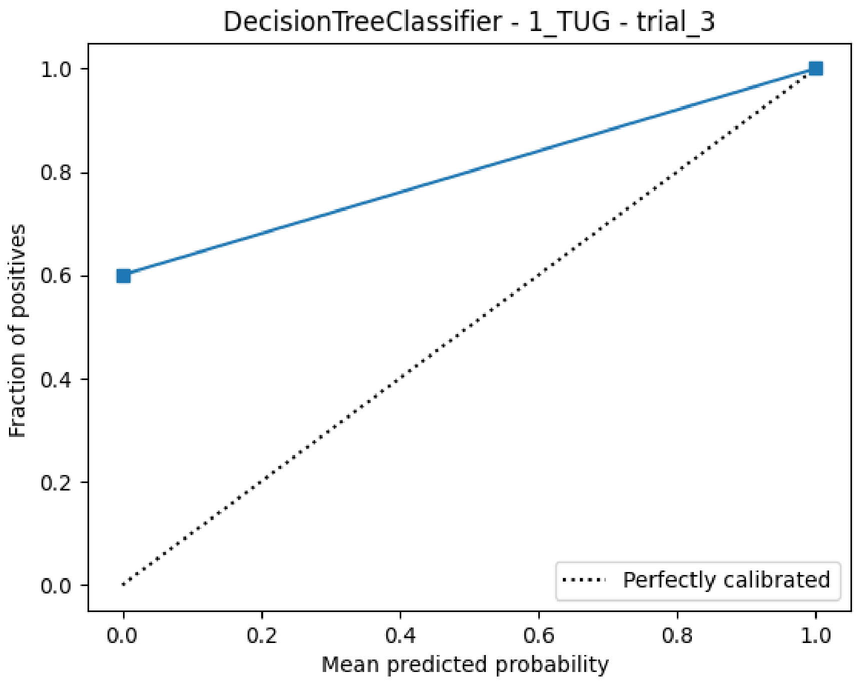

DecisionTreeClassifier - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

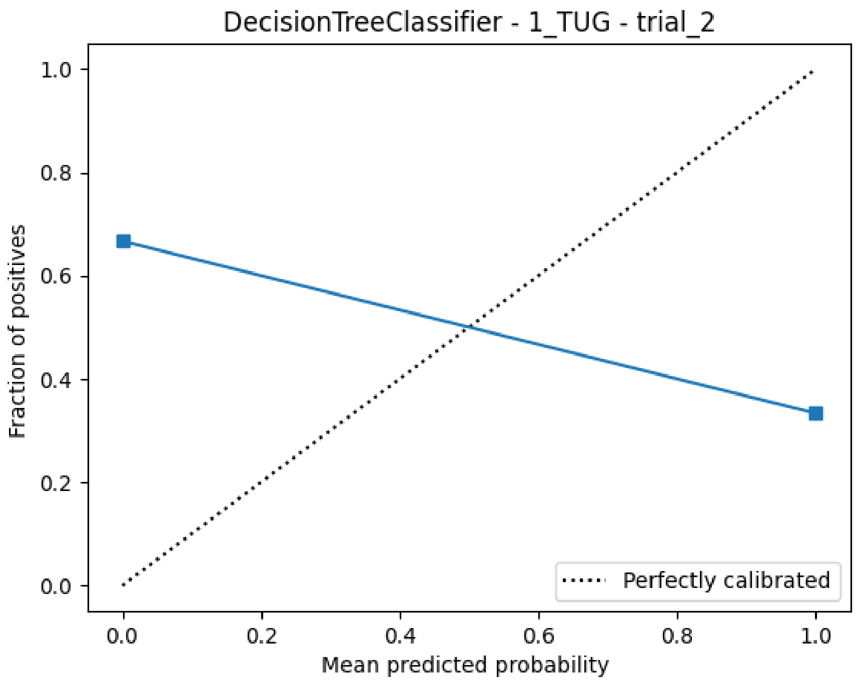

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

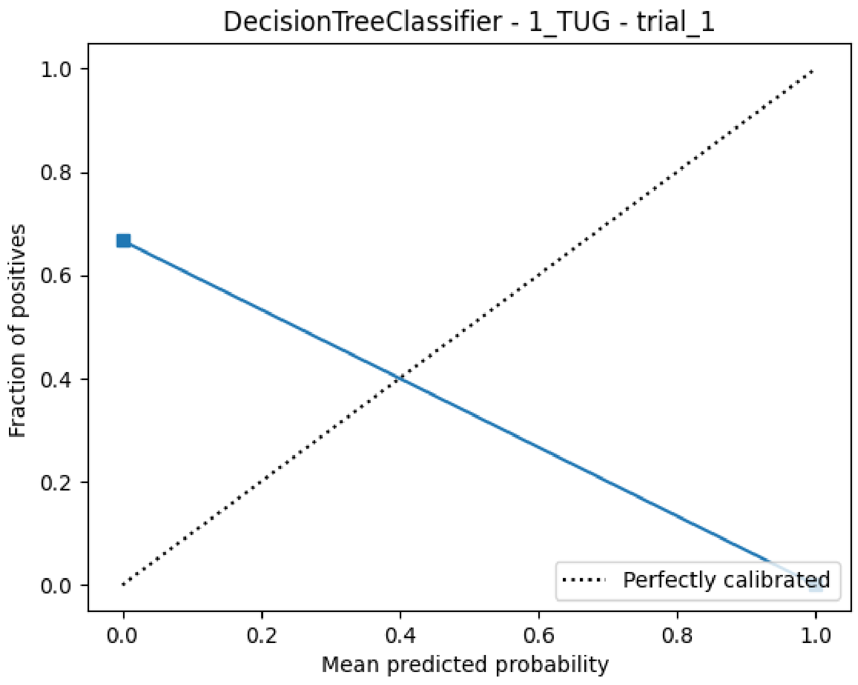

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

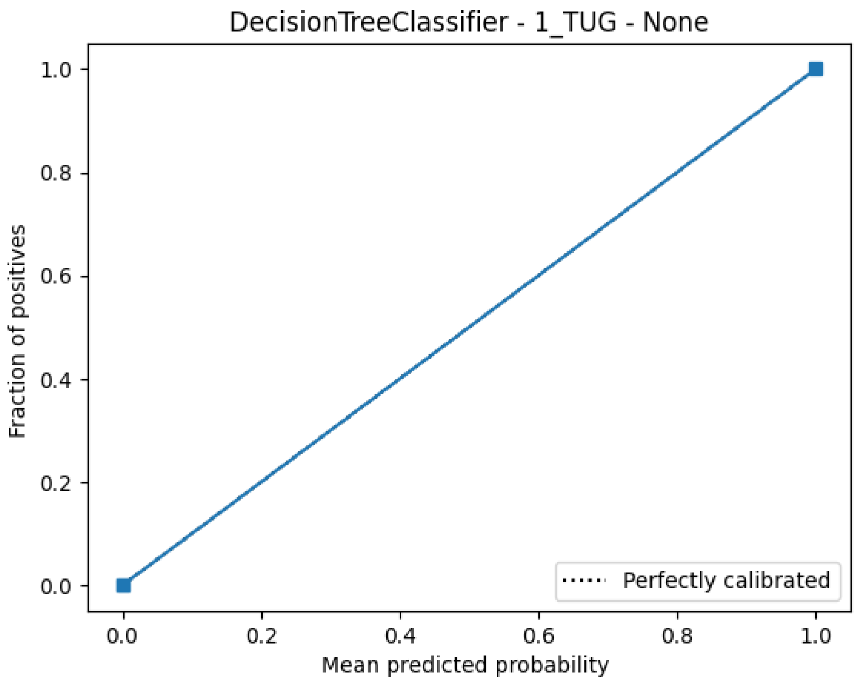

DecisionTreeClassifier - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

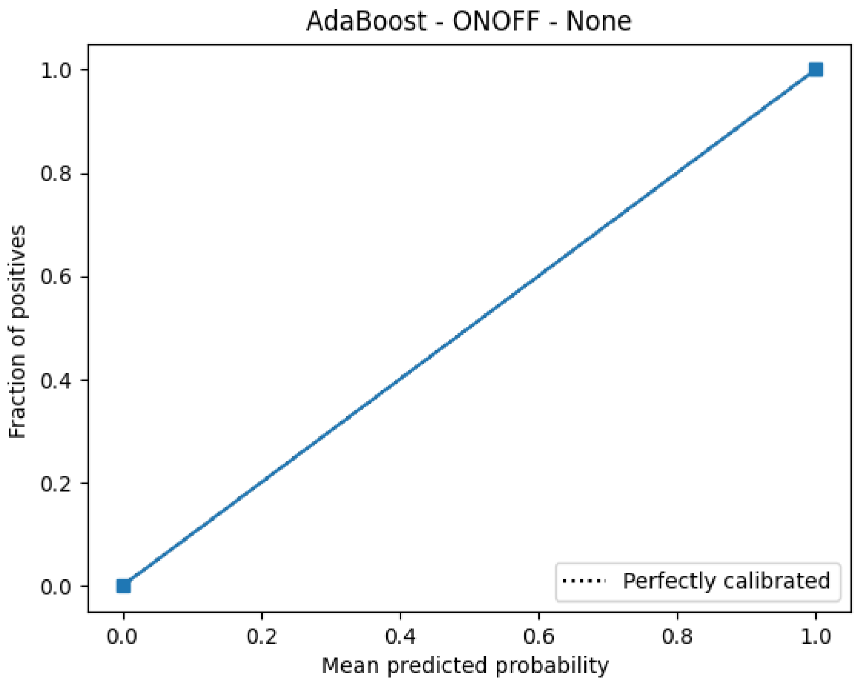

DecisionTreeClassifier - 2_CIRCUITOFISICO - None
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

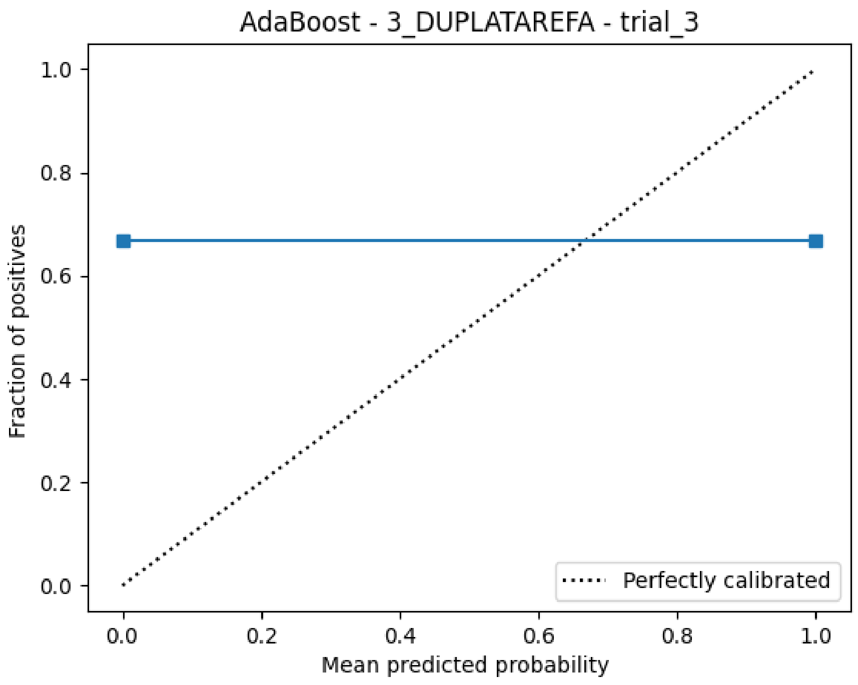

DecisionTreeClassifier - 1_TUG - trial_3
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability

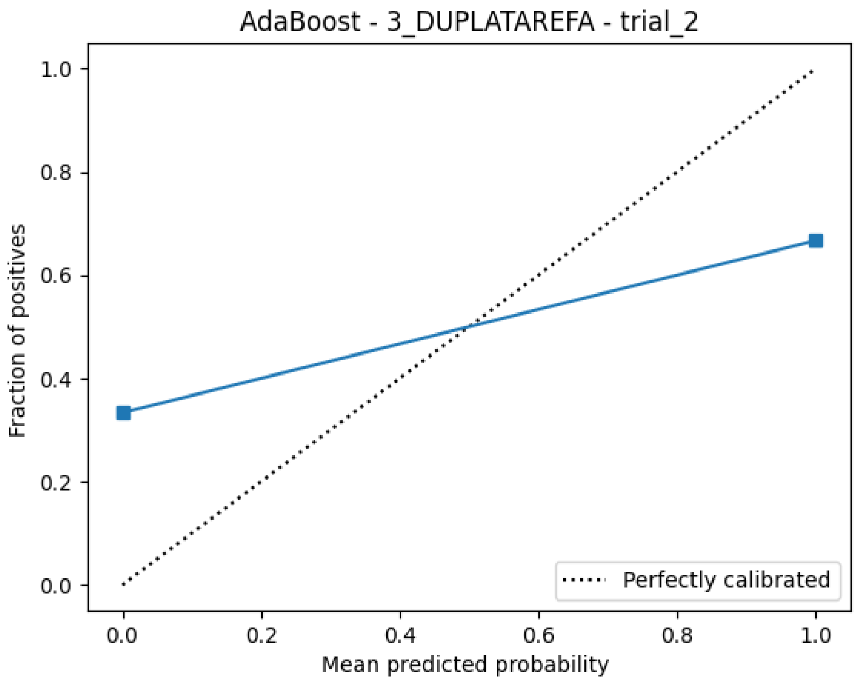

DecisionTreeClassifier - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

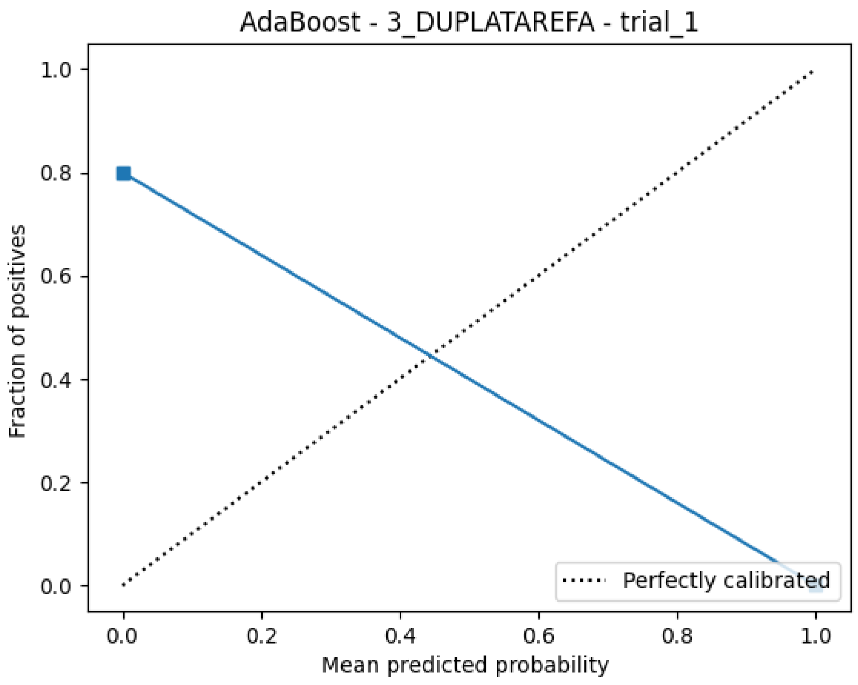

DecisionTreeClassifier - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

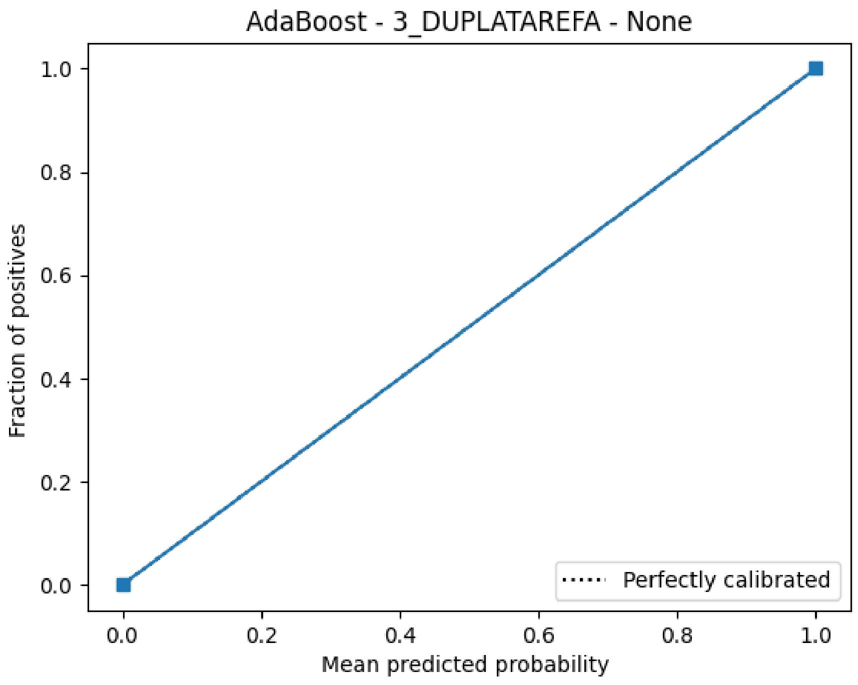

DecisionTreeClassifier - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

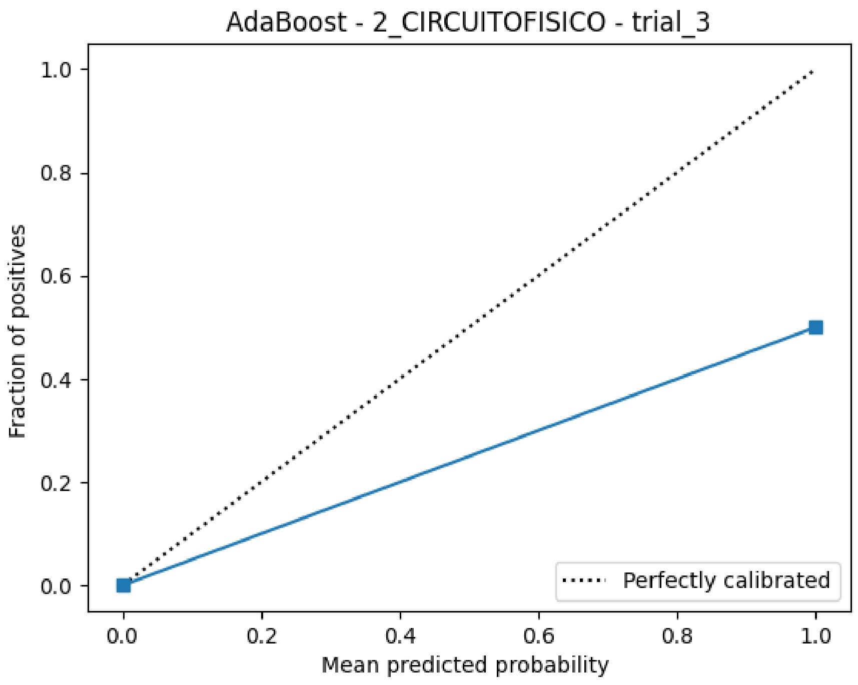

AdaBoost - ONOFF - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

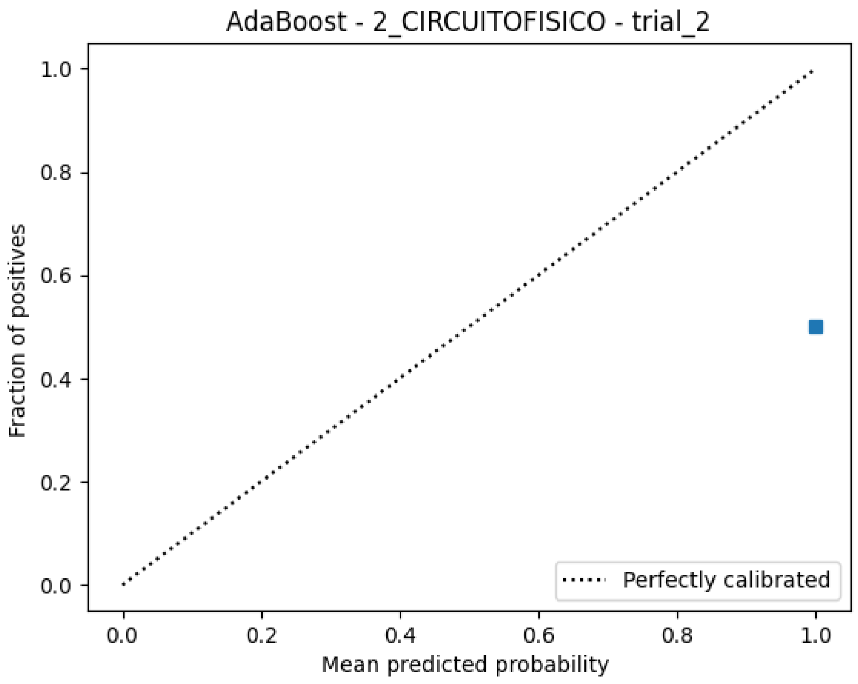

AdaBoost - 3_DUPLATAREFA - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

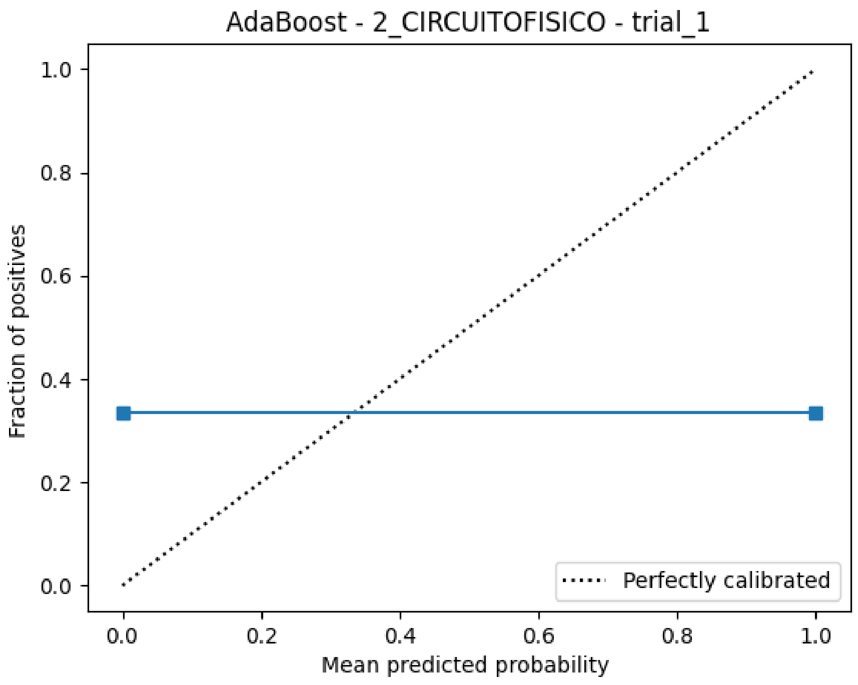

AdaBoost - 3_DUPLATAREFA - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

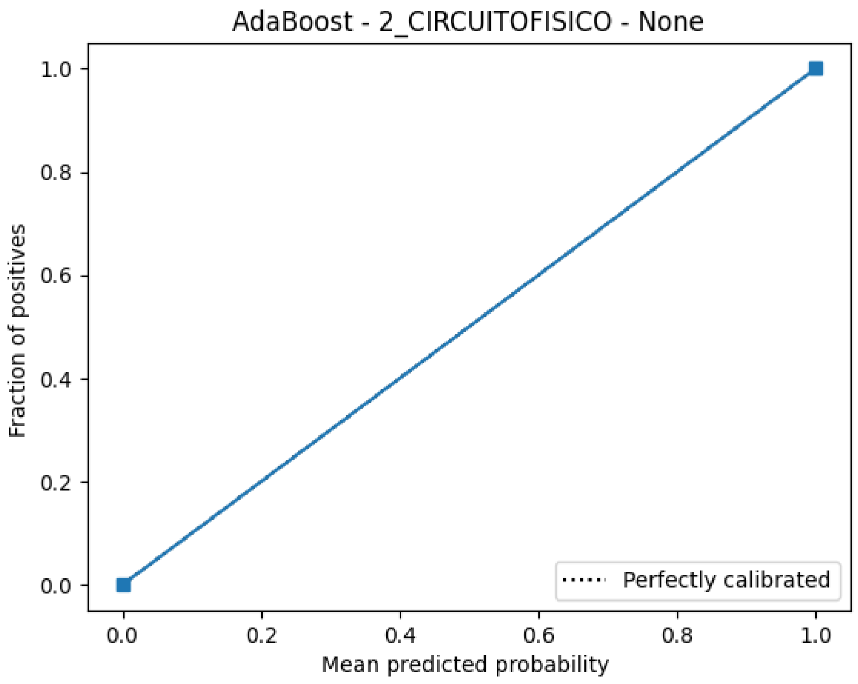

AdaBoost - 3_DUPLATAREFA - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

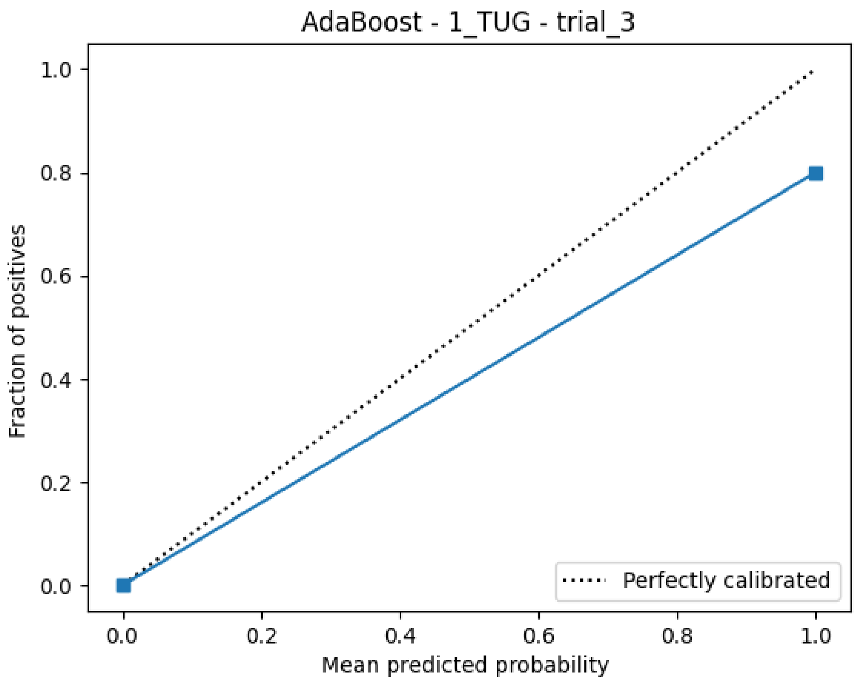

AdaBoost - 3_DUPLATAREFA - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

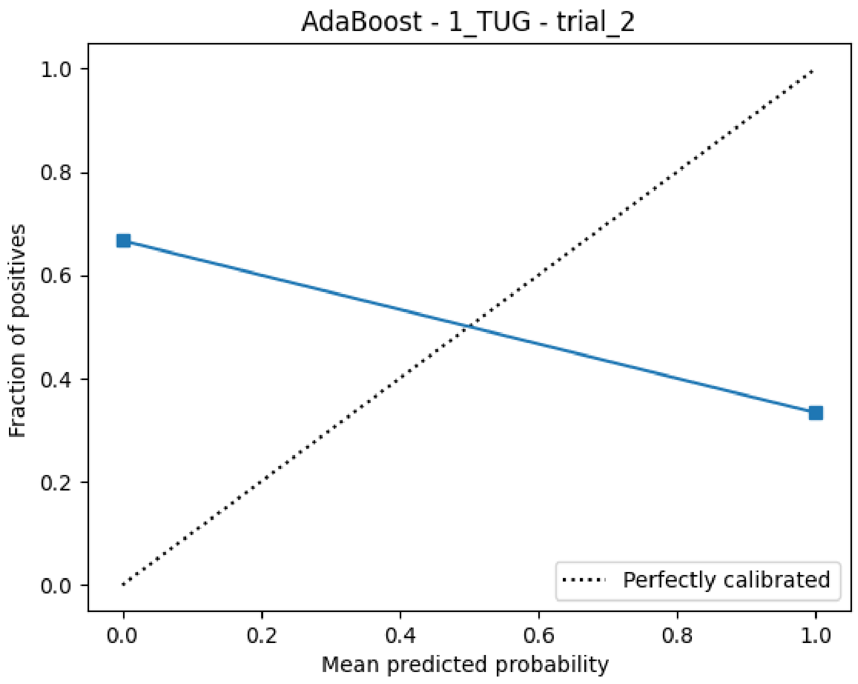

AdaBoost - 2_CIRCUITOFISICO - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

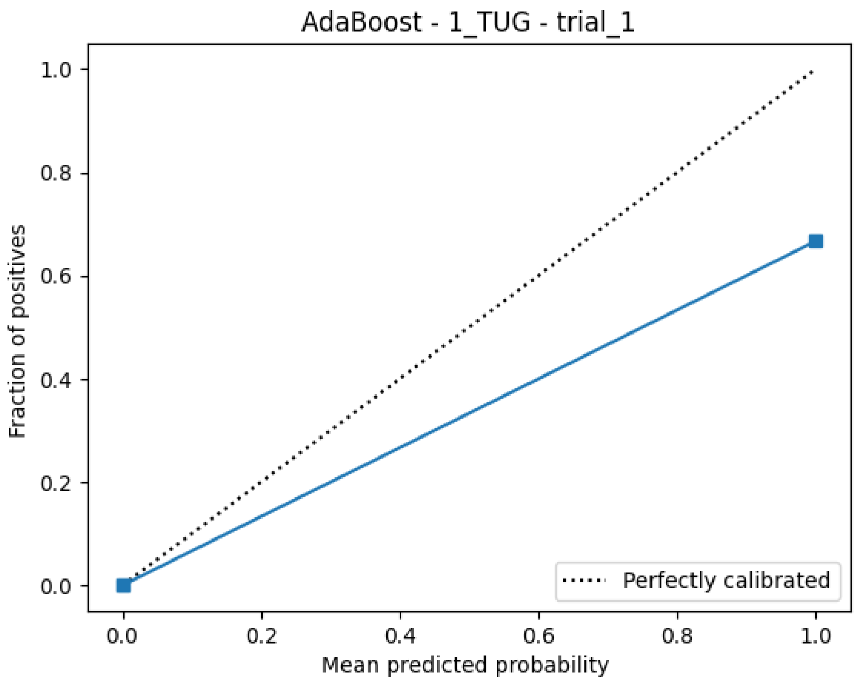

AdaBoost - 2_CIRCUITOFISICO - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

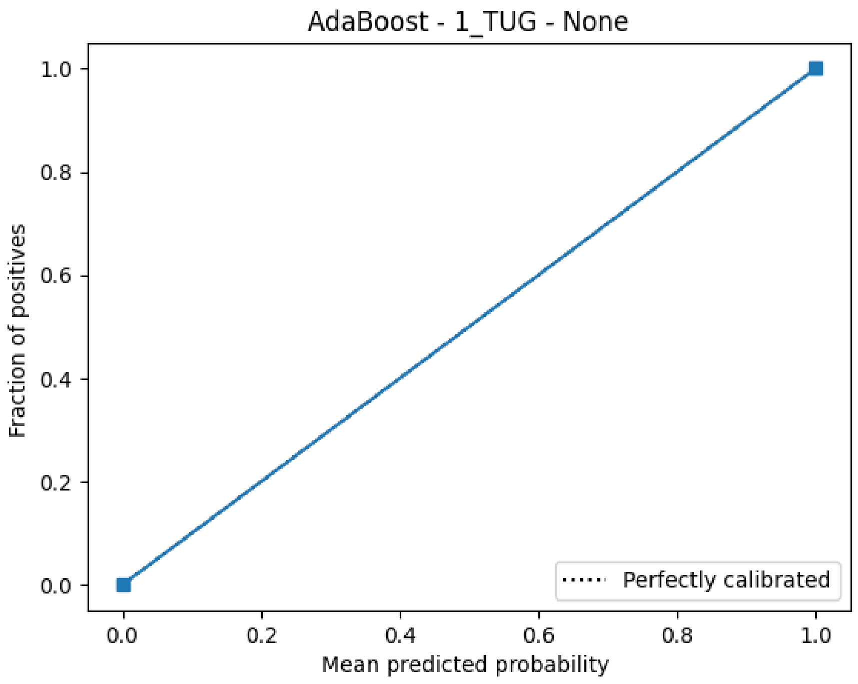

AdaBoost - 2_CIRCUITOFISICO - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0


AdaBoost - 2_CIRCUITOFISICO - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

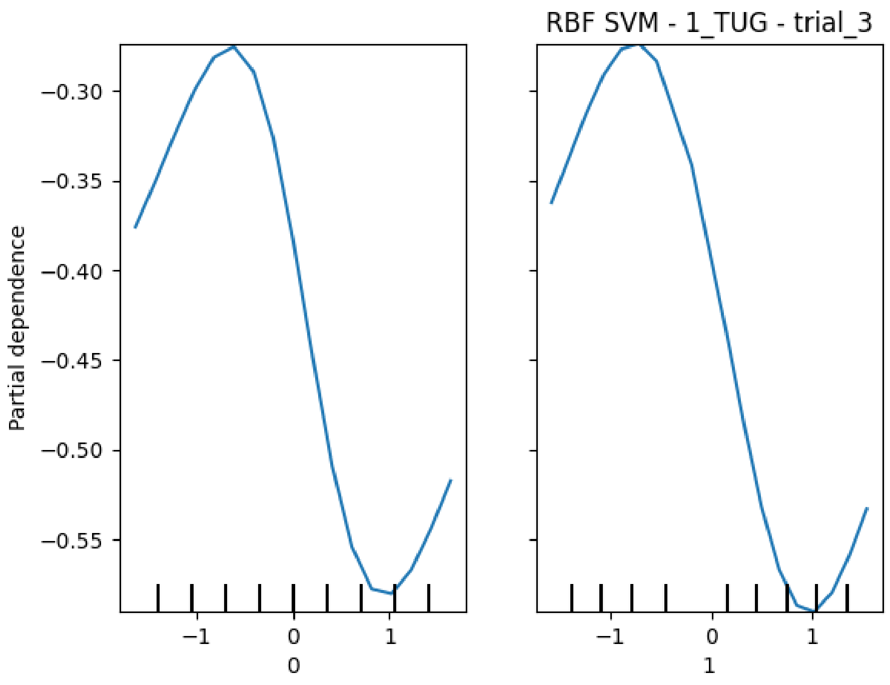

AdaBoost - 1_TUG - trial_3
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

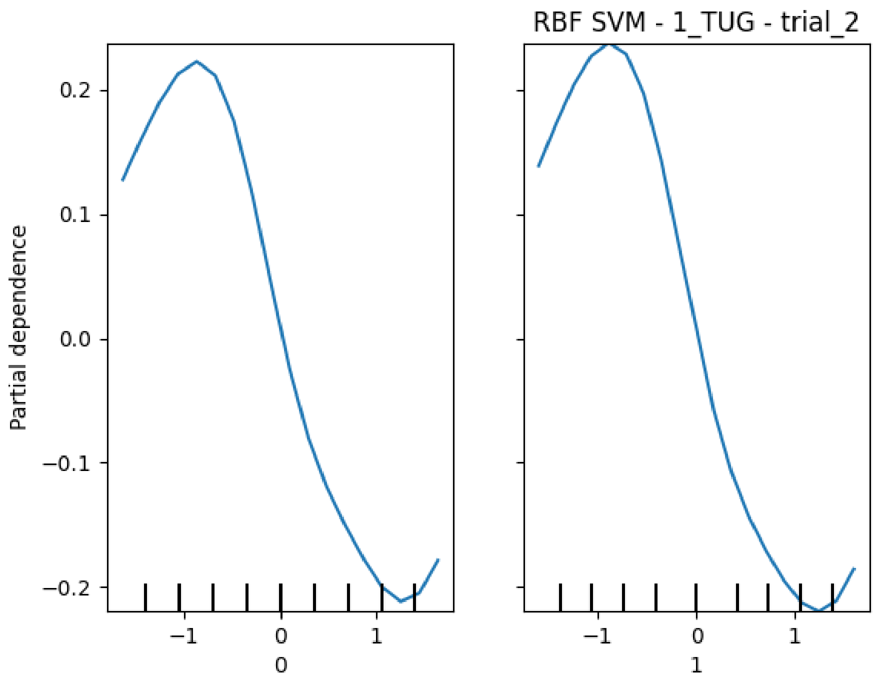

AdaBoost - 1_TUG - trial_2
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

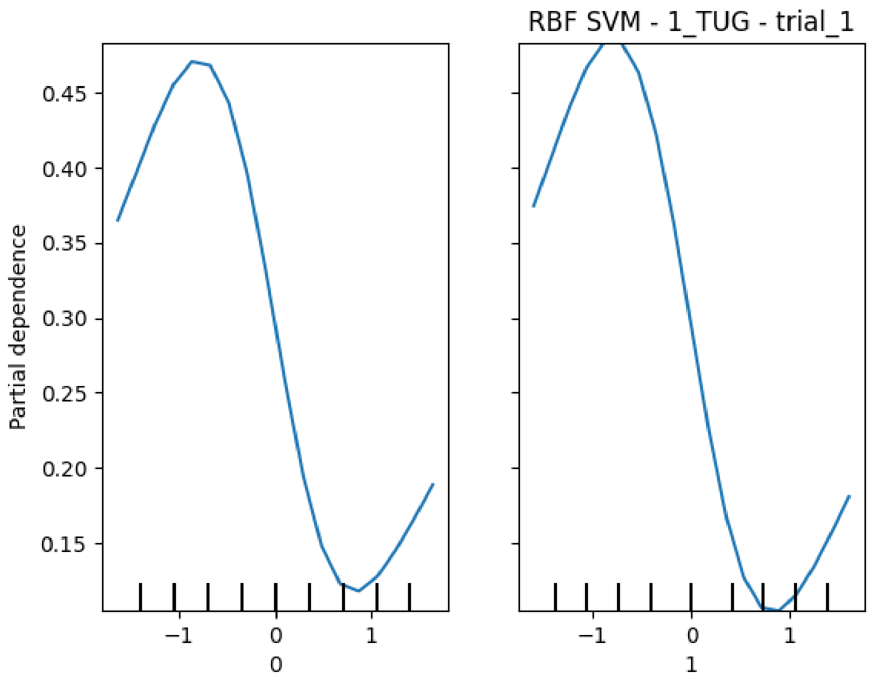

AdaBoost - 1_TUG - trial_1
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

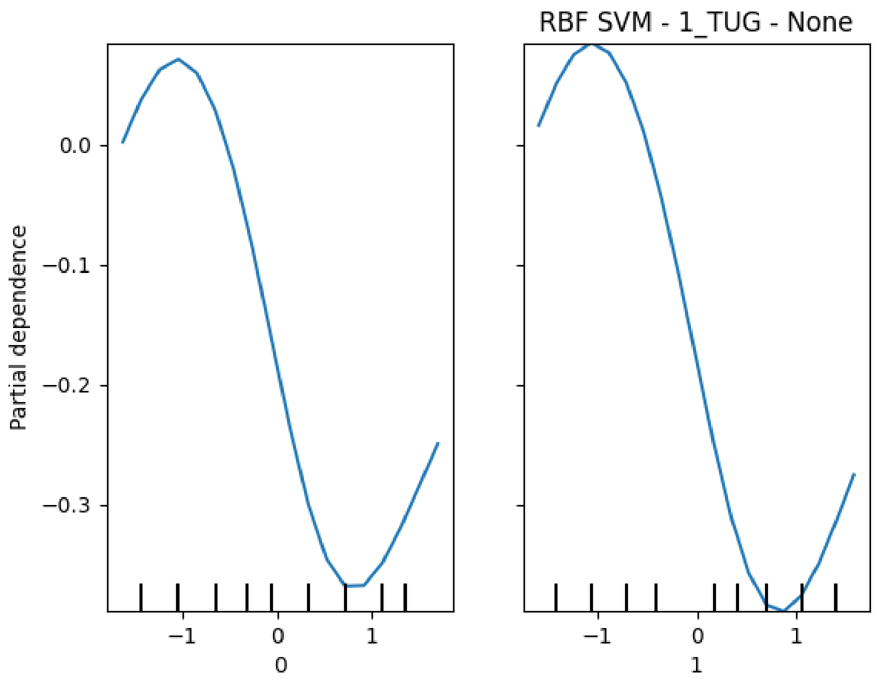

AdaBoost - 1_TUG - None
Fraction of positives
Mean predicted probability
Perfectly calibrated
0.0
0.2
0.4
0.6
0.8
1.0

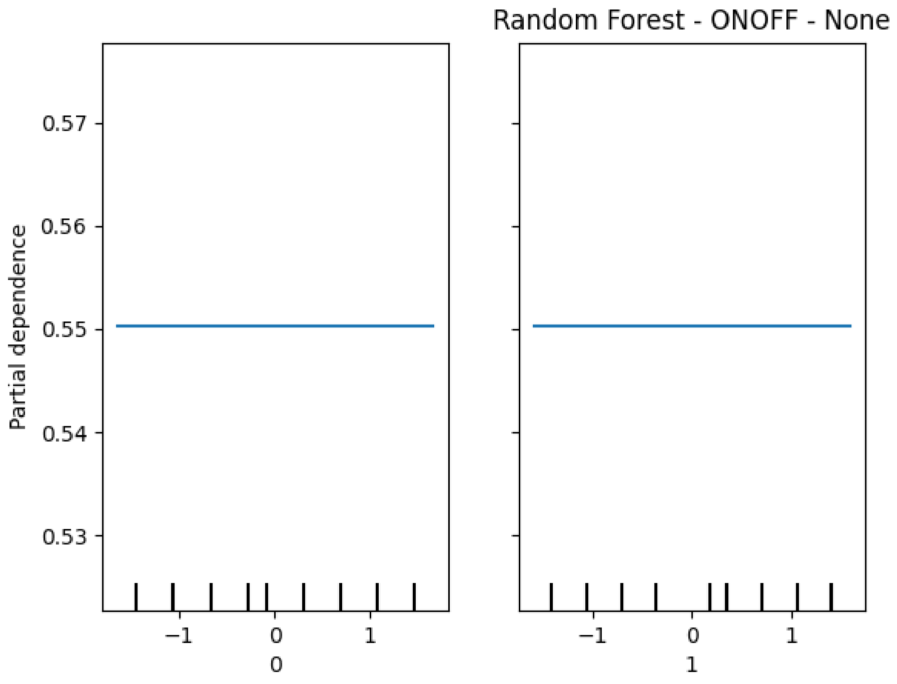
A.3 Partial Dependency Analysis

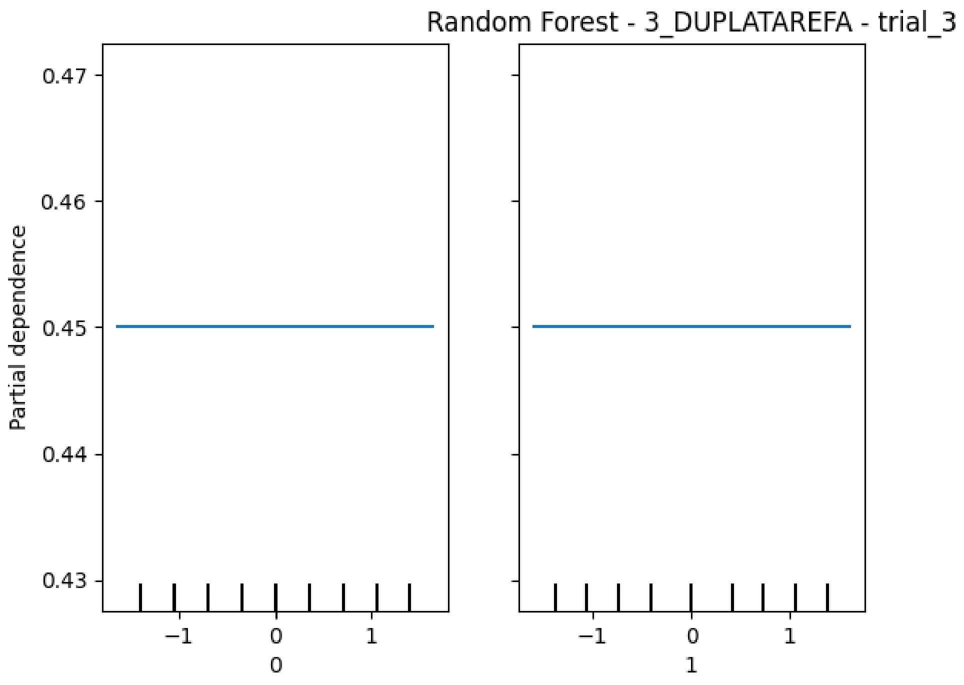


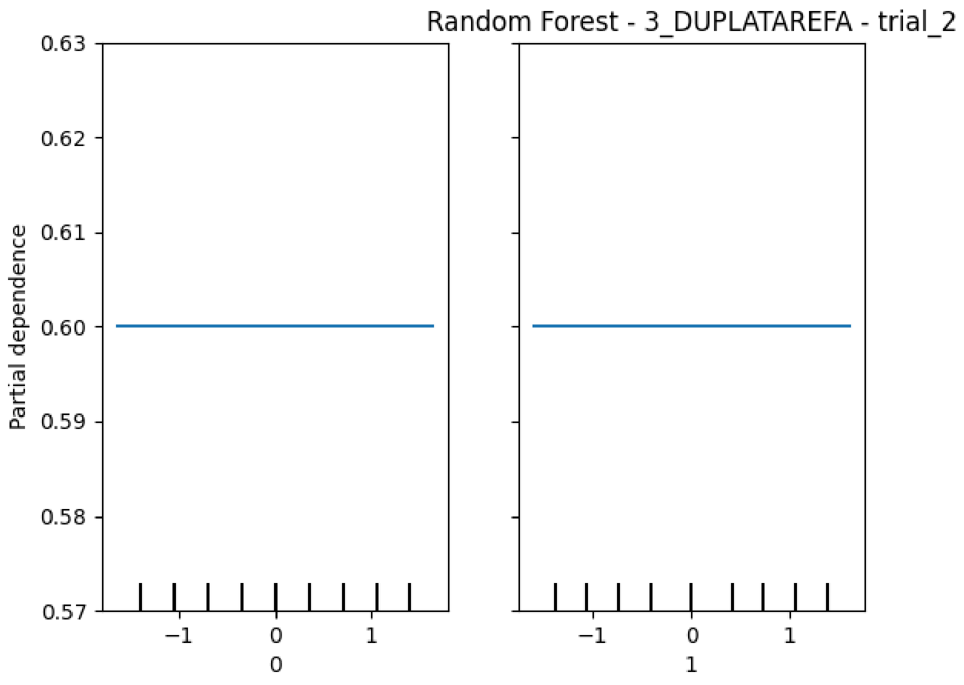

RBF SVM - 1_TUG - trial_2
Partial dependence
0.2
0.1
0.0
−0.1
−0.2
−1
0
1
0
1

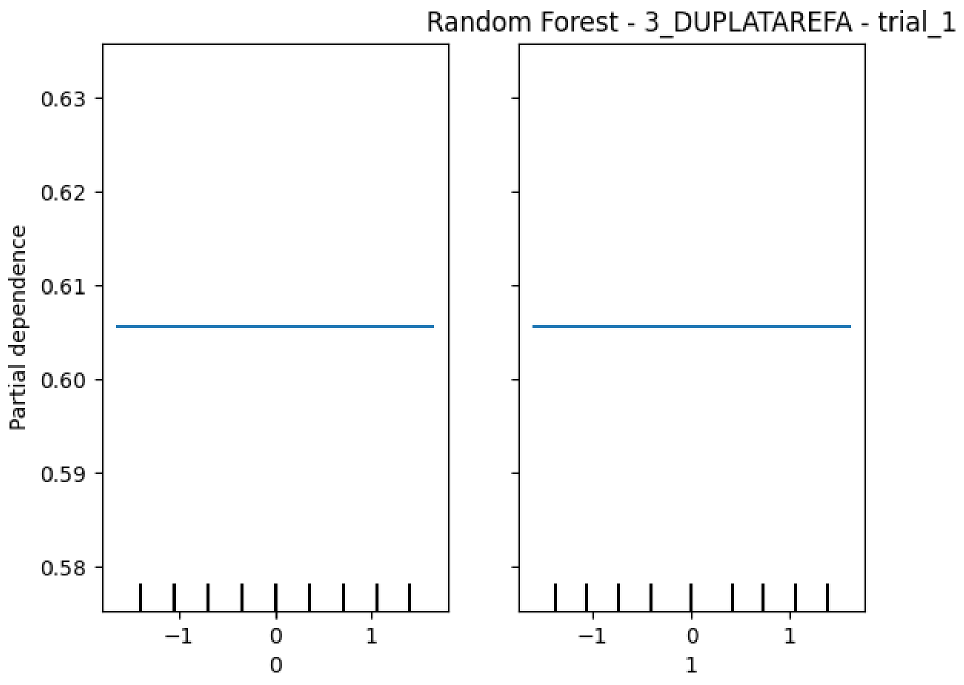

RBF SVM - 1_TUG - trial_1
Partial dependence
0.45
0.40
0.35
0.30
0.25
0.20
0.15
−1
0
1
0
1

