Map-Optimize-Learn: A Data Transformation, Feature Selection and Deep Learning Based Strategy to Improve Predictions on Tabular Dataset

Mario Tasso Ribeiro Serra Neto
Computer Science Doctoral Degree
Computer Science department
2024

Supervisor
Inês de Castro Dutra, Assistant Professor, University of Porto

Abstract

This work introduces Map-Optimize-Learn (MOL), a pipeline that leverages expert knowledge from the fields of Feature Selection (FS), Deep Learning (DL), and optimization to improve prediction quality and provide an additional layer of interpretation for results obtained from Tabular datasets. The objective is to investigate the effectiveness of incorporating this expert knowledge in addressing diverse Machine Learning (ML) classification problems.

The MOL pipeline incorporates various components, including FS strategies, Swarm Intelligence (SI) algorithms, and a Convolutional Neural Network (CNN) to construct different versions of MOL models. These models are specifically designed to tackle 2D tabular dataset problems by transforming the representation of 1D samples into 2D, thereby generating valuable patterns for classification tasks. MOL learns and adapts multiple information about the features to create a 3D dataset for the CNN architecture.

To validate the effectiveness of the approach, a range of benchmark and real-world problems are selected to evaluate MOL's performance in different scenarios. The proposed approach is applied to 13 datasets, including 10 benchmark problems and 3 real-world problems. Results are compared against state-of-the-art ML algorithms tailored for tabular datasets. Additionally, comparisons are made against popular FS strategies, as well as deep learning and data transformation-based approaches. This comprehensive evaluation aims to showcase the applicability of the proposed method across a wide range of problem domains.

The findings of this research highlight the potential of expert knowledge acquired from FS and optimization techniques to enhance interpretability and improve results for tabular datasets. The study also presents the advantages and disadvantages of the MOL strategy, providing valuable insights into its effectiveness.

Resumo

Este trabalho apresenta o Map-Optimize-Learn (MOL), um *pipeline* que aproveita o conhecimento especializado dos campos Feature Selection (FS), Deep Learning (DL) e tarefas de otimização para melhorar a qualidade da previsão e fornecer uma camada adicional de interpretação para os resultados obtidos de conjuntos de dados Tabulares. O objetivo é investigar a eficiência da introdução desse conhecimento para resolver diversos problemas de classificação com Machine Learning (ML).

O *pipeline* MOL incorpora vários componentes, incluindo estratégias FS, algoritmos Swarm Intelligence (SI) e um Convolutional Neural Network (CNN), para construir diferentes versões de modelos MOL. Esses modelos são projetados especificamente para lidar com problemas de conjunto de dados tabulares 2D, transformando a representação de amostras 1D em 2D, gerando assim padrões para tarefas de classificação. MOL aprende e adapta várias informações sobre os recursos para criar um conjunto de dados 3D para a arquitetura da CNN.

Para validar a eficácia do proposto, uma série de benchmarks e problemas do mundo real são selecionados para avaliar o desempenho do MOL em diferentes cenários. A abordagem proposta é aplicada a 13 conjuntos de dados, incluindo 10 problemas de *benchmark* e 3 problemas do mundo real. Os resultados são comparados com algoritmos de ML de estado da arte desenhados para conjuntos de dados tabulares. Além disso, são feitas comparações com estratégias populares de FS, bem como abordagens baseadas em DL e transformação de dados. Essa avaliação visa mostrar a aplicabilidade do método proposto em uma ampla gama de domínios de problemas.

Os resultados desta pesquisa destacam o potencial do conhecimento especializado adquirido de FS e algoritmos de otimização para melhorar a interpretabilidade e os resultados para conjuntos de dados tabulares. O estudo também apresenta as vantagens e desvantagens da estratégia MOL, fornecendo informações valiosas sobre sua eficácia,

Acknowledgments

First and foremost, I extend my deepest gratitude to my grandmother, Dirce. Her guidance, endless support, and faith in me have been the bedrock of my journey.

I owe a debt of gratitude to my parents, whose hard work, dedication, and resilience have not only been a model for me to emulate but have also instilled in me the values of perseverance and integrity.

To my dear girlfriend, your patience, encouragement, and belief in my dreams have been a source of strength. Your support has been a beacon, guiding me through moments of doubt and inspiring me to never give up.

I extend my gratitude to my advisor and friend, Professor Inês. Her invaluable guidance and steadfast support throughout my academic journey have been instrumental in my growth.

To my former professors in Brazil, I am grateful for your continual advice and assistance. Your commitment to teaching and willingness to extend a helping hand every day have impacted my educational experience.

Lastly, but certainly not least, I extend my thanks to my longstanding friends and the new ones I have made along this journey. Whether near or far, your companionship and the moments we have shared have enriched my life immeasurably.

Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ADAM	Adaptive Moment estimation
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
CV	Cross-Validation
DL	Deep Learning
DFFN	Deep Feed Forward Network
DT	Decision Tree
EPSO	Evolutionary Particle Swarm Optimization
FF	Feed Forward Neural Network
FS	Feature Selection
GS	Grid Search
GPU	Graphics Processing Unit
IE	Input Encoding
KNN	k-Nearest Neighbors
kCV	k-Fold Cross-Validation
LBFGS	Limited-memory Broyden–Fletcher–Goldfarb –Shanno
LGBM	Light Gradient Boosting Machine
LR	Logistic Regression
Net-DNF	Net-Disjunctive Normal Form
NS	Number of Solutions
MI	Mutual Information
MOL	Map-Optimize-Learn
ML	Machine Learning
PC	Pearson Correlation
PSO	Particle Swarm Optimization
SAINT	Self-Attention and Intersample Attention
SL	Supervised Learning
SVM	Support Vector Machine
SI	Swarm Intelligence
OHE	One-Hot-Encoding
OvA	One vs All
ResNet	Residual Neural Network
ReLU	Rectified Linear Unit
RF	Random Forest
RFE	Recursive Feature Elimination
TabNet	Attentive Interpretable Tabular Learning
XGB	Extreme Gradient Boosting

Chapter 1

Introduction

In the past decades, the concepts of adapting to new circumstances, detecting patterns and predicting new data found in the Machine Learning (ML) field were widely applied to solve a large variety of problems in multiple knowledge areas [1]. The field received even more attention with the advance on parallel computing, where several Graphics Processing Unit (GPU)'s are used to accelerate the training phase of the models. Several ML models were devised and each one has a set of properties and limitations regarding the type of features, target variable, time complexity, among other characteristics. Arguably, the most popular ML algorithm, the Artificial Neural Networks (ANN), has been widely used to learn from data in various domains and of various types: temporal, images, videos, among others.

The ANN model and others like Random Forest (RF), k-Nearest Neighbors (KNN) or Logistic Regression (LR) were initially constructed to handle tabular datasets, which are composed by a set of 1D instances with several features. In order to handle problems based on images, videos or text, the dataset has to be adapted into a tabular format. Later, with the advance of ML models research and the introduction of GPUs which could make it feasible, a field named Deep Learning (DL) emerged. DL is presented as an extension for the ANN models, where the most popular algorithm, the Convolutional Neural Network (CNN), has been outperforming other models [2]. However, its good performance has been mostly demonstrated through image datasets.

Inspired by the fact that CNN can capture signal from some hidden structure in the data, we propose to apply CNN to data in tabular format, but transforming this tabular data such that the 1D samples look like 2D images to the CNN. In order to generate these images, the Map-Optimize-Learn (MOL) strategy was devised [3]. In MOL, the dataset goes through a feature sorting step, followed by a data domain change that allows the data to be mapped into an image format. Last, the proposed method reduces the image shape through Feature Selection (FS), indicating the best subset of features that should be included to generate these images.

Feature Selection (FS) is a powerful technique employed to identify the optimal subset of features in a given dataset [4]. It involves the application of statistical methods or machine learning algorithms to gather valuable insights about the data and determine the importance of its features. By reducing the dimensionality of the data, FS greatly enhances the speed and complexity of the model. The statistical methods employed in the FS problem are also added as part of the MOL strategy, where they aid in determining the most effective order for incorporating selected features into the image format.

There have been a few attempts to take advantage of the success of neural network approaches on images by applying these models to tabular data [5]. They vary in the way the tabular data is represented [6], preprocessed [7] and in the neural network architecture used. None of them takes into account "informed" preprocessing, which can be useful in many domain areas, e.g., medicine.

When utilizing ML in practical applications, numerous domains necessitate extensive interpretation. Examples of such domains include Healthcare, Finance, Autonomous Vehicles, Legal and Ethical Implications, and more. Building upon this foundation, the objective of this work extends beyond delivering a pipeline with high classification accuracy. It aims to incorporate an additional layer of interpretation to the obtained results. By integrating knowledge into the image generation process, a deeper understanding of the model's behavior in relation to the classification task can be achieved. Furthermore, this approach enhances understanding of why a specific set of features holds significance for the dataset, thereby contributing to domain knowledge.