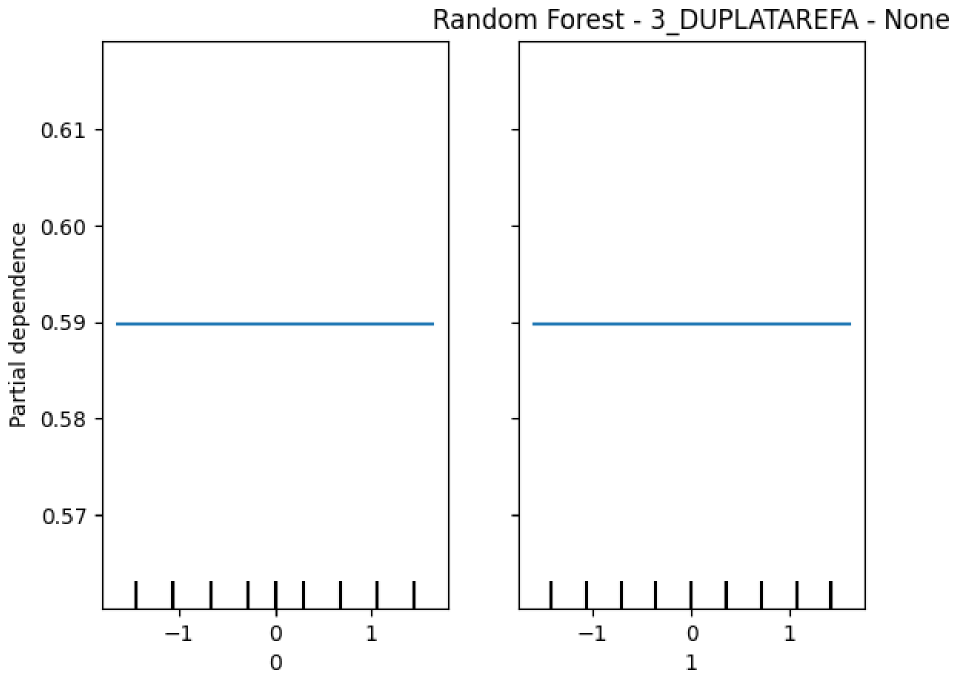

RBF SVM - 1_TUG - None
Partial dependence
0.0
−0.1
−0.2
−0.3
−1
0
1
0
1

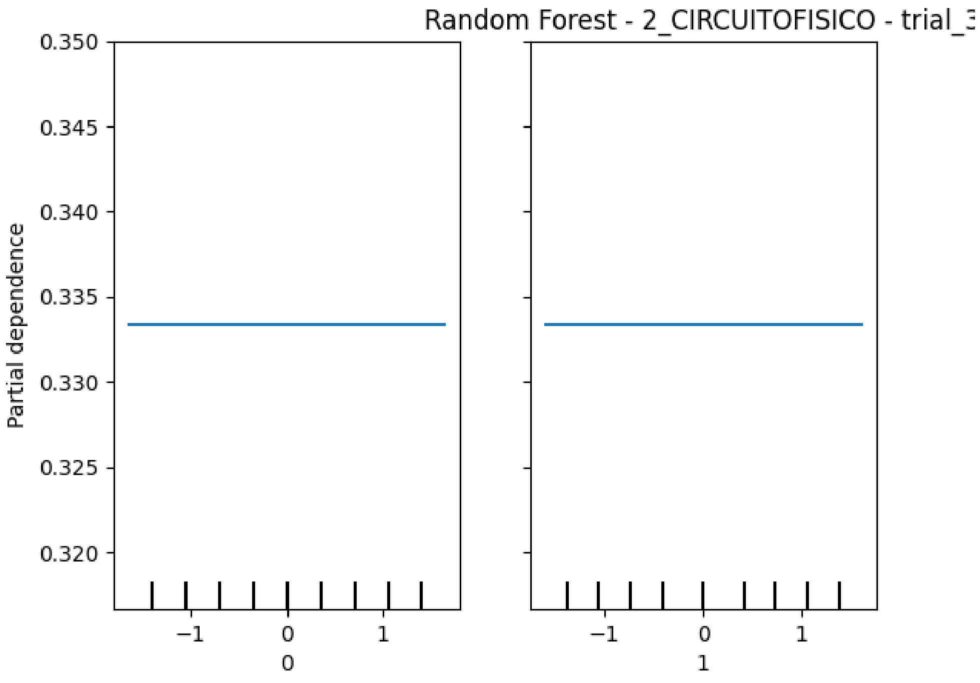

Random Forest - ONOFF - None
Partial dependence
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

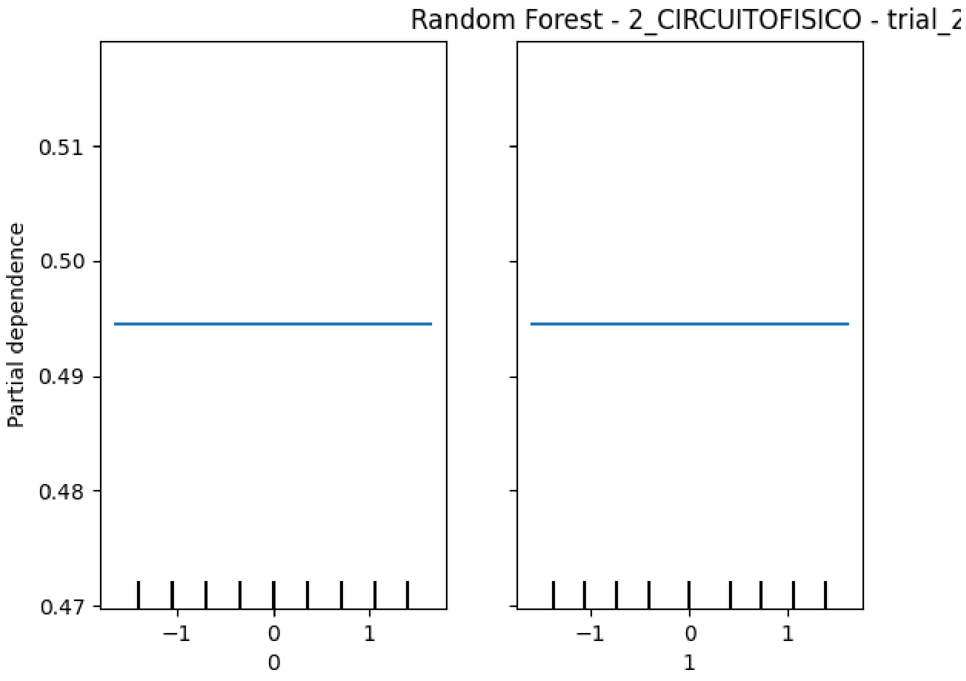

Random Forest - 3_DUPLATAREFA - trial_3
Partial dependence
0.47
0.46
0.45
0.44
0.43
−1
0
1
0
1

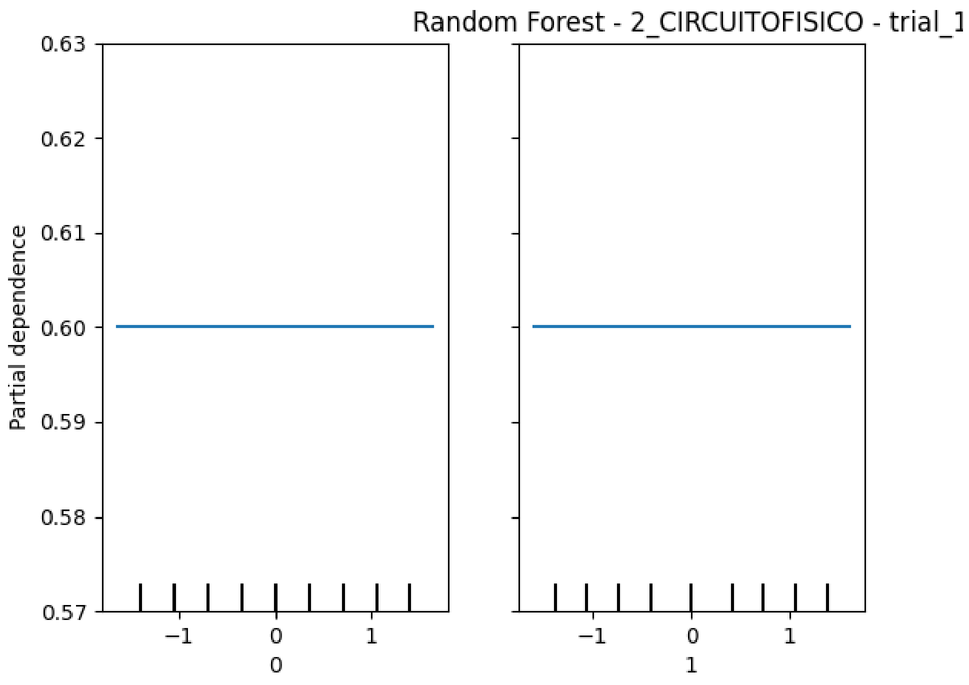

Random Forest - 3_DUPLATAREFA - trial_2
Partial dependence
0.63
0.62
0.61
0.60
0.59
0.58
0.57
−1
0
1
0
1

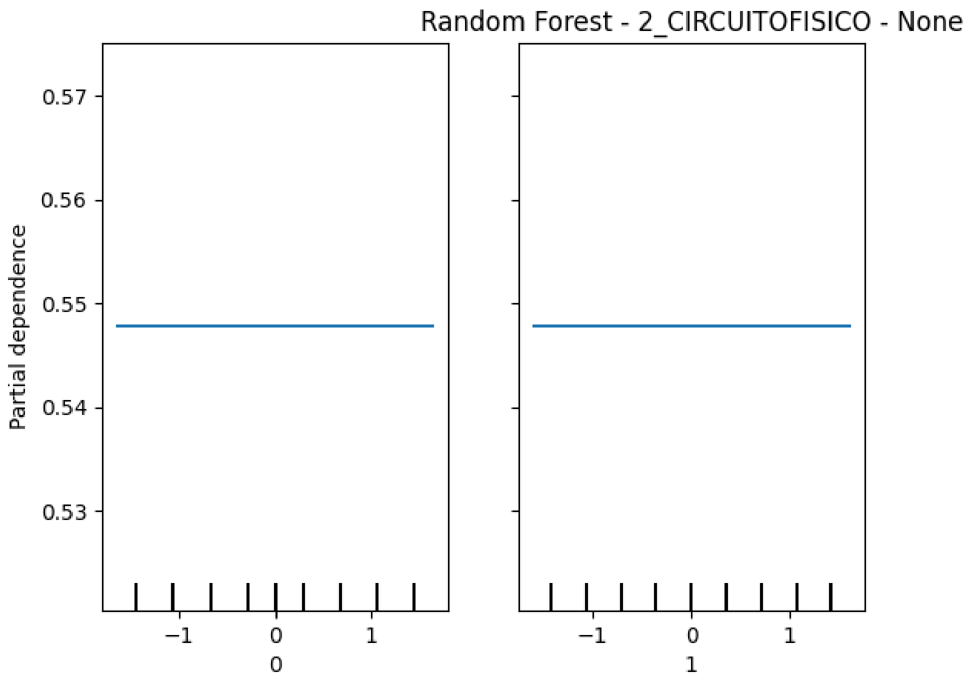

Random Forest - 3_DUPLATAREFA - trial_1
Partial dependence
0.63
0.62
0.61
0.60
0.59
0.58
−1
0
1
0
1

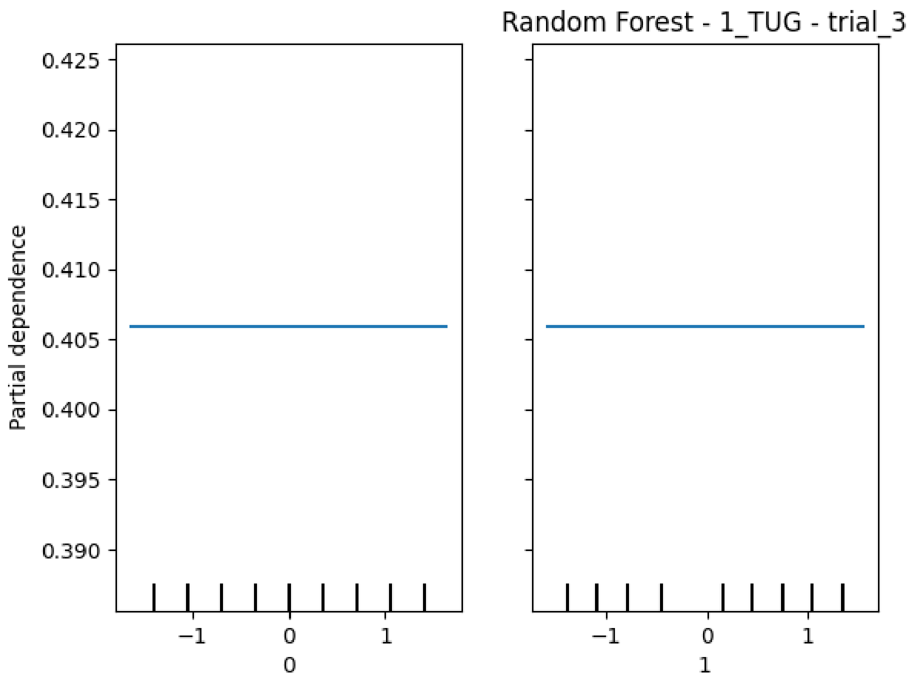

Random Forest - 3_DUPLATAREFA - None
Partial dependence
0.61
0.60
0.59
0.58
0.57
−1
0
1
0
1

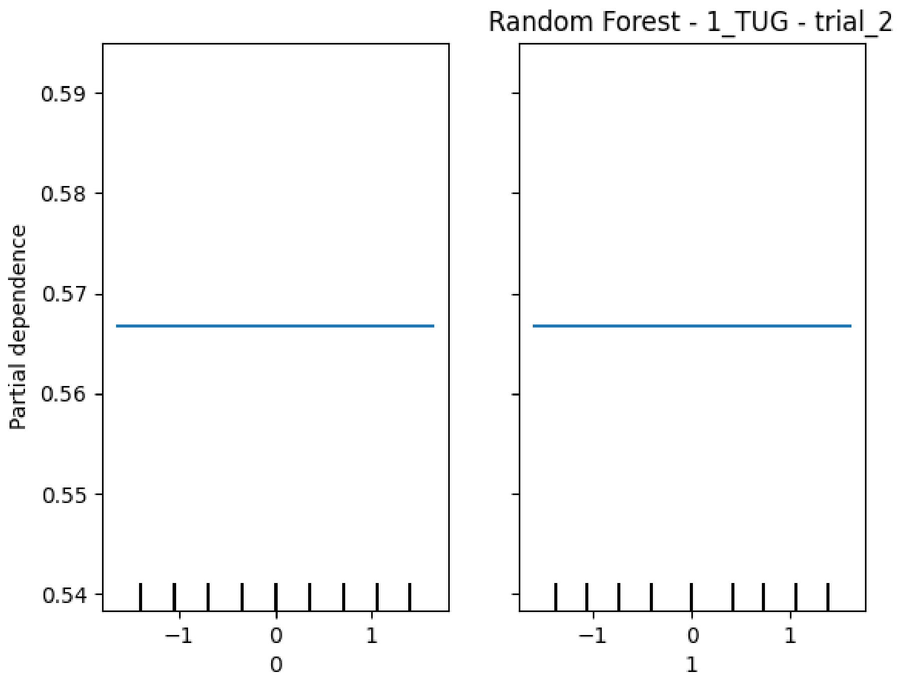

Random Forest - 2_CIRCUITOFISICO - trial_3
Partial dependence
0.350
0.345
0.340
0.335
0.330
0.325
0.320
−1
0
1
0
1

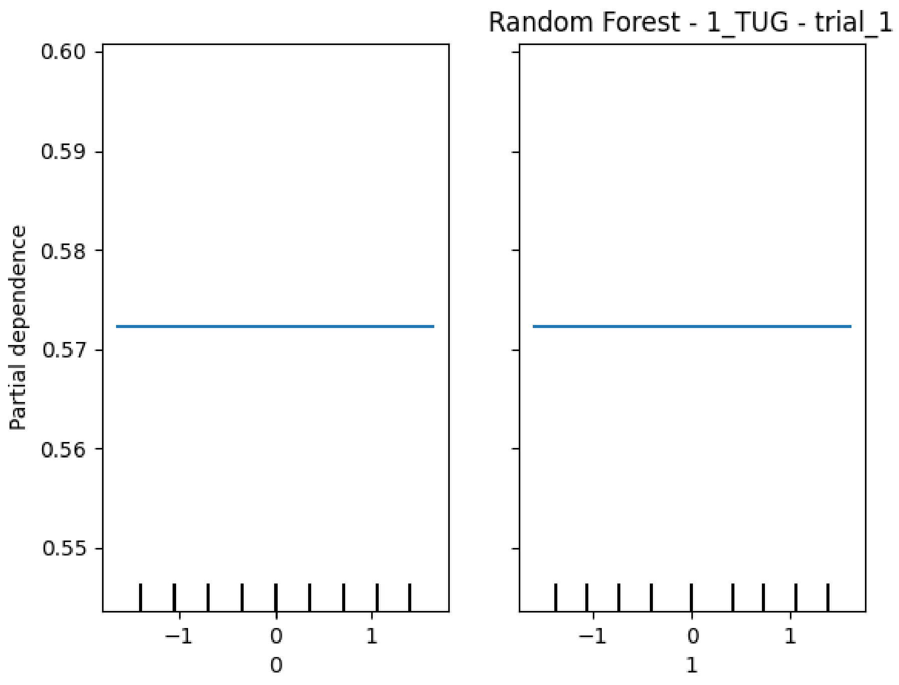

Random Forest - 2_CIRCUITOFISICO - trial_2
Partial dependence
0.51
0.50
0.49
0.48
0.47
−1
0
1
0
1

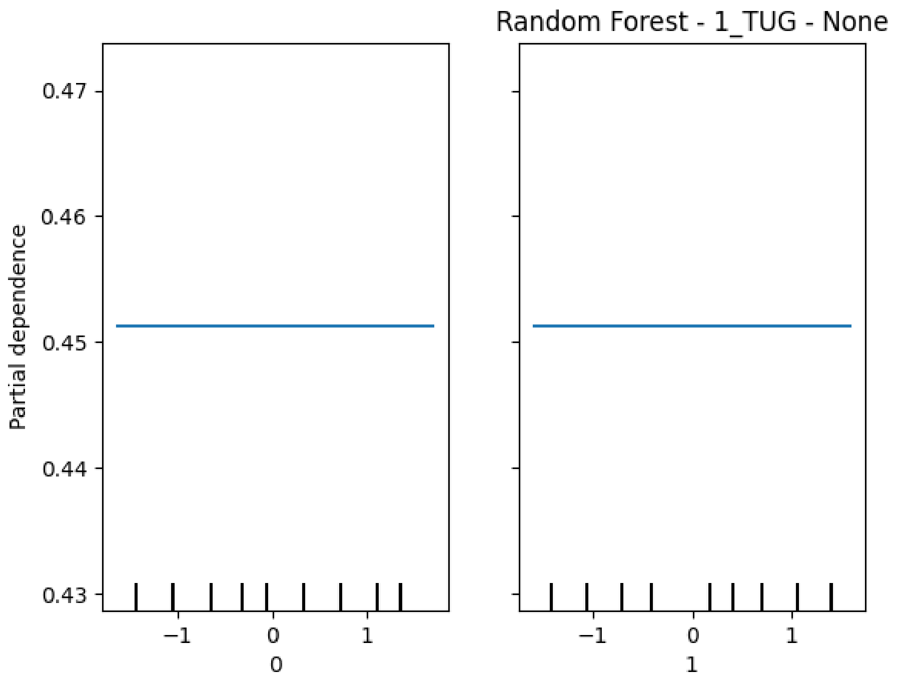

Random Forest - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.63
0.62
0.61
0.60
0.59
0.58
0.57
−1
0
1
0
1

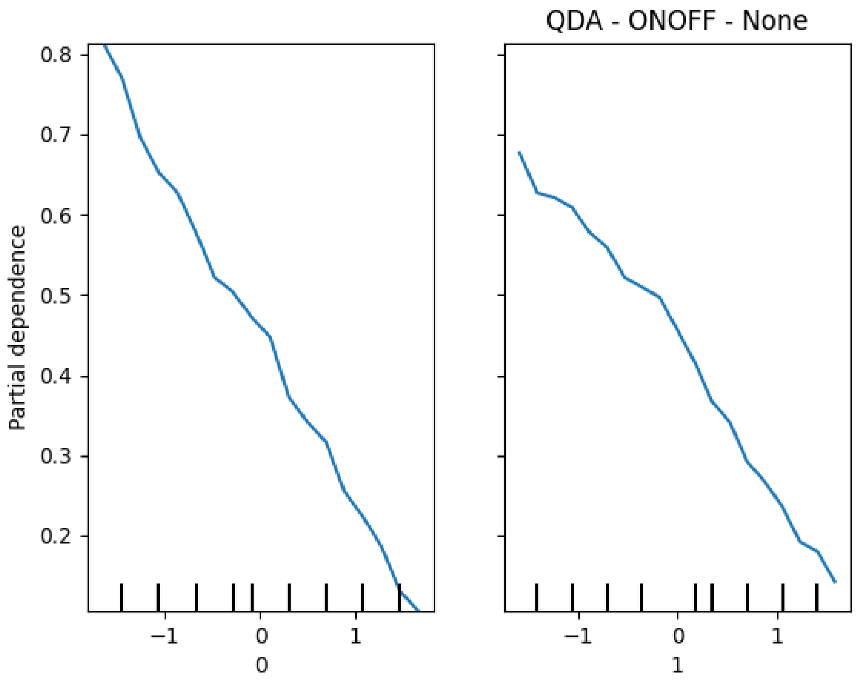

Random Forest - 2_CIRCUITOFISICO - None
Partial dependence
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

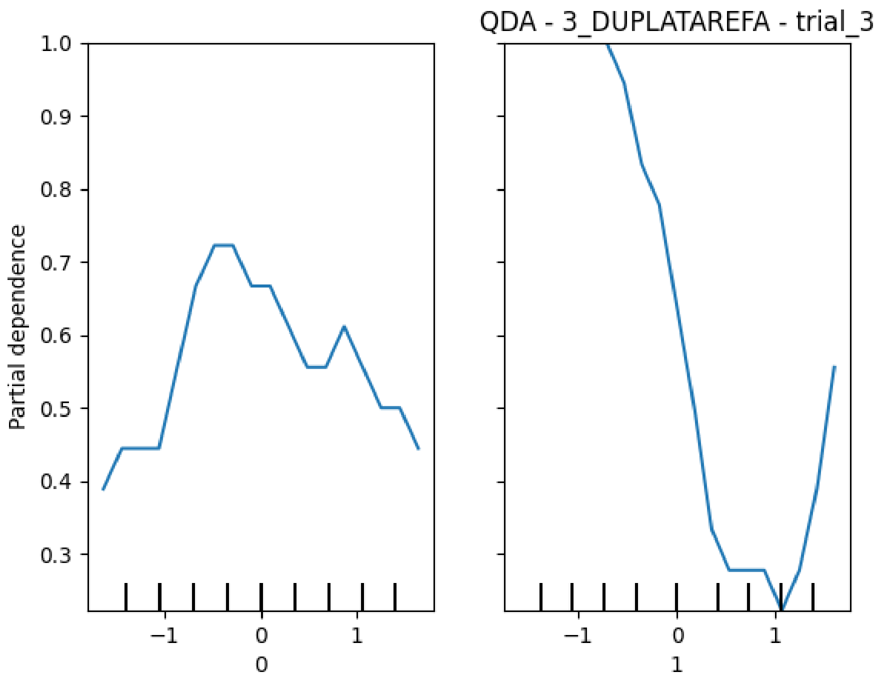

Random Forest - 1_TUG - trial_3
Partial dependence
0.425
0.420
0.415
0.410
0.405
0.400
0.395
0.390
−1
0
1
0
1

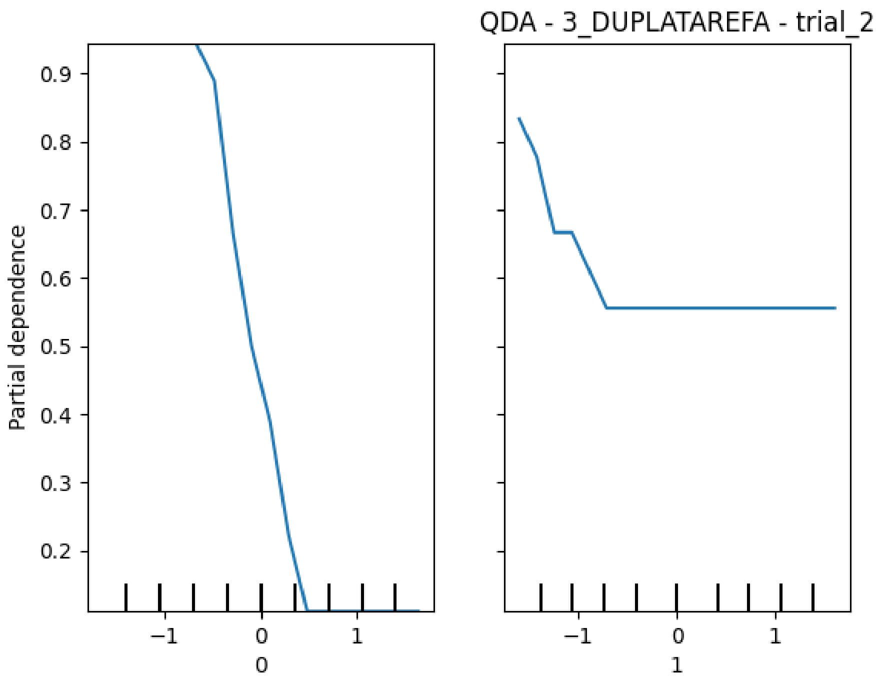

Random Forest - 1_TUG - trial_2
Partial dependence
0.59
0.58
0.57
0.56
0.55
0.54
−1
0
1
0
1

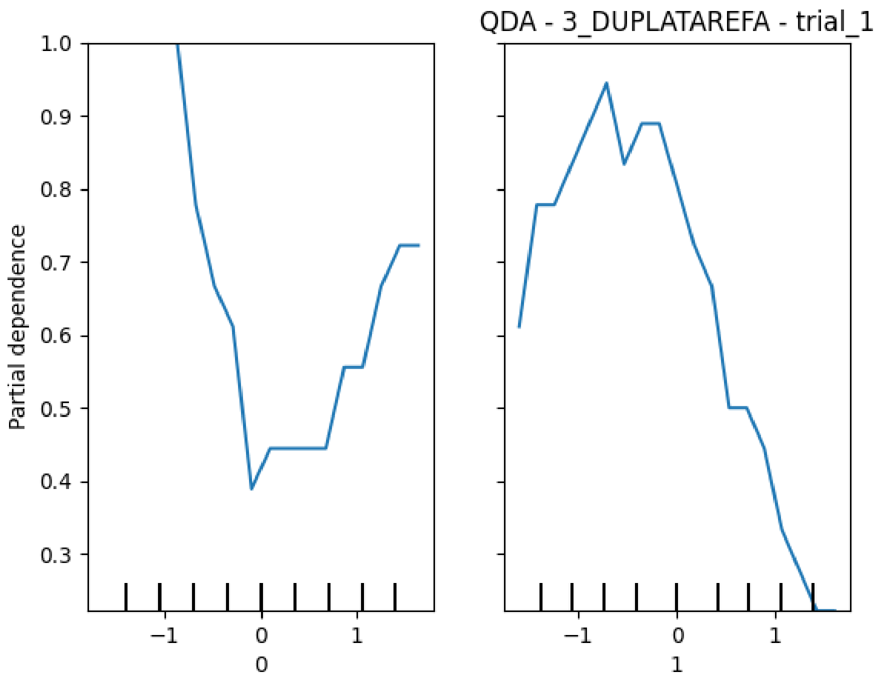

Random Forest - 1_TUG - trial_1
Partial dependence
0.60
0.59
0.58
0.57
0.56
0.55
−1
0
1
0
1

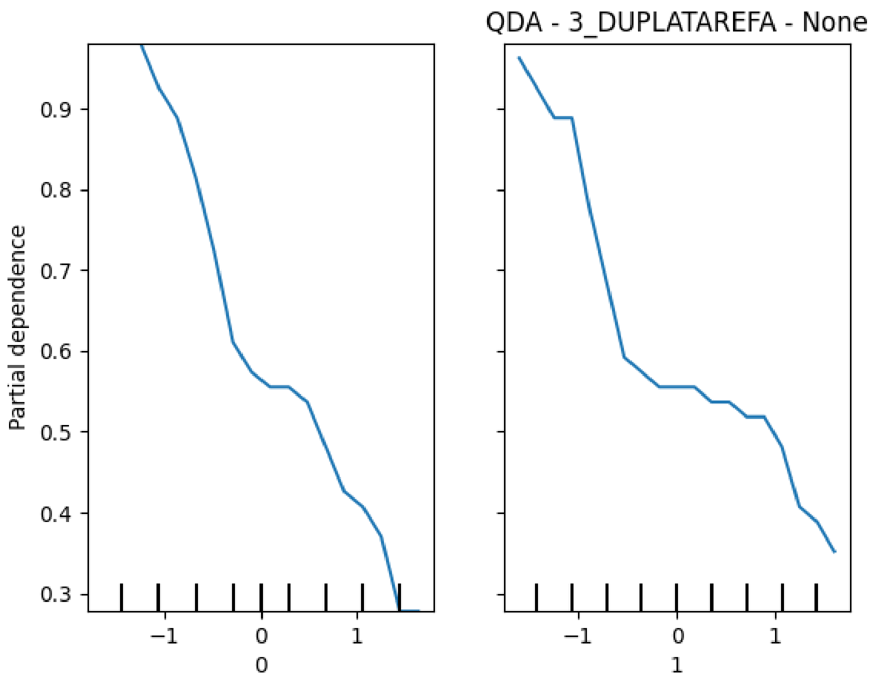

Random Forest - 1_TUG - None
Partial dependence
0.47
0.46
0.45
0.44
0.43
−1
0
1
0
1

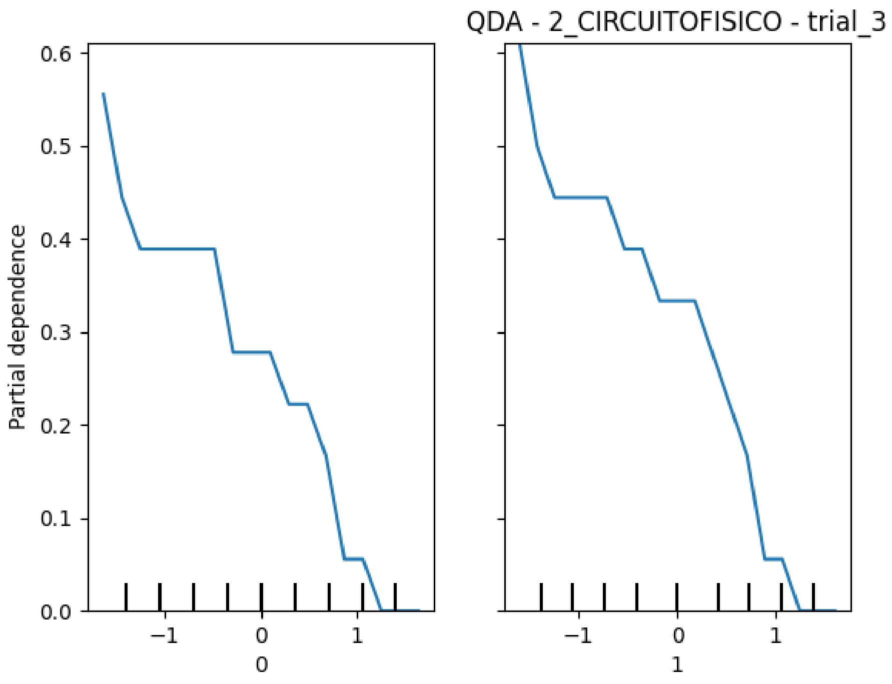

QDA - ONOFF - None
Partial dependence
0.8
0.7
0.6
0.5
0.4
0.3
0.2
−1
0
1
0
1

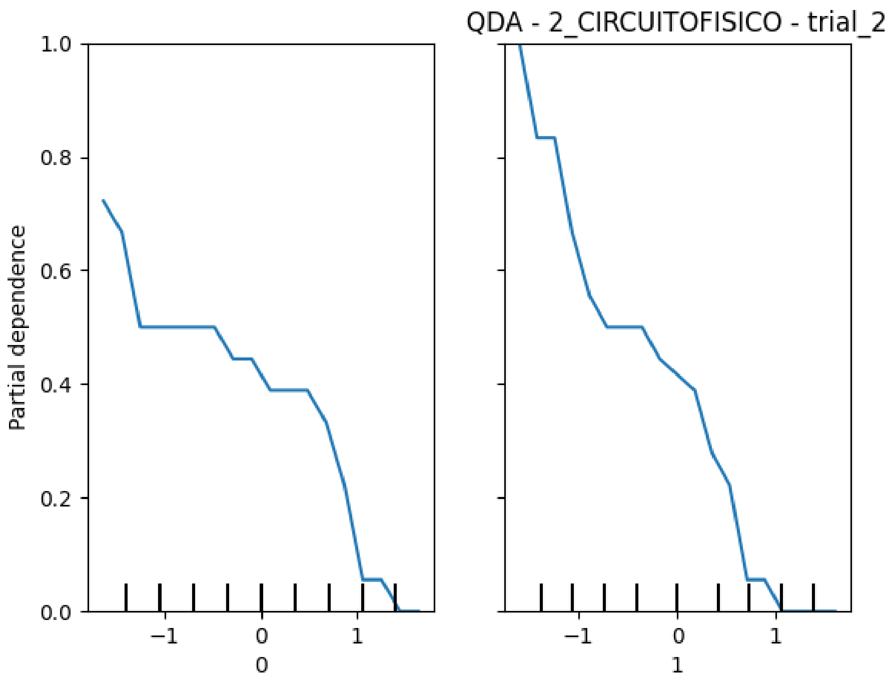

QDA - 3_DUPLATAREFA - trial_3
Partial dependence
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−1
0
1
0
1

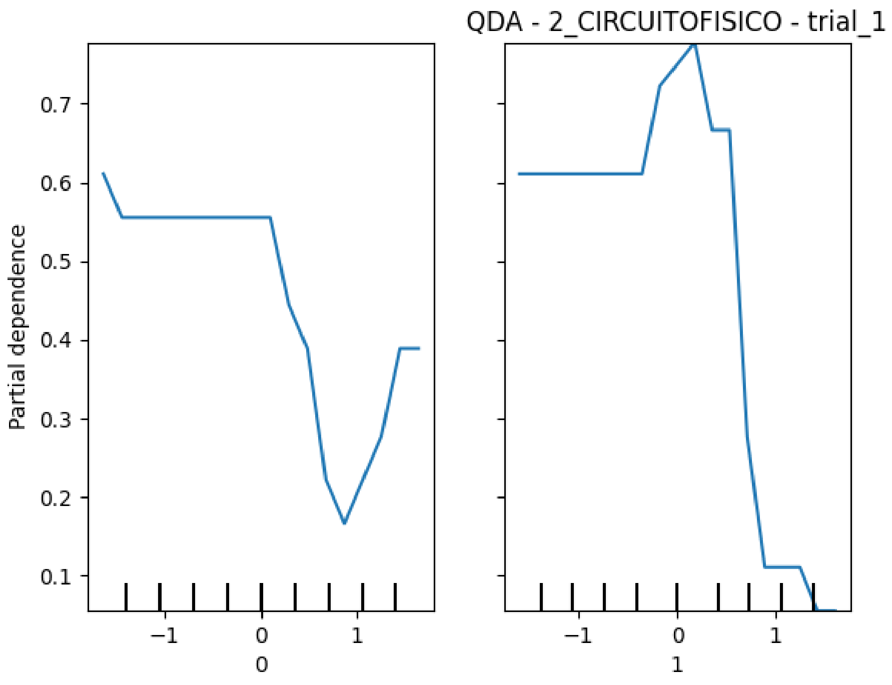

QDA - 3_DUPLATAREFA - trial_2
Partial dependence
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
−1
0
1
0
1

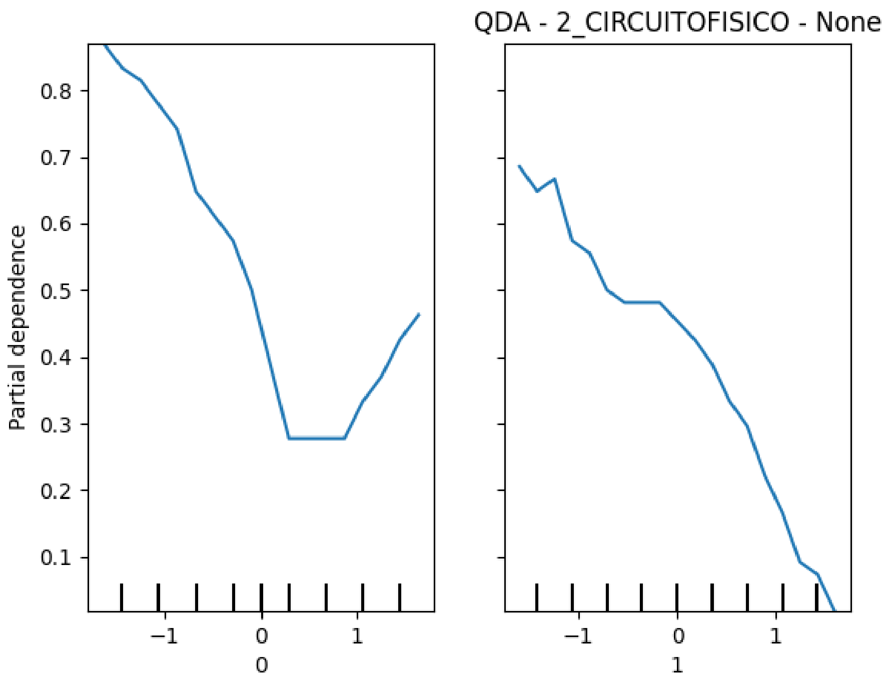

QDA - 3_DUPLATAREFA - trial_1
Partial dependence
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−1
0
1
0
1

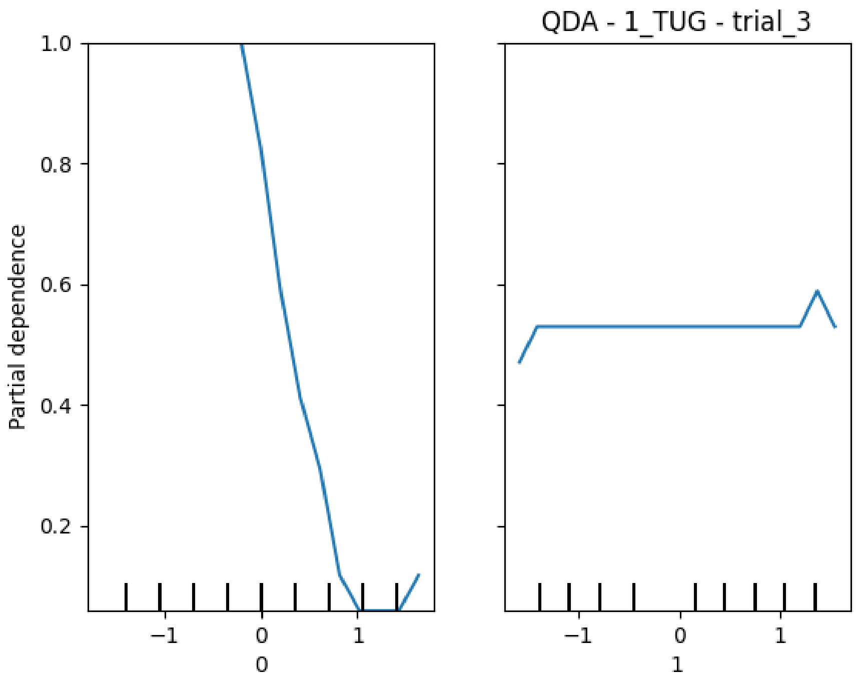

QDA - 3_DUPLATAREFA - None
Partial dependence
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−1
0
1
0
−1
0
1
1

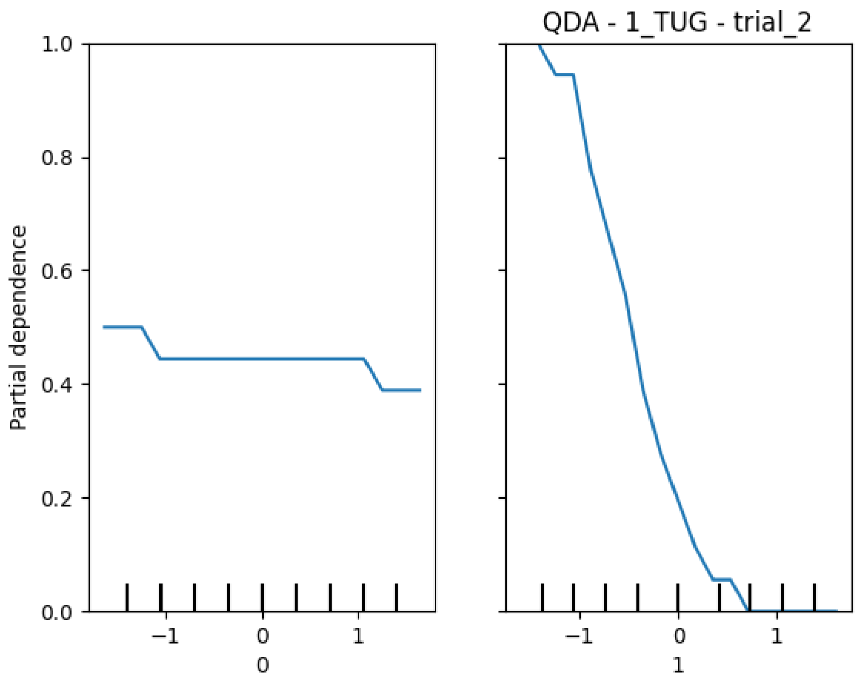

QDA - 2_CIRCUITOFISICO - trial_3
Partial dependence
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−1
0
1
0
−1
0
1
1

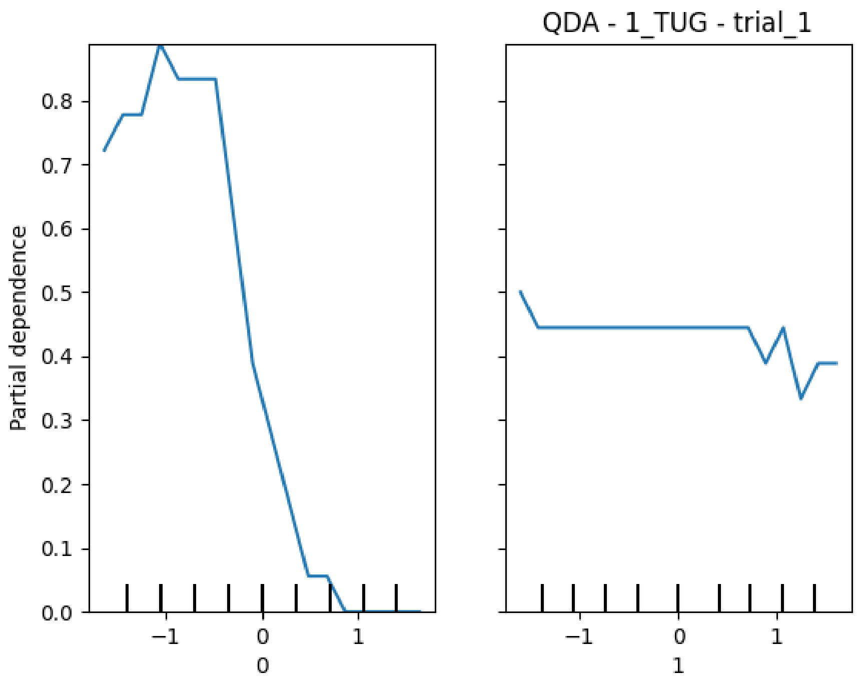

QDA - 2_CIRCUITOFISICO - trial_2
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

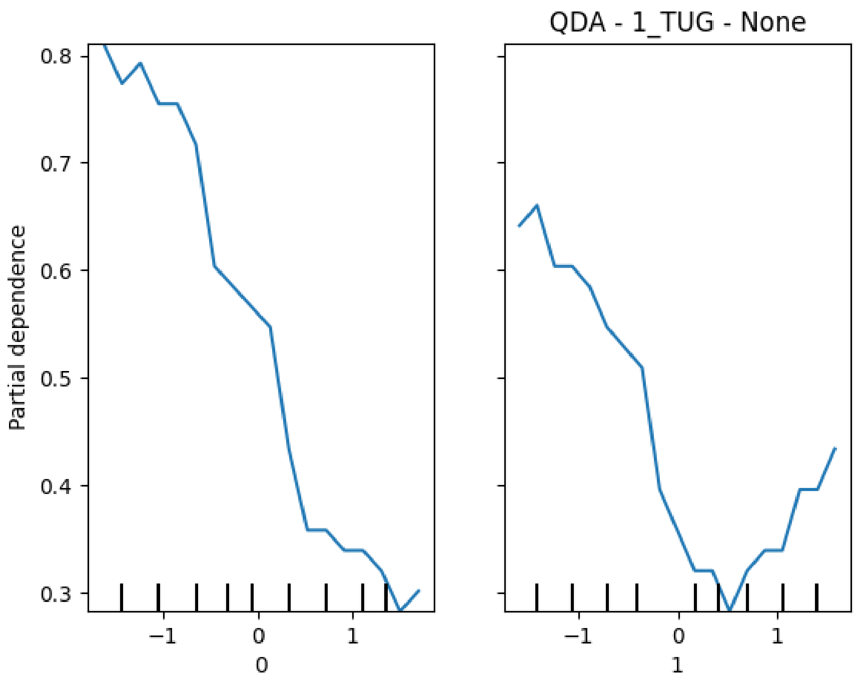

QDA - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.7
0.6
0.5
0.4
0.3
0.2
0.1
−1
0
1
0
1

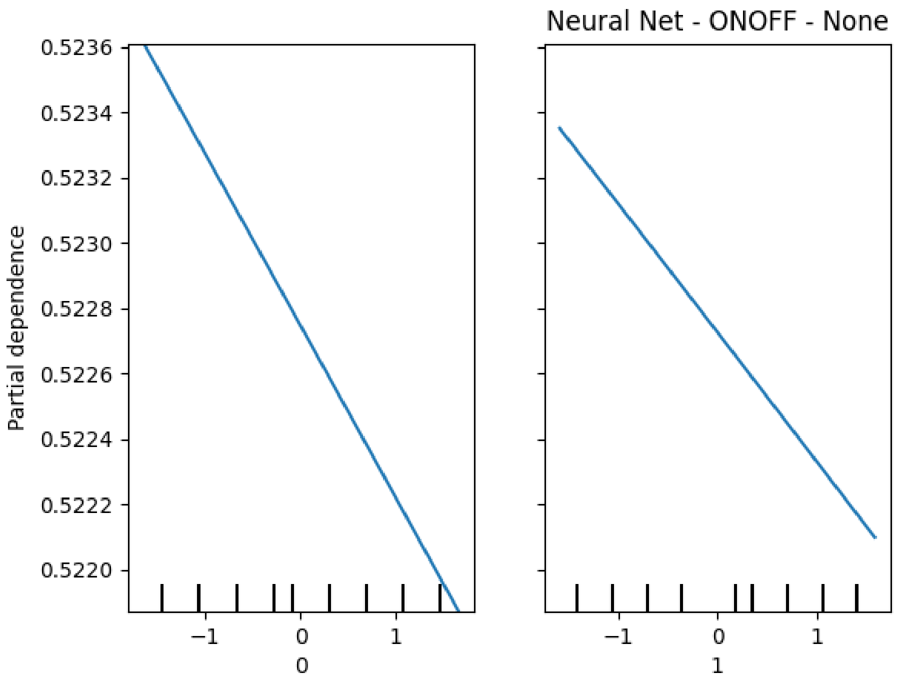

QDA - 2_CIRCUITOFISICO - None
Partial dependence
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
−1
0
1
0
1

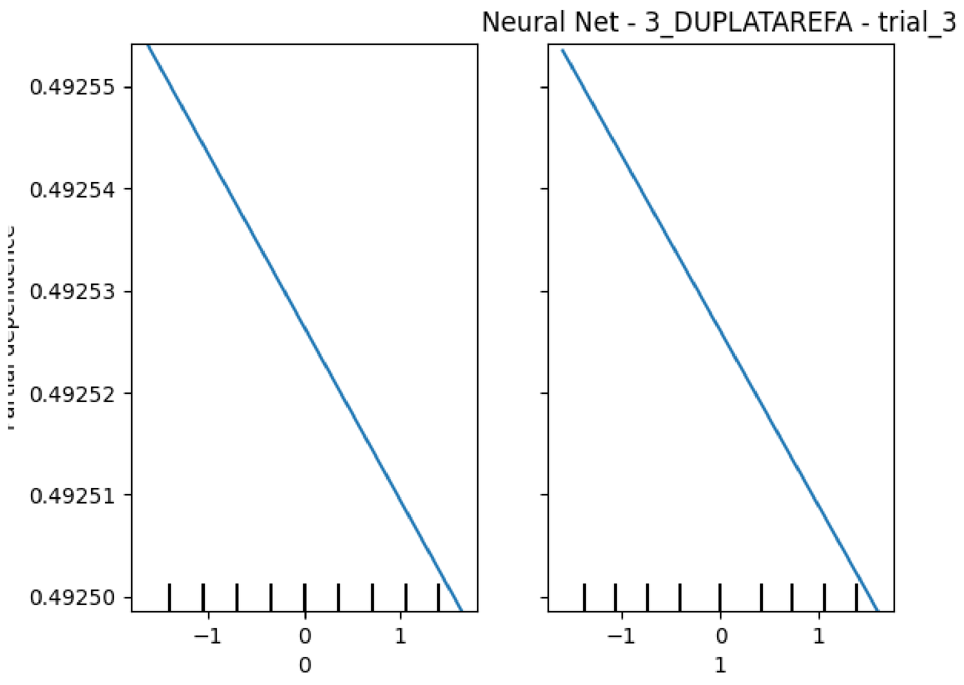

QDA - 1_TUG - trial_3
Partial dependence
1.0
0.8
0.6
0.4
0.2
−1
0
1
0
1

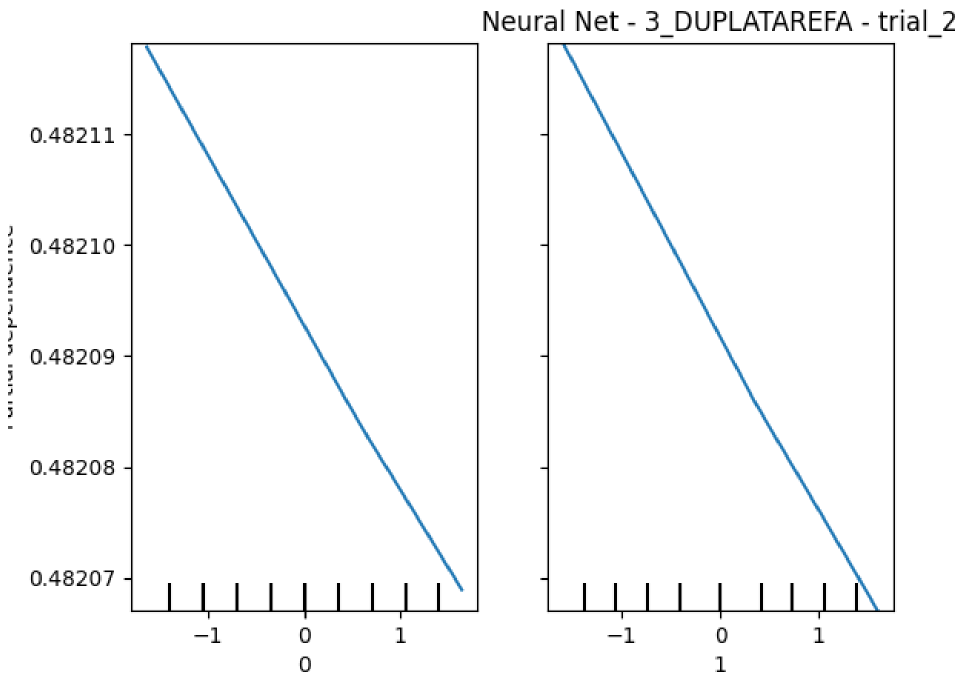

QDA - 1_TUG - trial_2
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
−1
0
1
1

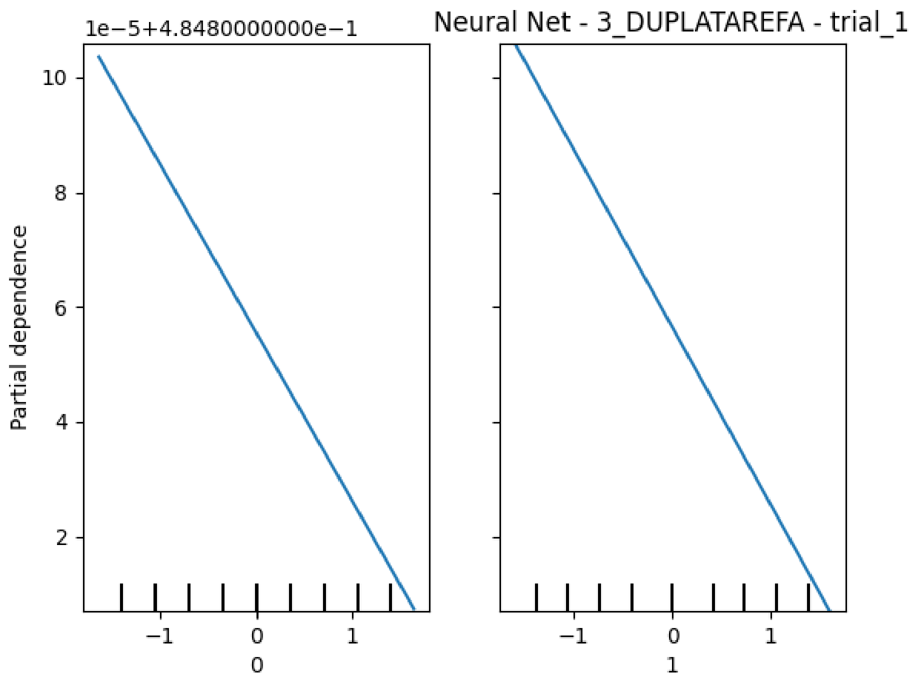

QDA - 1_TUG - trial_1
Partial dependence
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−1
0
1
0
−1
0
1
1

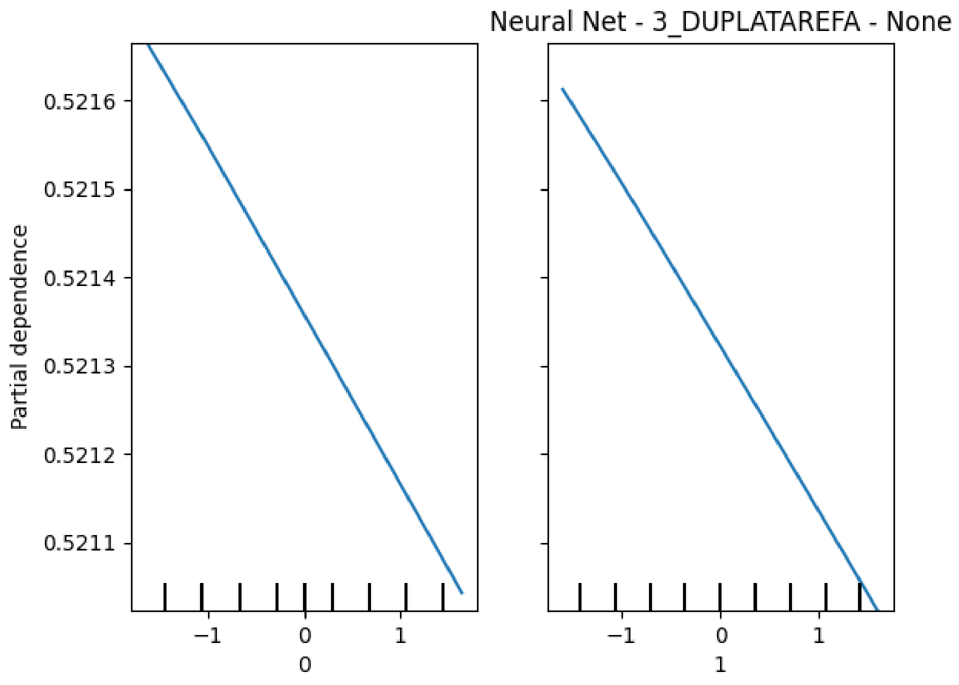

QDA - 1_TUG - None
Partial dependence
0.8
0.7
0.6
0.5
0.4
0.3
−1
0
1
0
1

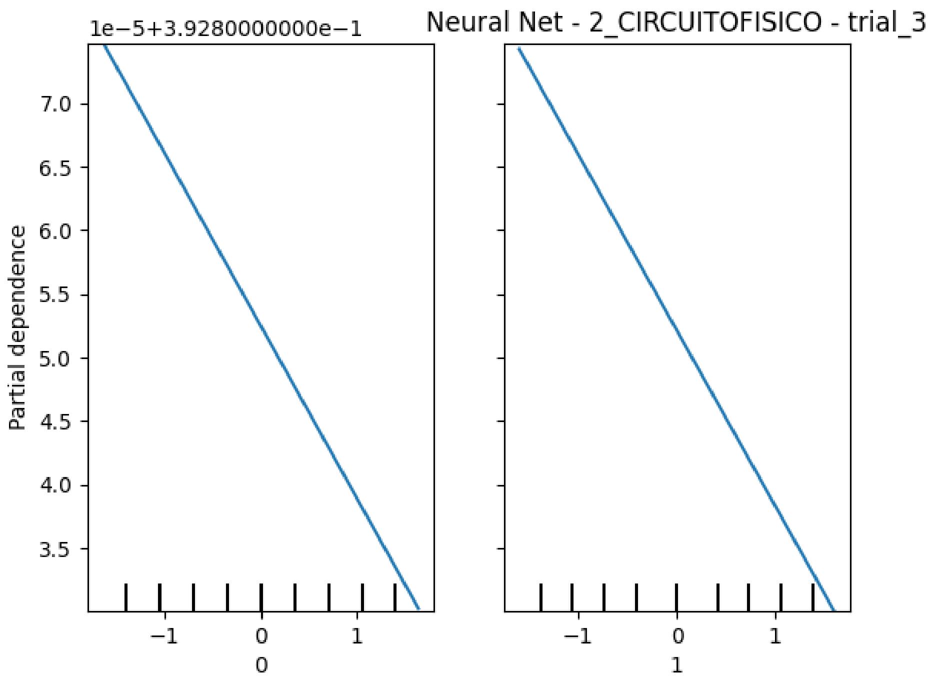

Neural Net - ONOFF - None
Partial dependence
0.5236
0.5234
0.5232
0.5230
0.5228
0.5226
0.5224
0.5222
0.5220
−1
0
1
0
1

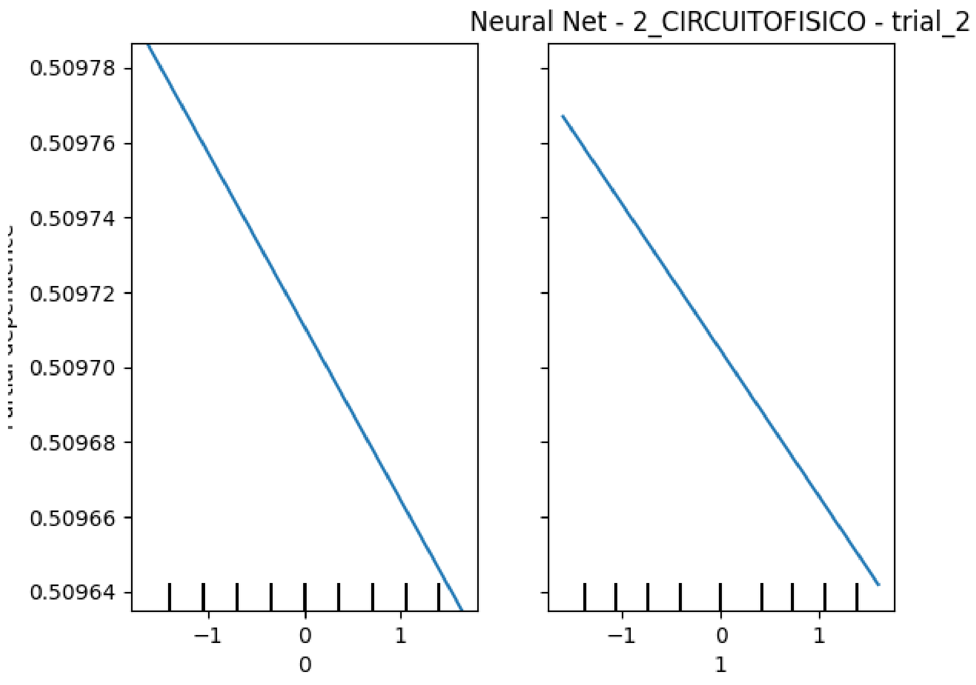

Neural Net - 3_DUPLATAREFA - trial_3
Partial dependence
0.49255
0.49254
0.49253
0.49252
0.49251
0.49250
−1
0
1
0
1

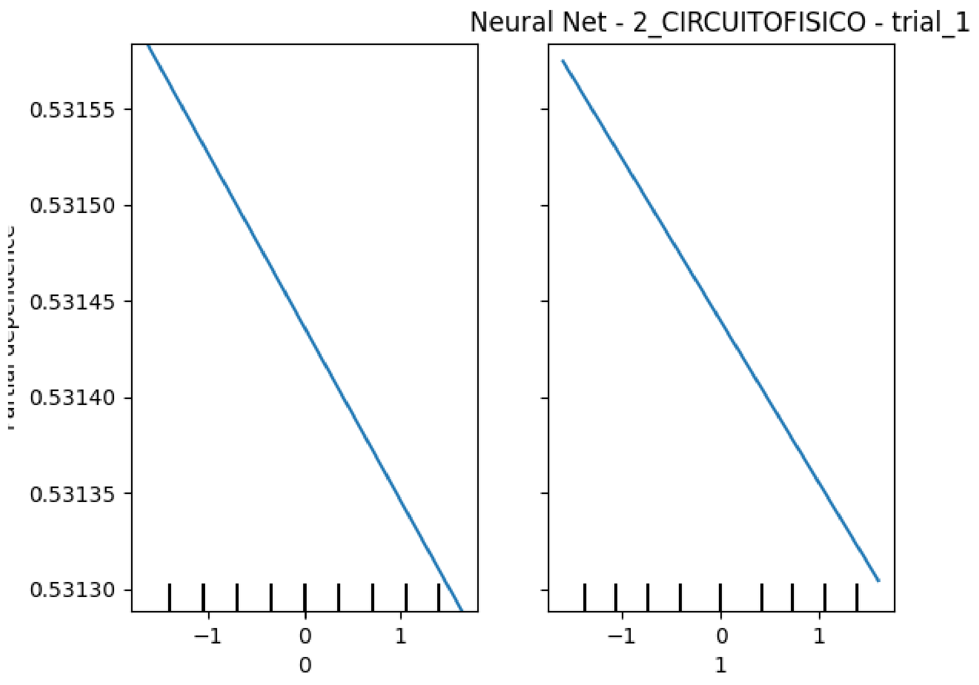

Neural Net - 3_DUPLATAREFA - trial_2
Partial dependence
0.48211
0.48210
0.48209
0.48208
0.48207
−1
0
1
0
1

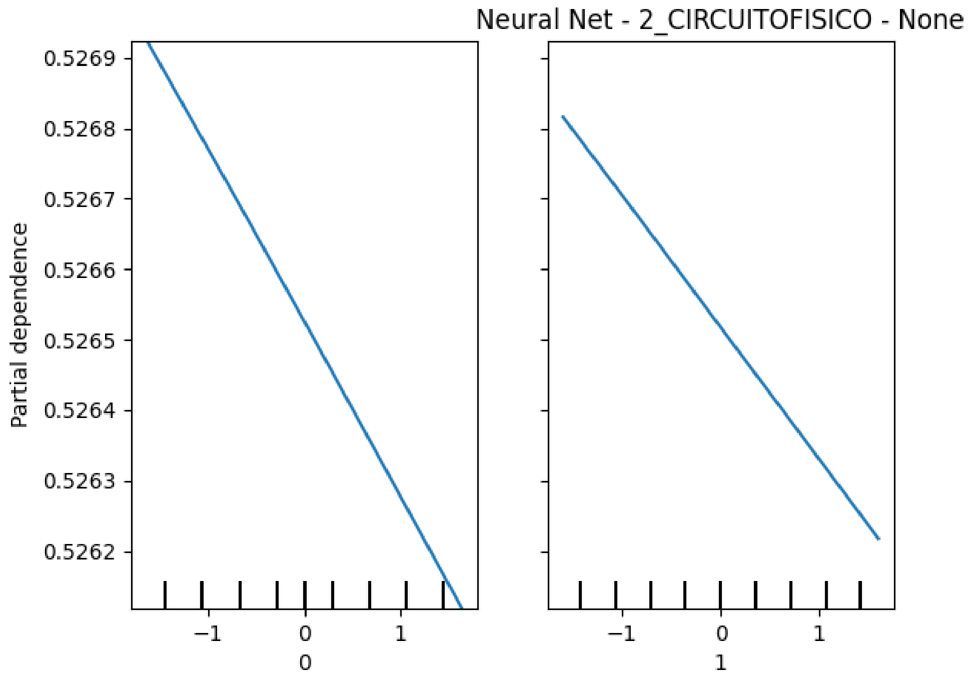

Neural Net - 3_DUPLATAREFA - trial_1
1e−5+4.8480000000e−1
Partial dependence
10
8
6
4
2
−1
0
1
0
1

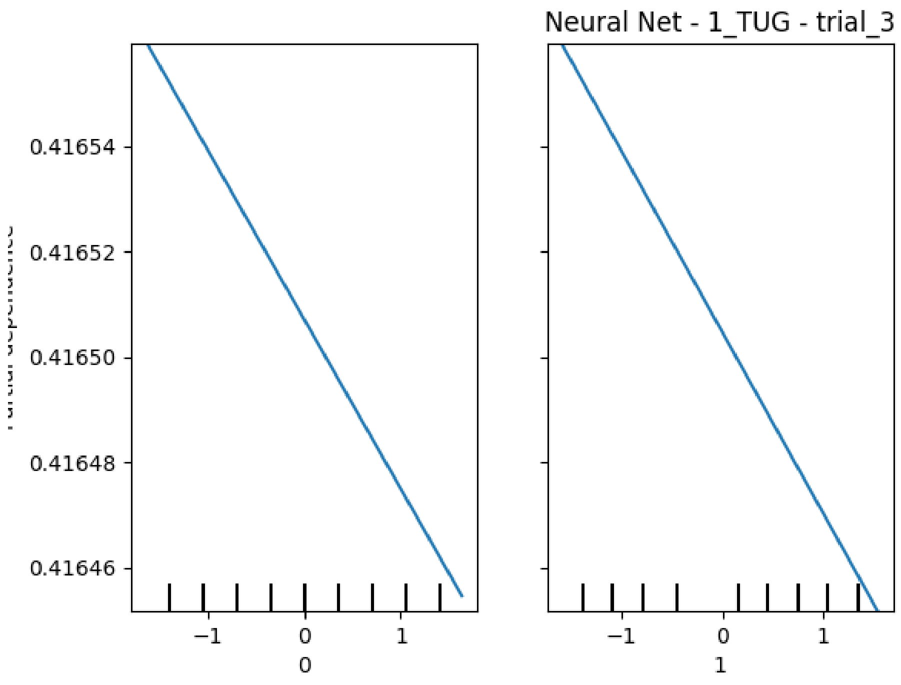

Neural Net - 3_DUPLATAREFA - None
Partial dependence
0.5216
0.5215
0.5214
0.5213
0.5212
0.5211
−1
0
1
0
1

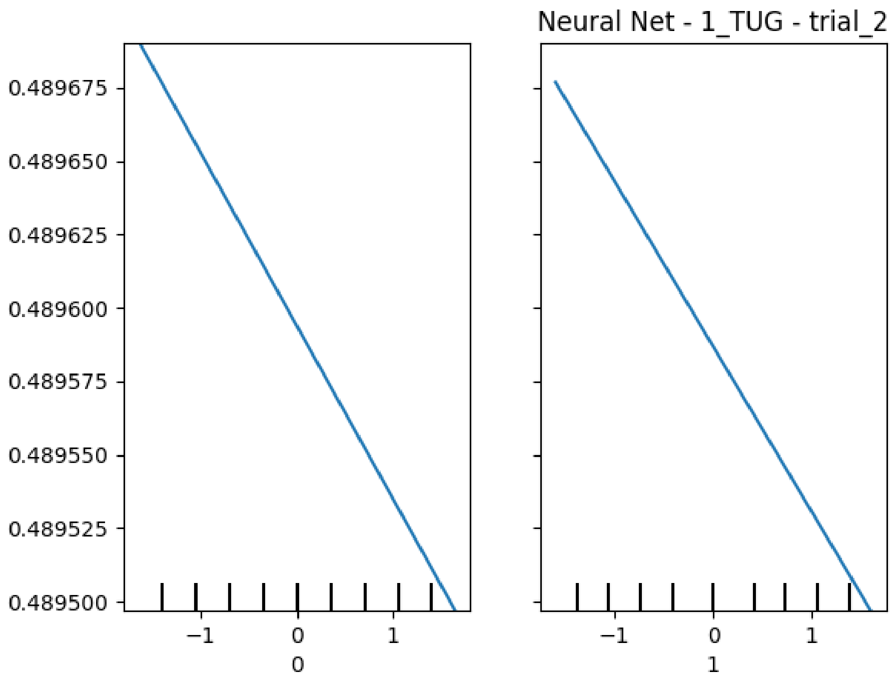

Neural Net - 2_CIRCUITOFISICO - trial_3
1e−5+3.9280000000e−1
Partial dependence
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
−1
0
1
0
1

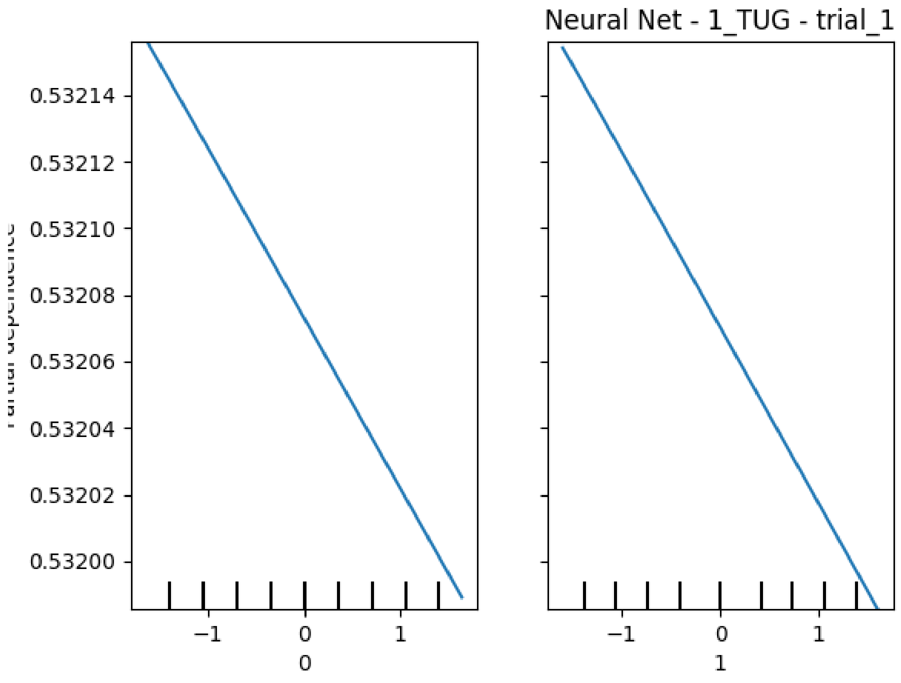

Neural Net - 2_CIRCUITOFISICO - trial_2
Partial dependence
0.50978
0.50976
0.50974
0.50972
0.50970
0.50968
0.50966
0.50964
−1
0
1
0
1

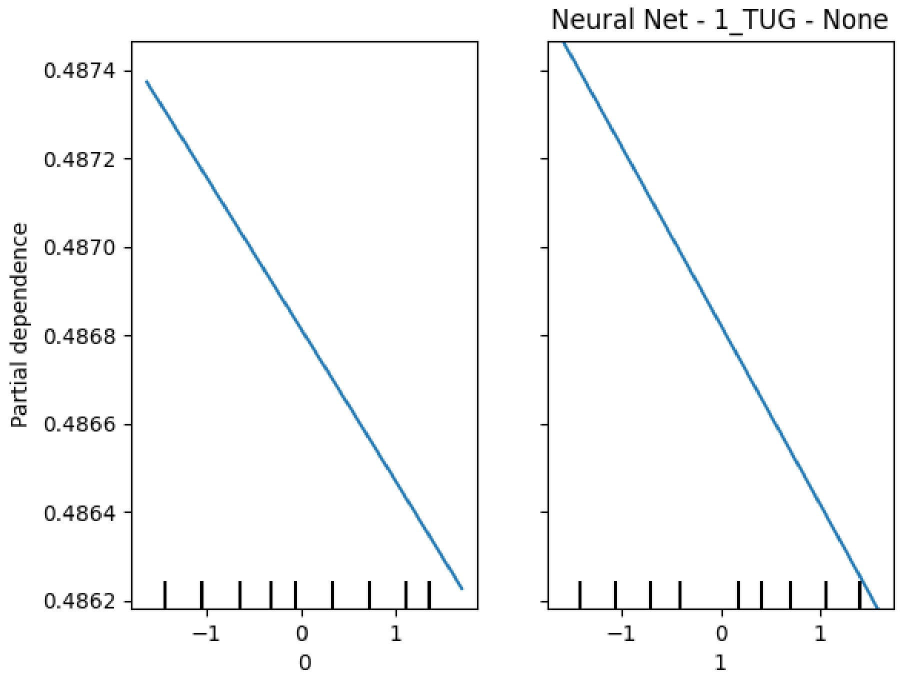

Neural Net - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.53155
0.53150
0.53145
0.53140
0.53135
0.53130
−1
0
1
0
1

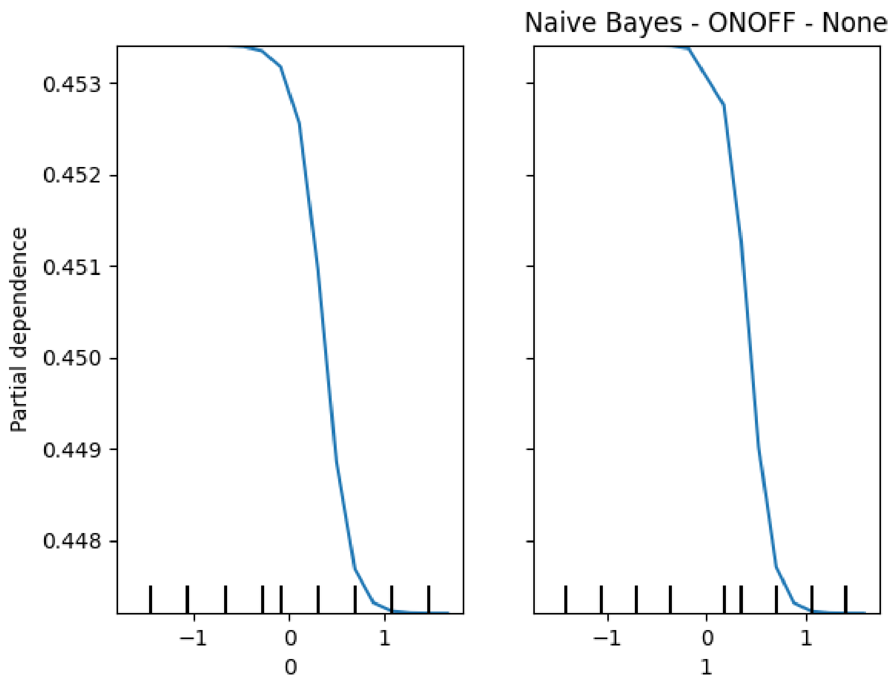

Neural Net - 2_CIRCUITOFISICO - None
Partial dependence
0.5269
0.5268
0.5267
0.5266
0.5265
0.5264
0.5263
0.5262
−1
0
1
0
1

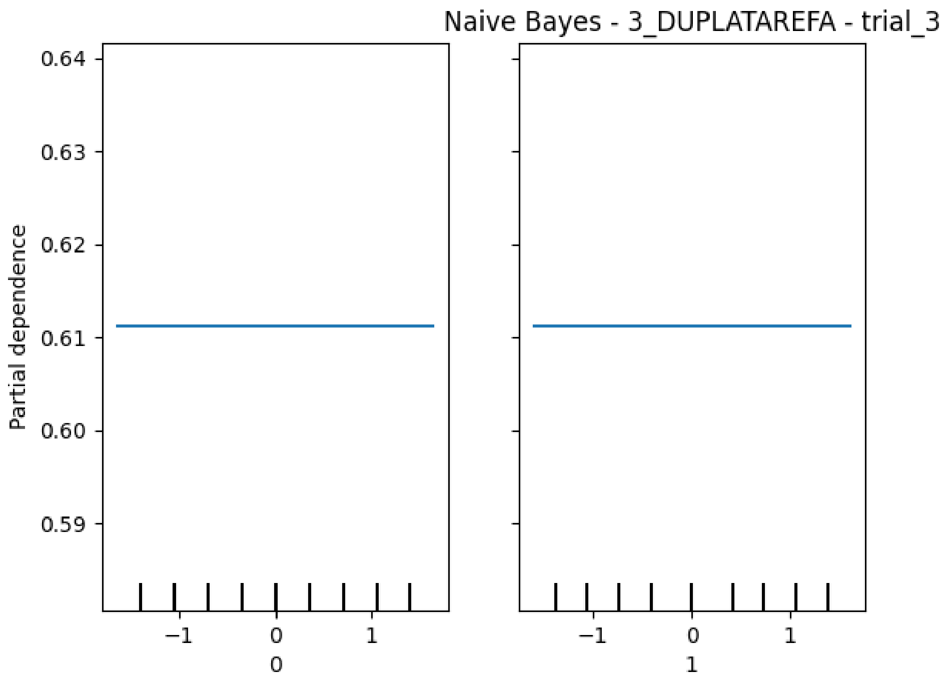

Neural Net - 1_TUG - trial_3
0.41654
0.41652
0.41650
0.41648
0.41646
−1
0
1
0
1

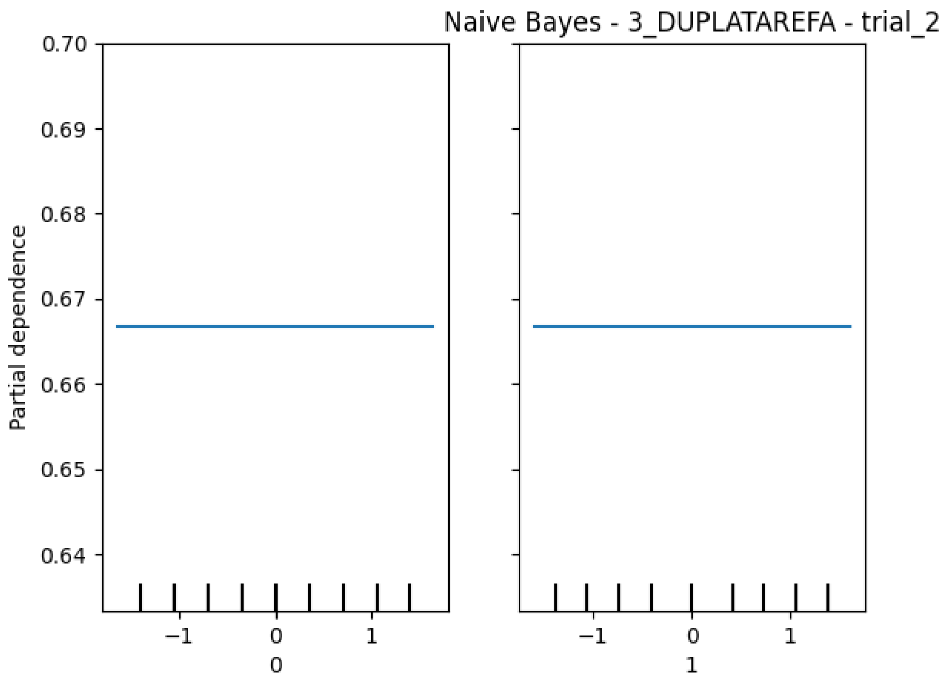

Neural Net - 1_TUG - trial_2
0.489675
0.489650
0.489625
0.489600
0.489575
0.489550
0.489525
0.489500
−1
0
1
0
1

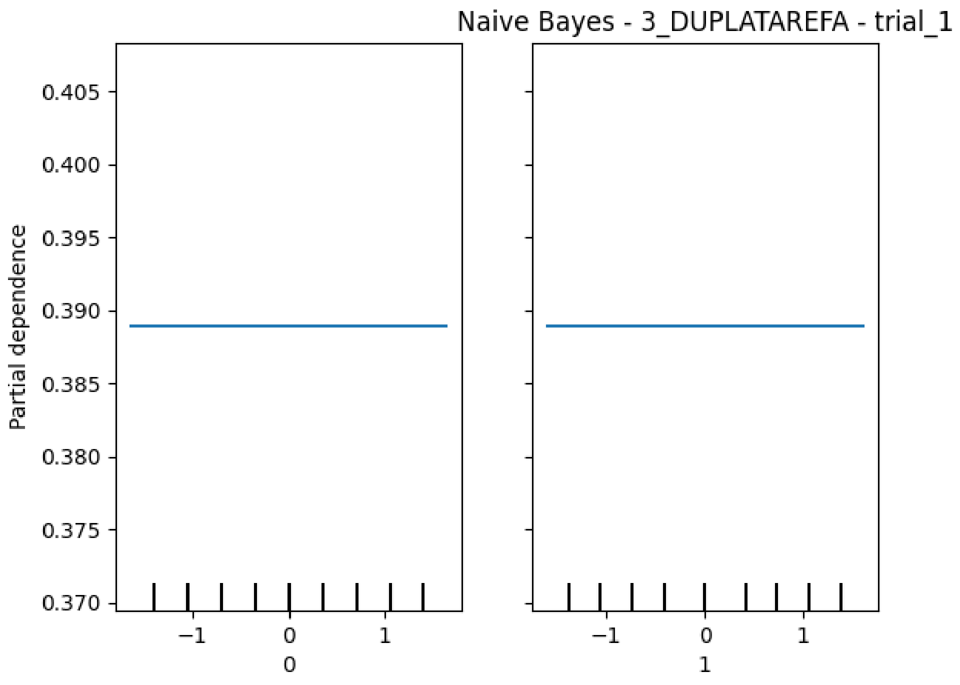

Neural Net - 1_TUG - trial_1
Partial dependence
0.53214
0.53212
0.53210
0.53208
0.53206
0.53204
0.53202
0.53200
−1
0
1
0
1

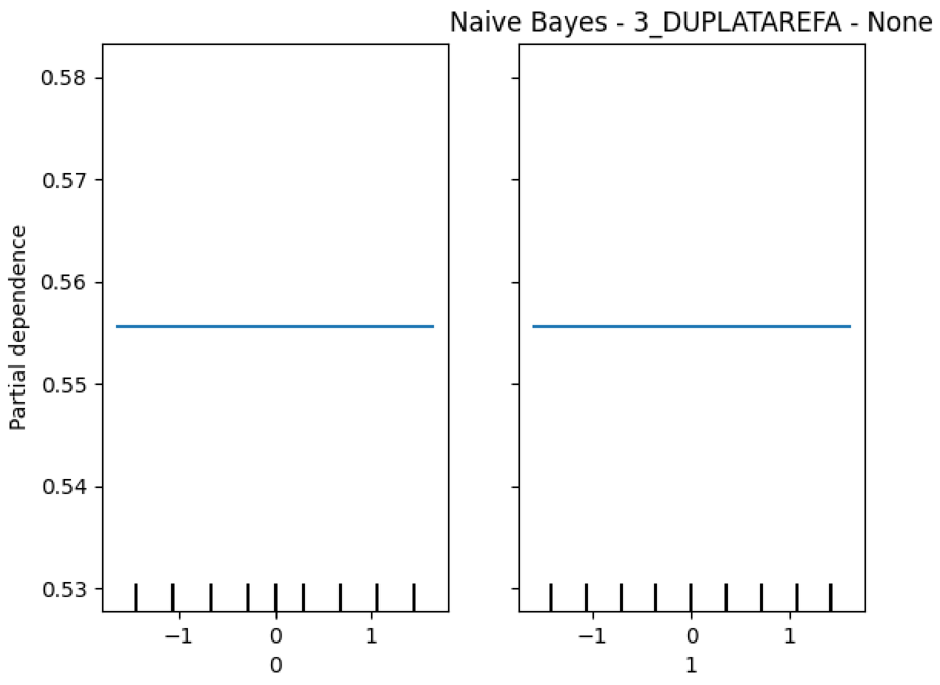

Neural Net - 1_TUG - None
Partial dependence
0.4874
0.4872
0.4870
0.4868
0.4866
0.4864
0.4862
−1
0
1
0
1

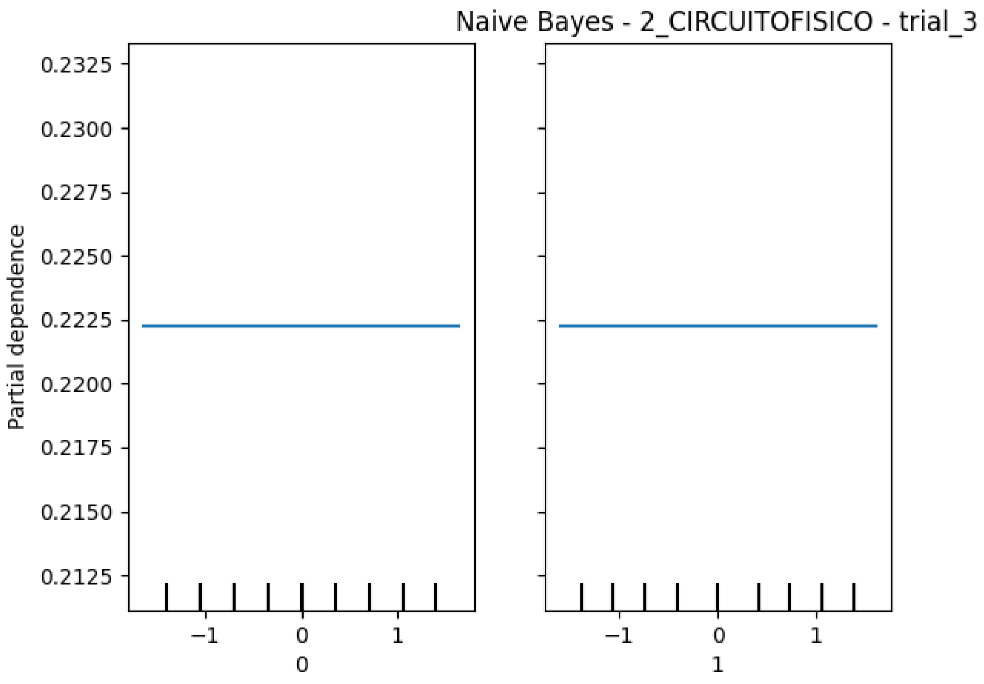

Naive Bayes - ONOFF - None
Partial dependence
0.453
0.452
0.451
0.450
0.449
0.448
−1
0
1
0
1

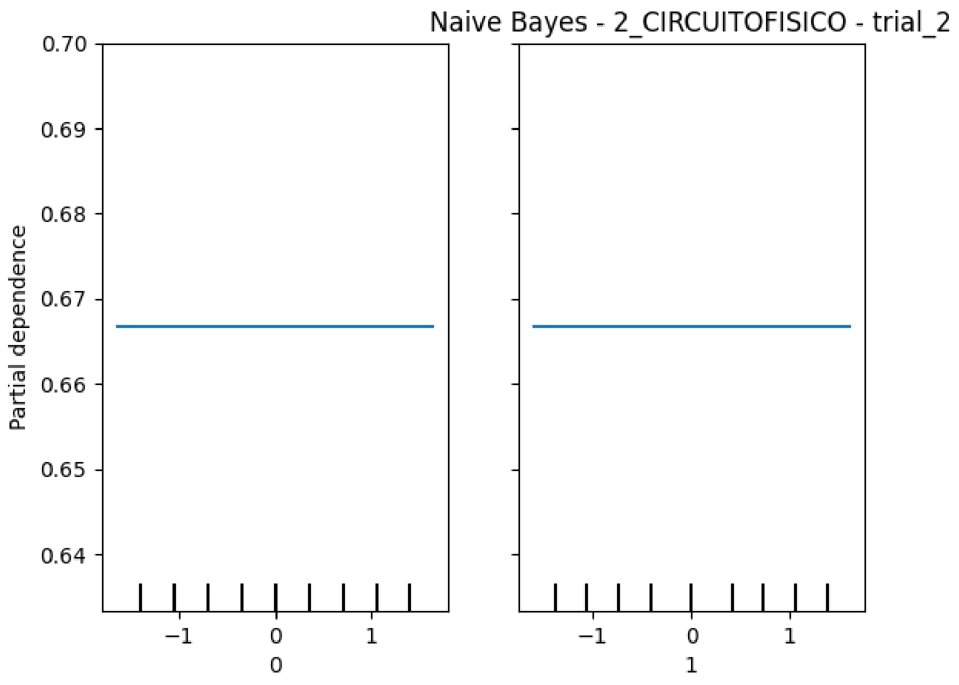

Naive Bayes - 3_DUPLATAREFA - trial_3
Partial dependence
0.64
0.63
0.62
0.61
0.60
0.59
−1
0
1
0
1

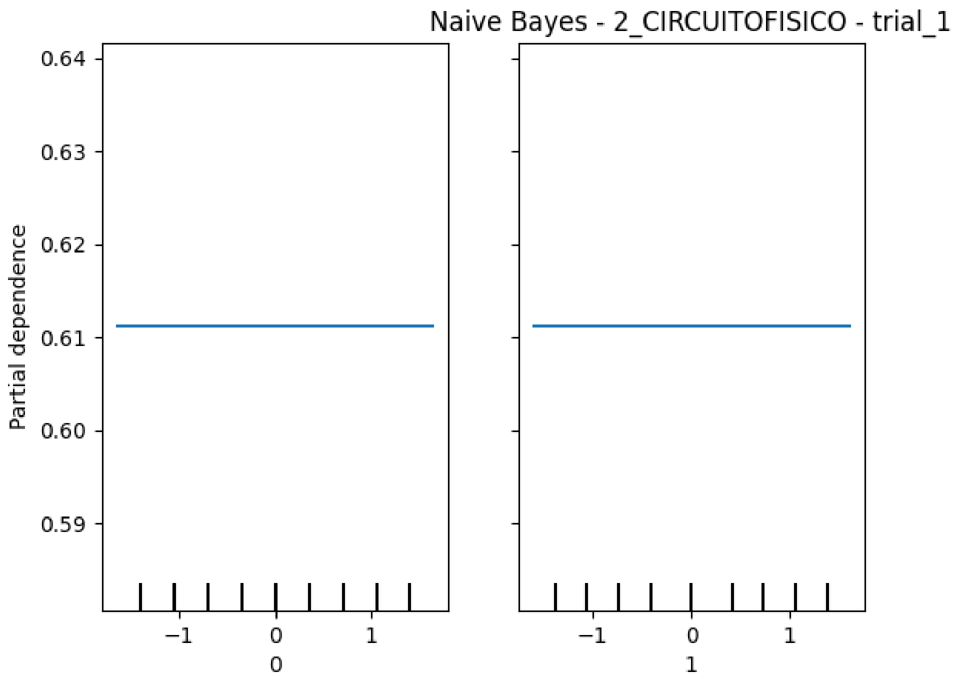

Naive Bayes - 3_DUPLATAREFA - trial_2
Partial dependence
0.70
0.69
0.68
0.67
0.66
0.65
0.64
−1
0
1
0
1

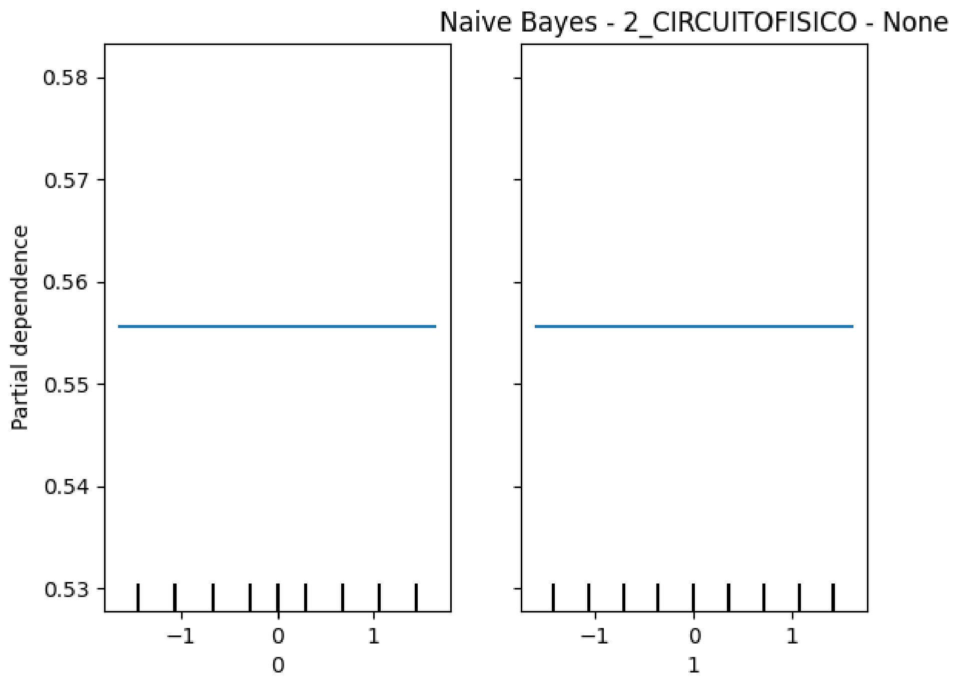

Naive Bayes - 3_DUPLATAREFA - trial_1
Partial dependence
0.405
0.400
0.395
0.390
0.385
0.380
0.375
0.370
−1
0
1
0
1