The purpose of this thesis is to investigate the potential of utilizing expert knowledge acquired from the field of FS and optimization tasks to enhance the accuracy of predictions, surpassing the performance of state-of-the-art ML algorithms and other data transformation methods. To demonstrate the versatility of this approach in solving diverse problems, benchmark and real-world problems are selected as the focal point. The problem is transformed using the MOL strategy, incorporating statistical methods, Swarm Intelligence (SI) algorithms, and a CNN to construct different versions of MOL aimed at improving prediction outcomes. The proposed approach is applied to each dataset, and the results are compared against baseline and state-of-the-art ML algorithms designed for tabular datasets. Furthermore, the noteworthy contributions include:

- Introducing MOL, a novel strategy that leverages the advantages offered by CNNs while being tailored for tabular data [8].
- Exploring a range of feature selection (FS) methods within MOL to identify important and relevant features within tabular datasets.
- Incorporating multiple swarm intelligence-based optimization techniques to separate sub optimal solutions, thereby enhancing overall performance.
- Introducing an interpretative layer to facilitate the understanding of the CNN's outcomes.

This thesis is organized as follows:

1. **Chapter 2 on page 5**: provides a comprehensive overview of essential concepts and algorithms in ML, DL, SI, and FS, foundational knowledge to understand the development of this thesis;

2. **Chapter 3 on page 37**: offers an in-depth exploration of the current state-of-the-art in data transformation and FS. These pivotal domains are integral to the broader scope of the MOL strategy;

3. **Chapter 4 on page 47**: presents the MOL strategy, offering an overview of the strategy. Details the Map, Optimize, and Learn phases, explaining how these phases interact with each other. Describes the pipeline employed to generate viable images and results. Also includes a brief discussion introducing MOL as a Tabular Learning Method;

4. **Chapter 5 on page 53**: denotes the conducted experiment, encompassing a selection of 10 Benchmark and 3 Real-World scenarios. Provides essential information concerning hyperparameters and relevant settings employed to configure SI and ML algorithms. Provides insights into each dataset, elucidating the variations in data types, instances, and features across the experiments;

5. **Chapter 6 on page 61**: discuss the achieved results for each dataset. These results were derived and compared against various combinations of FS strategies, ML algorithms, and the most effective tabular learning method. The analysis involves evaluating the simulations, generated images, and outcomes using ML metrics. Additionally, the significance among the strategies is determined through p-test analyses.

6. **Chapter 7 on page 101**: summarizes the thesis by presenting its conclusions, highlighting the significant knowledge gained during the experiments, and suggesting potential avenues for future research.

Chapter 2

Background

This chapter describes fundamental concepts that support the remainder of this text. Section 2.1 details the standards of Machine Learning (ML) and its application to tabular datasets. Section 2.1.3 on page 10 points out traditional ML algorithms used to solve diversified problems. Section 2.2 on page 21 details the Deep Learning (DL) field. Section 2.4 on page 29 details optimization algorithms used in this thesis. Finally, Section 2.3 on page 25 points out the fundamentals of the feature selection problem for tabular datasets.

2.1 Machine Learning

Machine learning is organized into a taxonomy, based on the desired outcome of the model [9]. It is also a sub field of Artificial Intelligence (AI) that explores a computer's ability to adapt to new circumstances, identify patterns, and make predictions with new data [1]. These capabilities are facilitated by models created through the knowledge acquired during the learning phase of one or multiple algorithms.

The field of ML is divided into various branches, with different approaches being employed depending on the specific problem at hand. These approaches heavily rely on data, which can be sourced from diverse origins and may encompass mixed data types. Often, data requires preprocessing and transformation to facilitate the learning process, rendering it comprehensible and meaningful. Subsequently, validation and evaluation phases are applied to assess the model's performance in making predictions with new data. These variations and processes for constructing a viable ML model are elaborated upon in the following subsections.

Among the types of ML, tasks can be categorized into four distinct types:

- **Supervised learning**: in this type, both the input data and the corresponding outputs or labels are available for the learning process.;
- **Unsupervised learning**: this category encompasses situations where there are no

predefined labels or hints regarding the correct outputs, and the algorithm seeks to discover inherent patterns or structures within the data.;

- **Semi-supervised learning**: semi-supervised learning combines elements of both supervised and unsupervised learning. While there are outputs or labels available for some of the data, the remainder lacks any hint or guidance;
- **Reinforcement learning**: In reinforcement learning, algorithms learn a policy that dictates how to interact with the environment based on observations. Actions taken have consequences, typically measured through a reward mechanism, guiding the algorithm toward optimizing its behavior.

This thesis primarily addresses Supervised Learning (SL) problems where the data does not consist of images, sound, video, text, etc. Consequently, the following concepts discussed are specifically related to the Supervised Learning paradigm.

2.1.1 Supervised Learning

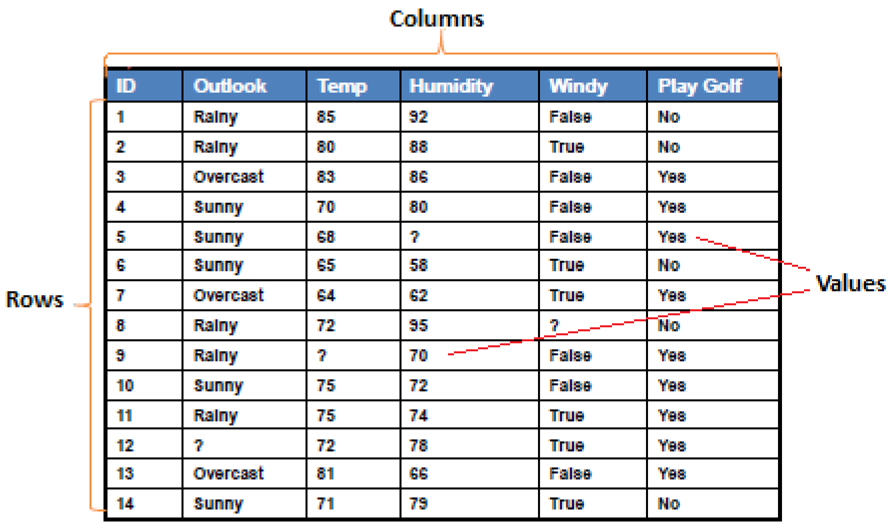
According to Witten et al. [10], SL approaches necessitate the presence of both inputs and outputs within the dataset. The inputs serve as the data that is fed into the chosen model to generate an outcome. However, the outputs within the dataset serve as indicators of the type of ML task that should be performed, where according to Goodfellow et al. [11], it can vary between:

- **Classification**: Among k categories, the model aims to identify which category is the correct for the input. The algorithm will produce a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$, where $y = f(x)$. The model will assign an input described by the vector x to a category identified by a number y. Goodfellow et al. [11] also mentions the multiclass classification problems, where f outputs a probability distribution over the number of classes found in the problem.
- **Regression**: In contrast to classification tasks, regression seeks to predict a numerical value based on given inputs. It operates similarly to classification but with a distinct output format. In regression, the algorithm generates a function $f : \mathbb{R}^n \to \mathbb{R}$ where it maps any n-dimensional input vector to a real number in $\mathbb{R}$.

As pointed out by Ayodele [12], classification problems align more closely with the Supervised Learning (SL) concept, whereas regression problems serve as an example of unsupervised learning embedded within the supervised framework.

To illustrate this, consider a tabular dataset as shown in Figure 2.1 on the next page, where weather conditions determine whether an individual, represented by the *ID* feature, will engage in golfing activities, indicated by the *PlayGolf* target feature.

The primary focus of this thesis revolves around classification problems. Nevertheless, for illustrative purposes, we can utilize the same dataset to exemplify regression problems. In this context, we would pivot from using the *PlayGolf* variable as the target and instead employ variables like Temperature or Humidity. This shift in perspective transforms the problem from determining whether a player would partake in golf to predicting humidity or temperature based on the conditions for golfing.


Columns
Rows
Values

ID	Outlook	Temp	Humidity	Windy	Play Golf
1	Rainy	85	92	False	No
2	Rainy	80	88	True	No
3	Overcast	83	86	False	Yes
4	Sunny	70	80	False	Yes
5	Sunny	68	?	False	Yes
6	Sunny	65	58	True	No
7	Overcast	64	62	True	Yes
8	Rainy	72	95	?	No
9	Rainy	?	70	False	Yes
10	Sunny	75	72	False	Yes
11	Rainy	75	74	True	Yes
12	?	72	78	True	Yes
13	Overcast	81	66	False	Yes
14	Sunny	71	79	True	No

Figure 2.1: Tabular data with instances and features (rows and columns)

Source: Sayad [13]

Assuming that this dataset is preprocessed, it would have 14 instances and 6 features, where the target variable *PlayGolf* will support the decision whether or not someone will play golf, therefore, splitting the data between inputs (set of features - X) and outputs (target variable - y). When observing each feature found in X, it is possible to notice the presence of *ID*, which is a simple identifier for each instance, which means that the input will have 4 features based on climate conditions.