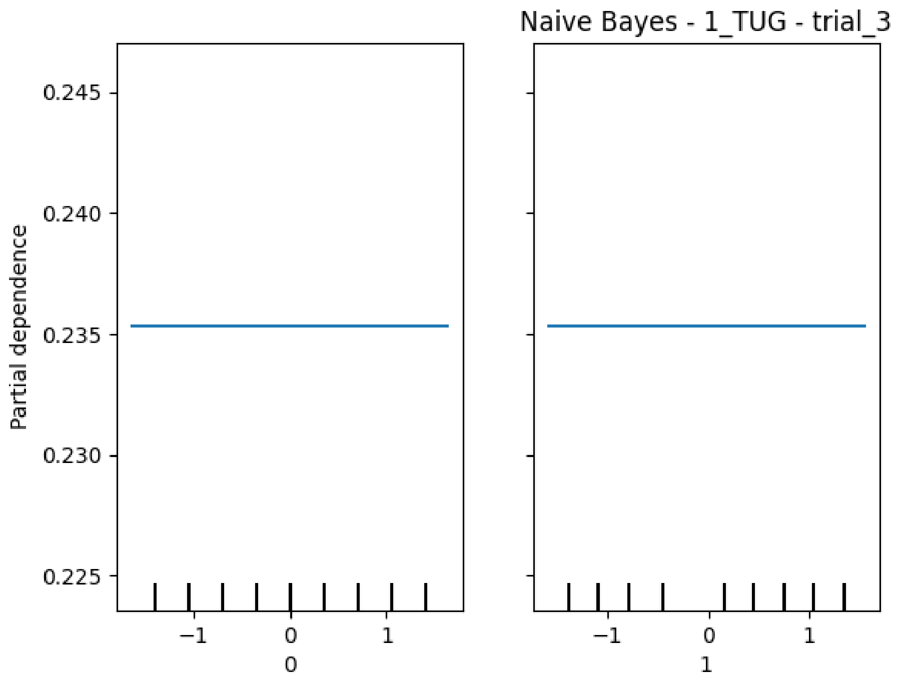

Naive Bayes - 3_DUPLATAREFA - None
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

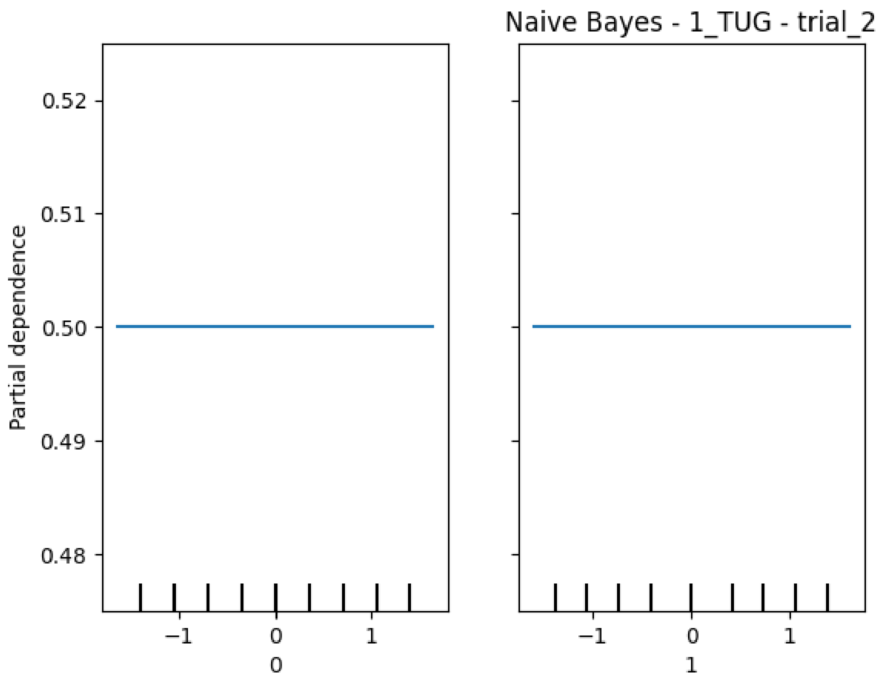

Naive Bayes - 2_CIRCUITOFISICO - trial_3
Partial dependence
0.2325
0.2300
0.2275
0.2250
0.2225
0.2200
0.2175
0.2150
0.2125
−1
0
1
0
1

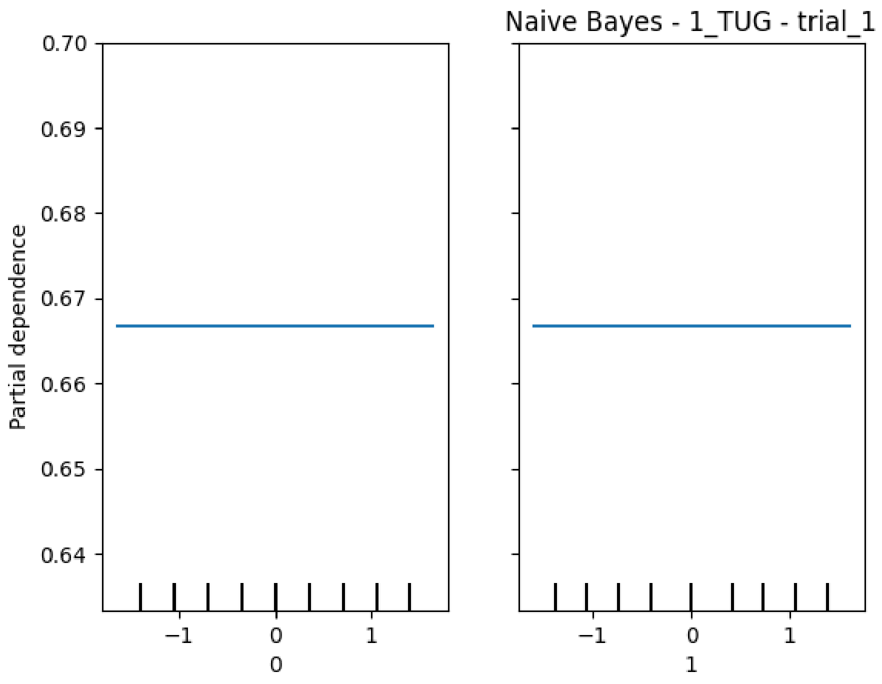

Naive Bayes - 2_CIRCUITOFISICO - trial_2
Partial dependence
0.70
0.69
0.68
0.67
0.66
0.65
0.64
−1
0
1
0
1

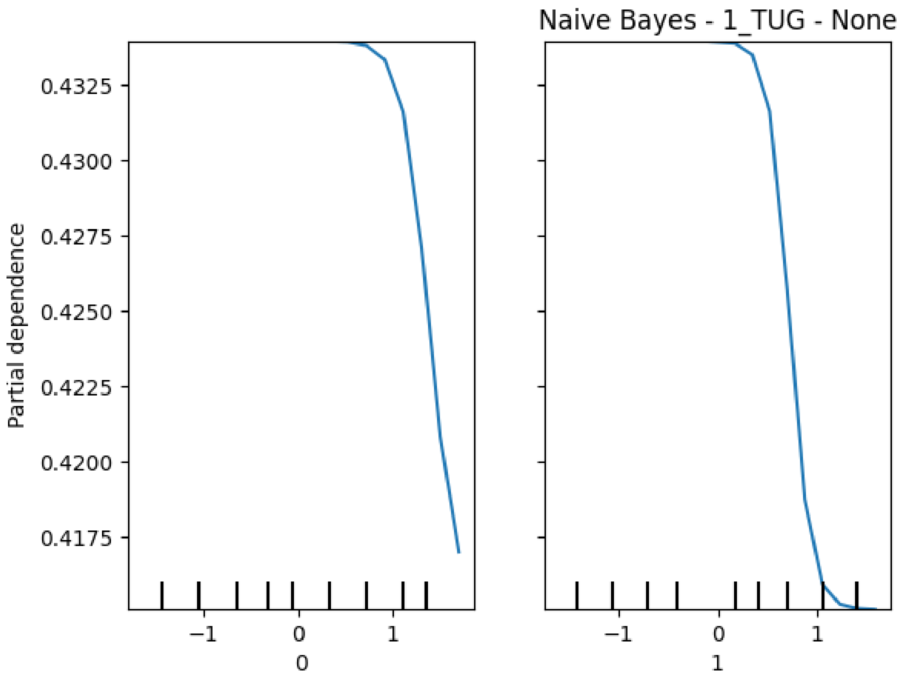

Naive Bayes - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.64
0.63
0.62
0.61
0.60
0.59
−1
0
1
0
1

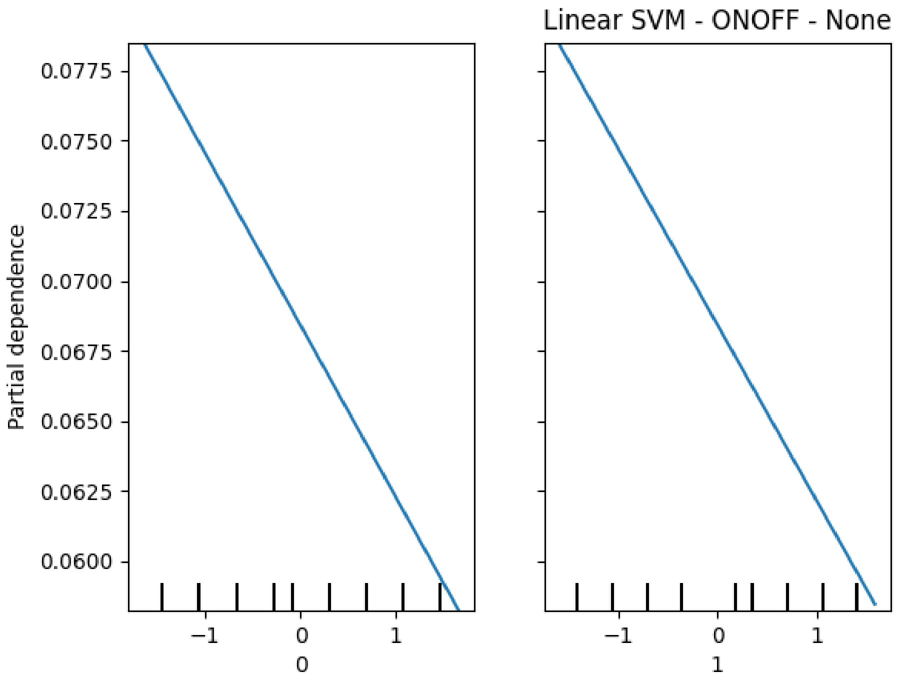

Naive Bayes - 2_CIRCUITOFISICO - None
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

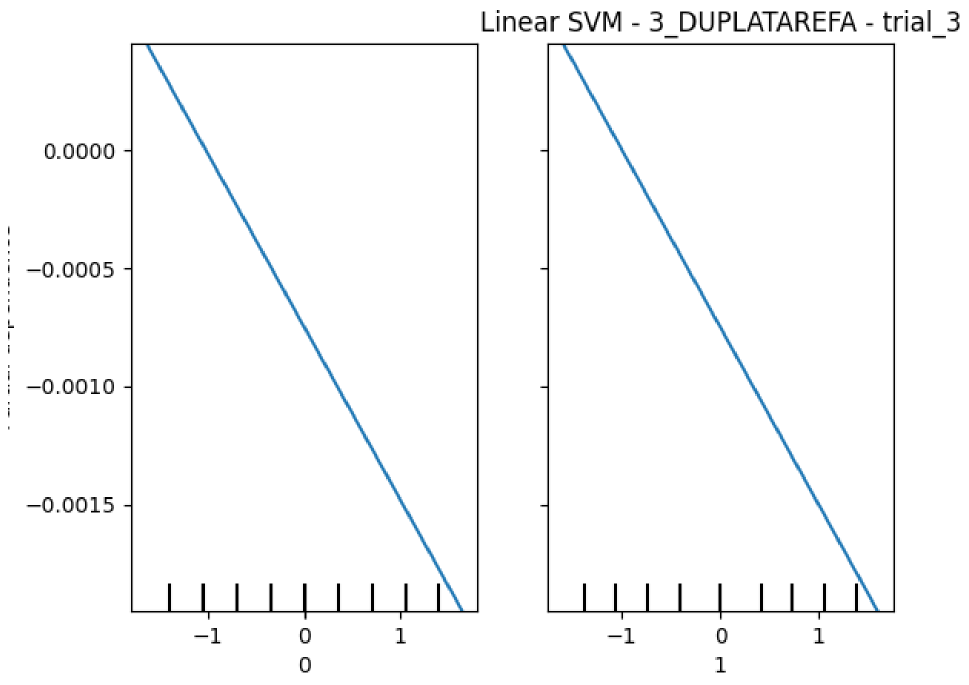

Naive Bayes - 1_TUG - trial_3
Partial dependence
0.245
0.240
0.235
0.230
0.225
−1
0
1
0
1

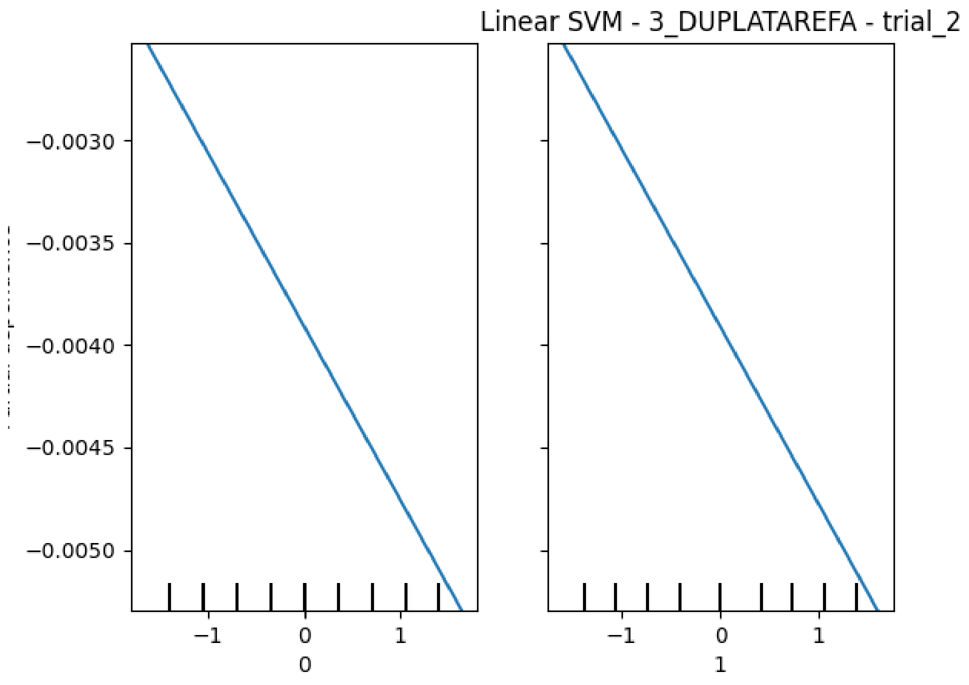

Naive Bayes - 1_TUG - trial_2
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

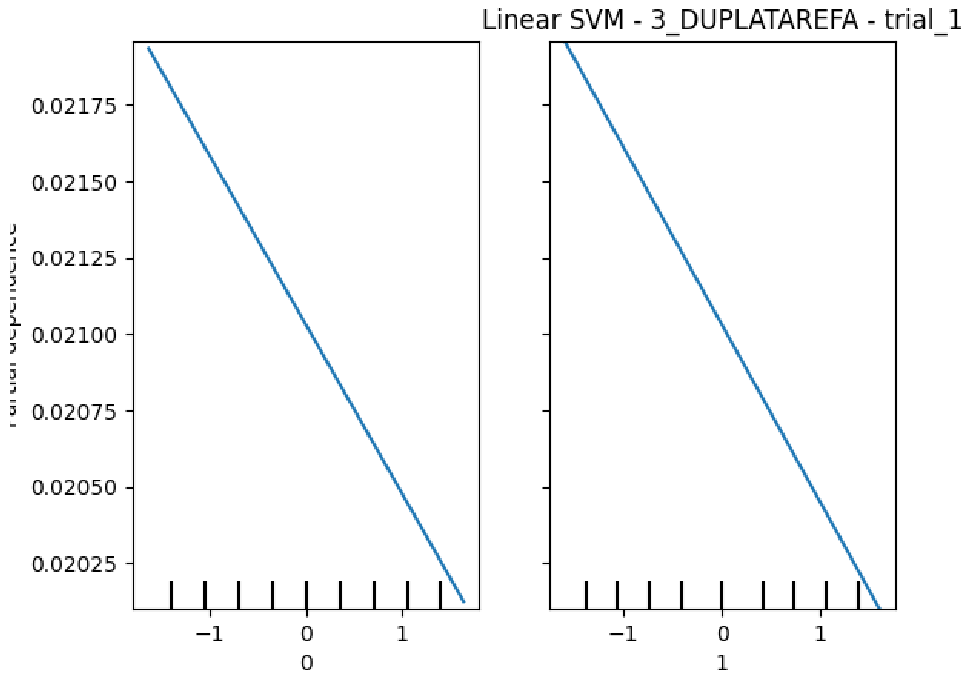

Naive Bayes - 1_TUG - trial_1
Partial dependence
0.70
0.69
0.68
0.67
0.66
0.65
0.64
−1
0
1
0
1

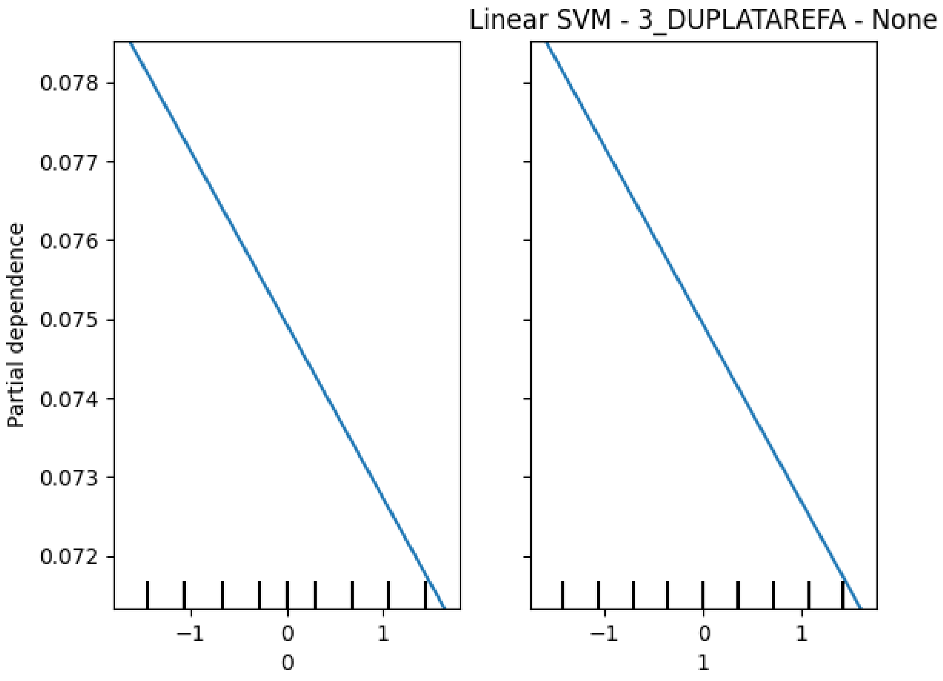

Naive Bayes - 1_TUG - None
Partial dependence
0.4325
0.4300
0.4275
0.4250
0.4225
0.4200
0.4175
−1
0
1
0
1

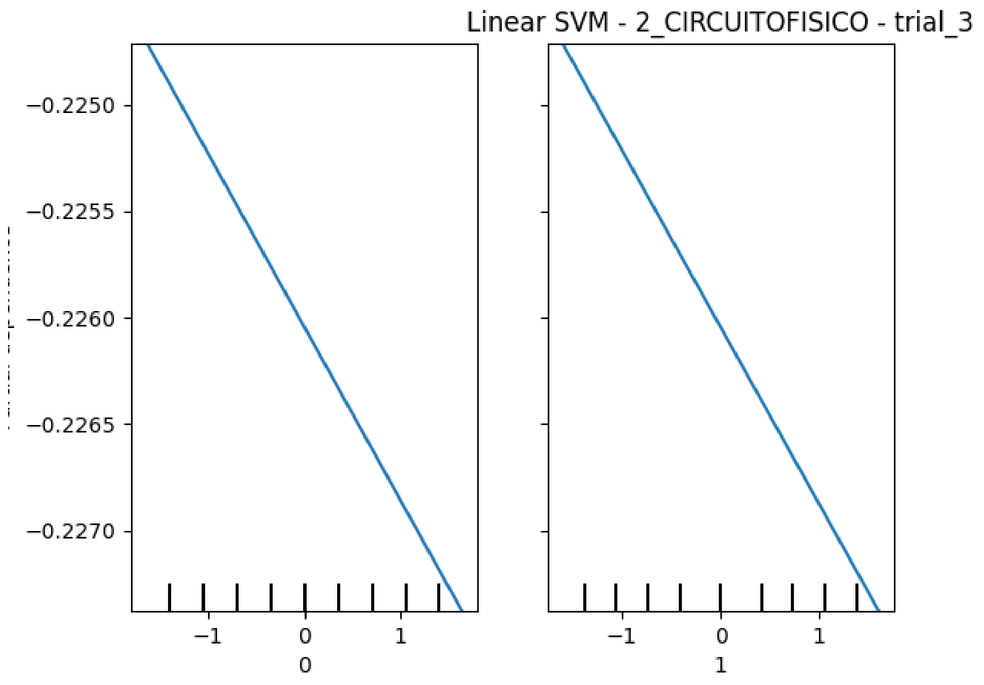

Linear SVM - ONOFF - None
Partial dependence
0.0775
0.0750
0.0725
0.0700
0.0675
0.0650
0.0625
0.0600
−1
0
1
0
1

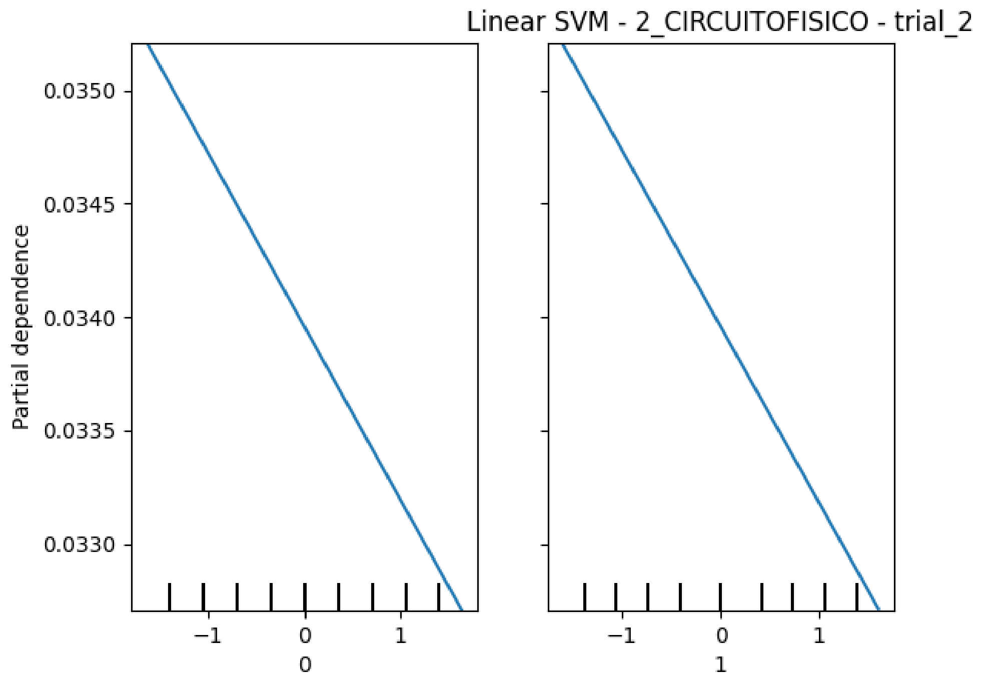

Linear SVM - 3_DUPLATAREFA - trial_3
0.0000
−0.0005
−0.0010
−0.0015
−1
0
1
0
−1
0
1
1

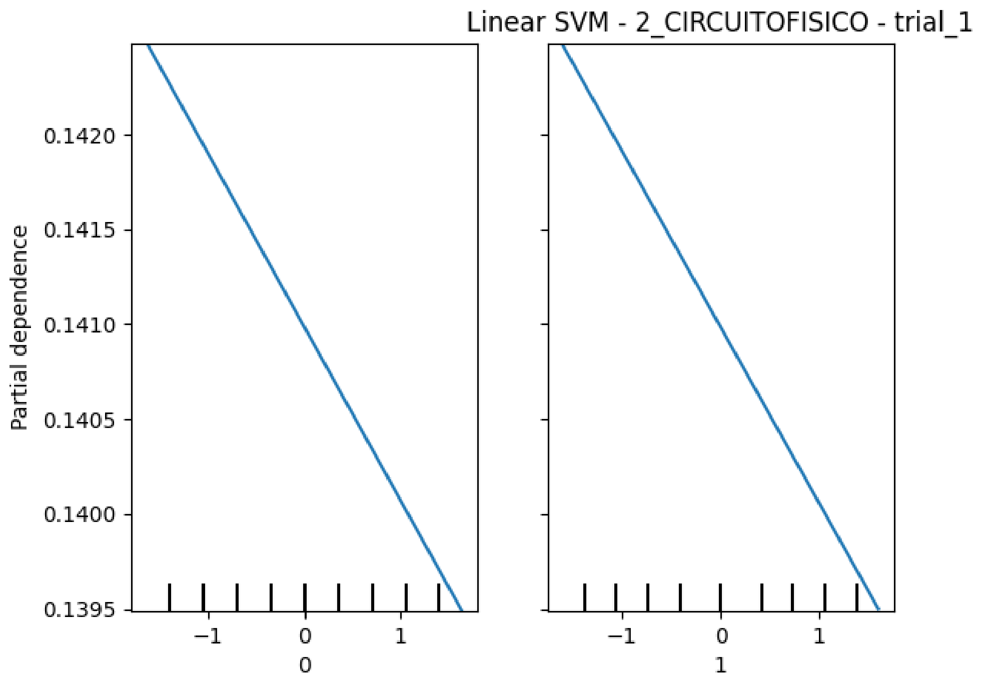

Linear SVM - 3_DUPLATAREFA - trial_2
−0.0030
−0.0035
−0.0040
−0.0045
−0.0050
−1
0
1
0
−1
0
1
1

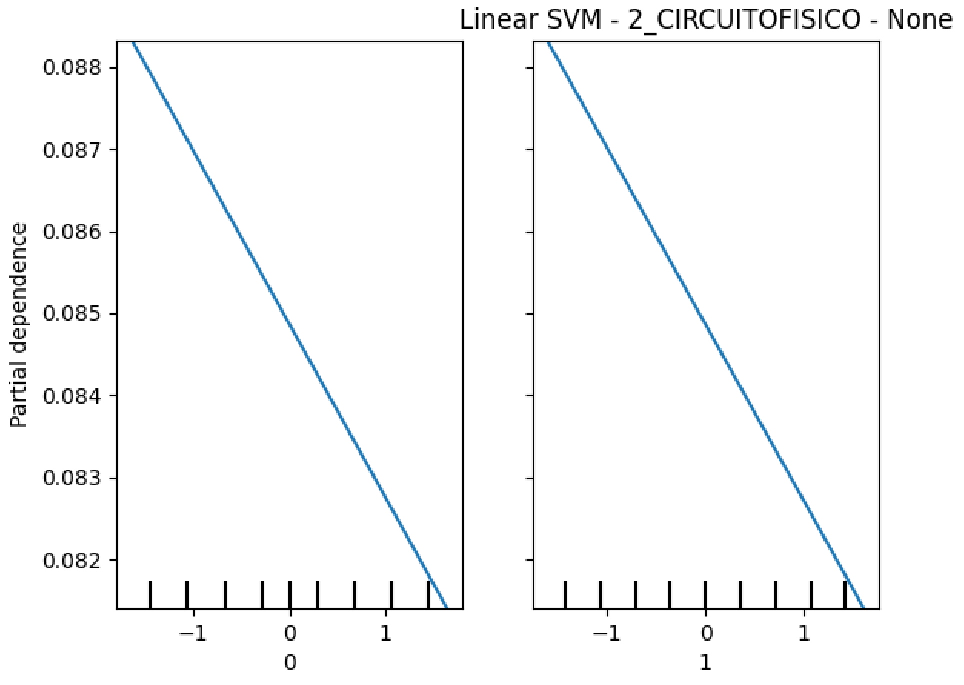

Linear SVM - 3_DUPLATAREFA - trial_1
Partial dependence
0.02175
0.02150
0.02125
0.02100
0.02075
0.02050
0.02025
−1
0
1
0
1

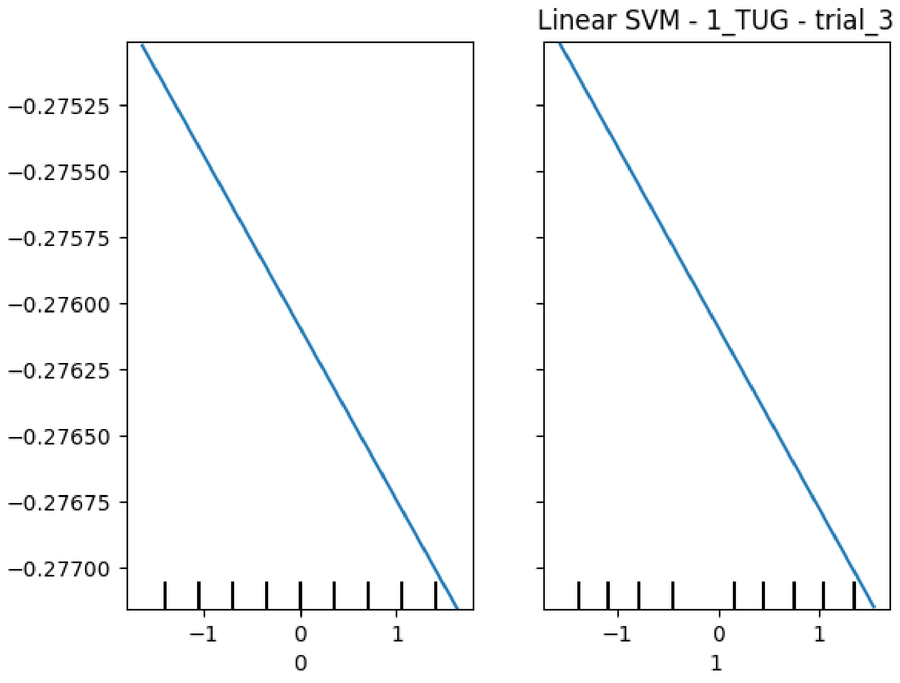

Linear SVM - 3_DUPLATAREFA - None
Partial dependence
0.078
0.077
0.076
0.075
0.074
0.073
0.072
−1
0
1
0
1

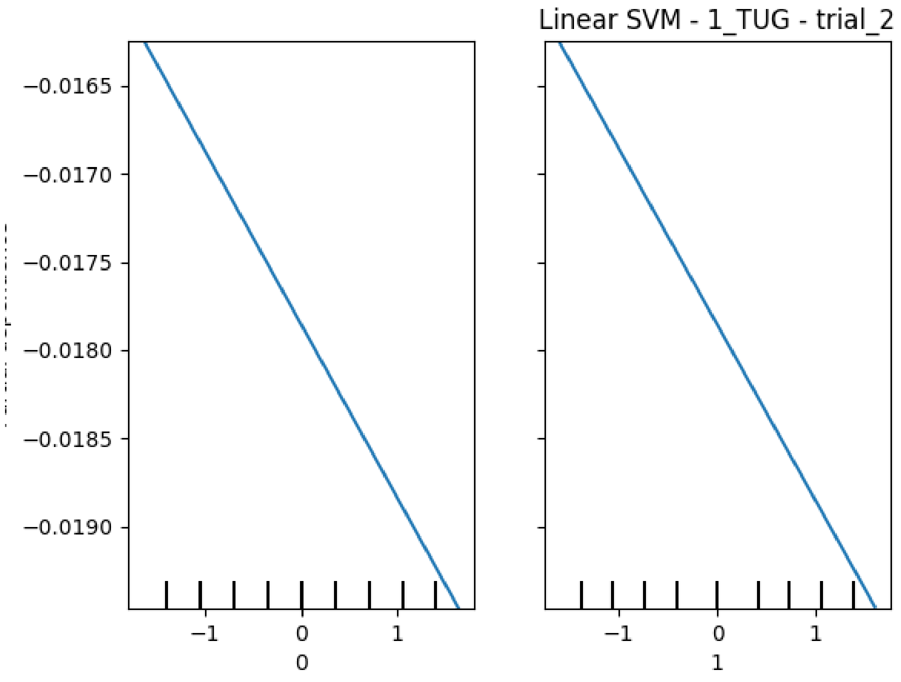

Linear SVM - 2_CIRCUITOFISICO - trial_3
−0.2250
−0.2255
−0.2260
−0.2265
−0.2270
−1
0
1
0
1

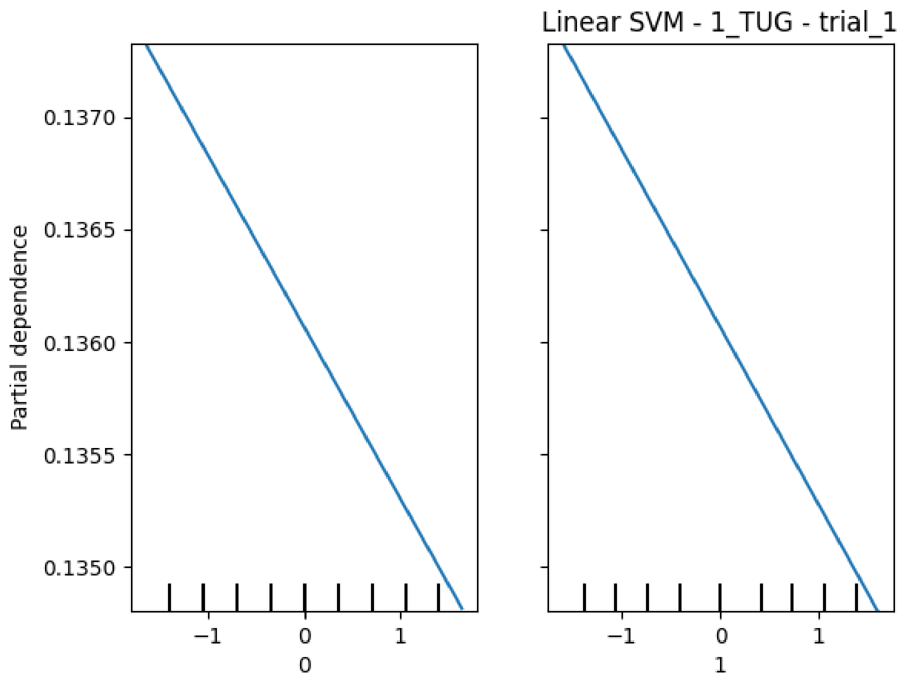

Linear SVM - 2_CIRCUITOFISICO - trial_2
Partial dependence
0.0350
0.0345
0.0340
0.0335
0.0330
−1
0
1
0
1

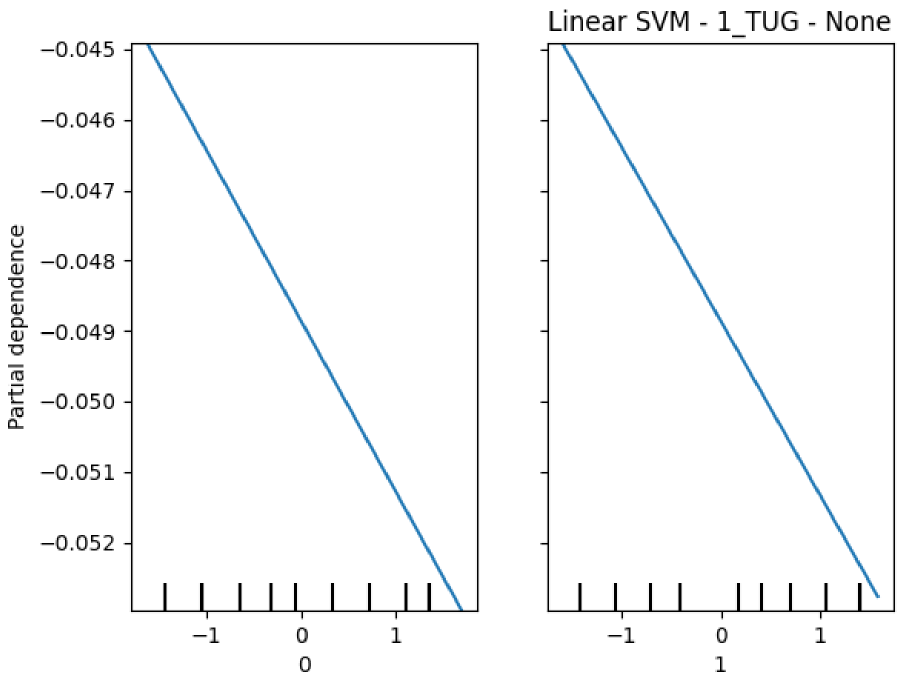

Linear SVM - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.1420
0.1415
0.1410
0.1405
0.1400
0.1395
−1
0
1
0
1

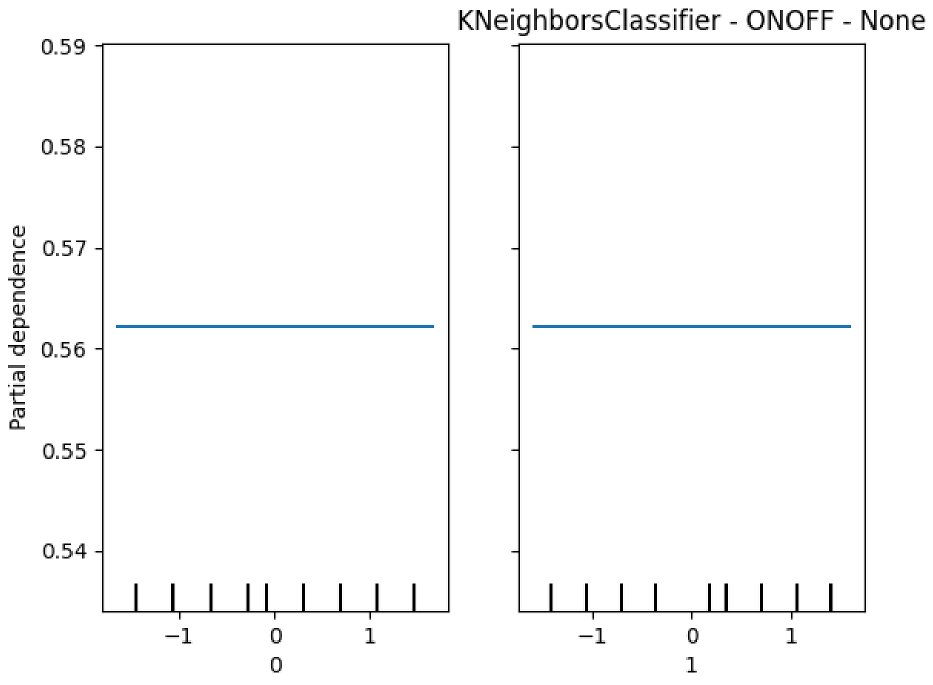

Linear SVM - 2_CIRCUITOFISICO - None
Partial dependence
0.088
0.087
0.086
0.085
0.084
0.083
0.082
−1
0
1
0
1

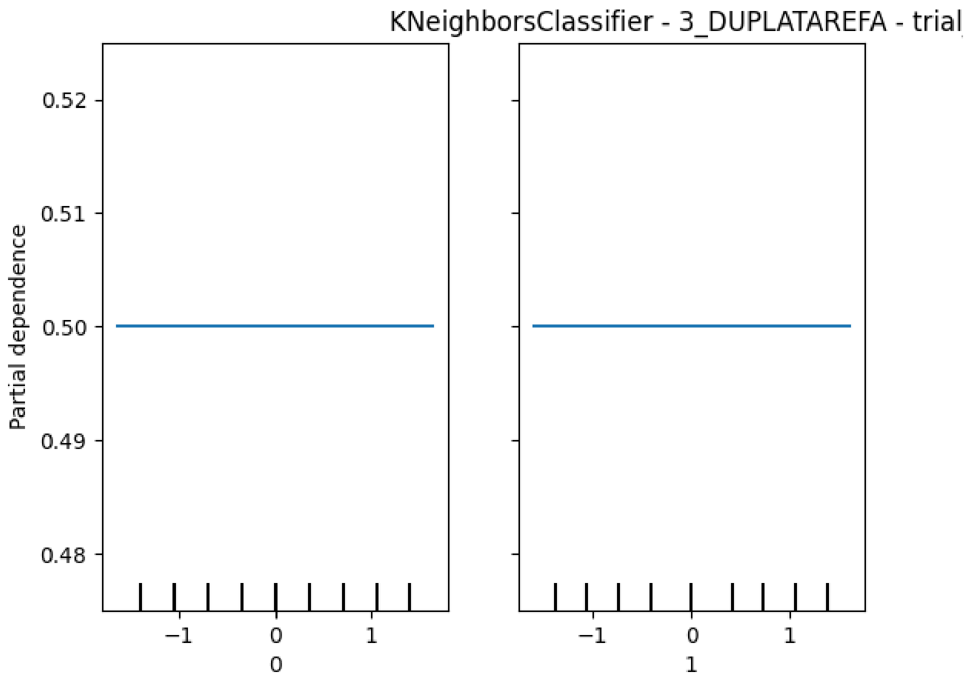

Linear SVM - 1_TUG - trial_3
−0.27525
−0.27550
−0.27575
−0.27600
−0.27625
−0.27650
−0.27675
−0.27700
−1
0
1
0
1

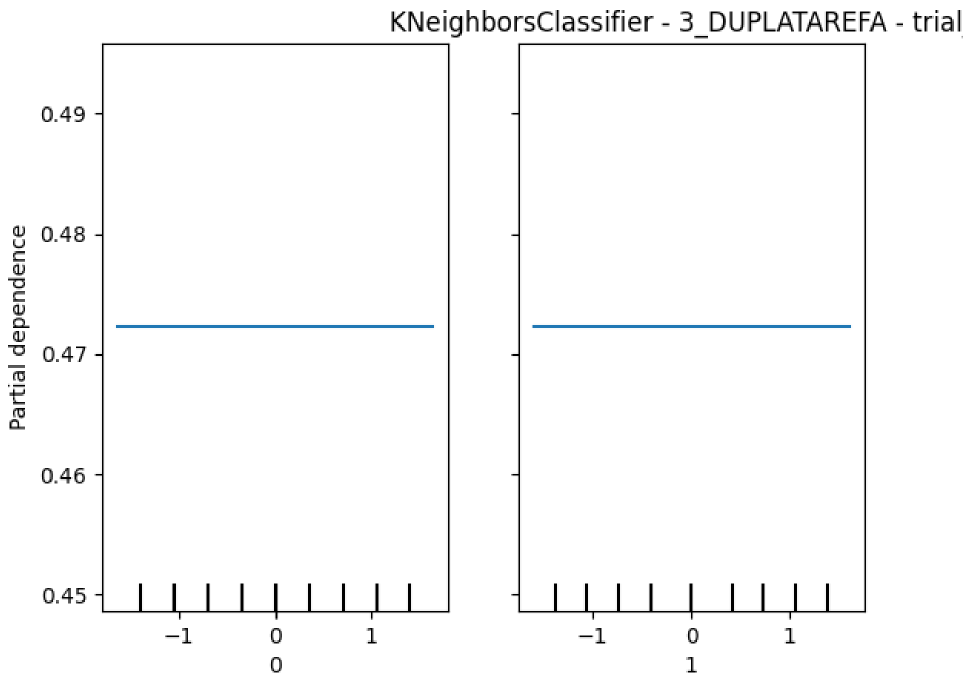

Linear SVM - 1_TUG - trial_2
−0.0165
−0.0170
−0.0175
−0.0180
−0.0185
−0.0190
−1
0
1
0
1

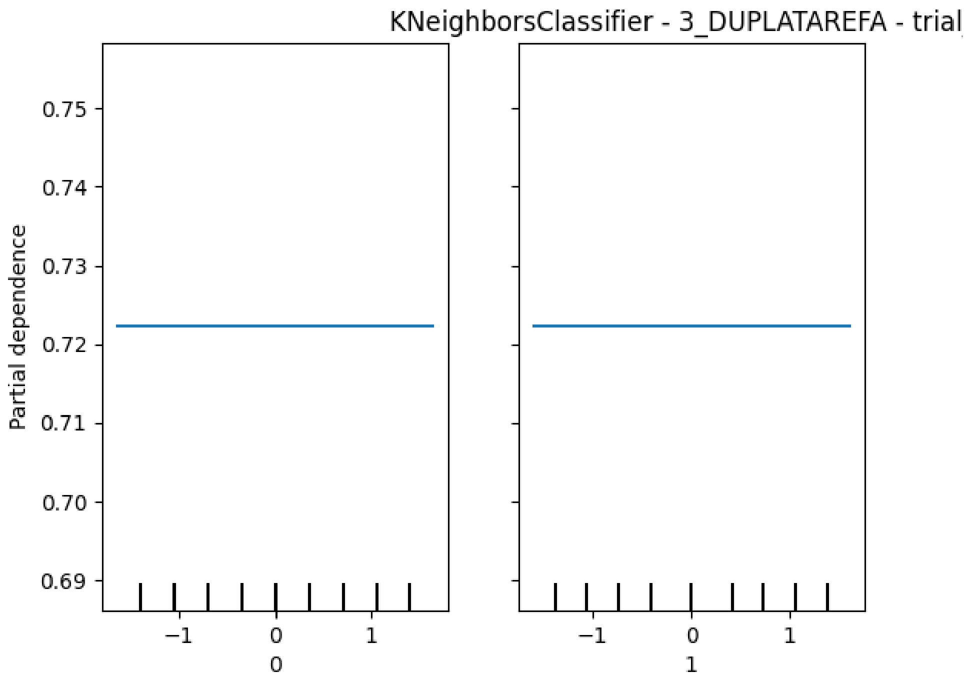

Linear SVM - 1_TUG - trial_1
Partial dependence
0.1370
0.1365
0.1360
0.1355
0.1350
−1
0
1
0
1

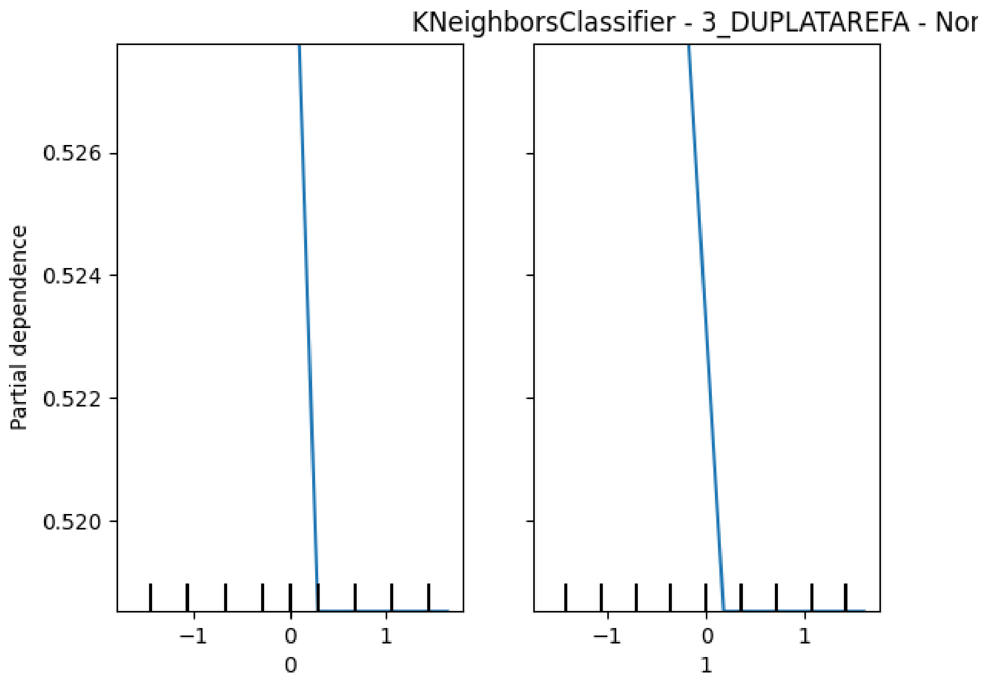

Linear SVM - 1_TUG - None
Partial dependence
−0.045
−0.046
−0.047
−0.048
−0.049
−0.050
−0.051
−0.052
−1
0
1
0
1

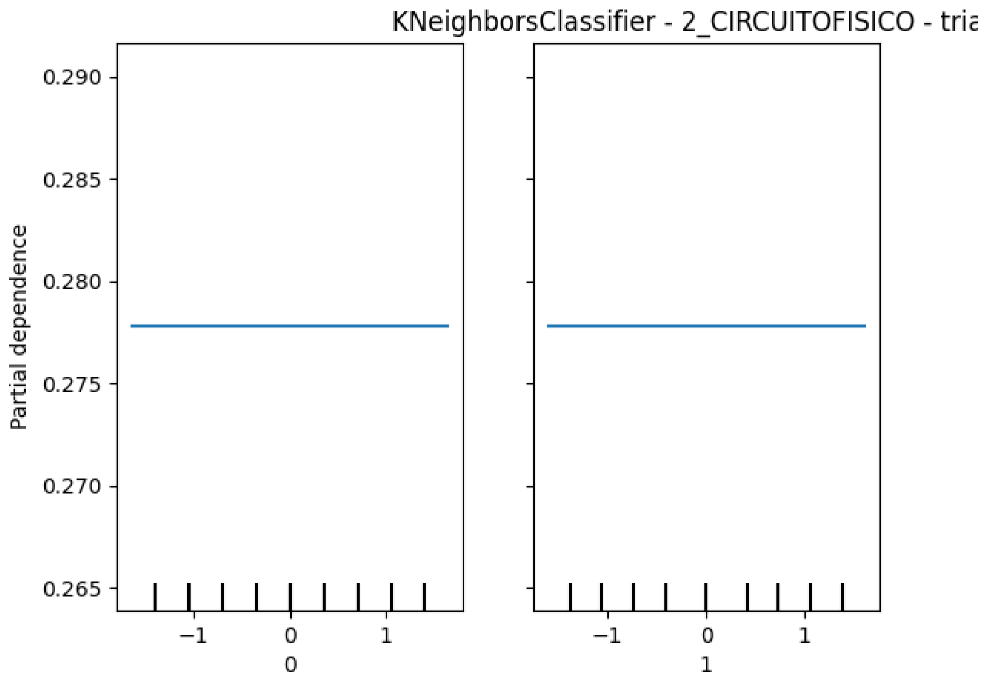

KNeighborsClassifier - ONOFF - None
Partial dependence
0.59
0.58
0.57
0.56
0.55
0.54
−1
0
1
0
1

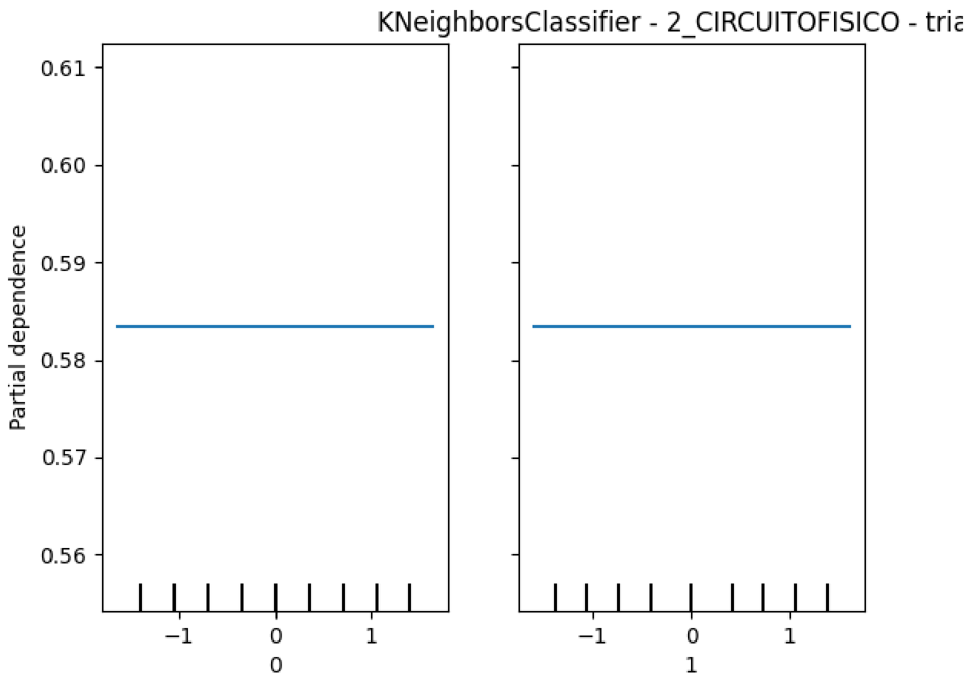

KNeighborsClassifier - 3_DUPLATAREFA - trial
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

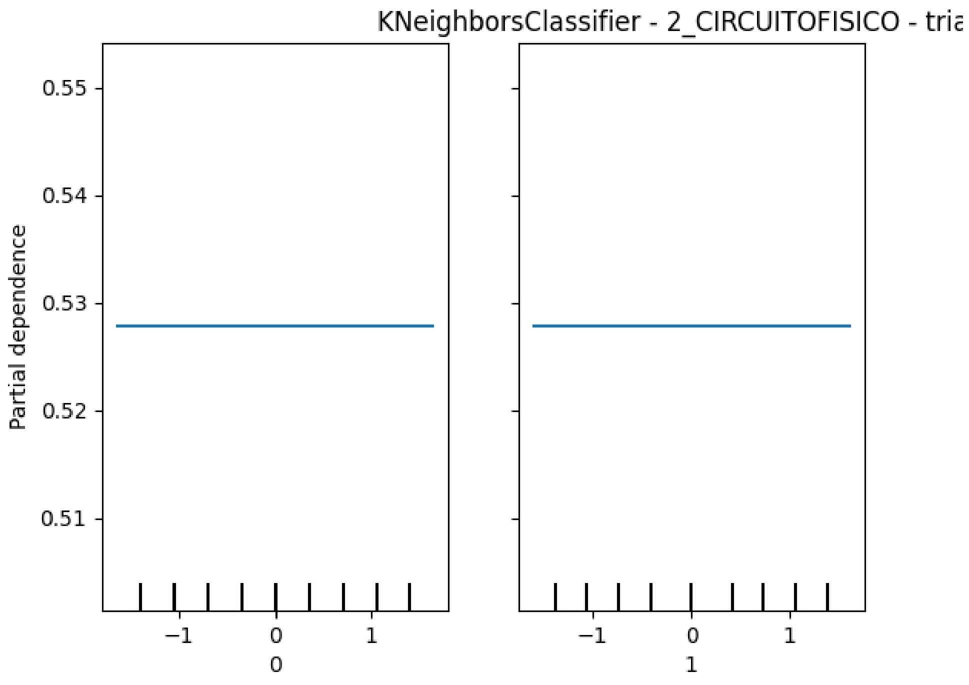

KNeighborsClassifier - 3_DUPLATAREFA - trial
Partial dependence
0.49
0.48
0.47
0.46
0.45
−1
0
1
0
1

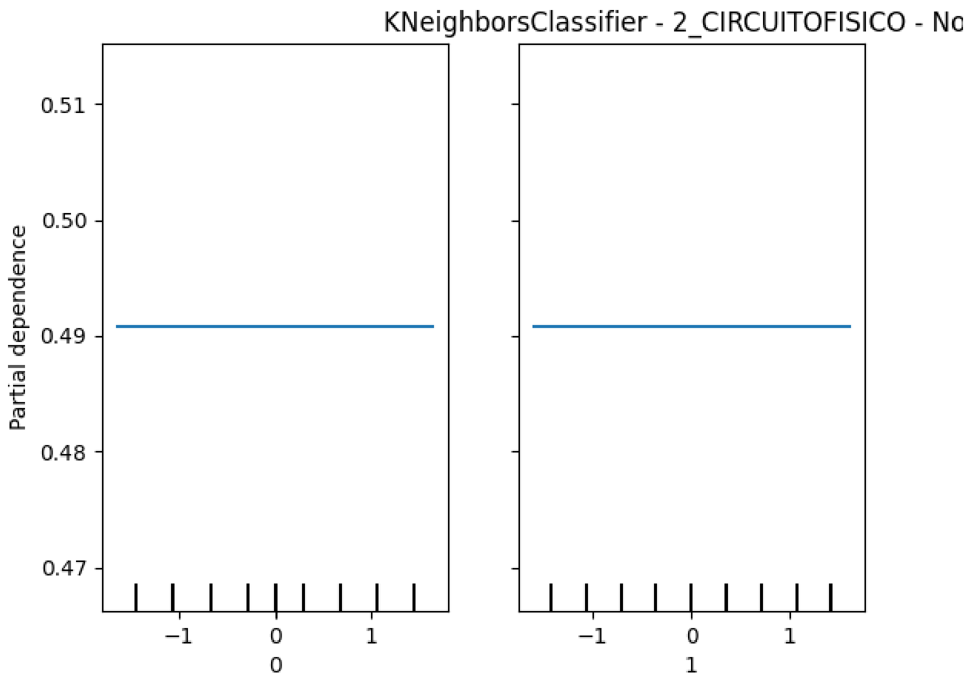

KNeighborsClassifier - 3_DUPLATAREFA - trial
Partial dependence
0.75
0.74
0.73
0.72
0.71
0.70
0.69
−1
0
1
0
1

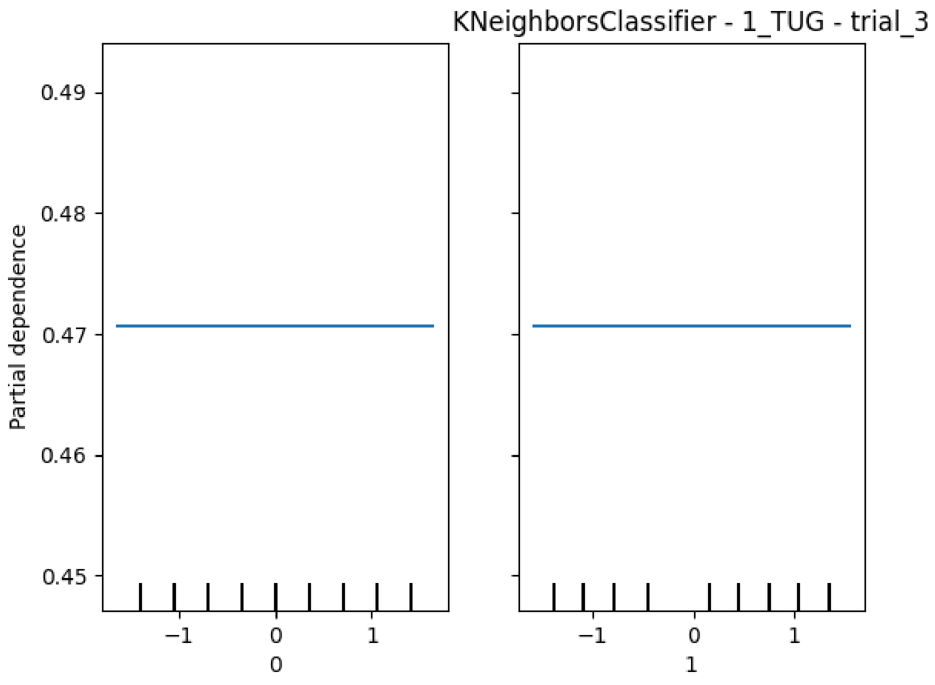

KNeighborsClassifier - 3_DUPLATAREFA - Nor
Partial dependence
0.526
0.524
0.522
0.520
−1
0
1
0
1

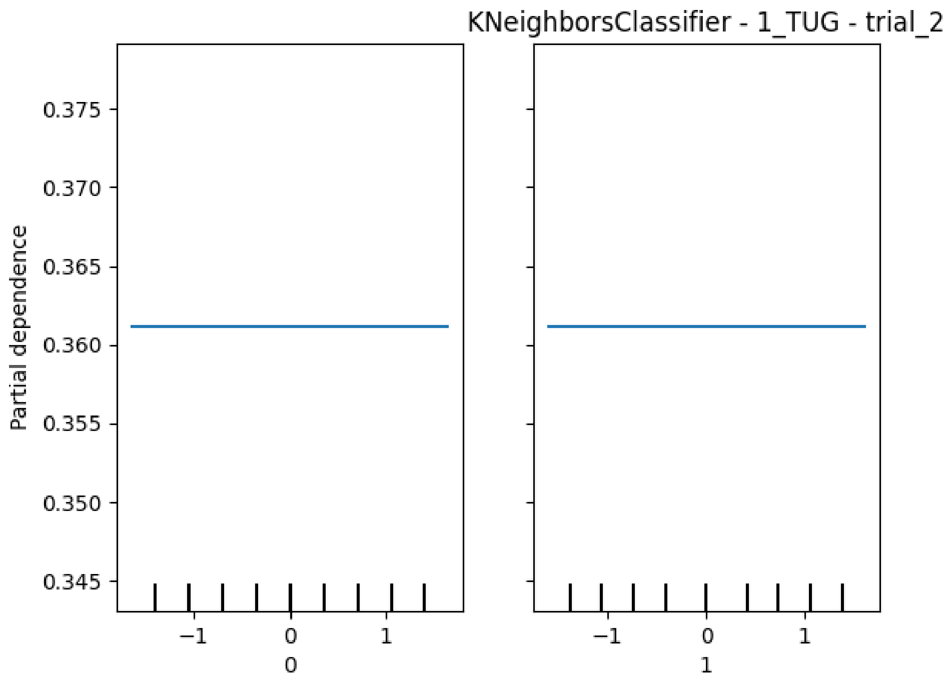

KNeighborsClassifier - 2_CIRCUITOFISICO - tria
Partial dependence
0.290
0.285
0.280
0.275
0.270
0.265
−1
0
1
0
1

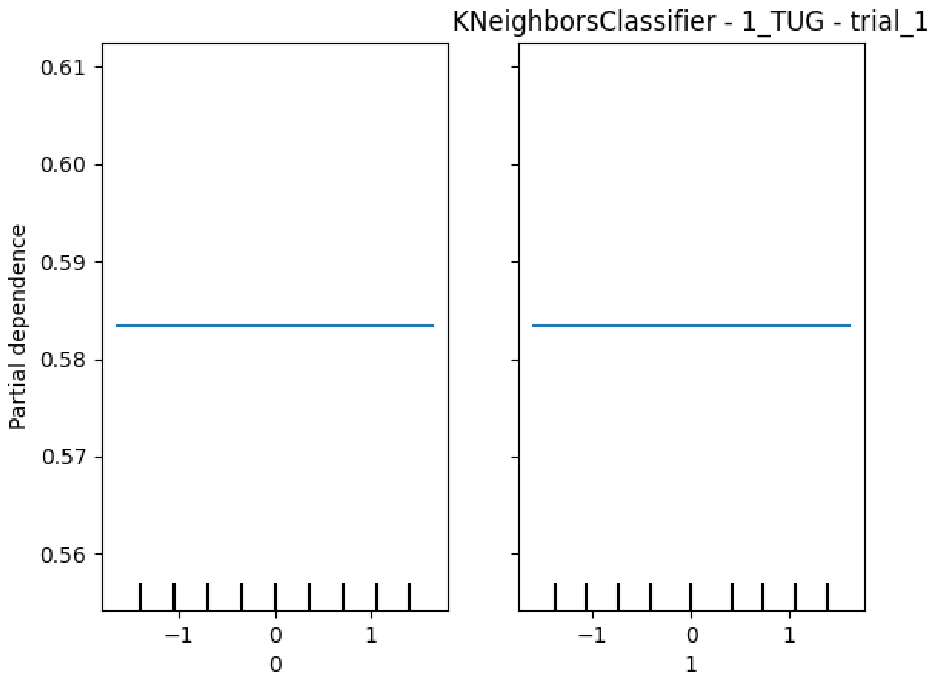

KNeighborsClassifier - 2_CIRCUITOFISICO - tria
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
1

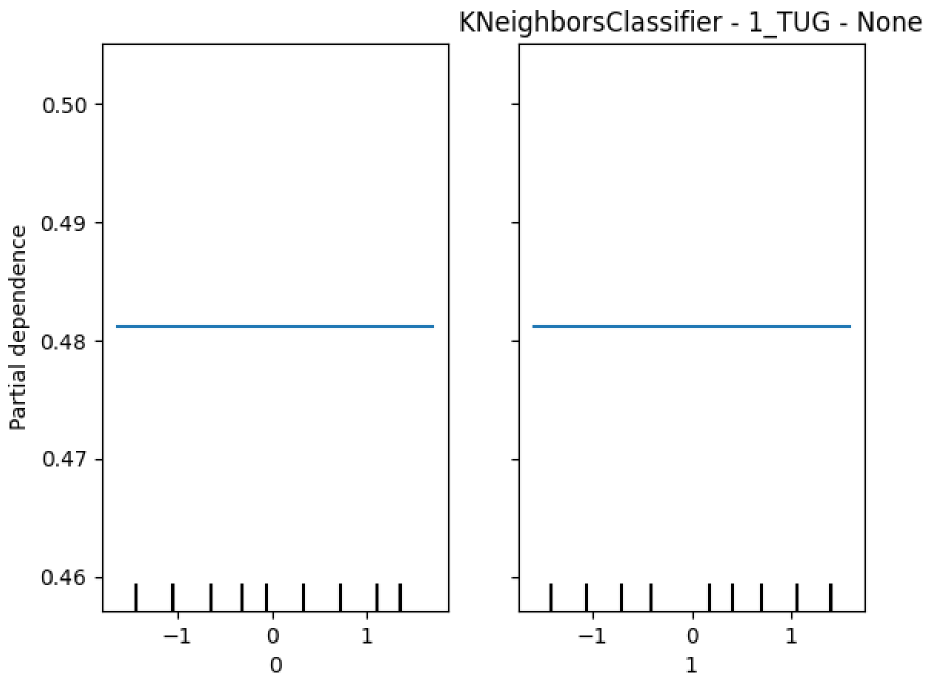

KNeighborsClassifier - 2_CIRCUITOFISICO - tria
Partial dependence
0.55
0.54
0.53
0.52
0.51
−1
0
1
0
1

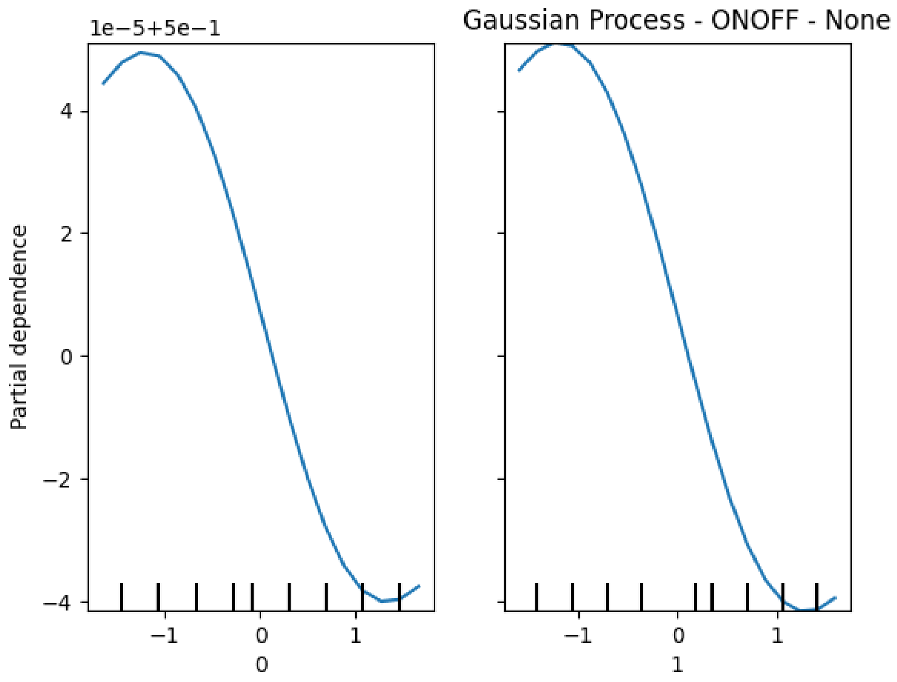

KNeighborsClassifier - 2_CIRCUITOFISICO - No
Partial dependence
0.51
0.50
0.49
0.48
0.47
−1
0
1
0
1

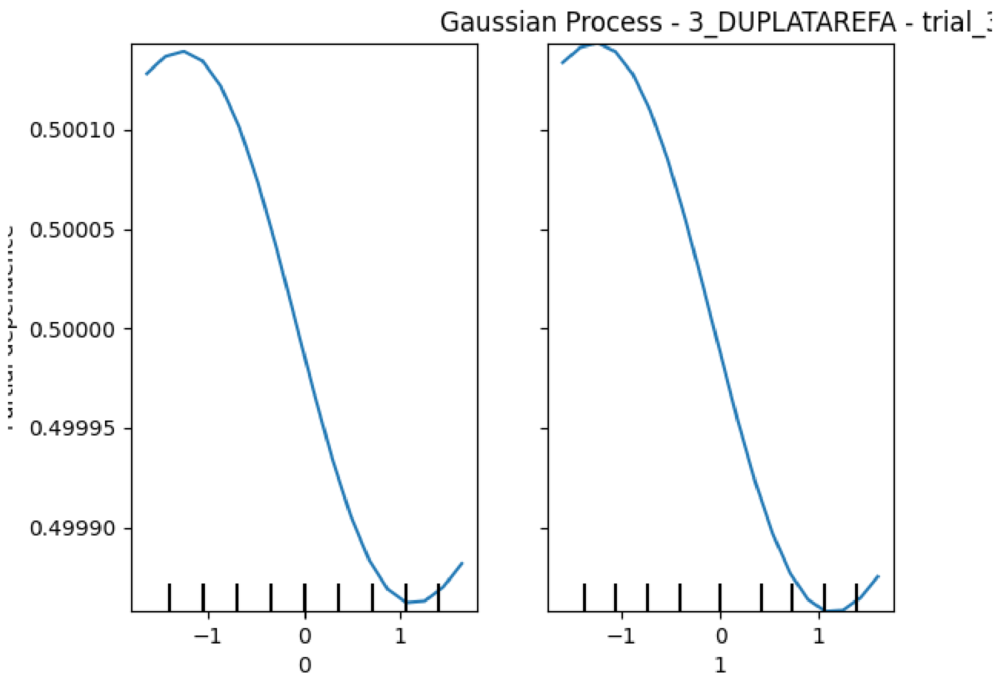

KNeighborsClassifier - 1_TUG - trial_3
Partial dependence
0.49
0.48
0.47
0.46
0.45
−1
0
1
0
1

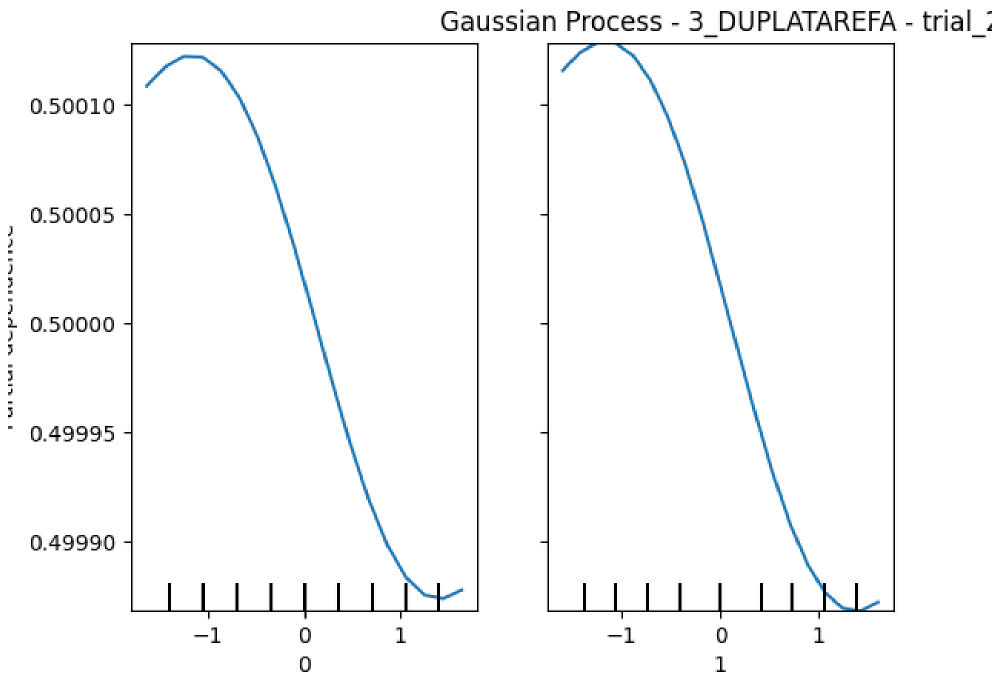

KNeighborsClassifier - 1_TUG - trial_2
Partial dependence
0.375
0.370
0.365
0.360
0.355
0.350
0.345
−1
0
1
0
1

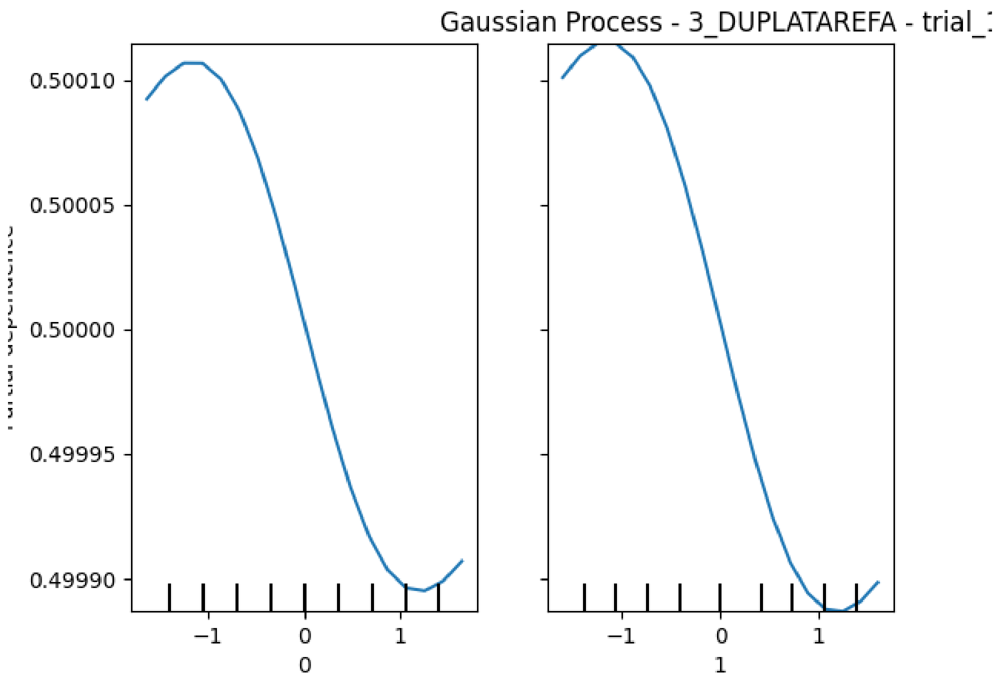

KNeighborsClassifier - 1_TUG - trial_1
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
1

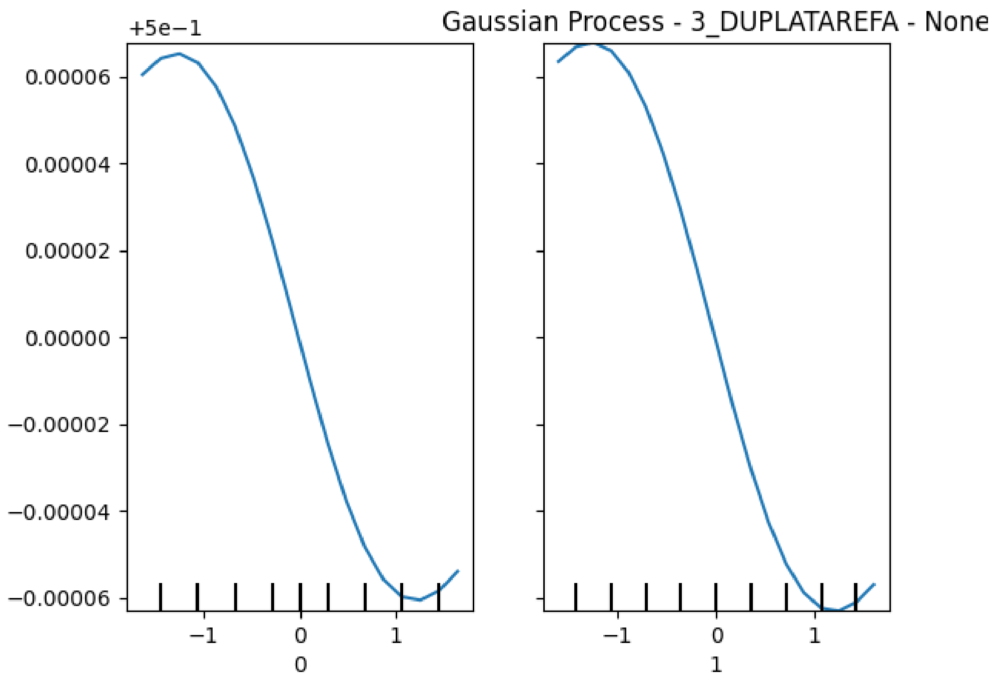

KNeighborsClassifier - 1_TUG - None
Partial dependence
0.50
0.49
0.48
0.47
0.46
−1
0
1
0
1

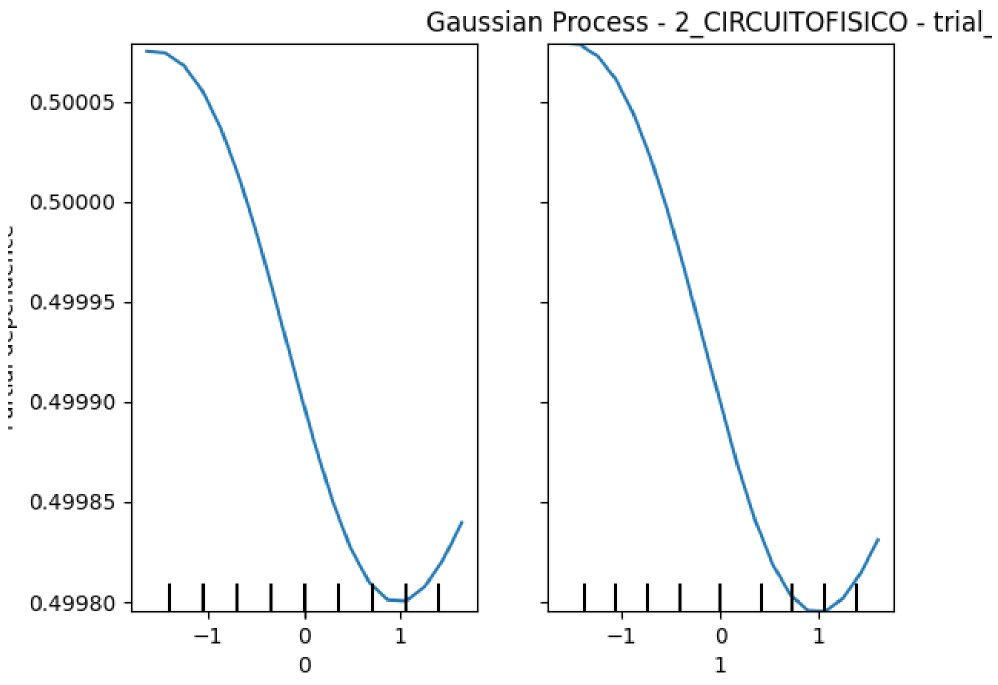

Gaussian Process - ONOFF - None
1e−5+5e−1
Partial dependence
4
2
0
−2
−4
−1
0
1
0
1

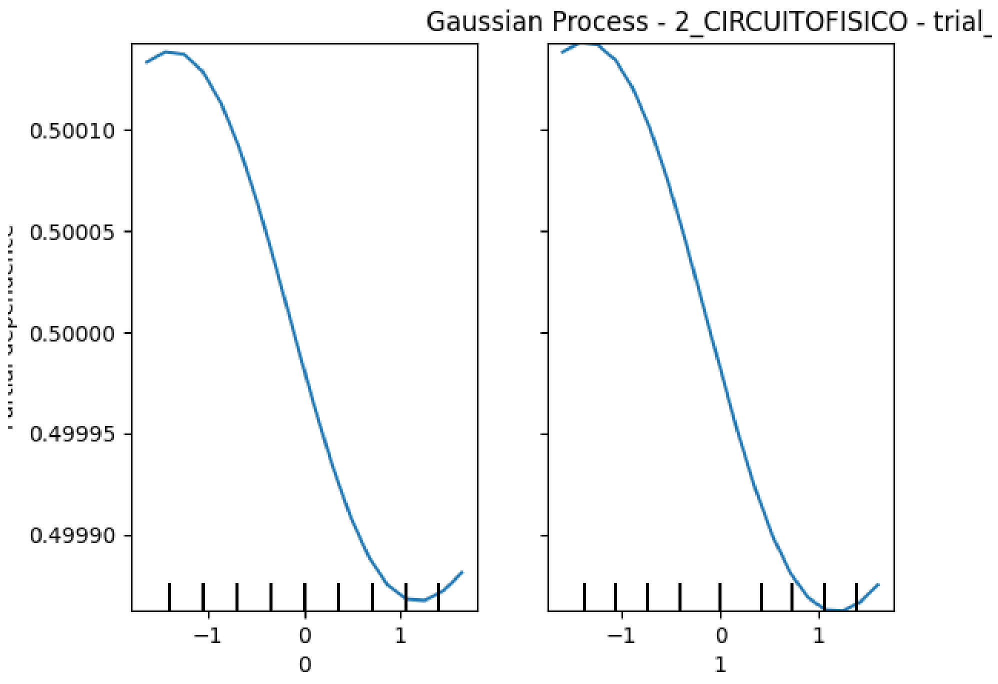

Gaussian Process - 3_DUPLATAREFA - trial_3
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1

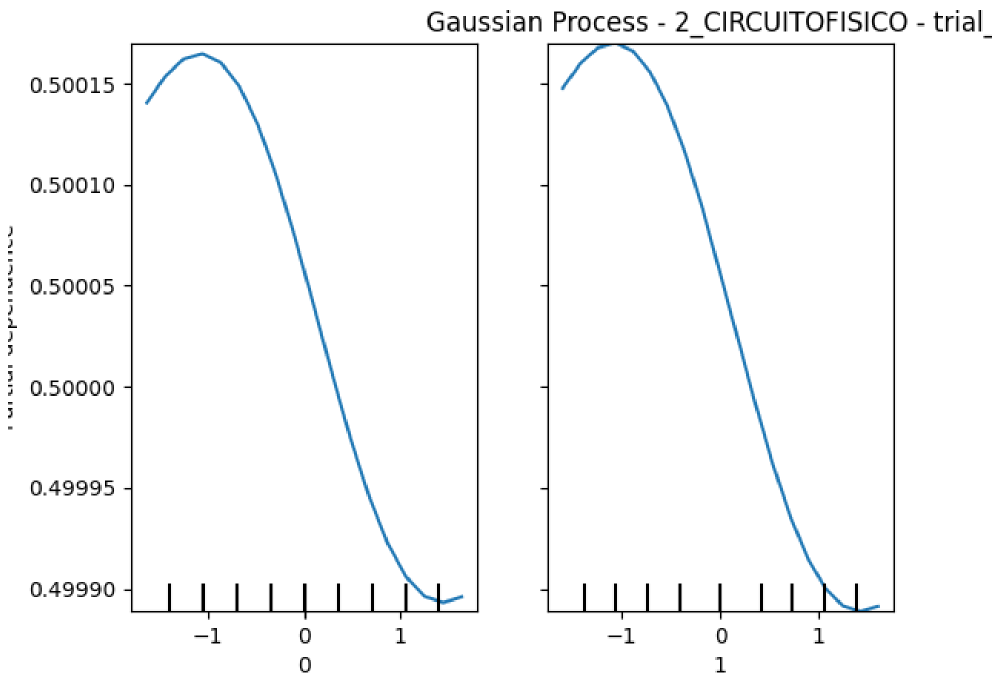

Gaussian Process - 3_DUPLATAREFA - trial_2
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1

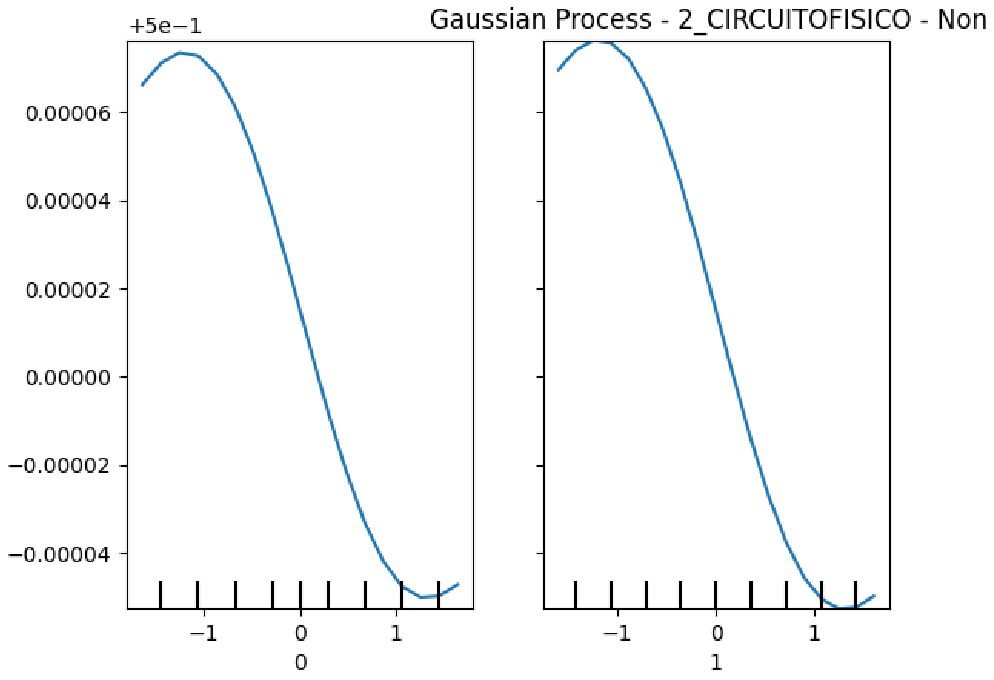

Gaussian Process - 3_DUPLATAREFA - trial_
Partial dependence
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1

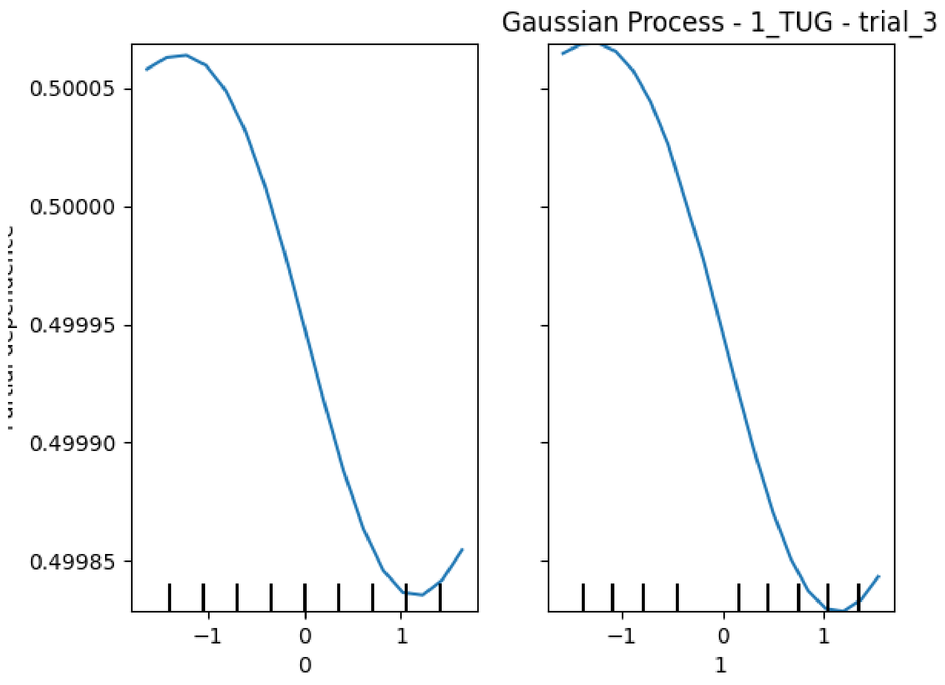

Gaussian Process - 3_DUPLATAREFA - None
+5e−1
0.00006
0.00004
0.00002
0.00000
−0.00002
−0.00004
−0.00006
−1
0
1
0
1

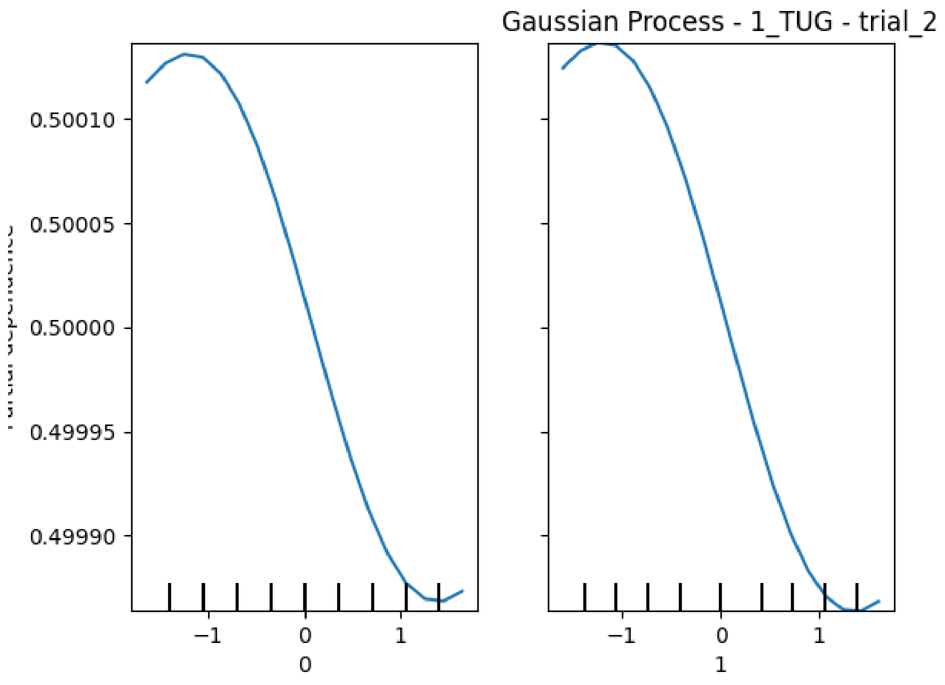

Gaussian Process - 2_CIRCUITOFISICO - trial_
Partial dependence
0.50005
0.50000
0.49995
0.49990
0.49985
0.49980
−1
0
1
0
1

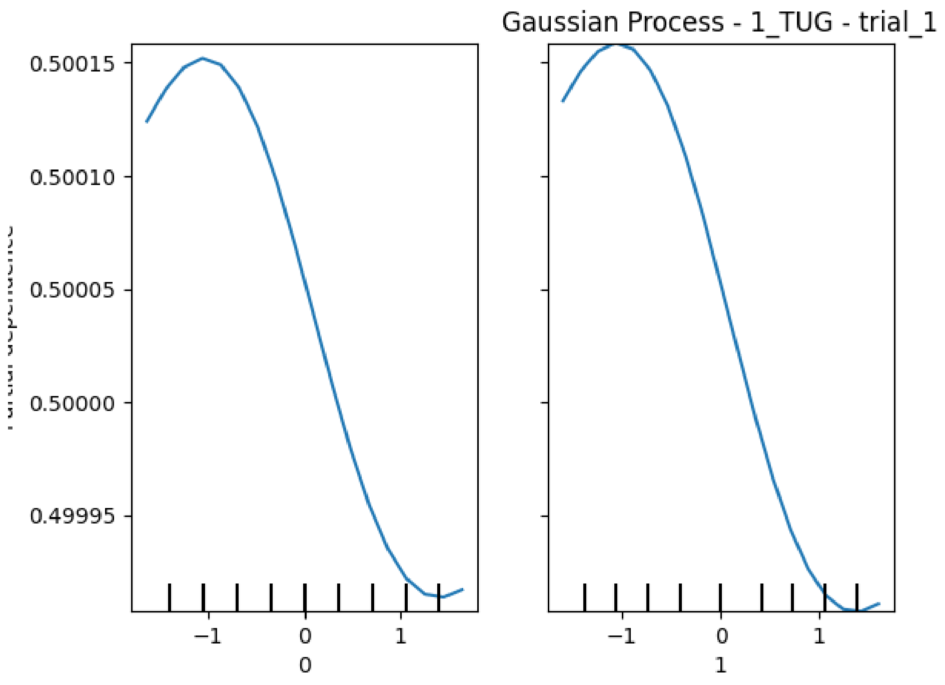

Gaussian Process - 2_CIRCUITOFISICO - trial_
Partial dependence
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1

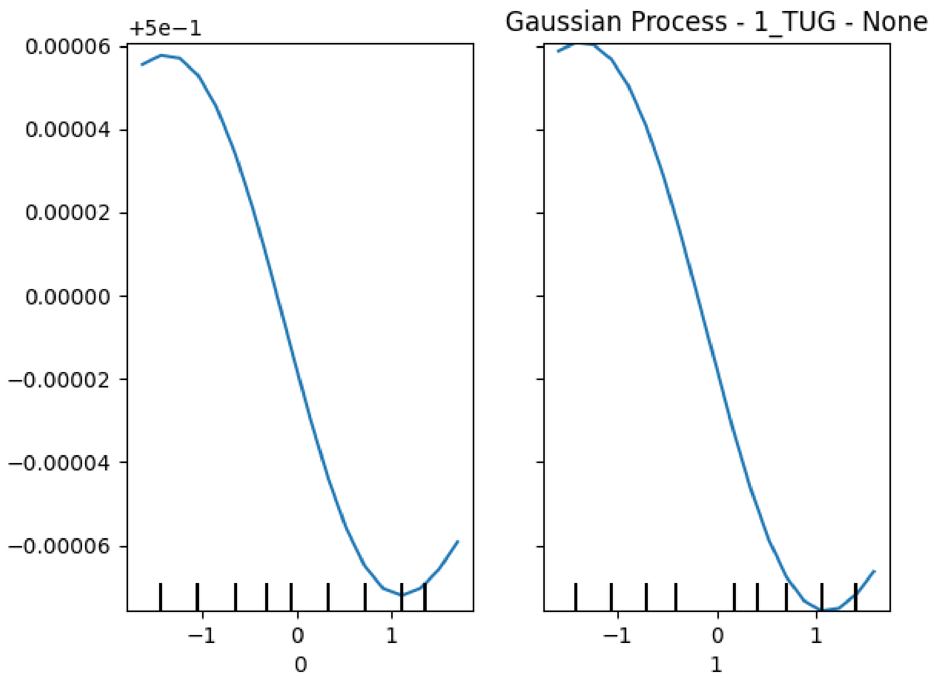

Gaussian Process - 2_CIRCUITOFISICO - trial_
Partial dependence
0.50015
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1