With the inputs and output defined for our example dataset, ML algorithms used for SL approaches often learn the probability for the inputs, in the example above, that indicate the probability to play golf. However, if there are missing values, several models are not able to infer anything since they are data dependent [12]. The dataset presented in Figure 2.1 shows a few missing values represented by the question mark. To circumvent this issue, a preprocessing step is applied to the dataset, where the strategies used in this phase rely on the types of data.

2.1.2 Types of Data

A tabular dataset represents a set of features, where these features can be acquired from multiple sources in multiple applications. In social media, data is acquired from posts such as text, images, videos, etc. Not limited to these sources, it is possible to store the social relations between users, user-post, etc [14].

Even for data acquired at the same place, website or any other source, there could be distinct types of information about the same entity, these may vary between two main flavors of data [15]:

- **Quantitative data**: numbers and variables that can be measured, e.g. height, width, length, temperature;
- **Qualitative data**: characteristics of the feature that can be counted, e.g. the state of a machine between working, not working, suspended or waiting or, in a game, if the player is in game, waiting for an opponent, away from keyboard, etc.

Quantitative data can also be divided into two other branches:

- **Discrete data**: represented as integer numbers standing as indivisible entities, for instance: the number of persons in a family (1, 2, ..., N);
- **Continuous data**: numbers that can be divided, represents non integer numbers, for instance: height, weight, etc.

Qualitative data is divided into three distinct branches:

- **Binomial data**: mutually exclusive categories: good/bad, true/false, accept/reject;
- **Nominal data**: data that set a label or name for a specific feature, it does not have a order. For instance with the following question: Which one is your favourite sport?, it is possible to get answers like ’Basketball’, ’Soccer’, ’Tennis’, etc.
- **Ordinal data**: a data that carries an ordering. For instance, it is possible to measure the quality of a restaurant as: 1- Very Bad; 2- Bad; 3- Neutral; 4- Good; 5- Very Good.

Regarding the target variable, it is possible to find at the ML literature distinct value types for classification problems:

- **Binary**: an output variable represented by two classes (Binomial data);
- **Multiclass**: when a variable has more than two classes, e.g. a car, motorcycle or a bike are represented as a vehicle and not individually;
- **Multilabel**: assigns to each sample a set of target labels, for instance finding a genre of a movie among Horror, Romance, Adventure, etc;
- **Multioutput-multiclass**: a single model has to handle several joint classification tasks.

For the golf dataset, Figure (2.1 on page 7): *ID* is a discrete feature; *Outlook* is a nominal data; *Temperature* in this situation has integer values, therefore is a *Discrete data*; *Humidity* is also a discrete data; finally, *Windy* and *PlayGolf* are binomial features. ML algorithms are used to learn from quantitative data, therefore, for most traditional ML algorithms, the variables *Outlook*, *Windy* and *PlayGolf* require a set of data transformations in order to allow an algorithm to infer probabilities for the input features.

2.1.2.1 Data Transformation

Basic data transformation is a minimum requirement when features are not quantitative. As part of the preprocessing, basic data transformation is applied in order to be able to feed each algorithm with the data. It is worth reminding that the considered strategies are commonly applied to supervised classification problems.

Among transformation strategies, the most straightforward is often named Label Encoder [16], which can be applied to binomial and nominal data. The strategy consists into creating a dictionary where each label is represented as a number, for example: represent 'No' as 0 and Yes as '1'. For multiple classes, the list of numbers will have the same size of the number of labels.

It is possible to visualize at data mining/machine learning that, when the numeric variable has a skewed distribution, categories are created depending on the problem. As an example of this strategy: the transformation of numerical data, representing a person age, being modified into age categories, where instead of numbers it starts to be identified as 'Young', 'Adult', 'Elder', transforming from numeric to nominal data.

The target variable also passes through transformations. The most popular approach applies a label encoder to it and them performs an encoding technique. For neural networks and deep learning, a standard approach is to apply One-Hot-Encoding (OHE) due to its simplicity: this transformation assumes a flat label space [17]. The OHE creates a binary vector with size equal to the number of distinct classes, assigning the index of the original label to this position. The process can be visualized in Table 2.1 for a 3 class (A, B, C) problem.

Table 2.1: One-Hot-Encoding for a 3 classes (A, B, C) dataset

y	y_{ohe1}	y_{ohe2}	y_{ohe3}
A	1	0	0
B	0	1	0
C	0	0	1
B	0	1	0
A	1	0	0
C	0	0	1

After transformations, the dataset with quantitative features is prepared and can be submitted to the validation and evaluation phases of the ML pipeline.

2.1.3 Machine Learning Algorithms

Numerous algorithms have been developed to address a wide spectrum of ML problems. Each algorithm possesses distinct characteristics and operates based on combinations derived from the input data presented to the model.

In the context of classification problems involving tabular datasets within the SL paradigm, as highlighted by Russell and Norvig [1], certain algorithms have demonstrated notable performance. Notably, Artificial Neural Networks (ANN), k-Nearest Neighbors (KNN), Logistic Regression (LR), and Random Forest (RF) have consistently delivered impressive results when applied to various data domains spanning multiple application areas.

2.1.3.1 Artificial Neural Network

ANN is a non-linear algorithm that tries to mimic the behavior of neurons in a human brain. The ANN can be represented by multiple distinct architectures, for instance: Hopfield network and Boltzmann machine [18]. This work focuses on fully connected networks named Feed Forward Neural Network (FF) ANN, where this architecture is represented as a group of connected layers where each layer may have multiple neurons. These layers are named input, hidden and output layers, where the input receives the data, propagating it to the hidden layers, reaching the output layer which will return a value based on the model. The training phase of the network is based on the propagation of data through its architecture [1]. Figure 2.2 shows an ANN architecture with 1 input layer with 2 neurons, 1 hidden layer with 2 neurons and 1 output layer with 1 neuron.

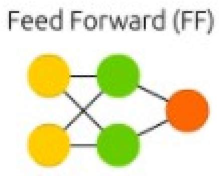


Figure 2.2: Feed Forward ANN architecture

Source: Adapted from Veen and Leijnen [19]

Regarding the number of neurons, the input layer and the output layer have the number of neurons equal to the number of inputs and outputs respectively, therefore, if a transformation technique is used to the target variable, the number of outputs will have the same size of the encoding. In the ANN architecture, the neurons have a value which can be propagated to another layer of neurons of the output. The values are calculated by using the following equation:

$$y_i = f(\sum_{j=1}^{m} w_{ij}x_{ij} + b) \tag{2.1}$$

where y_i is the value of the i-th neuron which has m neighbors from a previous layer (j). Also, w_{ij} is the corresponding weight that connects the previous to the next layer, x_{ij} is the propagated data from previous layer, f is an activation function, b is a bias that can also be a learning parameter during the model optimization [2].

After propagating the data, the network output ($\hat{y}$) is compared with the original output (y) and the model training is evaluated through an error metric [2]. For classification problems with both multiclass or binary, this function is usually the Log loss/Categorical Cross-Entropy given by:

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i * log_e(\hat{y}_i) + (1 - y_i) * log_e(1 - \hat{y}_i)] \tag{2.2}$$

where n stands for the number of instances found in the dataset. The objective is to minimize the network error by updating its weights. The process of training is usually performed by gradient based algorithms, since the derivative of each activation function can be easily calculated. Next subsection provides more information regarding these functions.

Activation Functions are used to determine the output of a neuron given an input or set of inputs, where these functions compute nontrivial problems using a small number of nodes, assisting the network to capture relations between received inputs and outputs [20]. The correct combination of activation functions provides a boost for the network predictive capability, decreasing the error while training the weights while increasing the accuracy when testing the model based on the network.

Among popular activation functions for supervised classification problems, it is the Sigmoid function:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

which describes the input probability of being positive (1) or negative (0). Since that the probability between infinite numbers are verified, the function output has a range between 0 and 1. This function is the most used [20], usually at the output layer when the problem is encoded by strategies like OHE, allowing the network to capture the most active neuron (neuron with the highest value at the output layer).

The network architecture is not limited to a single activation function. As mentioned earlier, each layer may have its own function, where the Rectified Linear Unit (ReLU) is usually used at hidden layers:

$$ReLU(x) = max(0, x) \tag{2.4}$$

which is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero [21].

When the dataset has multiple labels and is not prepared with an encoding strategy, the Softmax function can be used:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j))} \tag{2.5}$$

where a vector with exponential evaluated values are normalized by the summation of the exponential of all elements in the vector [22], allowing the model to extend the Sigmoid (Eq. 2.3 on the preceding page) to handle multiple labels. Many other activation functions can be used for multiple purposes, however, functions that are mentioned in this Section are constantly used at state of the art ML and Deep Learning (DL) applications.