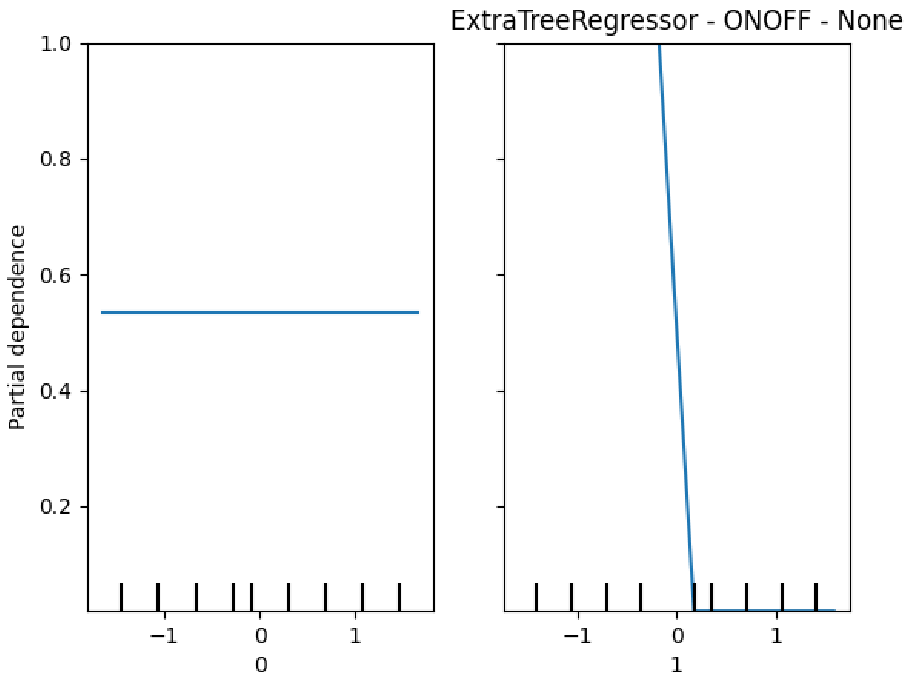

Gaussian Process - 2_CIRCUITOFISICO - Non
+5e−1
0.00006
0.00004
0.00002
0.00000
−0.00002
−0.00004
−1
0
1
0
1


Gaussian Process - 1_TUG - trial_3
0.50005
0.50000
0.49995
0.49990
0.49985
−1
0
1
0
1

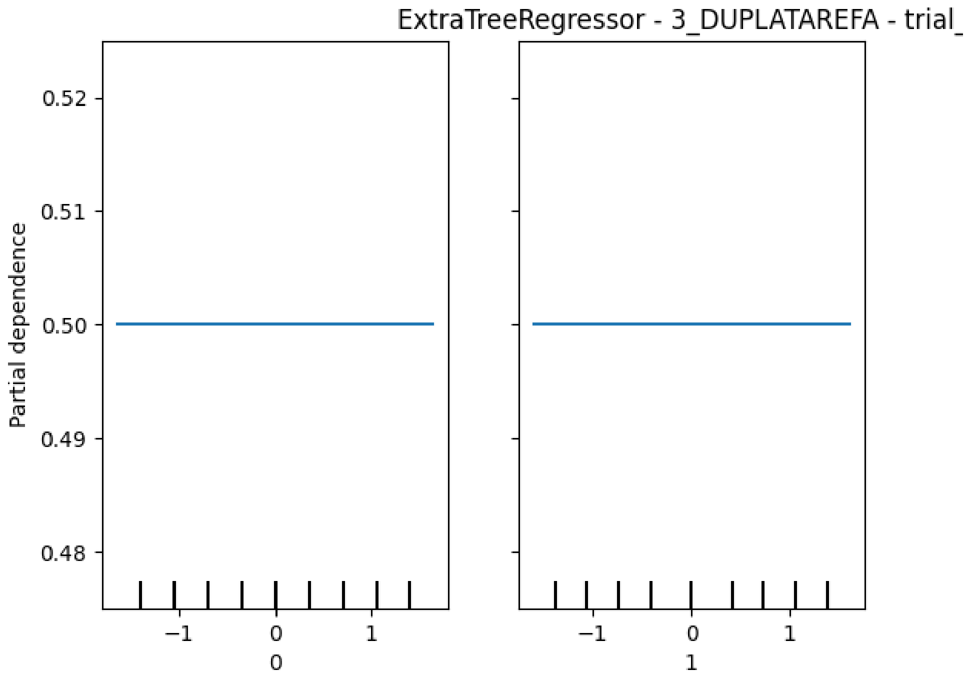

Gaussian Process - 1_TUG - trial_2
0.50010
0.50005
0.50000
0.49995
0.49990
−1
0
1
0
1


Gaussian Process - 1_TUG - trial_1
0.50015
0.50010
0.50005
0.50000
0.49995
−1
0
1
0
1

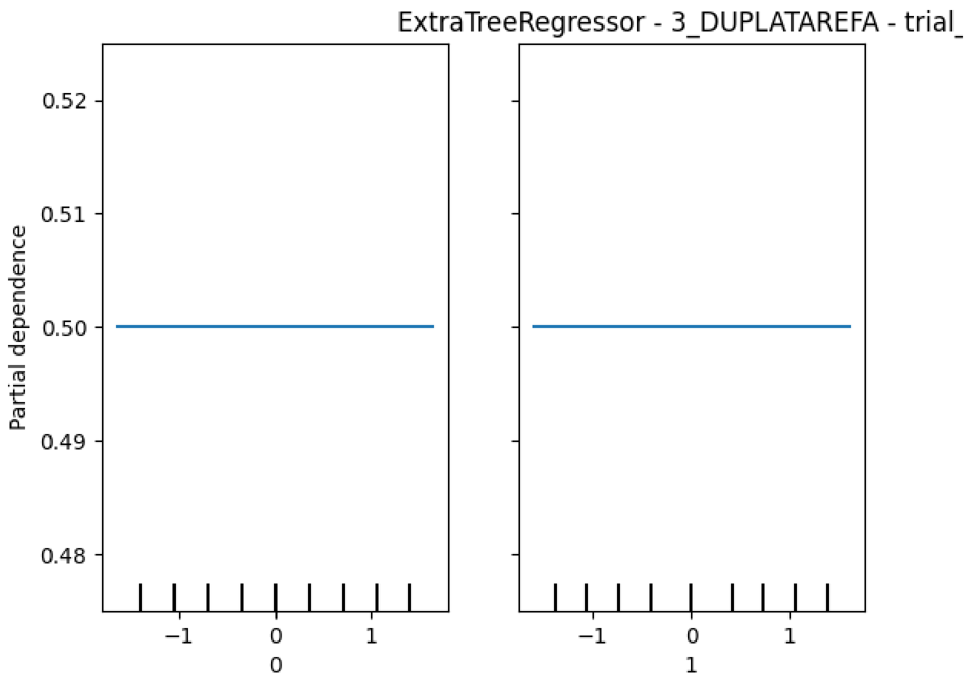

Gaussian Process - 1_TUG - None
+5e−1
0.00006
0.00004
0.00002
0.00000
−0.00002
−0.00004
−0.00006
−1
0
1
0
1

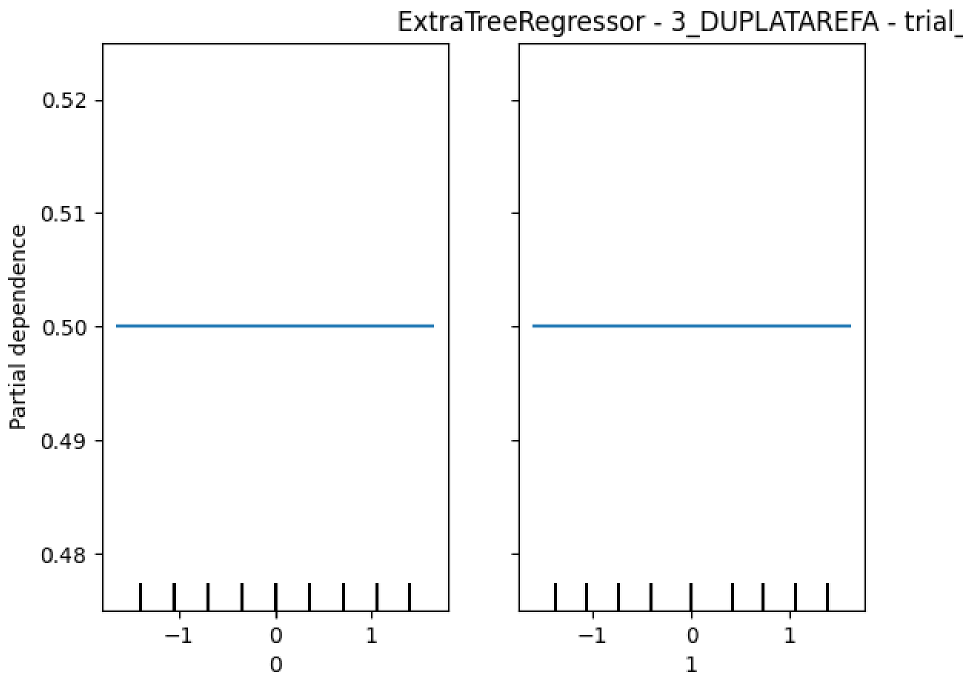


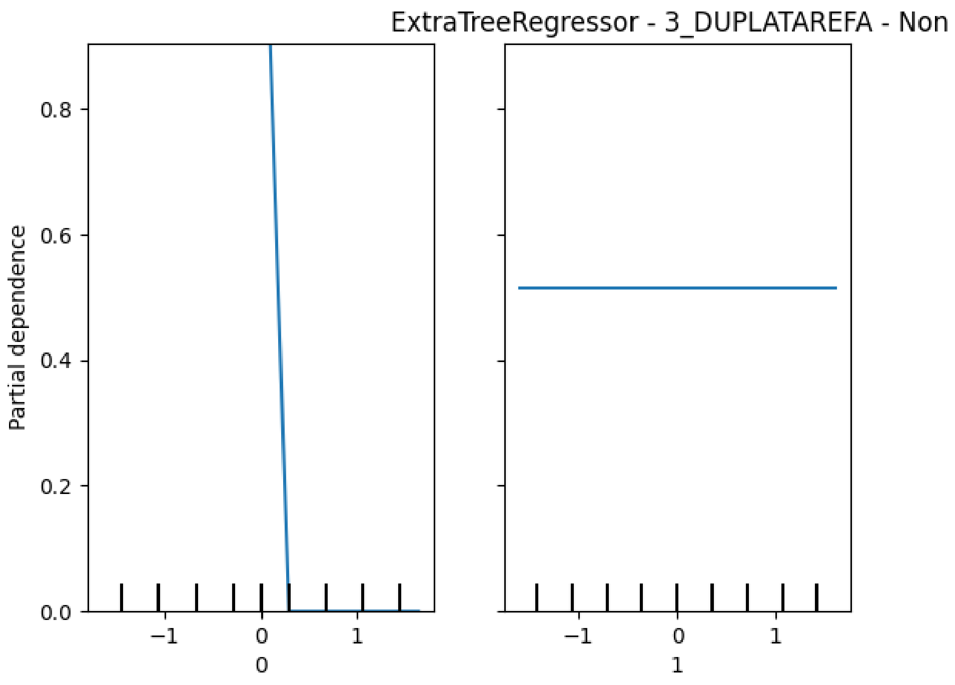
ExtraTreeRegressor - ONOFF - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
−1
0
1
0
1

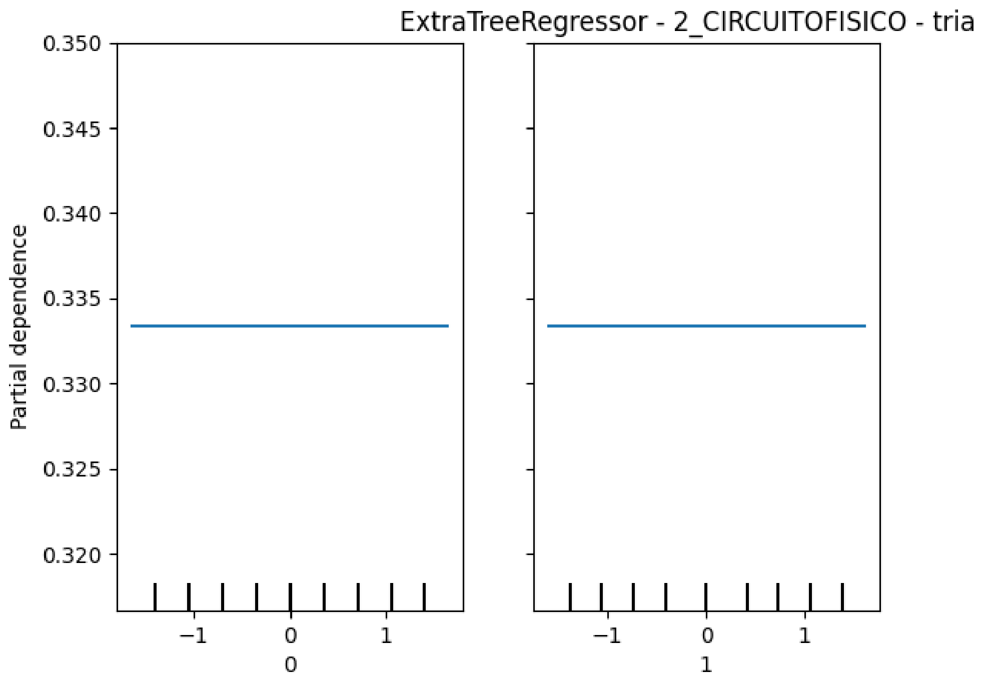


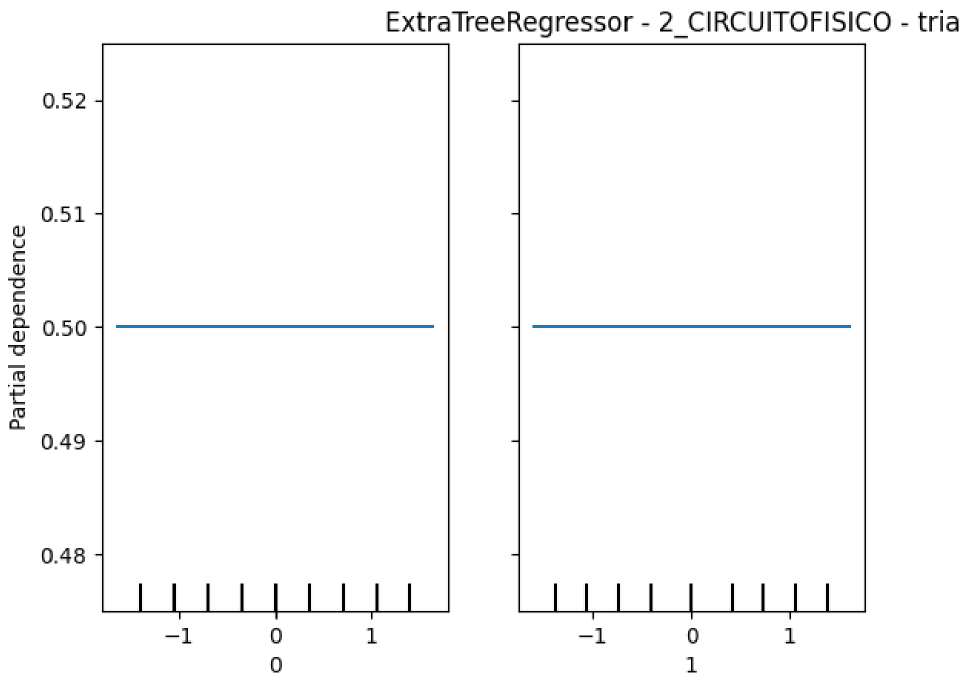
ExtraTreeRegressor - 3_DUPLATAREFA - trial_
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

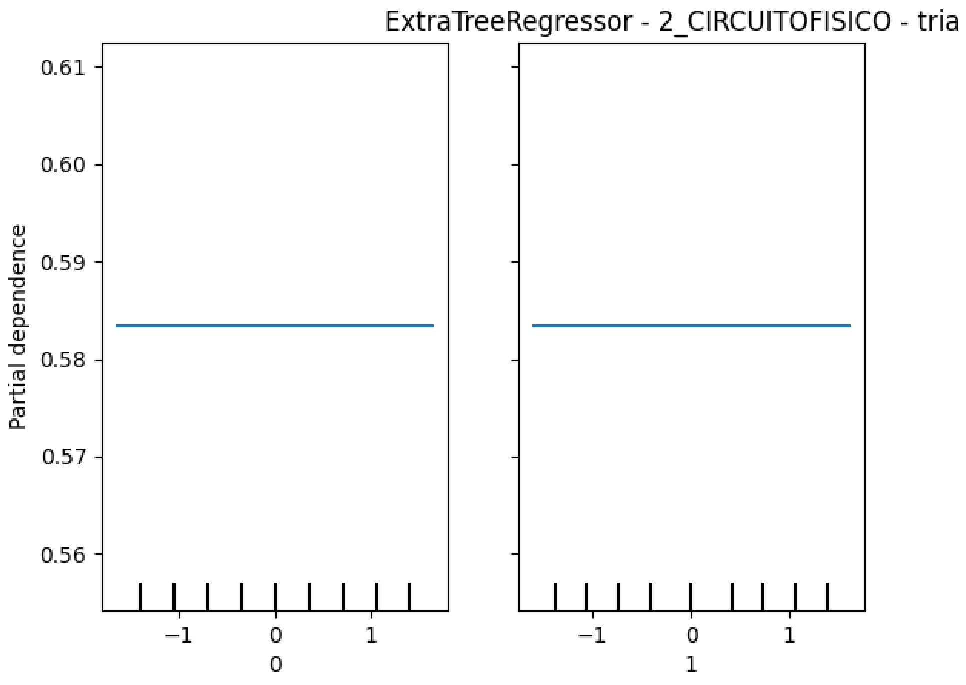

ExtraTreeRegressor - 3_DUPLATAREFA - trial_
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

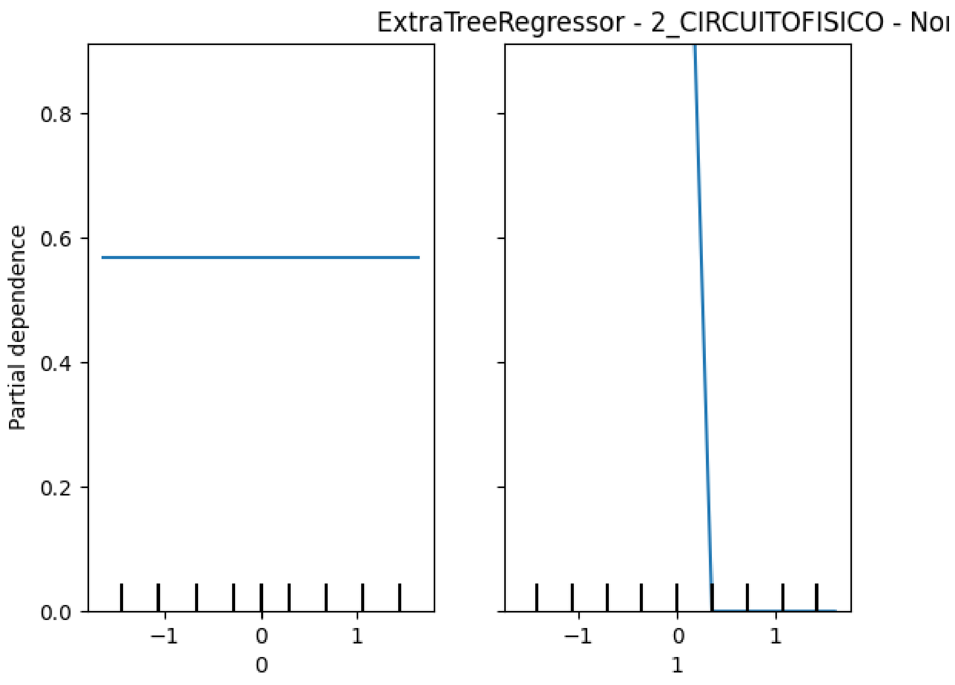

ExtraTreeRegressor - 3_DUPLATAREFA - trial_
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

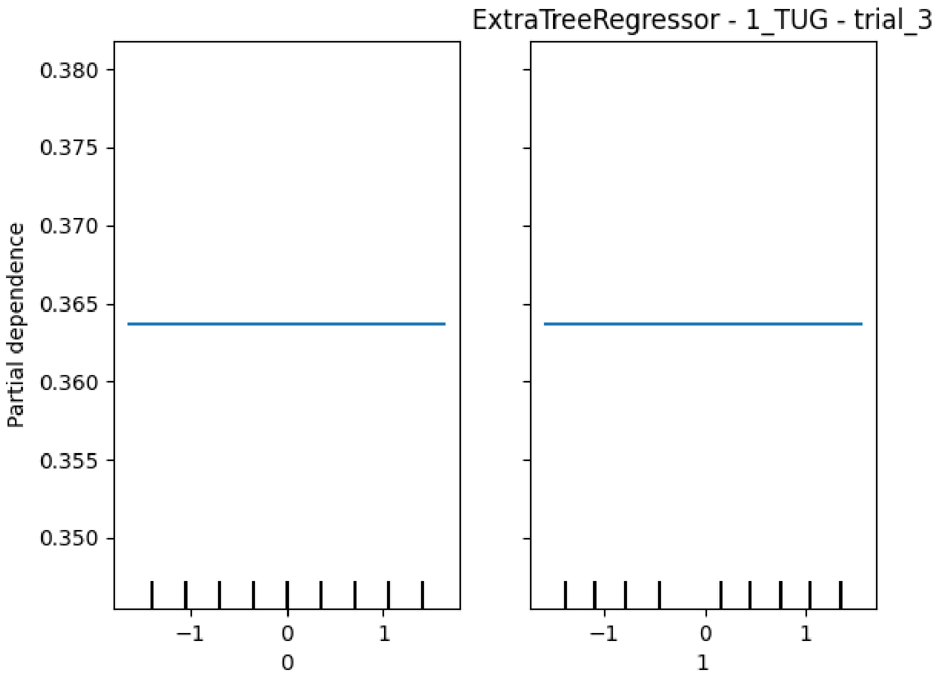

ExtraTreeRegressor - 3_DUPLATAREFA - Non
Partial dependence
0.0
0.2
0.4
0.6
0.8
−1
0
1
0
1

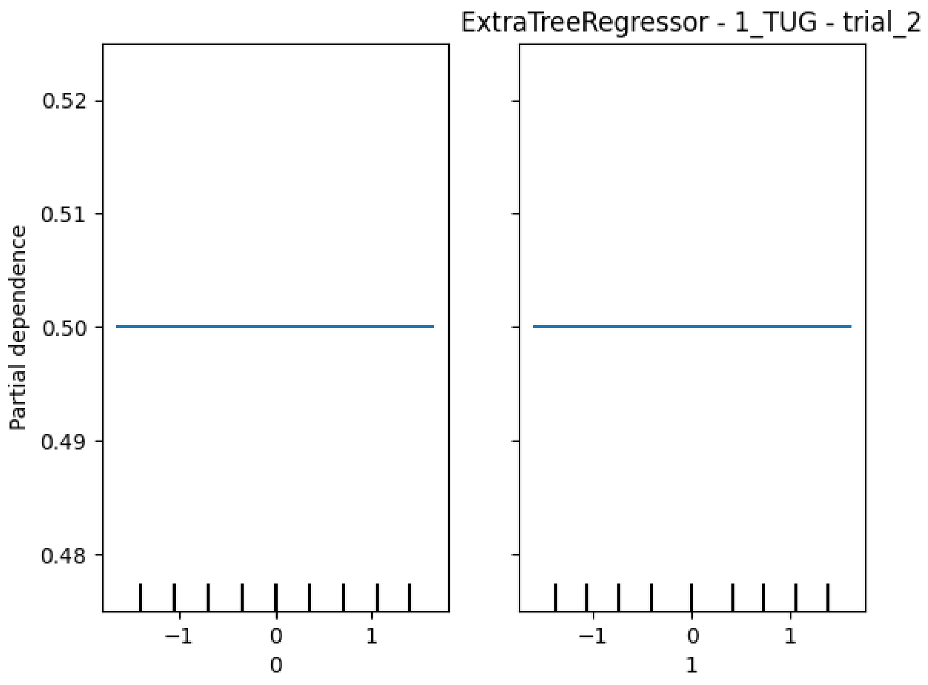

ExtraTreeRegressor - 2_CIRCUITOFISICO - tria
Partial dependence
0.320
0.325
0.330
0.335
0.340
0.345
0.350
−1
0
1
0
1

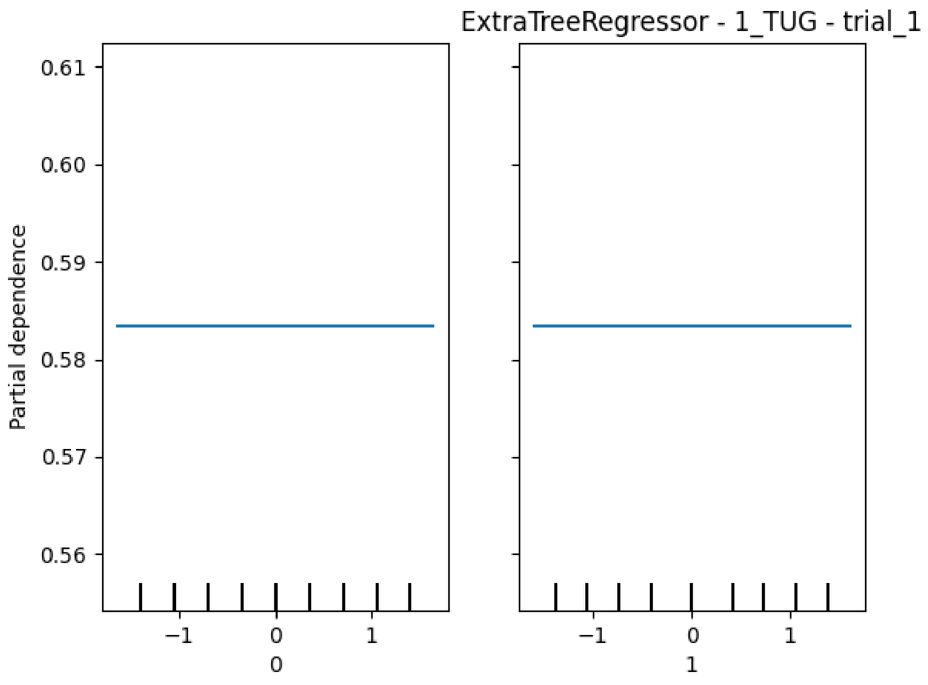

ExtraTreeRegressor - 2_CIRCUITOFISICO - tria
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

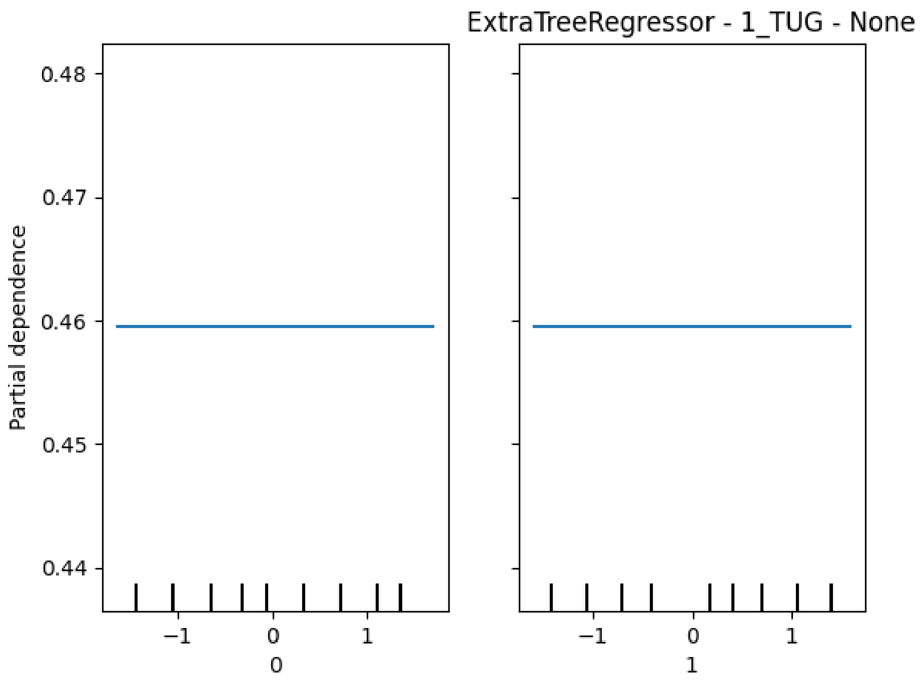

ExtraTreeRegressor - 2_CIRCUITOFISICO - tria
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
1

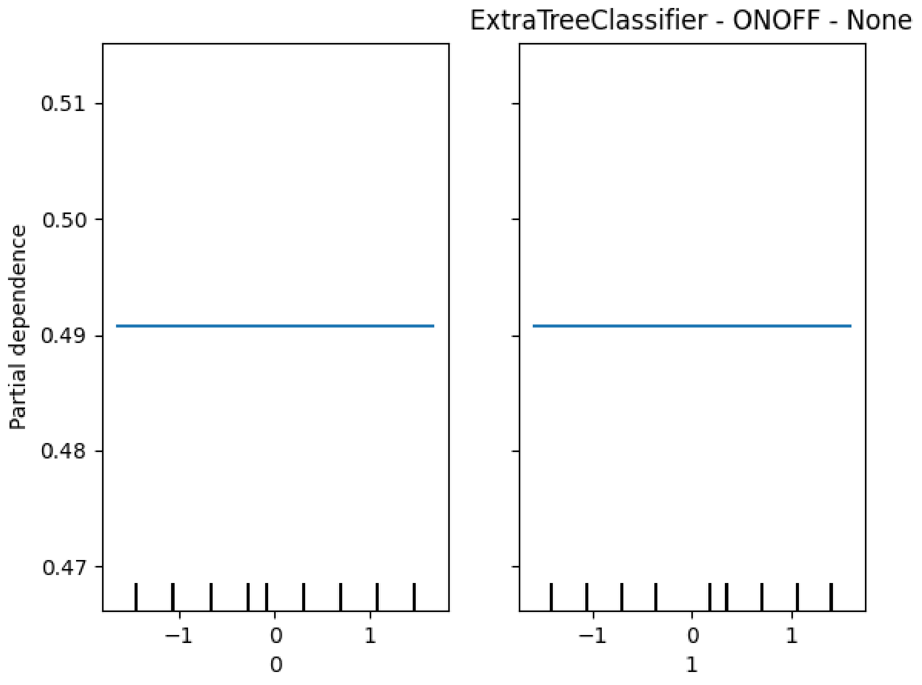

ExtraTreeRegressor - 2_CIRCUITOFISICO - Noı
Partial dependence
0.0
0.2
0.4
0.6
0.8
−1
0
1
0
1

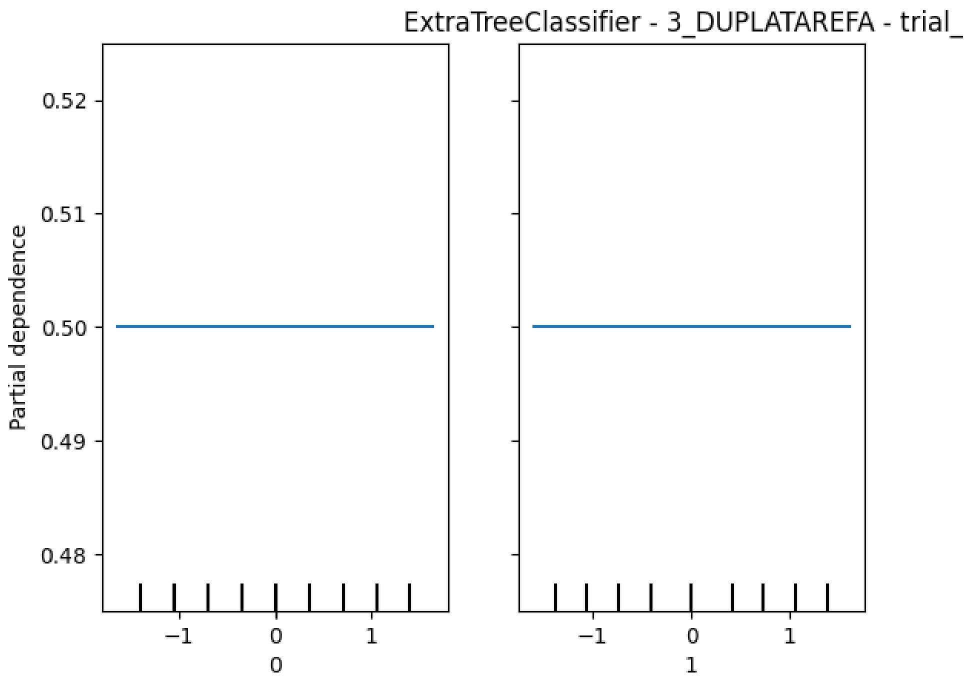

ExtraTreeRegressor - 1_TUG - trial_3
Partial dependence
0.350
0.355
0.360
0.365
0.370
0.375
0.380
−1
0
1
0
1

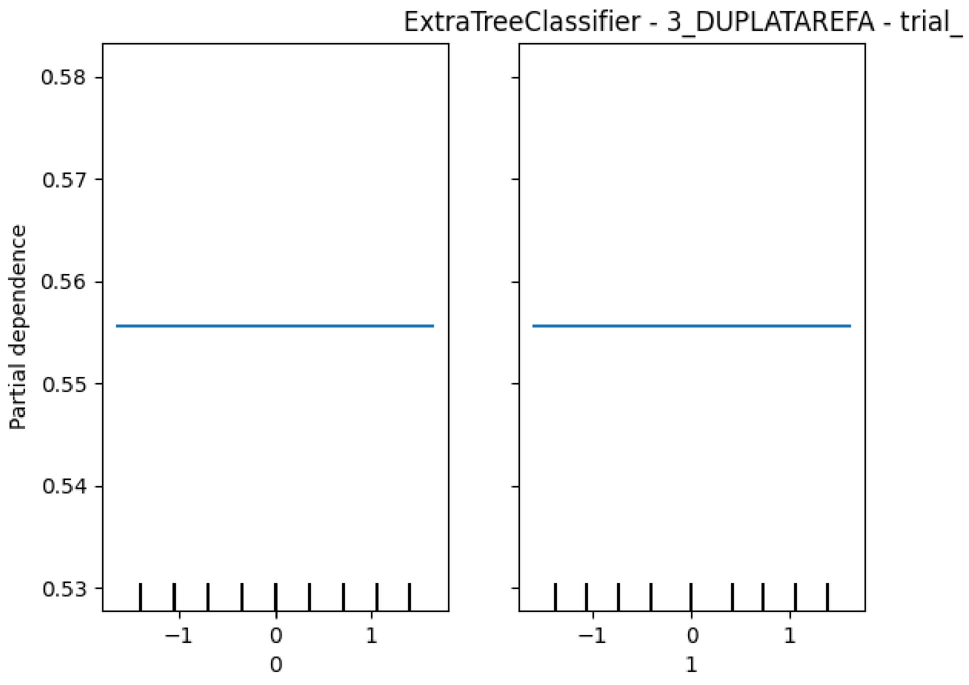

ExtraTreeRegressor - 1_TUG - trial_2
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

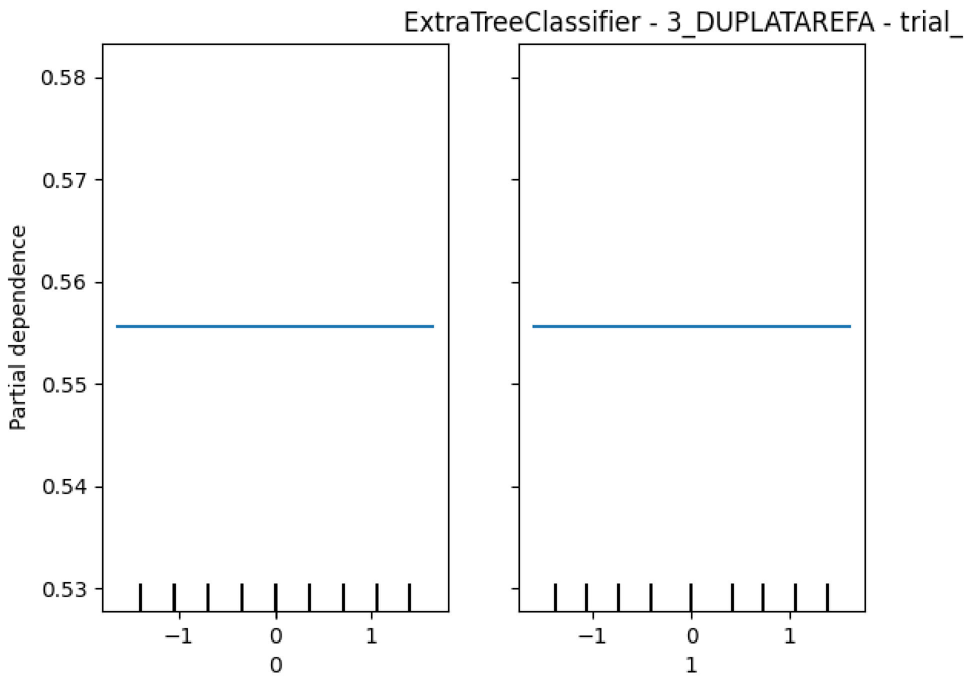

ExtraTreeRegressor - 1_TUG - trial_1
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
1

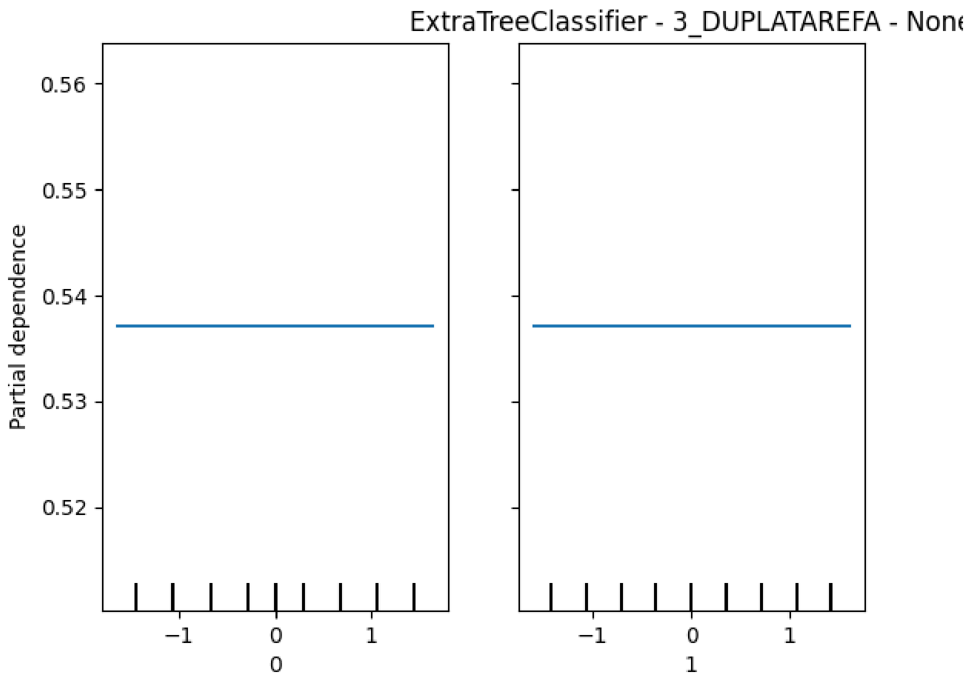

ExtraTreeRegressor - 1_TUG - None
Partial dependence
0.48
0.47
0.46
0.45
0.44
−1
0
1
0
1

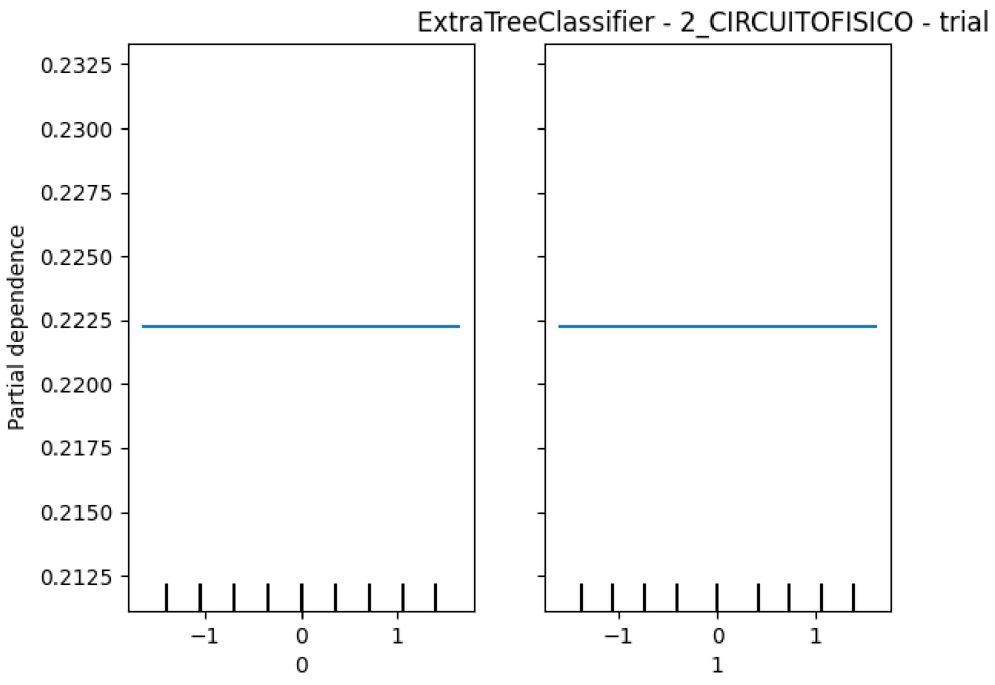

ExtraTreeClassifier - ONOFF - None
Partial dependence
0.51
0.50
0.49
0.48
0.47
−1
0
1
0
1

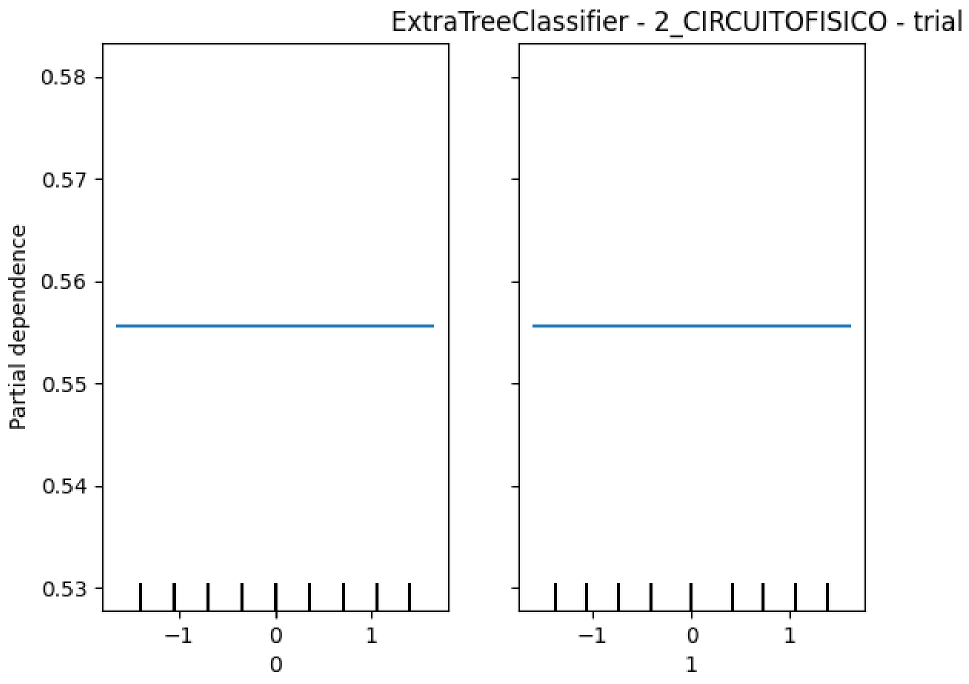

ExtraTreeClassifier - 3_DUPLATAREFA - trial_
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

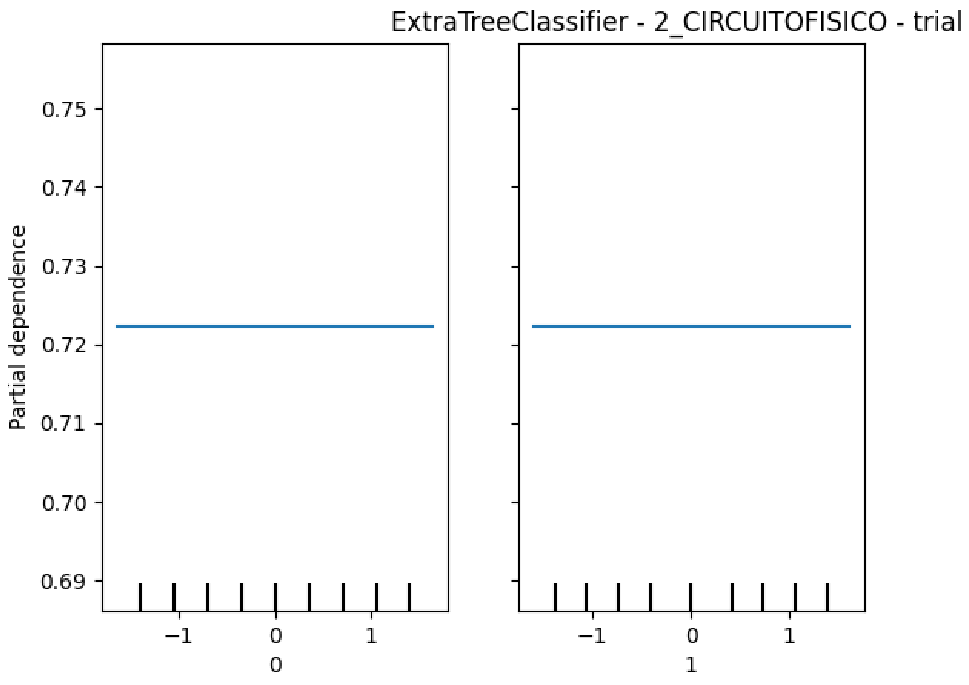

ExtraTreeClassifier - 3_DUPLATAREFA - trial_
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