2.1.3.2 k-Nearest Neighbors

Nearest Neighbors algorithm is a distance based algorithm, where a distance metric is used to cluster different instances in a group [23]. The algorithm does not require the data to be labeled, used for multiple learning tasks [24].

The KNN training phase starts with data points with unknown classification, where the following steps are applied until a stopping criteria is met [24]:

1. Calculate the distance between new and known points;
2. Check if each point is near the neighborhood, if it fits that condition, add the point as a nearest neighbor.
3. Count the most frequent class among k neighbors instances based on the amount of nearest neighbors;
4. Label new data points according to the frequency.

The frequency is also the mechanism used to predict data instances when evaluating the model with the test set. The stopping criteria might be associated with: the time, where the algorithm will stop after a certain amount of time; a number of iterations controlled by the user; the distance system, when the overall distance reaches a minimum precision or error.

The KNN has two major parameters found at the optimization phase. The first is used at the stage 2 of the training phase, where the value k will indicate the number of neighborhood points to be considered while inserting new data points to its neighbor; the second parameter is the distance metric used to measure the distance between data points and associate it to a neighborhood.

Distance Metrics Distance metrics were used for multiple purposes along the years, where many were devised and applied for specific conditions. For ML, these metrics are used according to the type of data, where the most used equation for the distance system is the Minkowski distance:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{2.6}$$

where x and y are two instances of data and p a parameter value that will transform this general equation into one of these metrics [25]:

- **Euclidean**: when p is equal to 2, it will calculate the shortest path between two points if they are continuously linear;
- **Manhattan**: when p is equal to 1, this metric assumes that the data has no linear dependence. When the data is restricted to a grid-like path with categorical variables, the distance can be analyzed as the amount of steps required to reach the other point.

KNN may also be used for other purposes, for example: to detect outliers in the data. When applying the algorithm for this task, the Mahalanobis distance might be useful [26]:

$$d(x, y) = ((\vec{x} - \vec{y})' \mathrm{C}^{-1} (\vec{x} - \vec{y}))^{\frac{1}{2}} \tag{2.7}$$

where x and y are two instances of data and C is the covariance matrix. In the Mahalanobis equation, the C matrix eliminates redundant information contained in correlated variables and can be applied without the normalization by the Euclidean distance.

As a general rule, any distance metric can be used by the KNN algorithm in the ML training phase, however, selecting this metric is not a straightforward process. Each metric has its own calculations and properties, making this process to rely on data. The bad selection of a metric may lead to an overfit or underfit, where the final model will not have a reasonable performance.

2.1.3.3 Logistic Regression

The Logistic Regression (LR) is a linear statistical algorithm built for problems where the target variable is binary. The algorithm applies a continuous, differentiable function on the linear regression output to circumvent prediction issues found at the base model [1].

According to [1], the hard threshold (e.g. a parameter to determine where the final output will be considered 0 or 1) and Sigmoid 2.3 on page 11 functions can be used. These functions will return the input probability of being positive (1) or negative, where the probability between infinite numbers are verified, giving a value that ranges between 0 and 1. It is possible to visualize both functions in Figure 2.3, that shows the hard threshold function on the left and the Sigmoid on the right.

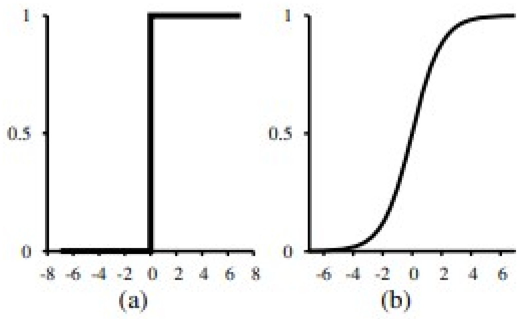


Figure 2.3: Output functions for a LR: a) Hard threshold function and b) Sigmoid

Source: Adapted from Russell and Norvig [1]

The learning process of a LR relies on fitting a set of weights in order to minimize the data loss. The model is represented in the following equation:

$$z = \alpha + \sum B_i X_i, \tag{2.8}$$

where B_i are the learning parameters of the model, X_i are the dataset inputs and α a constant to weight the summation. As mentioned in [1, 27], due to its mathematical properties, the Sigmoid function is preferred at the LR, therefore, the final output is given by the following equation:

$$P(X) = \frac{1}{1 + e^{-(\alpha + \sum B_i X_i)}}. \tag{2.9}$$

Similar to the ANN, since the derivative of the Sigmoid function can be easily calculated, the learning process of the LR is frequently performed by derivative methods, yet it might also be trained by general purpose optimization algorithms.

As mentioned earlier, the LR was built for problems where the data output ranges between 0 and 1. In order to be applied to solve multiclass problems, the algorithm can be inserted into the One vs All (OvA) strategy detailed in the next paragraph.

Logistic Regression and One vs All The OvA strategy can be defined as the application of many classifiers of the same type, where each predicts the input probability of a specific class [28].

The model fit process for multiple classes can be visualized in Figure 2.4. The train set is sent for several algorithms equal to the number of classes found in the data. These algorithms will pass through a training phase with the same parameters, resulting in multiple sub-models. These sub-models are grouped to make the final model, which will give predictions for new data. To perform predictions, the new data instance is sent to all sub-models, where the input probability for each class is calculated. In the end, the model with a higher probability of being the specific class is selected as output [28].

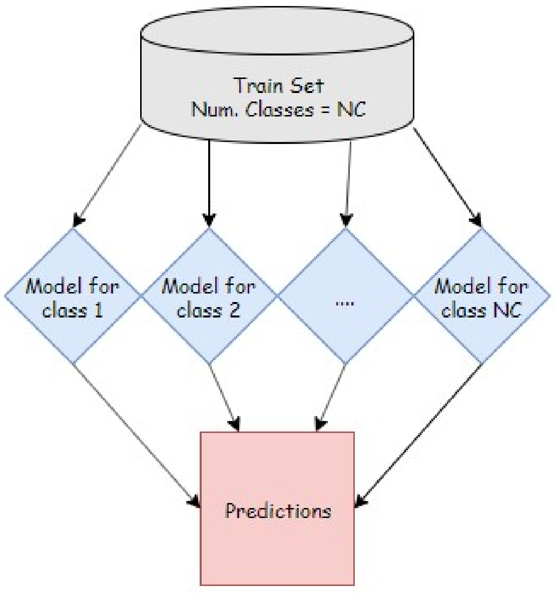


Figure 2.4: OvA strategy for multiclass classification

2.1.3.4 Random Forest

Random Forest (RF) is a term for ensemble methods composed by tree-type classifiers [29]. Each tree classifier is fitted on various sub-samples of the dataset to improve the predictive score and reduce the overfit [30].

The training phase of a RF is slightly different when compared to other algorithms mentioned in this section. The difference is located at the initial phase of the training, where a randomly selected subset of features and instances are split from the dataset and sent to multiple three-based algorithms [29, 30]. When applying this algorithm for a classification problem, the output is given by the majority vote of each tree. Regarding the model parameters, it is possible to mention the number of features to be included on each subset and the tree-type classifier used to gather outputs.

According to [31], when decreasing the number of selected features, the expectations are to reduce the required processing time and increase the correlation between trees, it is also possible to increase the predictive accuracy while creating a model less sensitive to handle outliers. Since this method selects specific subsets of features, the feature importance is calculated while the training phase is performed. The tree classifier depends on applications, where the most common algorithm used for both regression and classification tasks is the Decision Tree algorithm.

Decision Tree Decision Tree (DT) is an algorithm which mimics an up-side down real-world tree [32]. A DT built for the dataset shown at Figure 2.1 on page 7 can be visualized at Figure 2.5. The components of the DT are: root node; internal node; termination node; and branches.

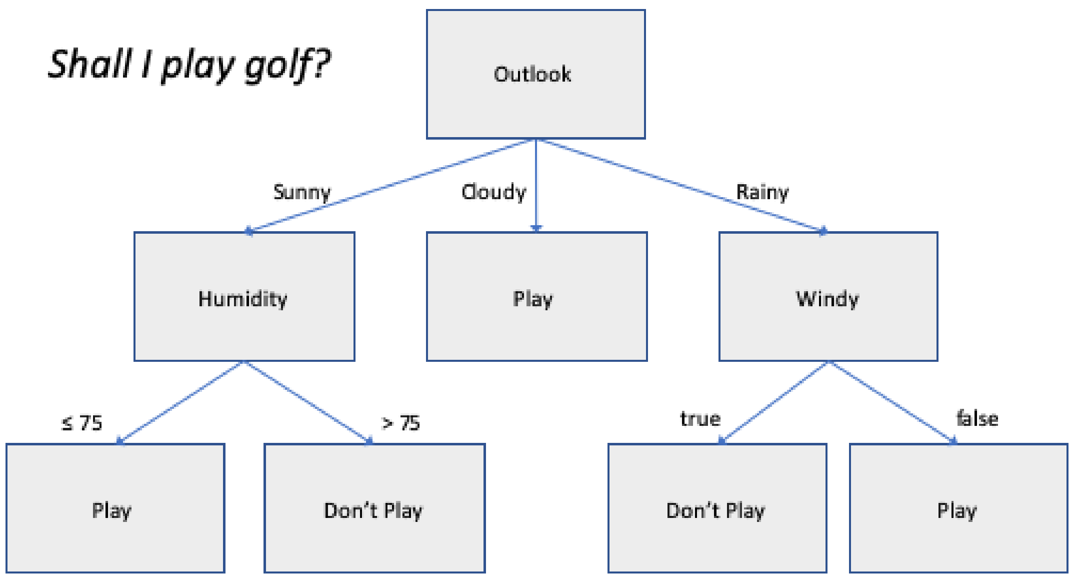


Figure 2.5: Decision Tree structure

Source: Raggett [33]

These components can be perceived in the Figure according to the following:

- **Root node**: is the root of the up-side down tree, represented here as the Outlook feature;
- **Branch**: each branch represents specific characteristics found in the data, where the *Outlook* feature branches are categorical with 3 possible outputs, *Humidity* branches are values within a range and *Windy* branches are binary;
- **Internal node**: are features found in the dataset, where the algorithm will have to make a decision based on the value received (*Humidity*, *Windy* features);
- **Termination node**: represents the final decision based on the target variable, for this example, whether or not the climate conditions are good to *Play* golf.

According to Ali et al. [31], the training phase propagates the data from the root until it reaches one terminal node located at the left of the tree, passing through internal nodes. Each internal node will try to optimize the best split for each feature depending on its type (value ranges, categories, etc). With the best data splits for the presented training set, the algorithm will evaluate the test set using the same structure.

Considering the current purpose of the DT, which is to insert multiple DT's into a RF algorithm when reducing the number of features used in the subset given by the RF, it is expected that this process makes ease the search for the best data split since there are fewer features and instances when compared to using the whole data. Regarding specific parameters for the DT algorithm, it varies between the criteria used to measure the quality of the split, where the entropy is usually selected:

$$Entropy\ H(X) = -\sum p(X) \log p(X) \tag{2.10}$$

measuring the information gain of the specific variable. Other parameters are the maximum depth of the tree, the maximum number of internal nodes, and which strategy will perform the data split. The quality of the split, created by the combination of these parameters and data, will assist to understand the performance of the DT output since the split quality, regarding the evaluation metrics, would increase the predictive quality of the DT, while a poor split will reduce its accuracy. Finally, the algorithm will output predictions that are compared against the real values and the performance can be compared against other algorithms.

2.1.4 Model Validation and Evaluation: Train and Test

The validation is performed under a training and testing phase, measuring and validating the capabilities of the algorithm to acquire the knowledge, building a model and predict new data [1, 10]. Figure 2.6 shows the validation process detailed in subsequent subsections.

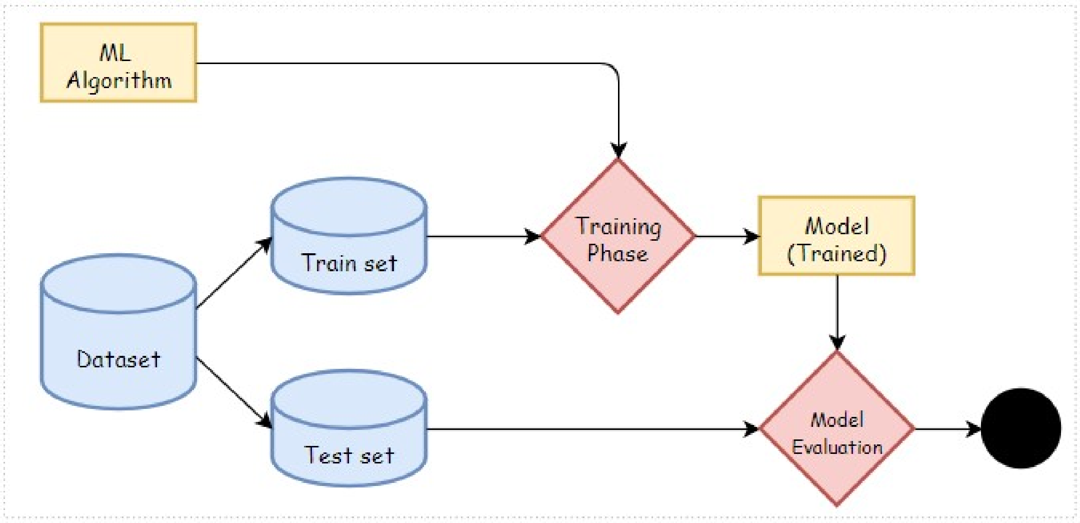


Figure 2.6: Summary of the validation

2.1.4.1 Training phase

Previous to the training phase, the dataset is split into 4 new sets: X_{train}; X_{test}; y_{train}; y_{test}, where X are inputs and y the output. Train sets are used to build and validate the model, while test sets are used to evaluate the predictions for unseen data. During the training phase, the X_{train} set is sent to the algorithm which will perform its learning mechanism and the algorithm output will be compared against the y_{train} where the performance is evaluated by a ML metric. After a series of iterations, the learning mechanism, weights, bias and others, should be optimized for that algorithm, generating a model. Finally, the model is used to generate predictions for the X_{test} set, where the predictions are compared against y_{test} and the performance is measured by the same, distinct or multiple ML metrics.

The standard training phase described in the previous paragraph performs a straightforward model evaluation, however, it is possible to apply other strategies to estimate the risk of a learner or to perform a better model selection. The most used strategy for this task is Cross-Validation (CV) [34] with its variation named k-Fold Cross-Validation (kCV) where 4 distinct steps are applied [35]:

1. Shuffle the dataset;
2. Split the data in k folds of the same length (same number of instances);
3. Create and evaluate a model for each fold;
4. Calculate the kCV score.

According to [35, 36], the parameter k is usually set to 5 or 10, depending on what is being observed. Increasing the value of k will reduce the variance and might increase the bias, therefore, the opposite happens when k has a lower value.

The kCV evaluation is a discussion topic where standard approaches calculate the mean of the metric at each fold, however according to Forman and Scholz [37], the best way to acquire the scores is by summing up correct and incorrect classifications per class, calculating them on the final score, leading to more realistic instead of optimistic results. In any case if the score obtained by the kCV is satisfactory, the data and algorithm can be propagated to the test phase.

2.1.4.2 Testing phase

The testing phase is responsible to measure the real performance of the model built at the training phase, using the unseen data (X_{test} and y_{test}) to analyze its predictive capability. If the model used a validation strategy during the training phase, it would require a refit, therefore, the X_{train} is used to fit the algorithm, generating a new model. The X_{test} is propagated to this model, which returns a set of predictions $\hat{y}$ that is compared against the y_{test} and its performance is evaluated by a ML metric.

The performance of the model at the test set is analyzed based on how well the function is being approximated [38], where this concept is applied to ML in the following characteristics:

- **Overfitting**: the model acquires knowledge for a specific set of data and may not be able to generalize;
- **Underfitting**: the model is not able to generalize the training data and has a poor predictive performance;
- **Good fit**: a mid term between over and underfitting, where the model acquired sufficient knowledge and is capable of generalizing while predicting new data.

To address overfitting and underfitting, validation strategies like the cross-validation mentioned in previous paragraphs can be used. Also, feature selection can improve the algorithms performance during training.

2.1.5 Evaluation Metrics

Evaluation metrics assess the performance of a model in the training and testing phases. In classification problems, the model returns the input probability of belonging to a class. The outcome, according to Sokolova et al. [39], is usually represented by a confusion matrix represented in Table 2.2, where: *TP* are True Positive; *FP* - False Positive; *TN* - True Negative; *FN* - False Negative. Accounting the results obtained by the model, each metric will calculate the score using the confusion matrix, but using a different mathematical model. It is worth mentioning that these metrics share the same goal, a maximization problem that has an optimal value in 1.0 and the worst value at 0.0.

Table 2.2: Confusion matrix for binary classification

Class x **Prediction**	**Positive**	**Negative**
Positive	TP	FN
Negative	FP	TN

2.1.5.1 Accuracy

Being the most popular metric used in the literature [39, 40], accuracy is defined by Equation 2.11,

$$accuracy = \frac{TP + TN}{TP + FP + FN + FP} \tag{2.11}$$

Regarding the mathematical model, accuracy has been criticized since it may lead to a biased score. An example described by [41], assumes an unbalanced dataset where, for each positive class, there are 100 negative classes. The same author indicates that a model with a good fit would produce an 99% accuracy, classifying correctly each negative class since it is majority, while misclassifying all positive cases.