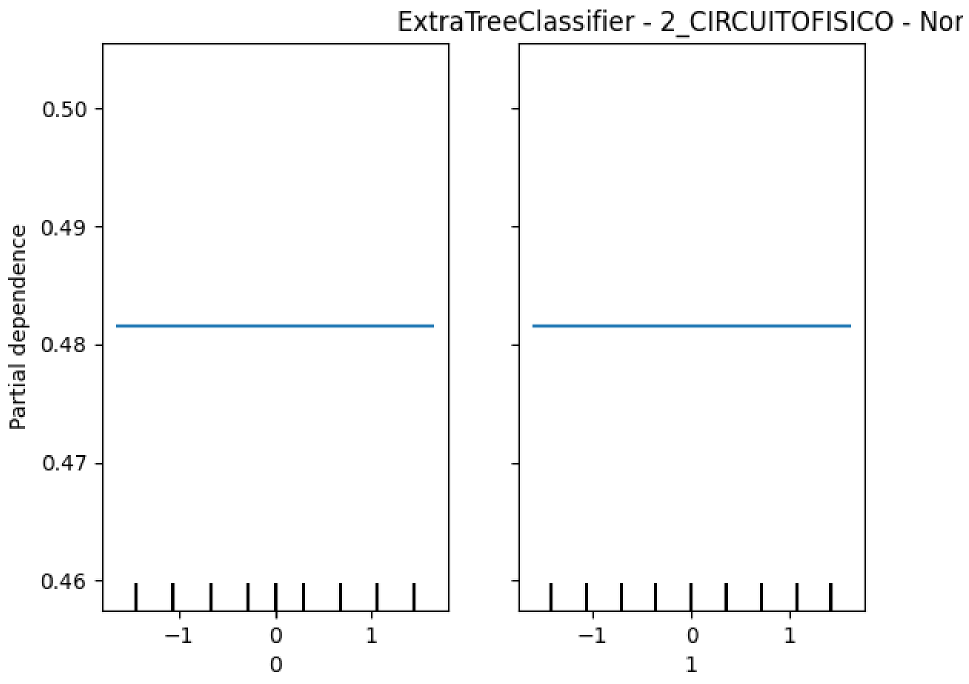

ExtraTreeClassifier - 3_DUPLATAREFA - trial_
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
−1
0
1
1

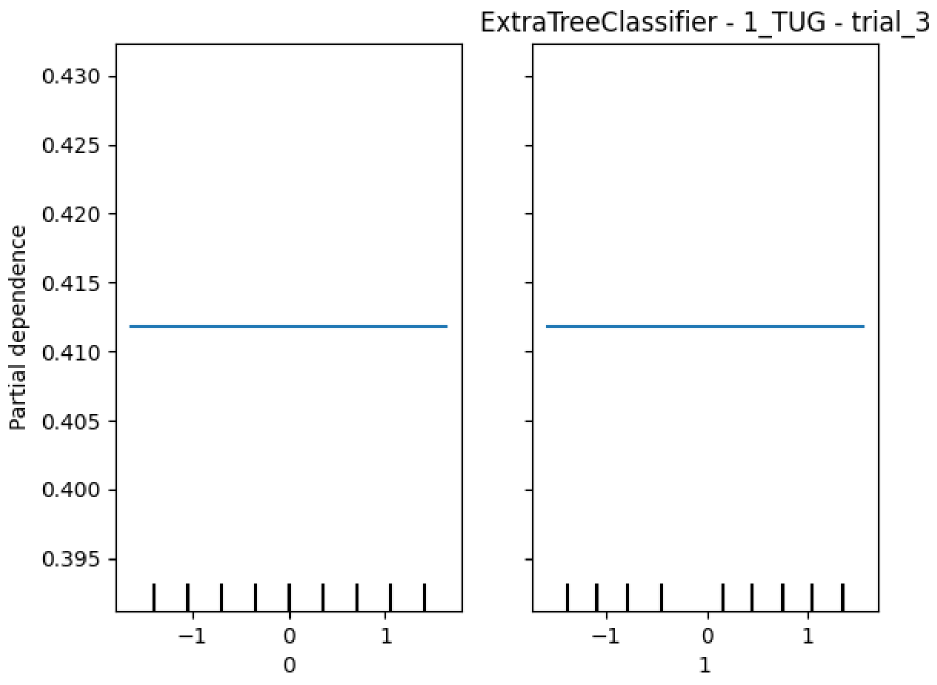

ExtraTreeClassifier - 3_DUPLATAREFA - None
Partial dependence
0.56
0.55
0.54
0.53
0.52
−1
0
1
0
−1
0
1
1

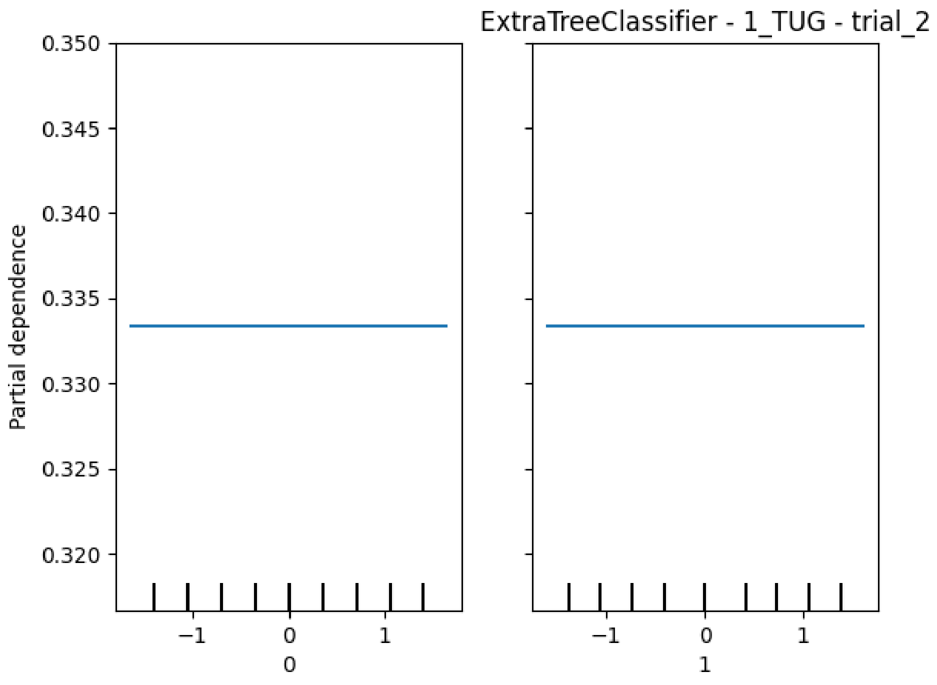

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial
Partial dependence
0.2325
0.2300
0.2275
0.2250
0.2225
0.2200
0.2175
0.2150
0.2125
−1
0
1
0
1

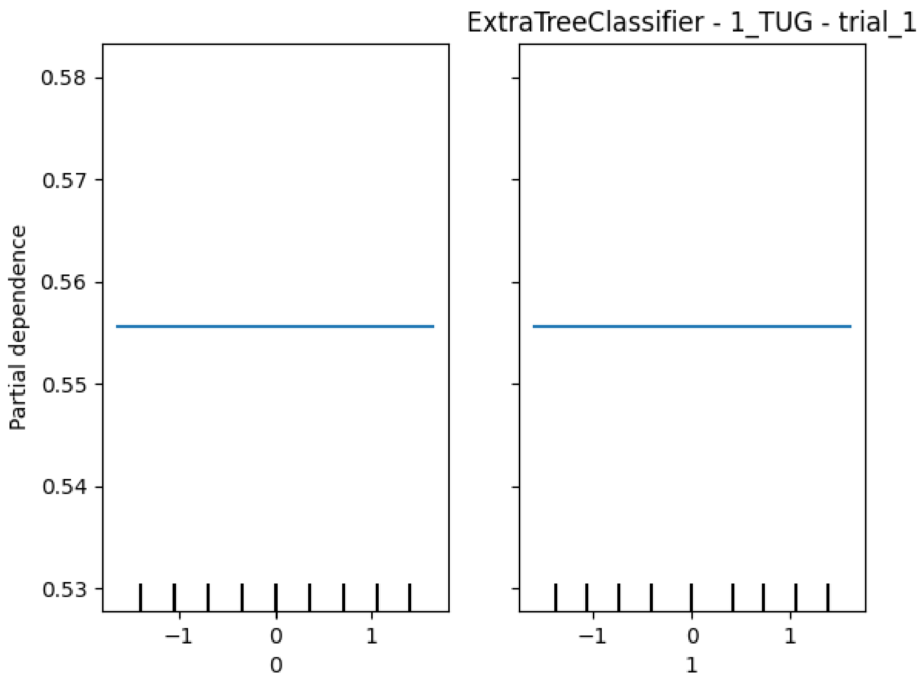

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

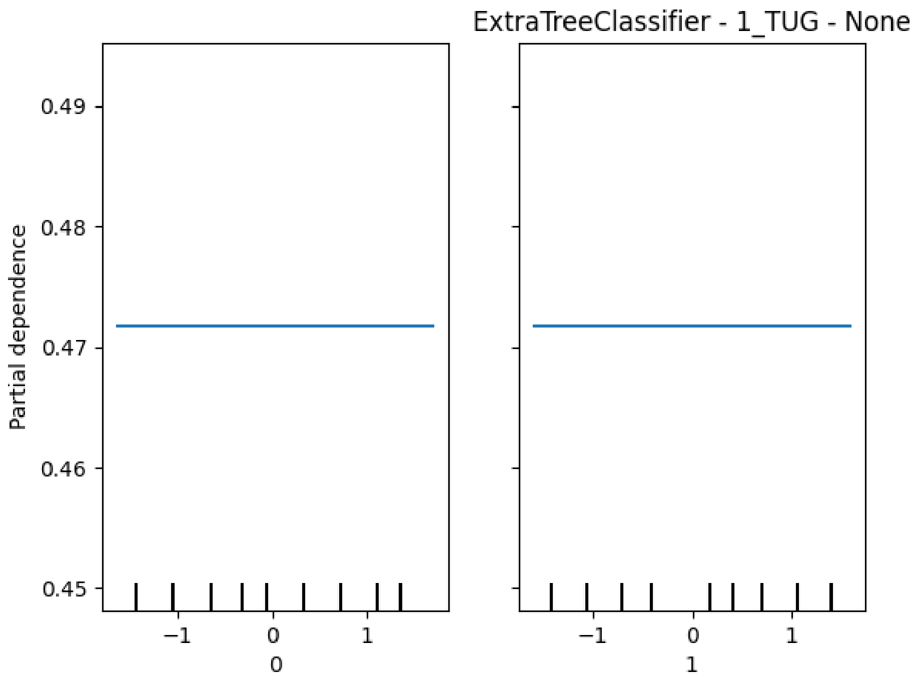

ExtraTreeClassifier - 2_CIRCUITOFISICO - trial
Partial dependence
0.75
0.74
0.73
0.72
0.71
0.70
0.69
−1
0
1
0
−1
0
1
1

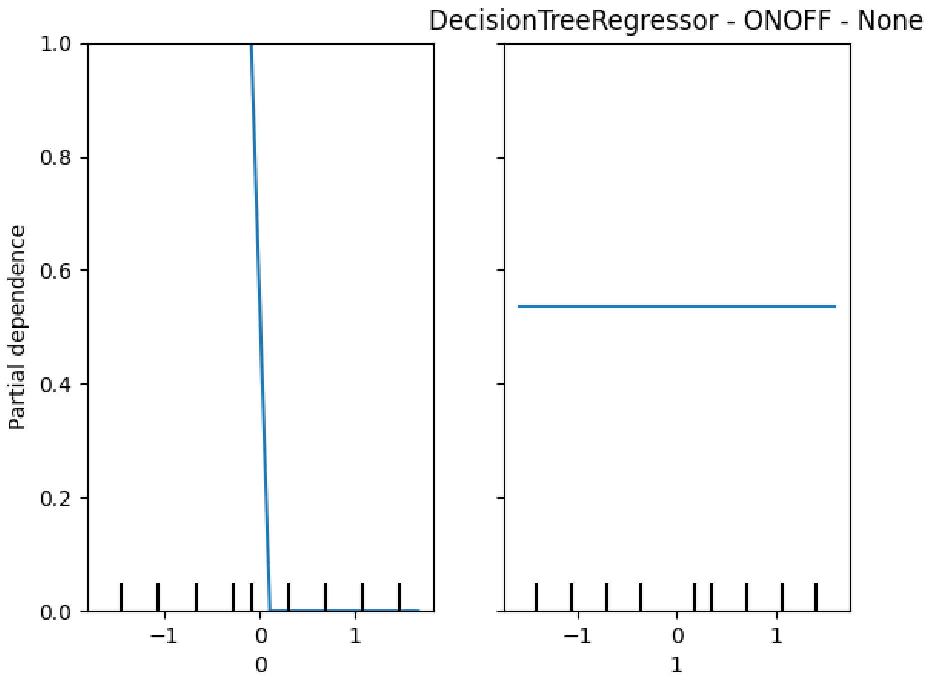

ExtraTreeClassifier - 2_CIRCUITOFISICO - Nor
Partial dependence
0.50
0.49
0.48
0.47
0.46
−1
0
1
0
−1
0
1
1

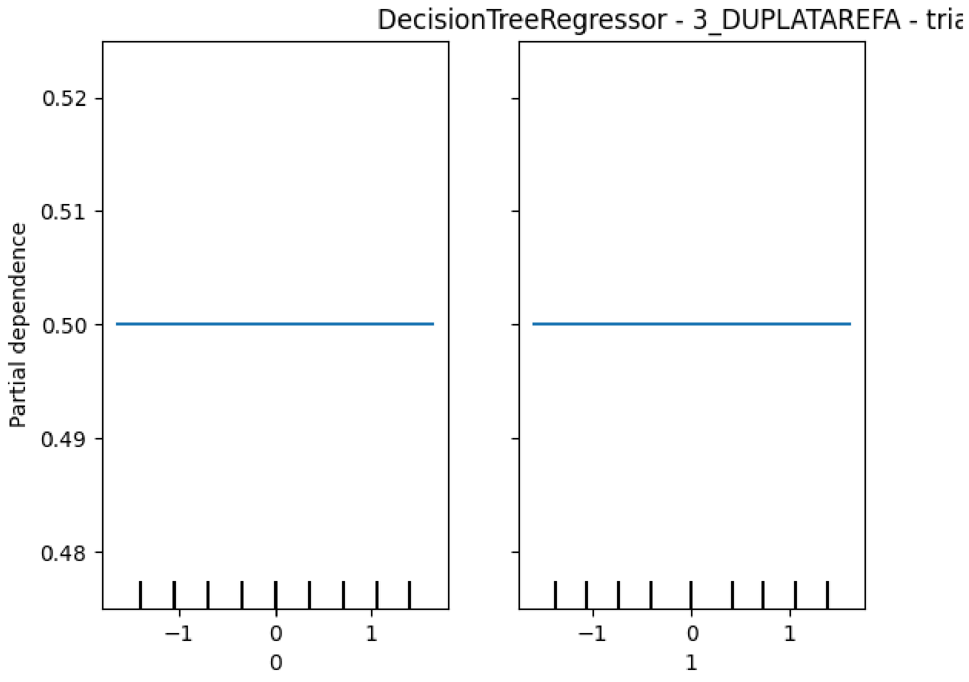

ExtraTreeClassifier - 1_TUG - trial_3
Partial dependence
0.430
0.425
0.420
0.415
0.410
0.405
0.400
0.395
−1
0
1
0
1

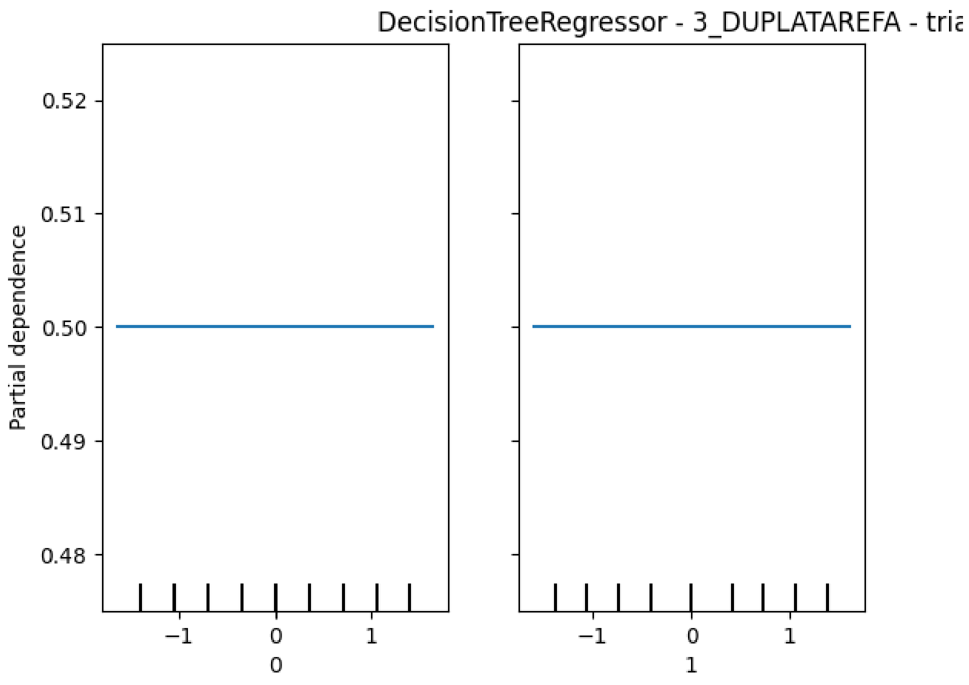

ExtraTreeClassifier - 1_TUG - trial_2
Partial dependence
0.350
0.345
0.340
0.335
0.330
0.325
0.320
−1
0
1
0
1

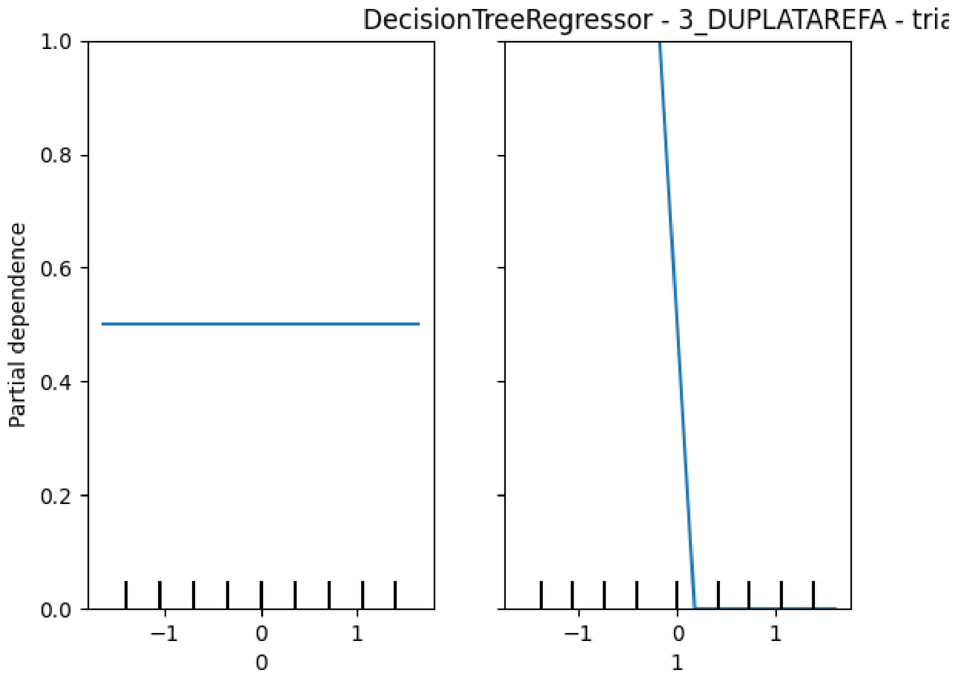

ExtraTreeClassifier - 1_TUG - trial_1
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

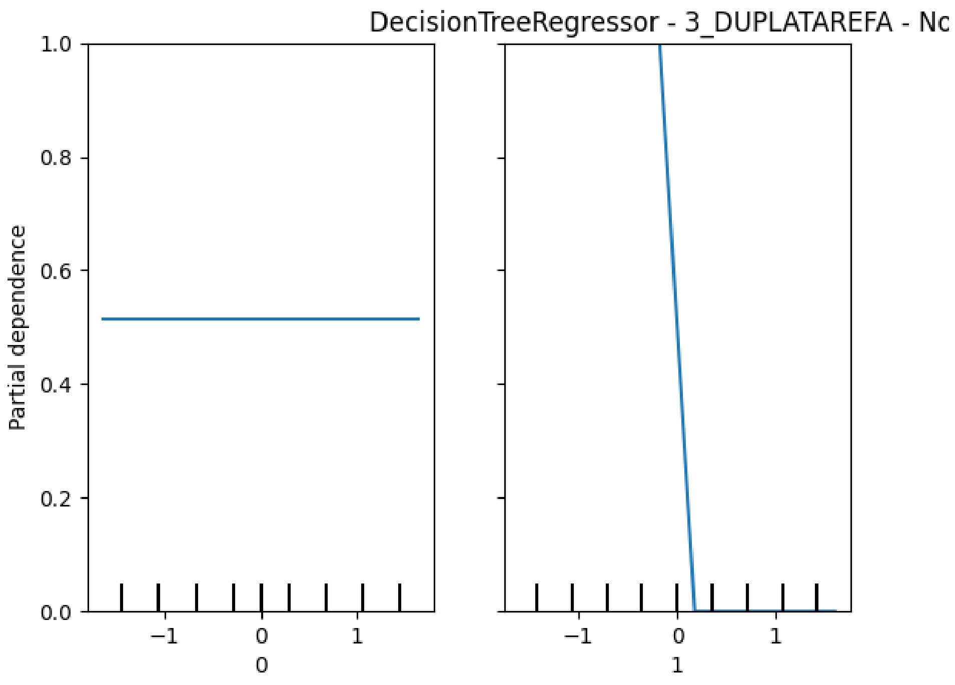

ExtraTreeClassifier - 1_TUG - None
Partial dependence
0.49
0.48
0.47
0.46
0.45
−1
0
1
0
1

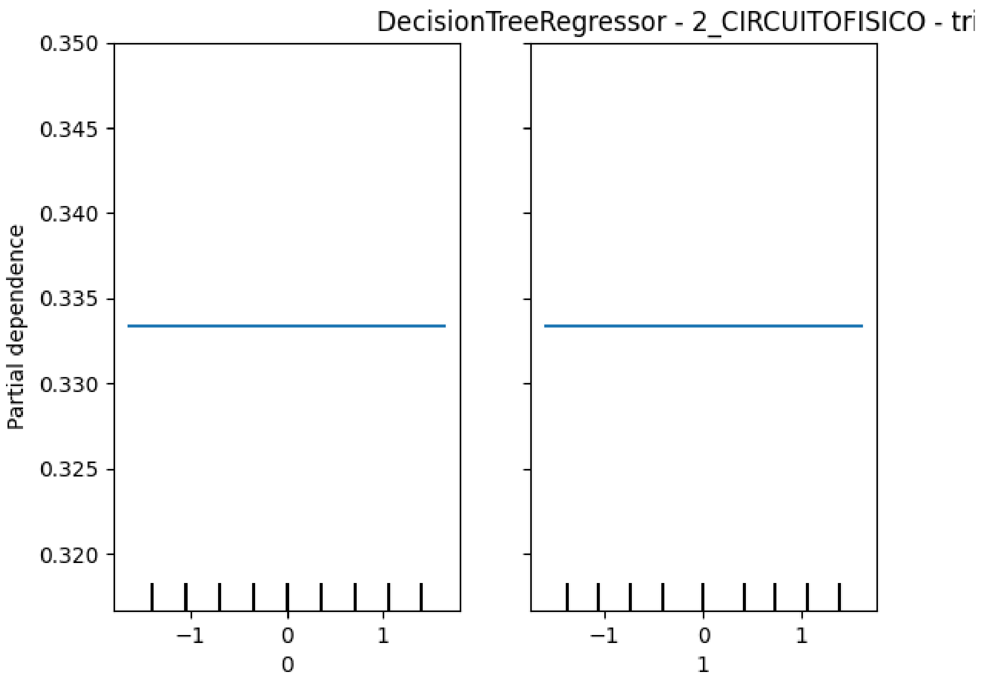

DecisionTreeRegressor - ONOFF - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

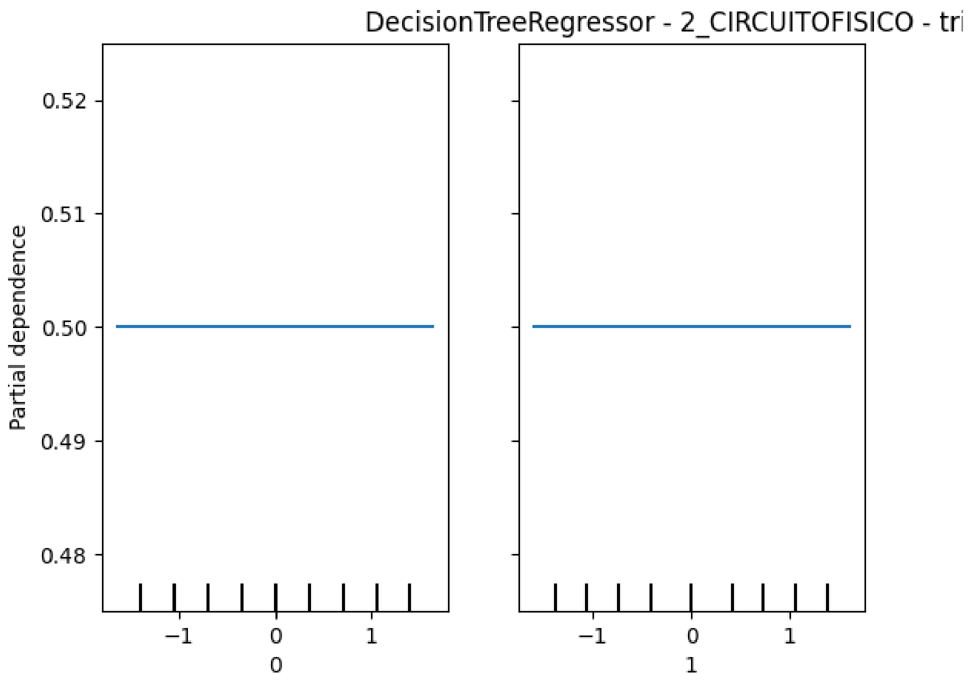

DecisionTreeRegressor - 3_DUPLATAREFA - tria
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

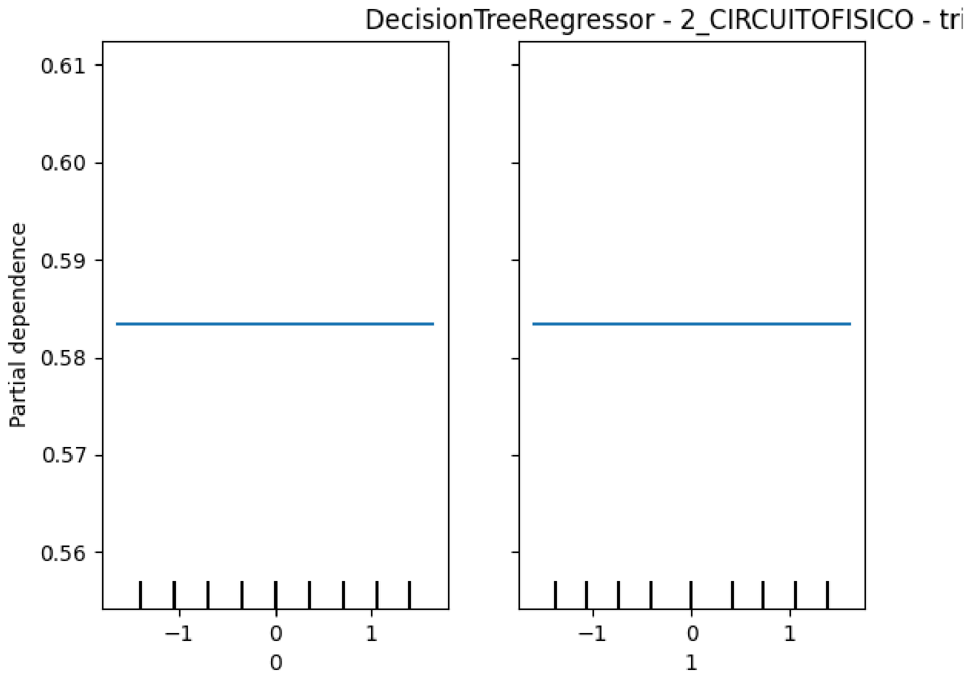

DecisionTreeRegressor - 3_DUPLATAREFA - tria
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

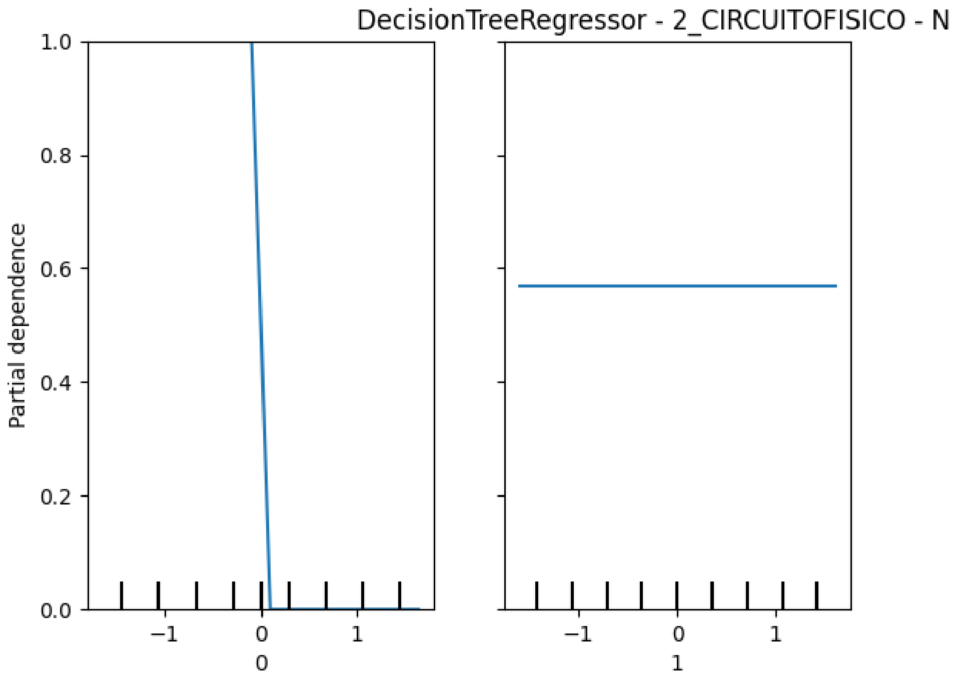

DecisionTreeRegressor - 3_DUPLATAREFA - tria
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

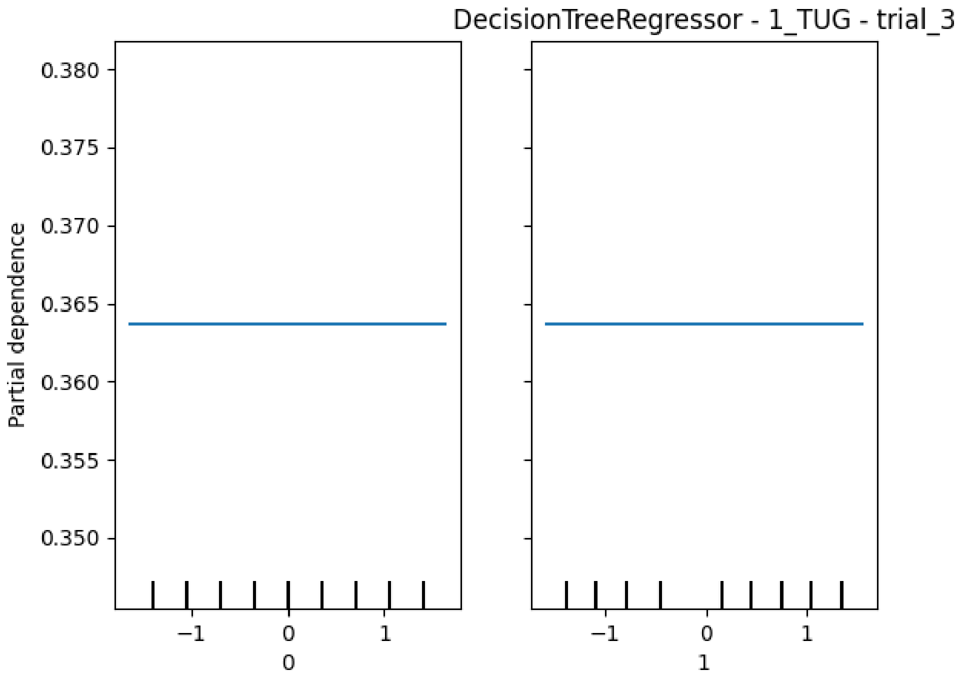

DecisionTreeRegressor - 3_DUPLATAREFA - Nc
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

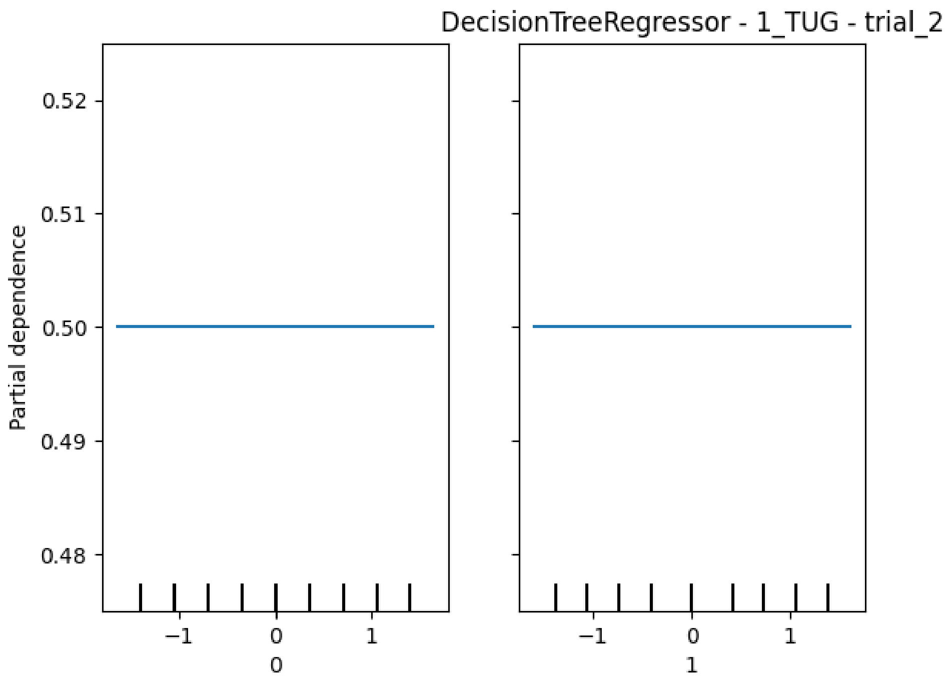

DecisionTreeRegressor - 2_CIRCUITOFISICO - tri
Partial dependence
0.350
0.345
0.340
0.335
0.330
0.325
0.320
−1
0
1
0
1

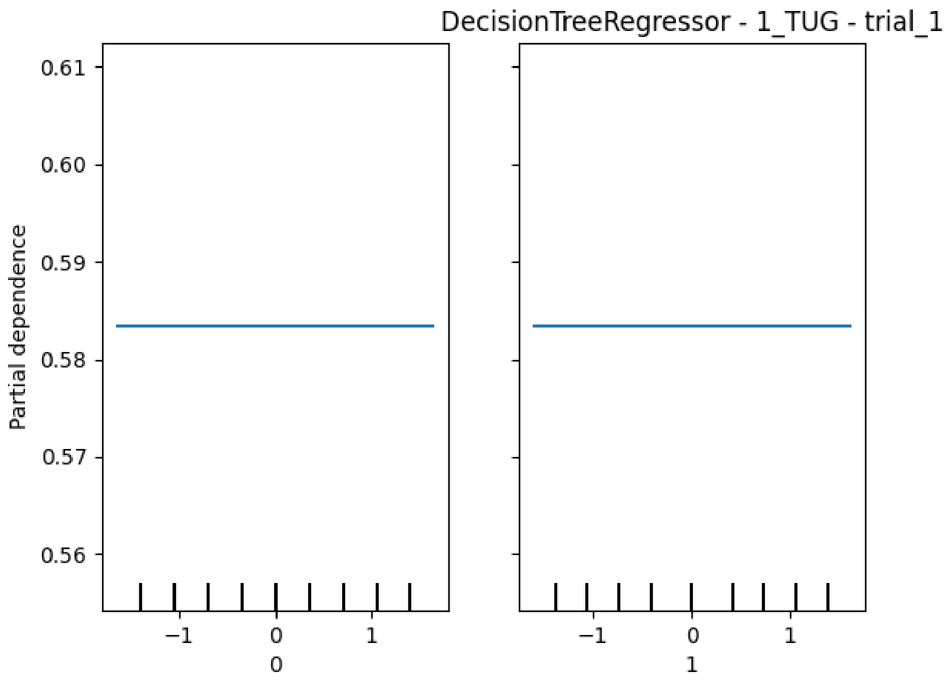

DecisionTreeRegressor - 2_CIRCUITOFISICO - tri
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
−1
0
1
1

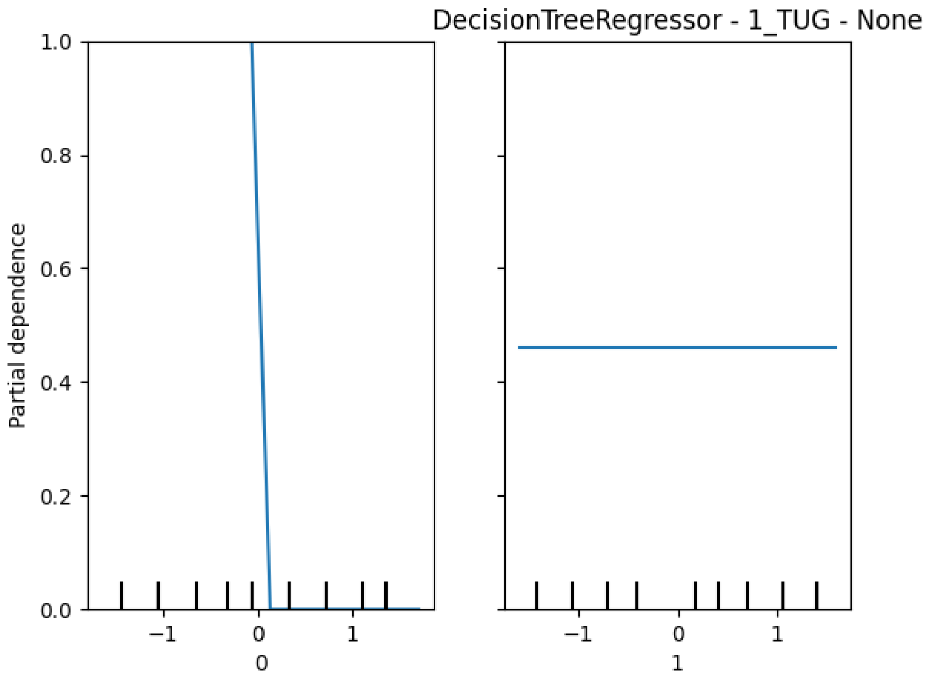

DecisionTreeRegressor - 2_CIRCUITOFISICO - tri
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
−1
0
1
1

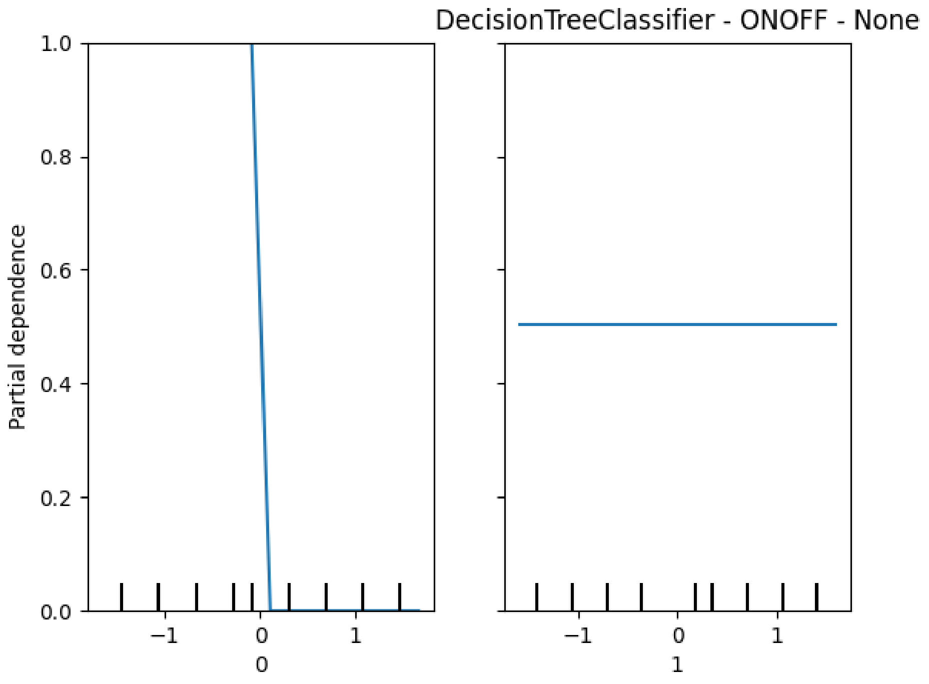

DecisionTreeRegressor - 2_CIRCUITOFISICO - N
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
−1
0
1
1

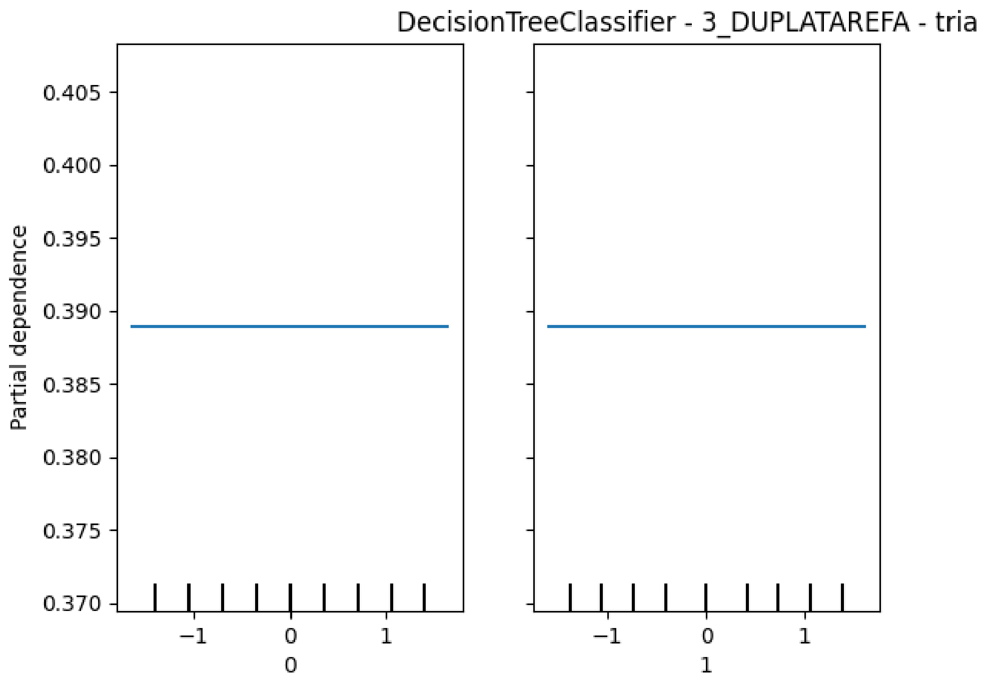

DecisionTreeRegressor - 1_TUG - trial_3
Partial dependence
0.380
0.375
0.370
0.365
0.360
0.355
0.350
−1
0
1
0
−1
0
1
1

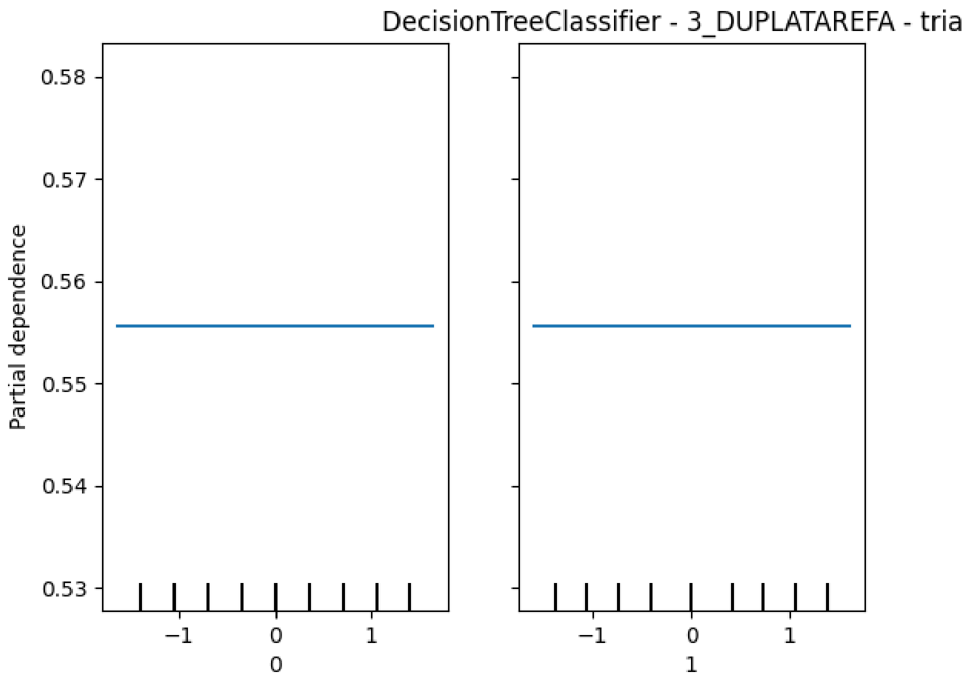

DecisionTreeRegressor - 1_TUG - trial_2
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
−1
0
1
1

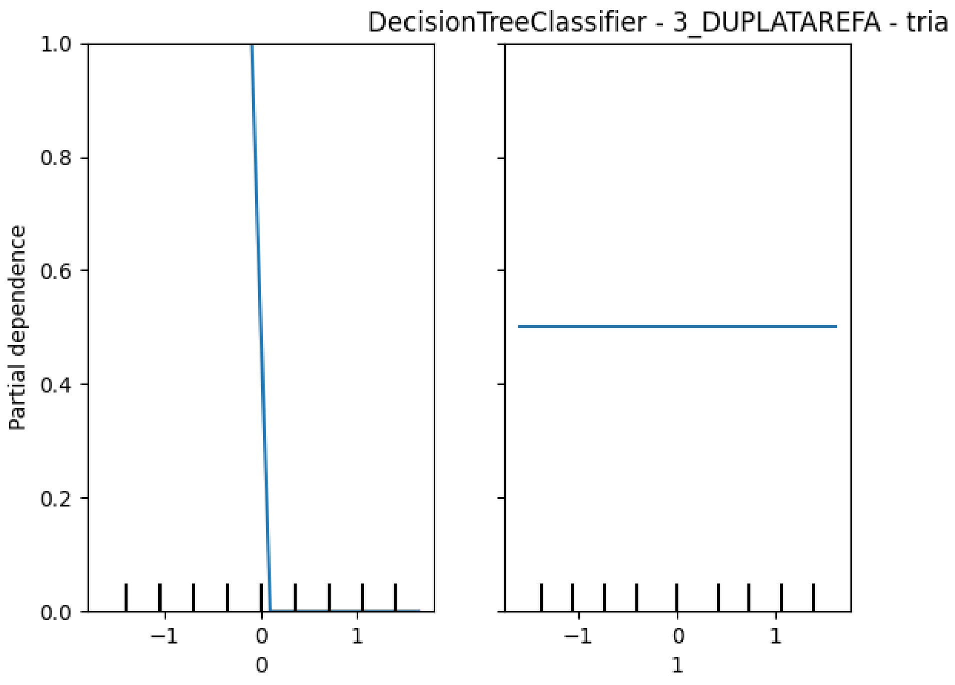

DecisionTreeRegressor - 1_TUG - trial_1
Partial dependence
0.61
0.60
0.59
0.58
0.57
0.56
−1
0
1
0
−1
0
1
1