To mitigate challenges stemming from imbalanced datasets, balanced accuracy serves as a valuable metric for assessing machine learning algorithms [42]. It is quantified by the following equation:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{Positives}} + \frac{\text{TN}}{\text{Negatives}}\right) \tag{2.12}$$

The macro-average of recall scores per class is computed by weighting each class's score based on the inverse prevalence of its true class. When the classifier performs equally well on both classes, this computation simplifies to the conventional accuracy.

2.1.5.2 Receiver Operating Characteristic

ROC is described as a comprehensive function to evaluate the performance of a model [39]. The mathematical model can be described by Equation 2.13:

$$ROC = \frac{P(x|positive)}{P(x|negative)} \tag{2.13}$$

where $P(x|C)$ denotes the conditional probability that a data entry has the class label C. As mentioned by [39], ROC results deal with the most positive to the most negative classification, being a metric that can be easily analyzed. Regarding unbalanced datasets, it has been proven that metrics based on precision and recall may return a more realistic result [43].

2.1.5.3 F-score

F-score is a precision-recall based metric, therefore, it is required to understand the mathematical model of both metrics in order to understand the main goal. Recall or sensitivity or true positive rate, is calculated by 2.14:

$$recall = \frac{TP}{TP + FN} \tag{2.14}$$

while precision is given by 2.15:

$$precision = \frac{TP}{TP + FP} \tag{2.15}$$

both metrics are used to calculate the F-score, presented in 2.16 on the facing page:

$$F\text{-}score = 2 * \frac{recall * precision}{precision + recall} \tag{2.16}$$

Recall may be defined as the model capability to predict positive results and precision is the proportion of positive results that are truly positive. The f-score would be the harmonic mean between these metrics and also can be used to achieve more realistic results for unbalanced datasets. However, the f-score can balance precision and recall by applying two parameters α and β.

2.2 Deep Learning

Traditional ML algorithms were used for multiple applications, where they were capable to detect patterns and adapt to new circumstances. In a counterpart, these algorithms were limited to process natural data in their raw form, which also require expert knowledge to transform raw data into a suitable representation to feed an algorithm [44].

According to LeCun et al. [44], methods that automatically discover patterns in raw data are methods of Representation learning, where Deep Learning (DL) are representation learning methods with multiple layers of representations. According to the same author, when dealing with classification tasks, when increasing the number of layers, it is expected that the network would capture discriminant and important features while suppressing irrelevant information, for instance: when learning images, layers are responsible to detect the presence or absence of edges at particular locations, and detect patterns regardless small variations in the image position.

LeCun et al. [44] and Bengio et al. [45] state that the important factor of DL is that each layer of feature is not designed by humans. They are, in fact, learned from the data using a learning mechanism. Among those learning mechanisms, it is possible to mention the basic DL architecture named Deep Feed Forward Network (DFFN). DFFN's are an extension to the FF architecture, where multiple hidden layers are inserted into the architecture. As an example, Figure 2.7 on the next page shows a 4 layer network where the input layer has 3 neurons, 2 hidden layers with 4 neurons each and an output layer with 2 neurons.

As per the research conducted by Nielsen [2], this particular architecture demonstrates greater computational power than a standard FF. However, its complexity poses challenges in the training process. Nonetheless, the DFFN has garnered substantial attention in both academic literature and industry circles. Moreover, various other DL models have been developed to address distinct challenges in handling diverse data sources. In the context of this thesis, emphasis is placed on the Convolutional Neural Network (CNN), which has exhibited superior performance compared to conventional models, particularly in analyzing image data domains.

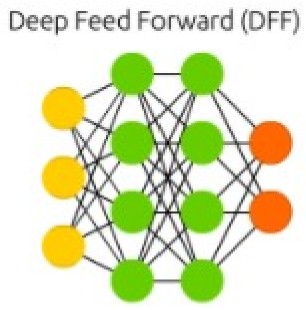


Figure 2.7: Deep Feed Forward Network structure

Source: Adapted from Veen and Leijnen [19]

2.2.1 Convolutional Neural Network

As mentioned in the previous paragraph, CNN outperformed traditional ML models and also its basic structure (ANN) for data represented as images, however, the CNN still share the same concepts of an ANN, where the first layer will receive the data and the final layer will output a variable according to the activation function and classes defined at the problem [46].

According to Nielsen [2], it is possible to achieve a great classification result using a DFFN, however it does not take into account the spatial structure of the image. Also considering a problem with a (28 x 28 x 1) image shape, only the first layer will have 784 weights to be optimized, requiring a significant computational time due to its complexity [2, 46]. Nevertheless, what differs the CNN from the ANN architecture is that convolution is used instead of matrix multiplication in at least one layer [45]. The layers of a CNN are [46]:

- **Input layer**: responsible to store the image pixels, following the same guidelines of the traditional FF;
- **Convolutional layer**: will determine the output of neurons which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume;
- **Pooling layer**: will then simply perform down sampling along the spatial dimension of the given input, further reducing the number of parameters within that activation;
- **Fully-connected layer**: standard approach based on the basic structure (FF), where weights and bias are learned to return an output.

An example of the basic structure of a CNN can be visualized in Figure 2.8 on the facing page, where an image of 28x28 pixels is propagated to the first convolutional layer, where its output is sent to a pooling layer and finishing this architecture with 1 hidden layer of 100 neurons and one output layer of 10 neurons. The next two subsequent sections will detail the mechanisms of the Convolutional and Pooling layers that are inserted into the CNN model.

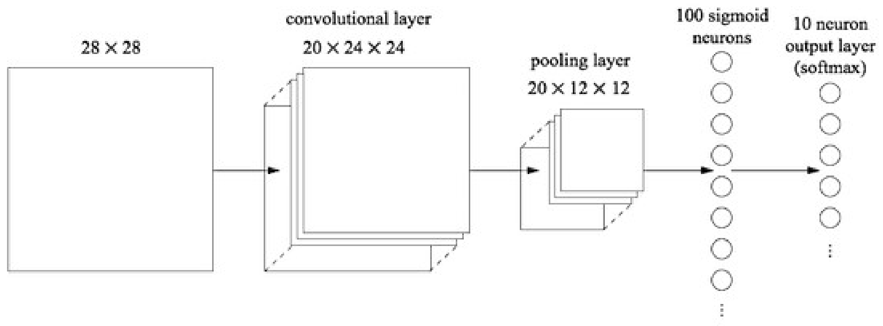


Figure 2.8: An example of a CNN structure

Source: Nielsen [2]

2.2.1.1 Convolutional Layer

As mentioned by O'Shea and Nash [46], convolutional layers are the key aspect of a CNN, where parameters of these layers are based on small kernels. Kernels are used to produce 2D activation maps, acquired from the convolution of each kernel across the spatial dimension when the data pass through it.

As shown in Figure 2.9 on the next page, a kernel passes through an image and moves according to its dimension, the scalar product of each value of the kernel is calculated [46]. Each kernel is used by the network to learn structures, where the kernel map will be activated if it passes through a specific feature in the same position, therefore, the model will learn from an area named receptive field instead of connecting each image pixel to the hidden layer.

In LeCun et al. [44] and O'Shea and Nash [46], the authors mention three hyperparameters that can be optimized through the process, these are named as:

- **Depth**: the volume produced by the convolutional layer, which is set by the number of neurons within the input area;
- **Stride**: used to set the depth around the spatial dimension of the input;
- **Zero-padding**: process of padding the border of the input.

According to Nielsen [2], the CNN is also based in the shared weights and bias idea, which means that all neurons in the first hidden layer will learn exact the same feature, however in distinct locations. The same author points out that this idea is useful to assist the CNN to detect movements or other modifications applied to the image, where the author mentions a rotation to a cat image, where it will still be a cat but in a different position. The concept of shared weights and bias provides a huge advantage when compared to the standard FF network, as it assists to reduce the number of parameters to be learnt at the training phase. The example given in [2]

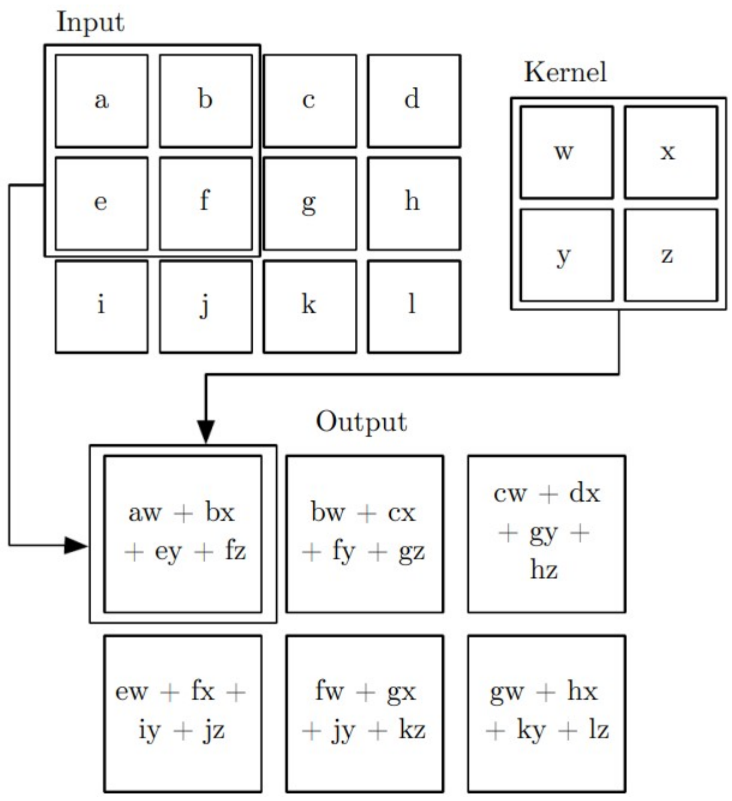


Figure 2.9: An example of a 2D convolution

Source: Bengio et al. [45]

shows that a network for the MNIST dataset (28x28 image shape) requires 23,550 parameters, while the CNN with 25 feature maps requires 520 parameters. It is also worth mentioning that feature maps are the common expression found to indicate the output of the hidden neurons in the layer.