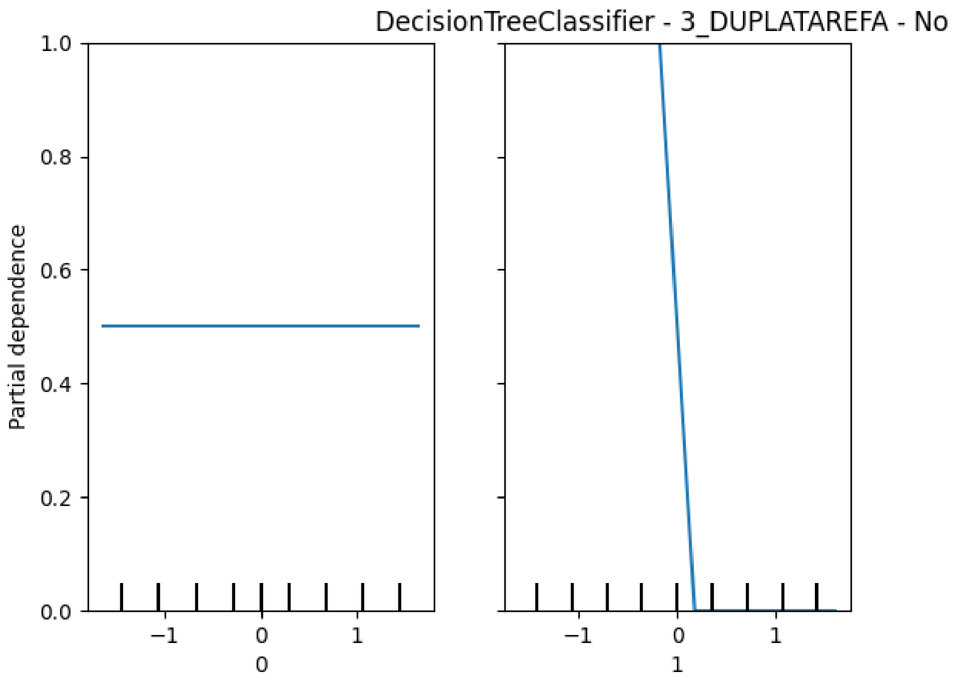

DecisionTreeRegressor - 1_TUG - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

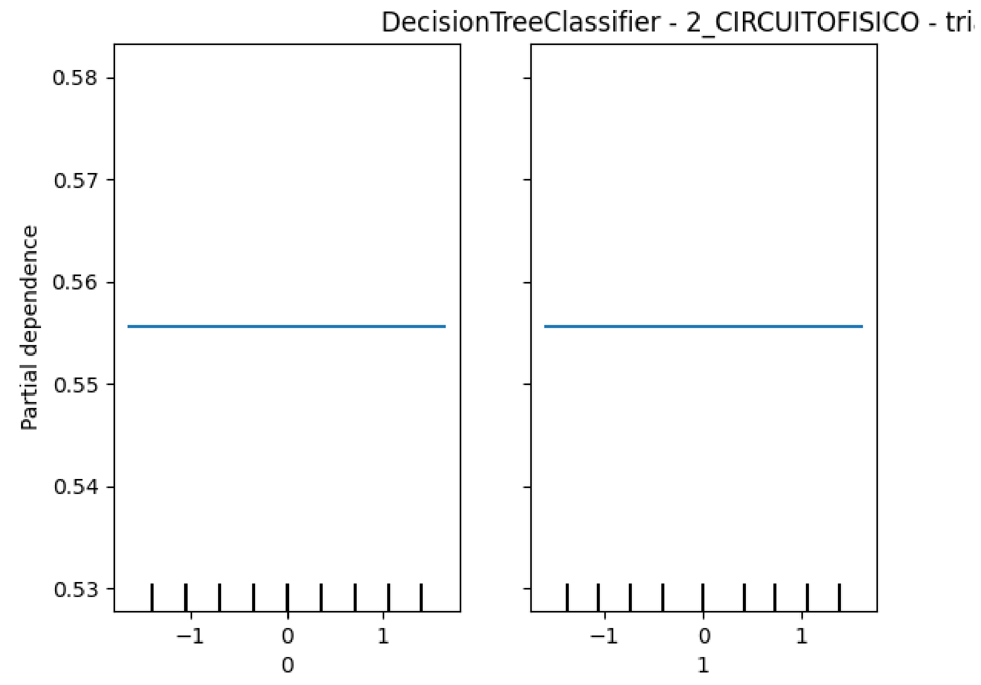

DecisionTreeClassifier - ONOFF - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

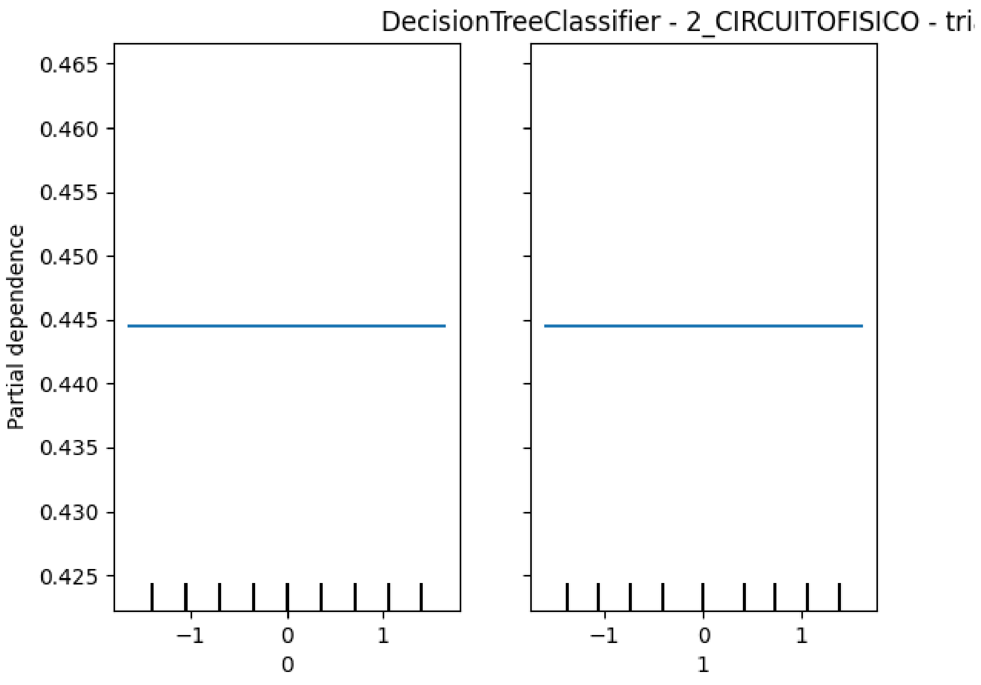

DecisionTreeClassifier - 3_DUPLATAREFA - tria
Partial dependence
0.405
0.400
0.395
0.390
0.385
0.380
0.375
0.370
−1
0
1
0
1

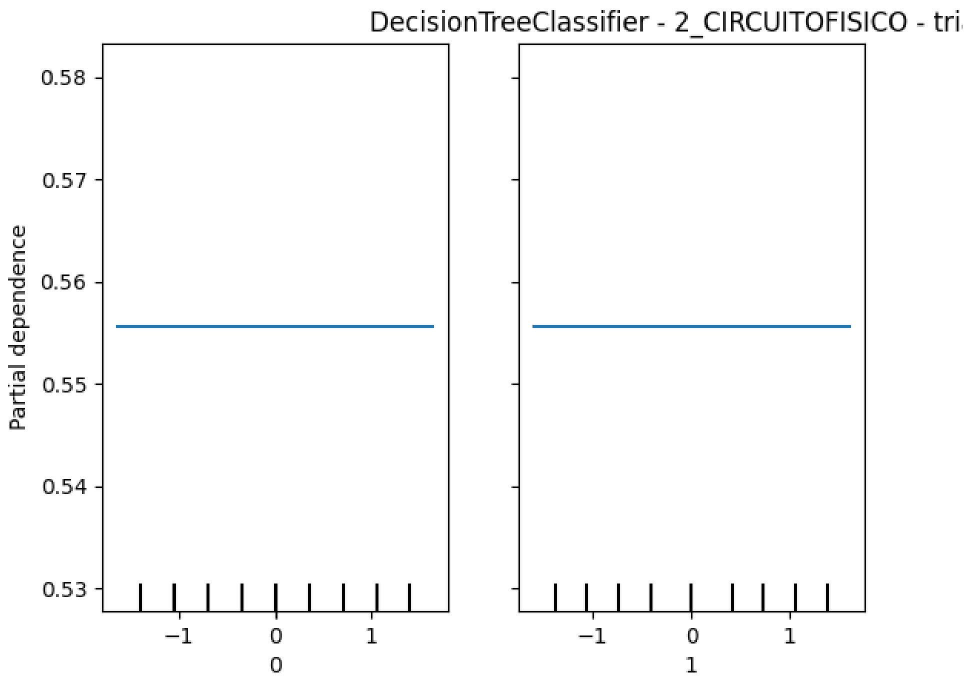

DecisionTreeClassifier - 3_DUPLATAREFA - tria
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

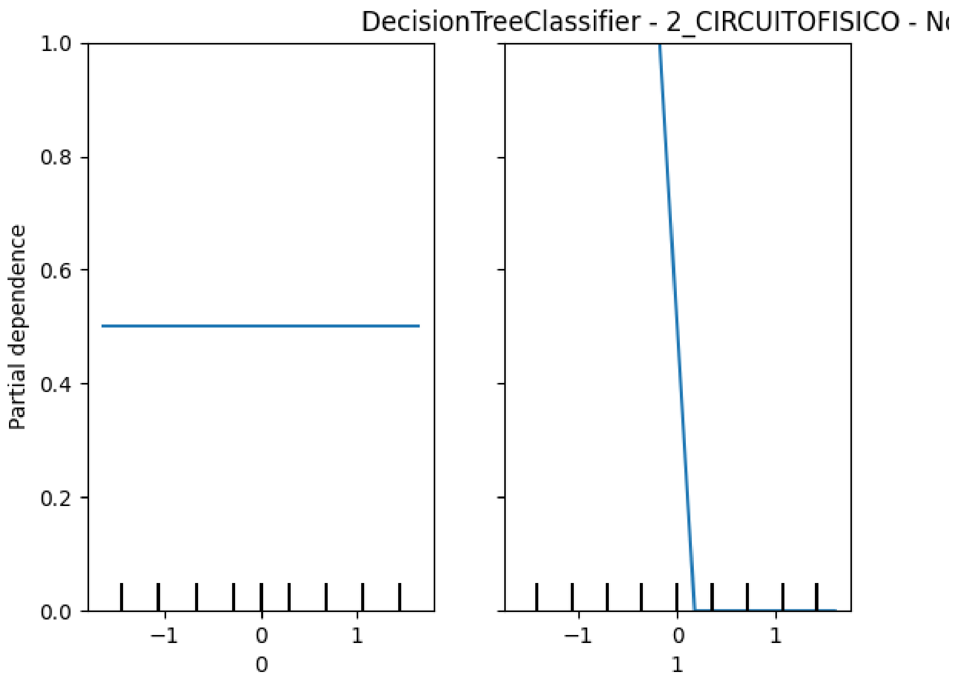

DecisionTreeClassifier - 3_DUPLATAREFA - tria
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

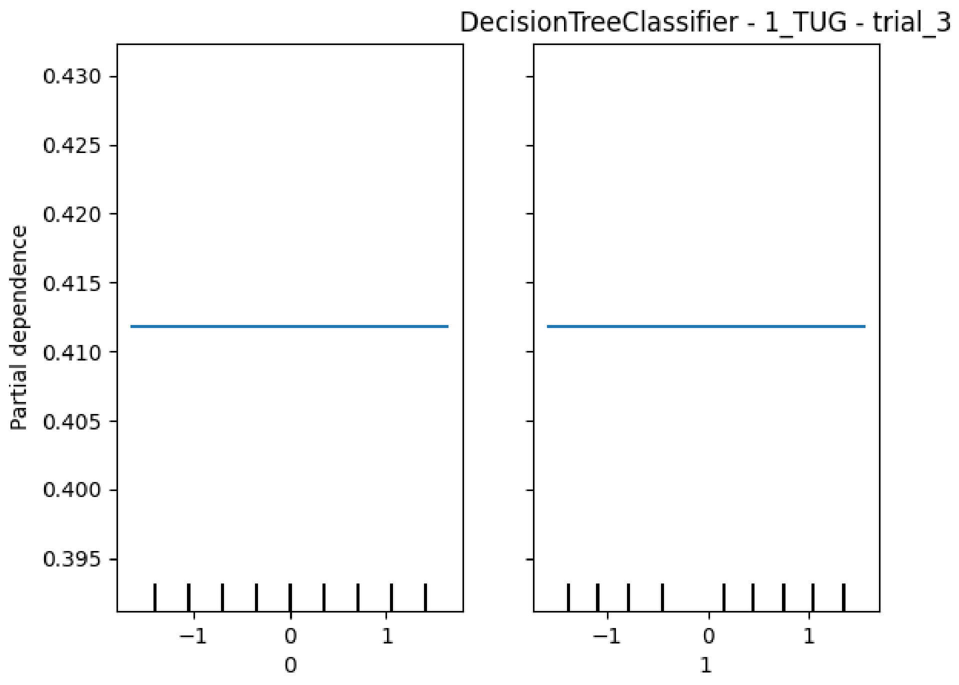

DecisionTreeClassifier - 3_DUPLATAREFA - No
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

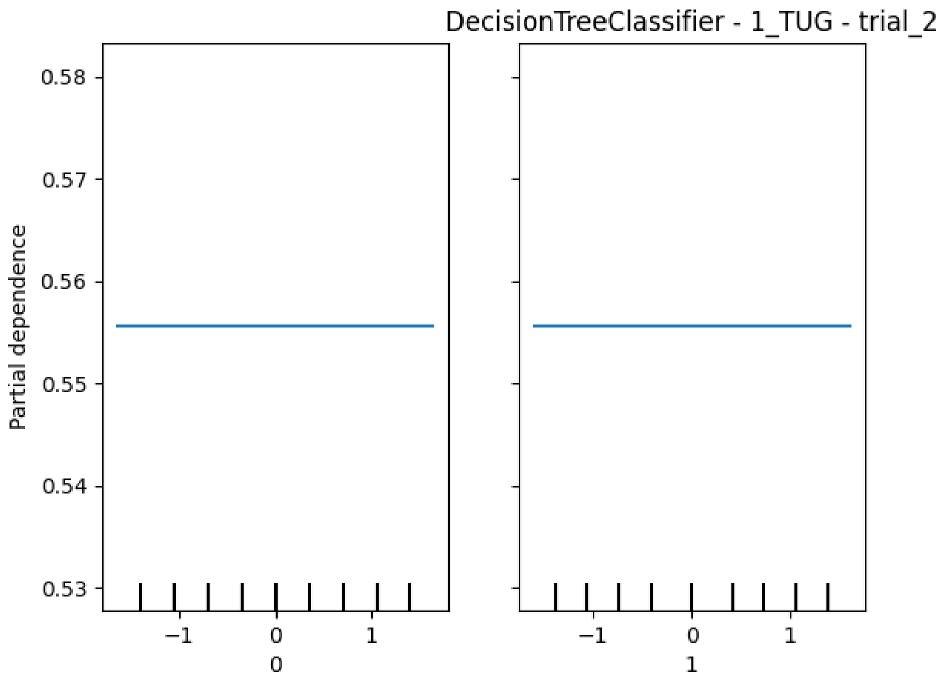

DecisionTreeClassifier - 2_CIRCUITOFISICO - tri
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

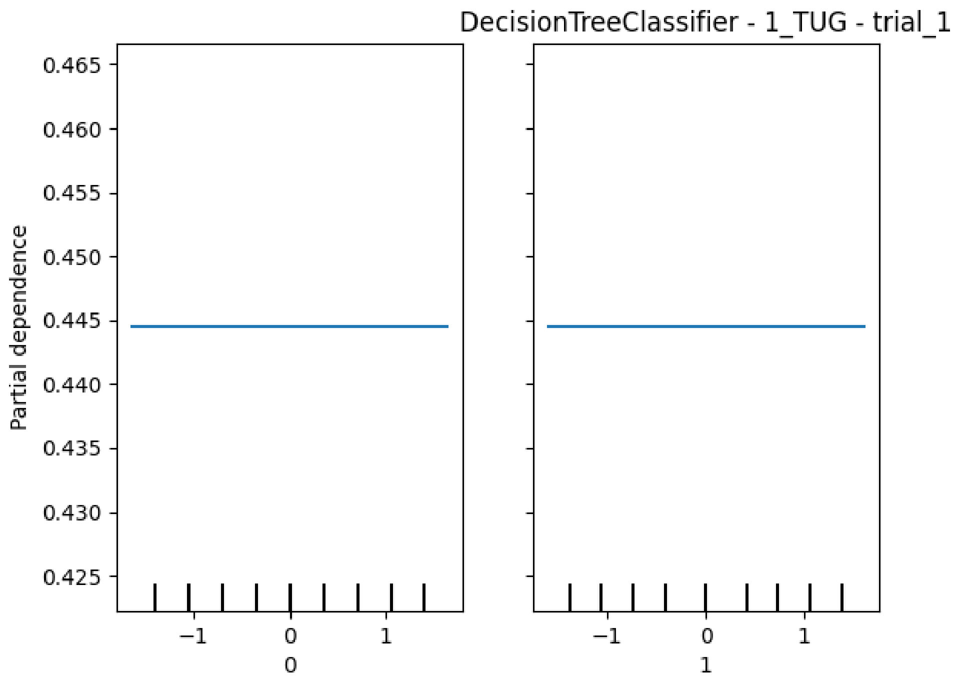

DecisionTreeClassifier - 2_CIRCUITOFISICO - tri
Partial dependence
0.465
0.460
0.455
0.450
0.445
0.440
0.435
0.430
0.425
−1
0
1
0
1

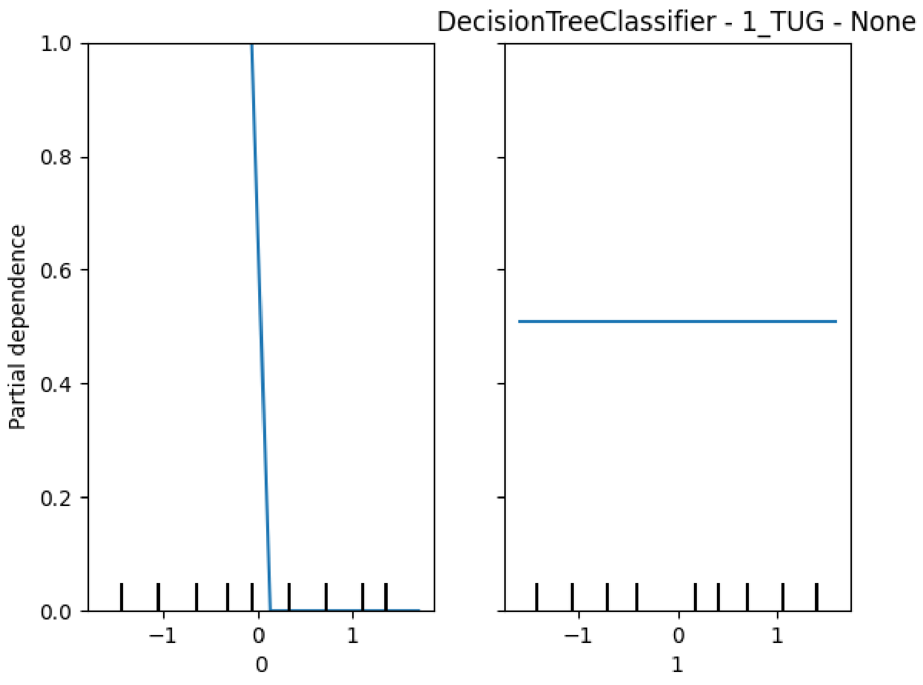

DecisionTreeClassifier - 2_CIRCUITOFISICO - tri
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

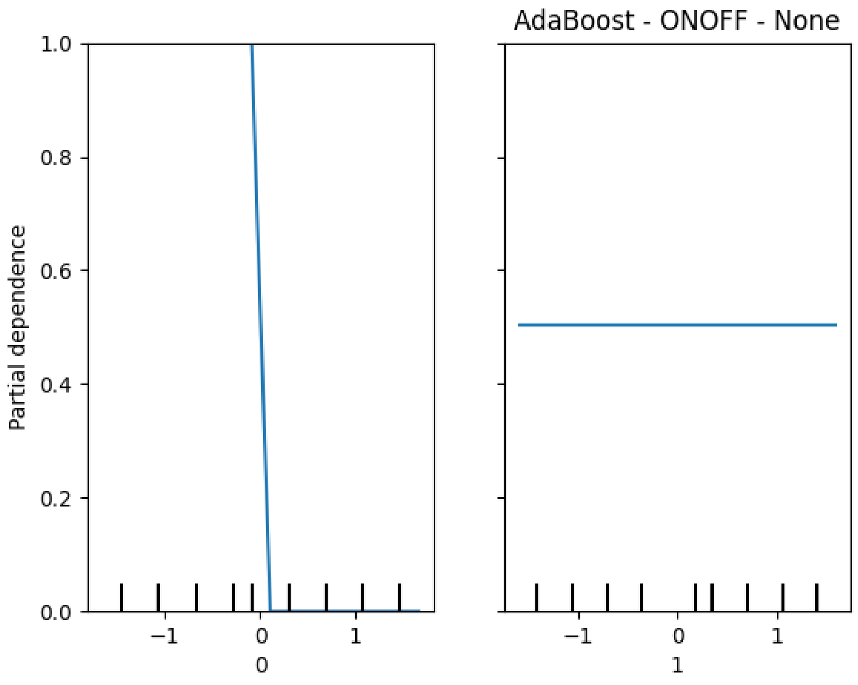

DecisionTreeClassifier - 2_CIRCUITOFISICO - N
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

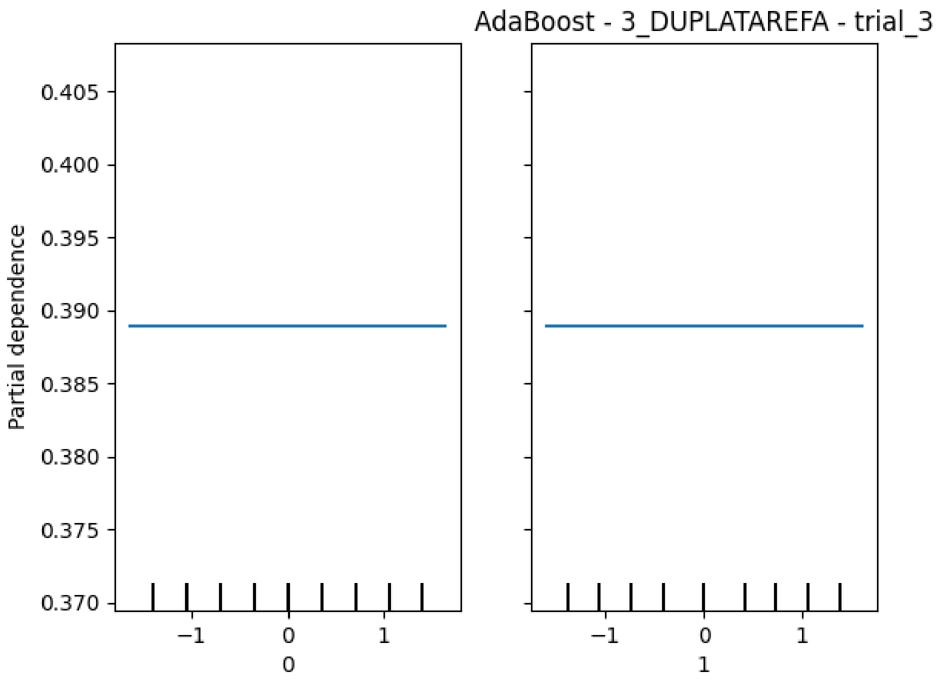

DecisionTreeClassifier - 1_TUG - trial_3
Partial dependence
0.430
0.425
0.420
0.415
0.410
0.405
0.400
0.395
−1
0
1
0
1

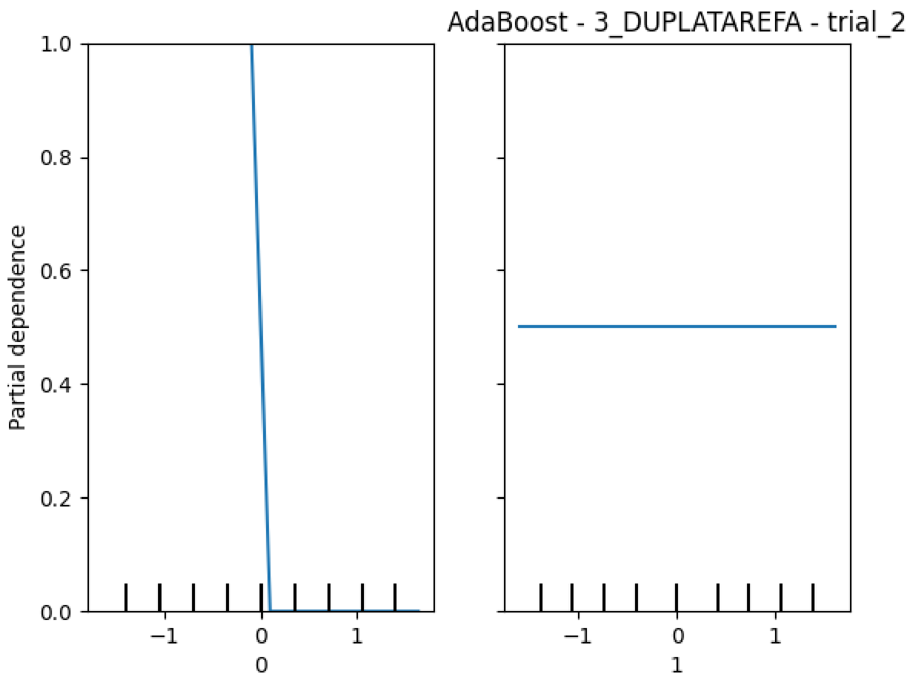

DecisionTreeClassifier - 1_TUG - trial_2
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
1

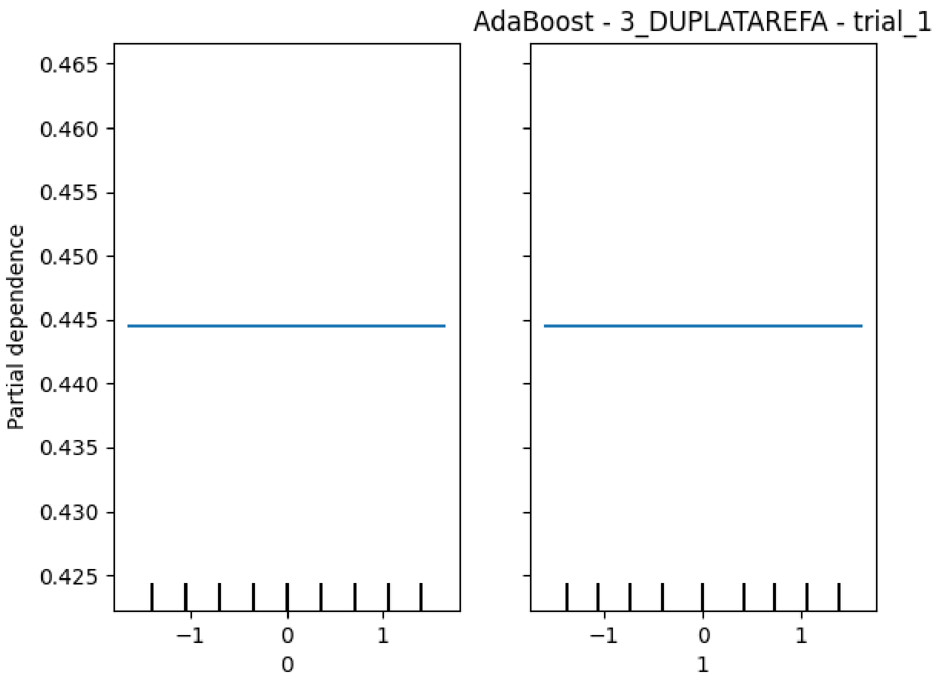

DecisionTreeClassifier - 1_TUG - trial_1
Partial dependence
0.465
0.460
0.455
0.450
0.445
0.440
0.435
0.430
0.425
−1
0
1
0
1

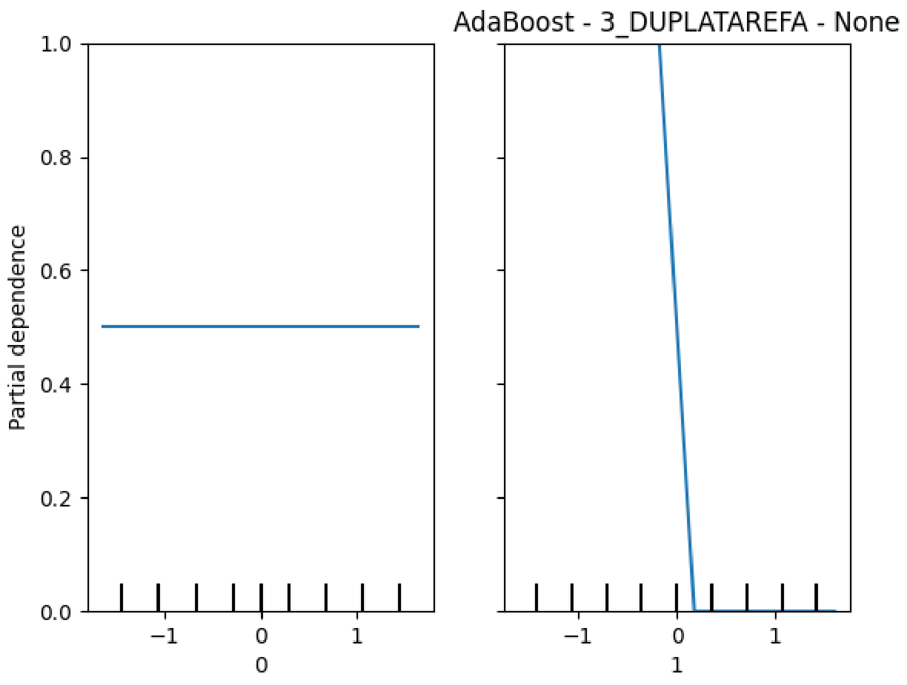

DecisionTreeClassifier - 1_TUG - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

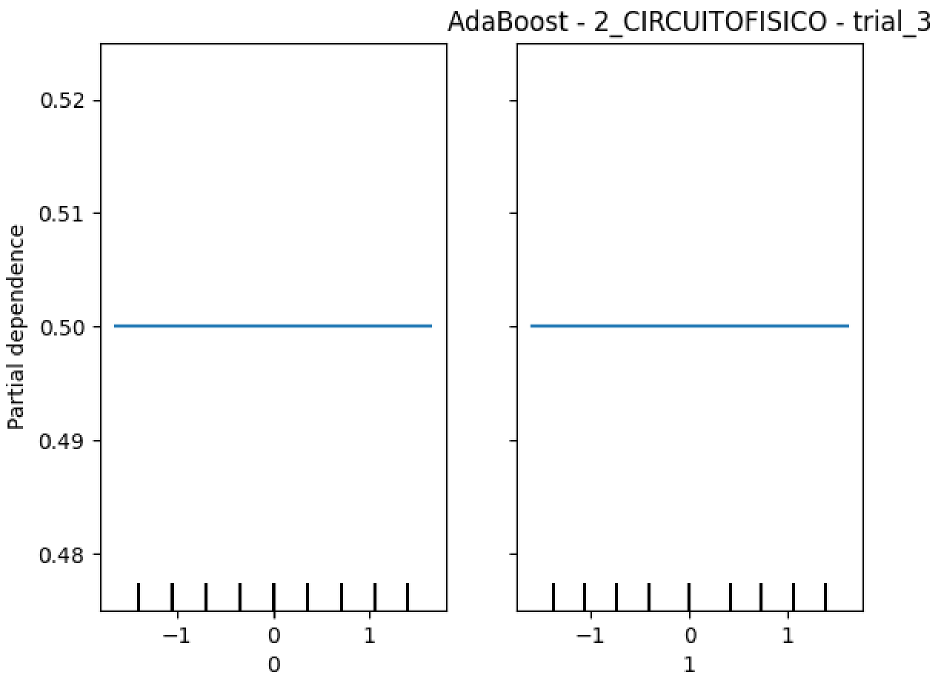

AdaBoost - ONOFF - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

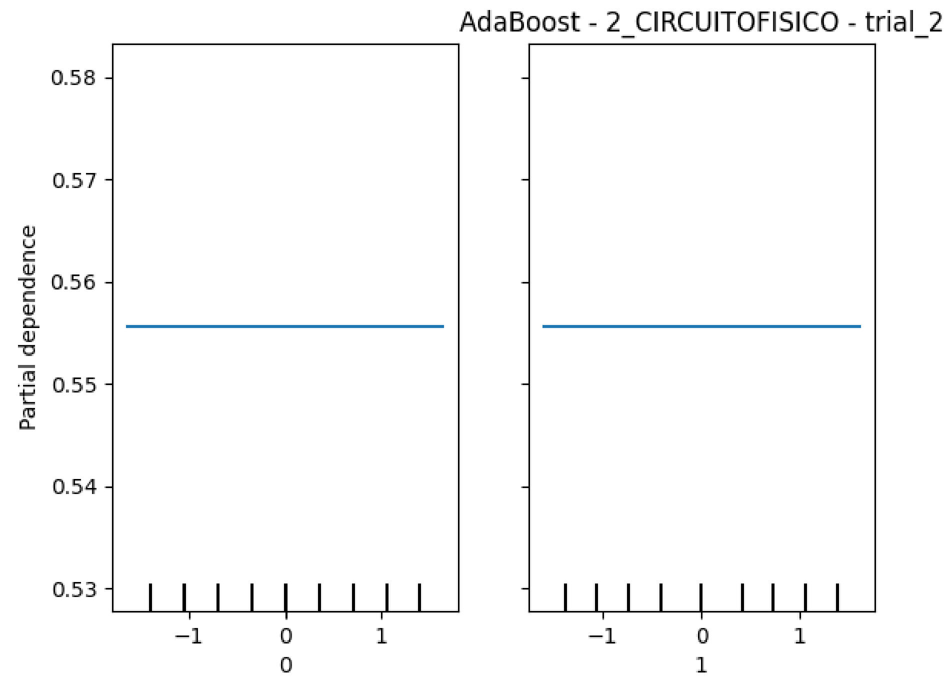

AdaBoost - 3_DUPLATAREFA - trial_3
Partial dependence
0.405
0.400
0.395
0.390
0.385
0.380
0.375
0.370
−1
0
1
0
1

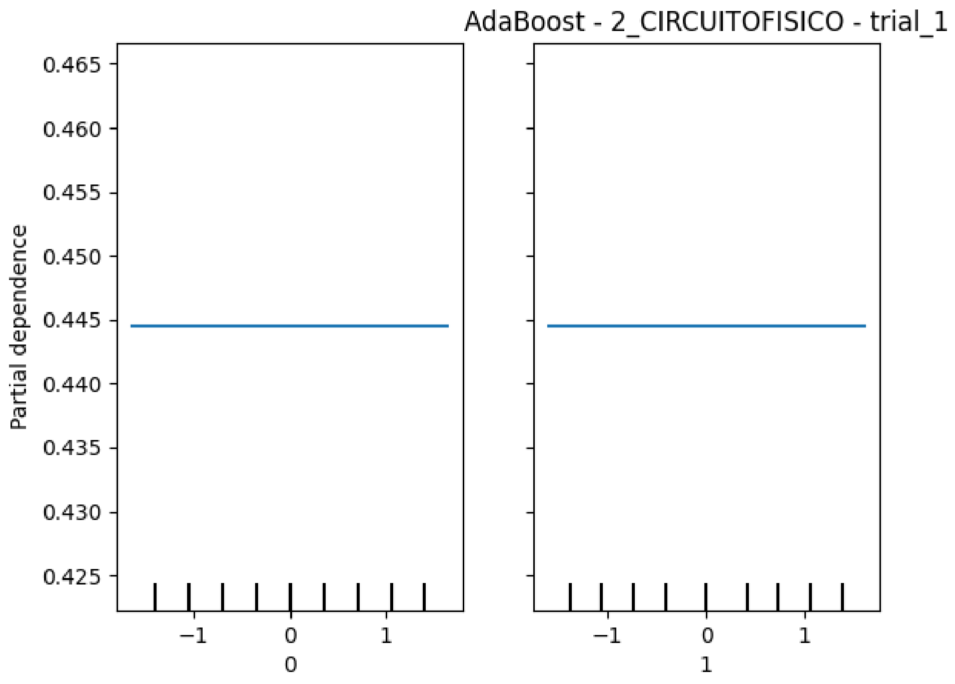

AdaBoost - 3_DUPLATAREFA - trial_2
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

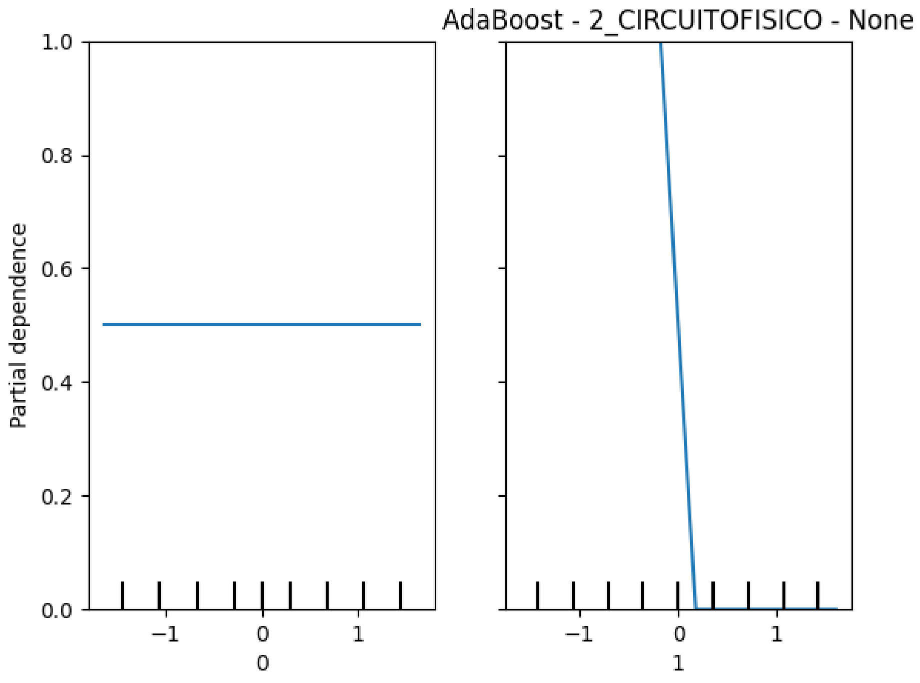

AdaBoost - 3_DUPLATAREFA - trial_1
Partial dependence
0.465
0.460
0.455
0.450
0.445
0.440
0.435
0.430
0.425
−1
0
1
0
1

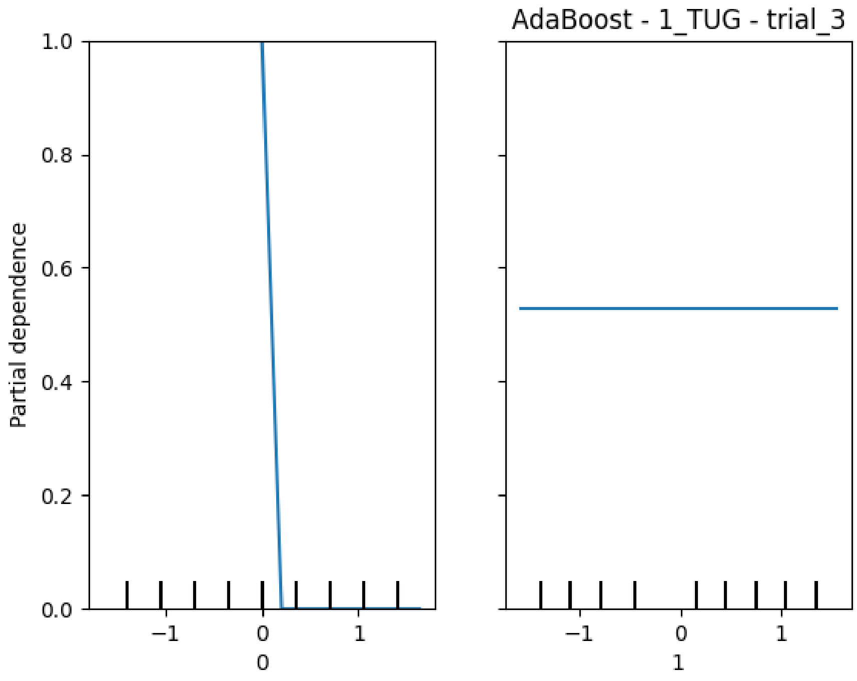

AdaBoost - 3_DUPLATAREFA - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1

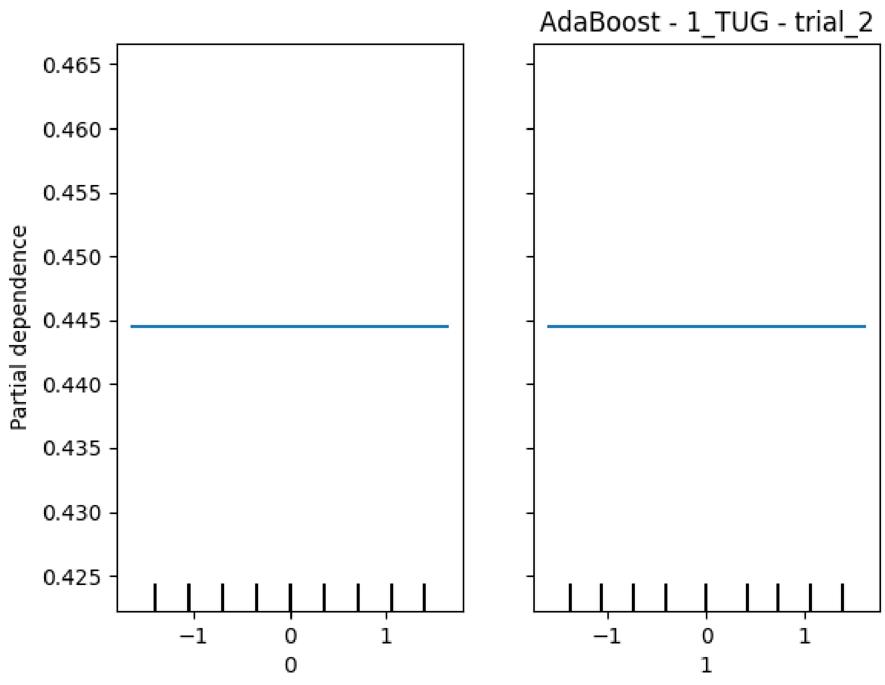

AdaBoost - 2_CIRCUITOFISICO - trial_3
Partial dependence
0.52
0.51
0.50
0.49
0.48
−1
0
1
0
1

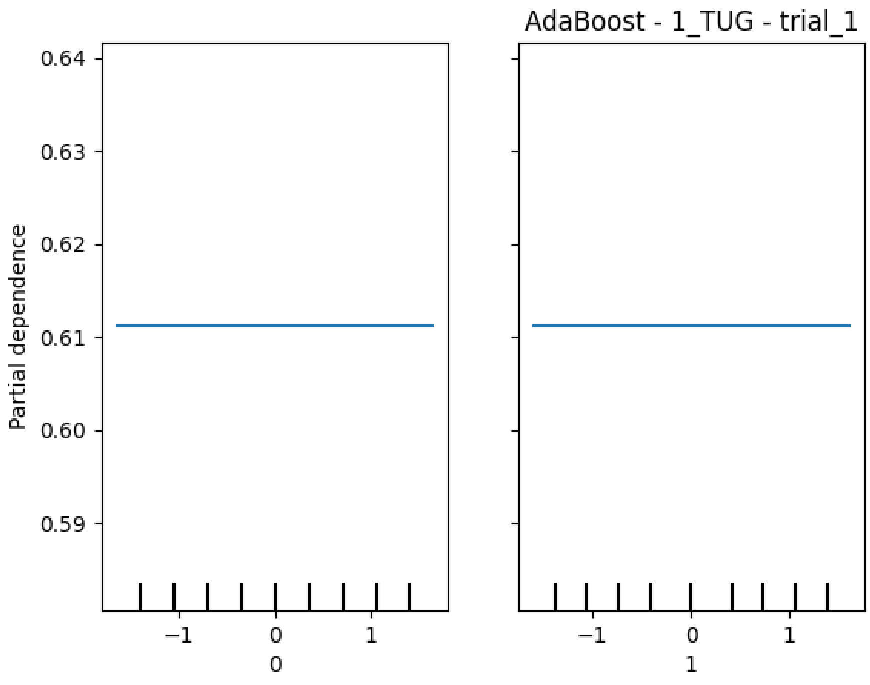

AdaBoost - 2_CIRCUITOFISICO - trial_2
Partial dependence
0.58
0.57
0.56
0.55
0.54
0.53
−1
0
1
0
−1
0
1
1

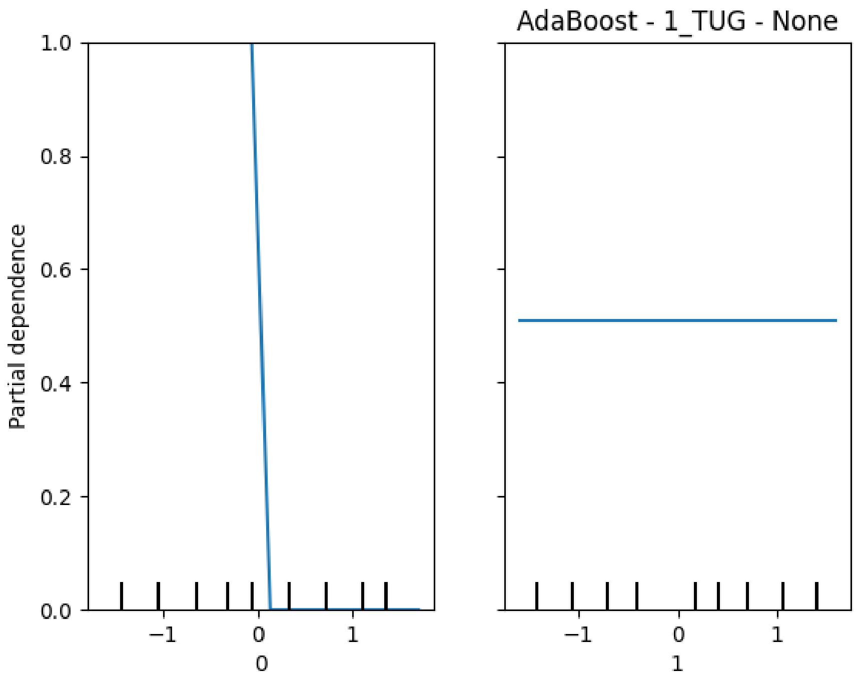

AdaBoost - 2_CIRCUITOFISICO - trial_1
Partial dependence
0.465
0.460
0.455
0.450
0.445
0.440
0.435
0.430
0.425
−1
0
1
0
−1
0
1
1


AdaBoost - 2_CIRCUITOFISICO - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1


AdaBoost - 1_TUG - trial_3
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1


AdaBoost - 1_TUG - trial_2
Partial dependence
0.465
0.460
0.455
0.450
0.445
0.440
0.435
0.430
0.425
−1
0
1
0
−1
0
1
1


AdaBoost - 1_TUG - trial_1
Partial dependence
0.64
0.63
0.62
0.61
0.60
0.59
−1
0
1
0
−1
0
1
1


AdaBoost - 1_TUG - None
Partial dependence
1.0
0.8
0.6
0.4
0.2
0.0
−1
0
1
0
1