2.2.1.2 Pooling Layer

Pooling layer is a distinct type of layer applied right after the convolution layer, where the goal is to reduce the representation dimension, reducing the number of parameters and model complexity [2, 45, 46]. As can be visualized in Figure 2.10 on the facing page, the units of the pooling layer summarize the information at a specific region of the feature map (2x2 at Figure 2.10 on the next page) according to a specific criteria [2]. Shown in the same figure and caption, it indicates that a max-pooling layer is applied, which means that the maximum value of the 2x2 region is propagated to the pooling unit. It is worth mentioning that the pooling is applied to each feature map generated at the convolution layer.

The max-pooling is the most used pooling strategy with a 2x2 size, which will reduce the map by the half of its original size. For instance, a feature map with dimensions 24x24 is reduced to 12x12 after pooling. Nevertheless, max-pooling is not the only pooling strategy. The following

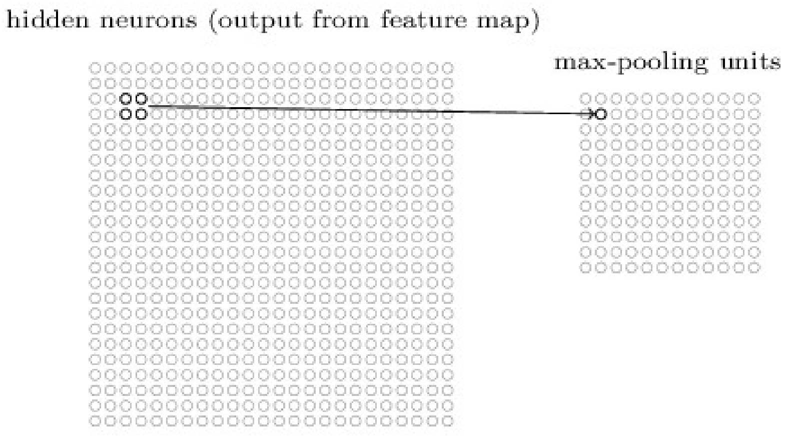


Figure 2.10: Example of a max-pooling layer

Source: Nielsen [2]

strategies can be used [2, 46]:

- **L2 pooling**: calculate the square root of the sum of the squares at the region;
- **Average pooling**: calculate the average of the region;
- **Overlapping pooling**: the stride is set to 2 while the kernel size is set to 3;

2.3 Feature Selection

At Section 2.1 on page 5, it was possible to notice that a tabular dataset is a representation for multiple features of the same instance. When the dataset goes through a preprocessing, the data can be considered ready to be used by ML algorithms. Algorithms like the Random Forest reduce the size of the dataset, aiming to increase the predictive accuracy of the algorithm when grouping multiple tree-based algorithms. The process of selecting which features should be inserted into the subset of features to be further evaluated by the ML algorithm is named Feature Selection (FS).

FS is an optimization problem which emerges with the necessity to reduce the number of features found in a dataset, creating the best possible subset of features [47]. According to the same author, there are several benefits to apply FS techniques to the data:

- **Faster training phase**: less features require less computational time and also a ML algorithm with less complexity;
- **Reduces overfitting**: redundant features are excluded from the entire set;
- **Improve the predictive capability**: less misleading data would increase the obtained score by the algorithm.

Drawbacks are associated to the fact that FS does not guarantee that a subset of features will give a better result when compared to using the complete set. If an algorithm combined with a specific FS strategy fails to enhance its predictive capability, the time required for the FS phase plus training phase will be greater when compared with a simple training phase, being an unnecessary process.

Regarding the optimization task, FS is a bi-objective problem which searches for a solution that increases the algorithm predictive capability while decreases the number of features. The solution encoding of an algorithm for the FS task is a binary array with size equal to the number of features in the dataset, where the i-th variable represents whether or not the i-th feature of the dataset should be used in the subset. Functions devised to handle the FS problem penalize the number of features and the score, leading to the traditional FS objective function:

$$f(x) = \alpha Score(X_{train}, y_{train}) + \beta \frac{|S|}{|D|}, \tag{2.17}$$

where: the score stands for an ML metric acquired from the training stage of an ML model with the X input with selected features and y the target variable; S stands for the selected subset of features; D is the full set of features; α is a real number between [0, 1] range and β is given by (1 - α). The goal for Equation 2.17 is to penalize the learning algorithm score with the number of selected features, minimizing the score while minimizing features, where both score and features are weighted by two distinct parameters. However, in order to apply the same strategy for classification problems, a slightly modification should be performed, modifying the plus sign to minus, resulting in:

$$f(x) = \alpha Score(X_{train}, y_{train}) - \beta \frac{|S|}{|D|}. \tag{2.18}$$

The modification would maximize the score while it still minimizes the number of features, allowing the algorithms to be compared regarding the obtained score applying the classification metrics as basis. With the definitions of a solution and the cost function to evaluate it, according to Guyon and Elisseeff [47], this problem is usually solved by three distinct approaches that will be explained on subsequent subsections: Filter; Wrapper and Embedded.

2.3.1 Filter

Filter approaches are based on a feature rank which suggests which features are the most prominent in the dataset [4]. The feature rank is calculated according to a criteria based on a statistical test or strategy which will be able to sort the features according to its importance. Filters do not require to fit a ML model in order to acquire the feature rank, therefore, it is considered a quick method for FS, however, the final model performance might suffer a decrease on the predictive capability since it does not consider the baseline model for FS.

As mentioned in previous paragraph, the process to acquire the feature rank depends on the strategy, however, with the calculated feature rank, the features are selected according to a threshold defined by the user, where this threshold will indicate which features should be used in the feature subset. Among popular filter approaches, it is possible to mention the Correlation criteria and Mutual Information (MI).

2.3.1.1 Correlation criteria

The correlation criteria is a simple method for feature selection that captures the linear dependencies between the features and the target variable. The base correlation model is Pearson Correlation (PC) coefficient, defined here as:

$$R(X_i) = \frac{cov(X_i, y)}{\sqrt{var(x_i) * var(y)}} \tag{2.19}$$

where *cov* is the covariance and var is the variance. Correlation can vary in the range [-1, 1], where positive correlation indicates that both variables move towards the same direction and negative correlation indicates that both variables move towards the opposite direction.

To be able to select the most correlated features at the FS task, the absolute value of each correlation is calculated, where according to the defined threshold, features will appear in the selected subset. The threshold for FS can vary between a single real value between [0, 1] representing a percentage of features to be included or statistical measures, for instance: mean, median, min, max, etc.

2.3.1.2 Mutual Information

While the PC captures the linear dependency between pairs of variables, the MI uses Shannon Entropy between two variables. As defined in [4], MI uses the Shannon entropy as basis represented here by Equation 2.20,

$$H(y) = -\sum_{x} p(Y) log(p(Y)) \tag{2.20}$$

which stands for the information content in the target variable. When observing the X variable in relation to the target, the conditional entropy is defined as:

$$H(Y|x) = -\sum_{x}\sum_{y} p(x, Y) log(p(Y|x)). \tag{2.21}$$

Equation 2.21 implies that by observing a variable X, the uncertainty in the output Y is reduced [4]. The decrease in uncertainty is given as:

$$\mathrm{MI}(Y, X) = H(Y) - H(y|X). \tag{2.22}$$

Equation 2.22 gives the mutual information between X variable and target. If these two variables are independent, the MI score will be zero, otherwise the result will be greater than zero. Similar to the correlation criteria, the threshold can be a statistical measure and a percentage of features, however, the results of the MI vary between range $[0, \infty)$, requiring the results to be normalized in order to be able to select the feature subset.

2.3.2 Embedded

The Embedded approach, different from the Filter approach, requires a baseline model. The strategy makes a trial to insert the FS at the training phase of the algorithm [4]. This approach is done by selecting an objective function which will evaluate the process, for instance an ML metric to determine if the generated subset helps the model to increase its performance.

At the training phase, an algorithm is used to gather knowledge from the data and optimize its learning mechanism in order to become a model capable to generalize the data for multiple scenarios. In the Embedded approach, the learning mechanism (weights, bias, thresholds, etc) is used to rank the selected features, where a threshold or statistical measure is also used to select features to be used in the subset.

The Embedded approach offers several advantages, notably in potentially streamlining the process of enhancing model performance by automatically eliminating redundant features through its intrinsic learning mechanism. However, it's worth noting that a feature deemed unimportant by the algorithm might still contribute to identifying novel patterns when combined with other features. On the other hand, as only a single new subset is generated, there could be insufficient diversity between the subset and the original dataset. This limitation may result in wasted processing time when attempting to fit the model to this new subset.

2.3.3 Wrapper

The Wrapper approach is an extension for the Embedded model, using a baseline model which will assist to select the feature subset [4]. The strategy fits a model with the complete dataset and will repeat the following instructions until a stopping criteria is reached:

1. Compute the feature importance;
2. Reduce the dataset, e.g. create the subset, according to a criteria;
3. Fit a new instance of the same model with the subset.

The feature importance is computed from the model learning mechanism, which can be used to indicate the less relevant set of features. The criteria used to reduce the dataset can be the feature importance or a percentage of features to be selected. Finally, the stopping criteria is defined as a minimum number of features to be found at the subset or a maximum number of iterations or time.

The benefits of using a Wrapper strategy are related to the number of distinct subsets generated along the process, exploiting multiple combinations of data until it finds a good subset of features for the specific model. In a counterpart, the strategy requires a model to be fit, therefore, when dealing with a large scale dataset, this strategy might have a performance downgrade regarding the required computational time. Also, the control parameters for this strategy: learning algorithm parameters; stopping criteria; and criteria to create the feature subset, are problem dependent, requiring a parameter tuning to achieve the best performance, also leading to more computational time.

2.4 Swarm Intelligence Optimization

The optimization field is mainly located in the mathematical background based on the assumptions to solve any problem involving a decision making process. According to Chong and Zak [48], this area received a huge attention in the past years due to the progress of the computer technology and the flexibility of application in multiple knowledge areas.

At the optimization process, the algorithm searches for a feasible solution according to optimization criteria, and results are evaluated by an objective function whose purpose is to measure the quality of the solution [48]. The best solution achieved is interpreted as the best decision based on the algorithm criteria. Standard optimization problems are defined on the following format:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & l_j \leq x_j \leq u_j, \quad j = 1, \ldots, n. \end{aligned} \tag{2.23}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is nonlinear, multimodal and can be nondifferentiable or noncontinuous. We assume each variable is box-constrained, otherwise $l_j = -\infty$ and $u_j = \infty$. Also, other problems may vary according to their objective function, where the function can be a bi-objective cost function or have several other constraints.

In this thesis, the optimization area is summarized in the Swarm Intelligence (SI) field, where bio-inspired algorithms are developed for a general purpose optimization process. The algorithms found in this branch were applied in interdisciplinary scenarios which assisted researches and the industry to acquire and develop high-end solutions for multiples problems.

SI is a branch of Artificial Intelligence (AI) which encompasses bio-inspired algorithms to solve diversified optimization problems [49]. In this category, each algorithm simulates the collective intelligence of species that can be found in real world, for instance, the bees foraging behavior, the movement of flocking birds, the ant colony system, etc. Prominent algorithms that are commonly used in the literature are: Artificial Bee Colony (ABC) [50]; the Ant Colony Optimization (ACO) [51]; and the Particle Swarm Optimization (PSO) [52].

Besides the theoretical environment of SI algorithms, the No Free Lunch theorem [53] corroborates the fact that there is not an algorithm capable of solving all problems, being one of the reasons to focus in algorithms for a specific area. For instance: ACO algorithm is a well known strategy to solve the traveling salesman problem and job scheduling; ABC and PSO presented good results when applied to continuous and discrete encoding in multiple domains [50, 54].

Regarding SI's applications, surveys found in Karaboga et al. [50] and Wei and Qiqiang [54] have presented a huge variety of them in economics, engineering, computer science, etc. In the data mining area, it is shown by [55] that ACO and PSO are the most used algorithms, where the approaches vary on parameter tuning, training small or medium scale ML models and feature selection. However, the optimization process of ACO does not circumvent the problem created in the thesis, which will be presented in the next Chapter. Therefore, the following subsections will detail the standards of SI algorithms and the optimization process of ABC and PSO.

2.4.1 Swarm Intelligence Base Algorithm

The standard mechanism is applied to each algorithm based on the swarm intelligence. The basic structure is represented by a 2-Dimensional matrix of shape Number of Solutions (NS) x Problem Dimension and a vector of shape NP. Each row of the matrix is the candidate solution of a specific problem, which is evaluated by a function f subject to constraints g.

The main structure is generated in a common phase named Initialization phase, where the initial values of the matrix are generated according to 2.24,

$$x_{ij}^{t=0} = l_j + \phi * (u_j - l_j), \tag{2.24}$$

where x_{ij}^{0} is the j-th problem variable of the i-th particle in the swarm, ϕ is a random number sampled from a uniform distribution and set into a range between [0, 1], u_j and l_j are respectively the upper and lower bounds of the problem for the j-th variable. With the initialized matrix, each solution will have its respective fitness value calculated (x_{if}), finishing the initialization phase that leads to the individual mechanism performed by the SI algorithm until a stopping criteria is reached. The stopping criteria is usually the number of function calls, which stands by the number of times a solution was evaluated by the objective function.

2.4.2 Artificial Bee Colony

Inspired by the mathematical modeling of honey bees forage pattern proposed by Tereshko and Loengarov [56], Karaboga and Basturk [57] devised the ABC algorithm. The ABC encompasses three distinct group of bees: Employed; Onlooker; and Scout bees, which search for food sources in the environment while manage to trade information with other bees inside the hive. The optimization process is controlled by three parameters: NS; a *limit* variable for each solution; and a stopping criteria. The solutions are modified in three distinct phases where each phase is named with the name of each bee group.

Employed Bees: at this phase, bees are sent to explore the neighborhood searching for more promising food sources. For each candidate solution (x_i), a local search is combined with a greedy selection process, the solution is modified according to the following:

$$x_{ij}^{t+1} = x_{ij}^{t} - \phi \times (x_{ij}^{t} - x_{kj}^{t}) \tag{2.25}$$

where ϕ is a random uniform value between range [-1, 1], k is a random integer between range [0, NS] and the indexes are the j-th variable of the i-th solution and the k-th solution. After the information exchange, the new solution is evaluated according to the objective function, calculating its fitness value. If the fitness value of the new solution is better than its previous state, the old solution is replaced, otherwise, the limit value of the i-th solution is increased by 1.

Onlooker Bees: at this phase, the bees are selected according to a criteria and sent to trade information with a distinct bee from the hive. The common process for the selection process is the roulette wheel selection, however, a tournament among bees can be applied [50]. The roulette selection is similar to a fitness proportionate roulette, where the probability is calculated according to the following:

$$P_i = \frac{f(x_i)}{\sum_{n=1}^{NS} f(x_n)} \tag{2.26}$$

The selected bee is sent to search for more promising food sources, where it will perform the same process of the employed bee phase. The same greedy selection process is applied. If the solution fails to converge, the limit value of the selected bee is increased by 1.

Scout Bees: the last phase is a random motion searching for a food source. This process is performed when a solution has its limit equal or higher than a maximum limit. It is worth mentioning that, in the canonical ABC, only one bee can be considered a scout at each iteration. Summarizing this phase as the optimization process, it is a reset process, where the scout bee will have its values re-initialized according to Equation 2.24 on the facing page.

The cycle of Employed, Onlooker and Scout bees is applied while a stopping criteria is not reached. This stopping criteria is usually the number of iterations or number of function calls, e.g. the number of times a solution is evaluated. When the process is finished, the algorithm outputs the best solution found for the problem which can be compared or used to solve the specific problem. The complete procedure of the ABC algorithm can be found in Algorithm 1.

Algorithm 1: ABC pseudocode

```
Input: Objective function $f(x)$, $D$, $LB$, $UB$, $NFE$, $NS$, $LIMIT$
Output: Best solution found $P_g = \{xg_1, xg_2, xg_3, ..., xg_D\}$
// After each function evaluation:
// 1) Increment FEs counter;
// 2) Check for a possible new global best;
// 3) Check the stopping criteria.
1  $FEs \leftarrow 0$
2  $MaxV \leftarrow abs(UB - LB)$
3  $MinV \leftarrow MaxV * -1$
   // Initialization phase
4  for $i \leftarrow 0$ to to $NP$ do
5      Initialize the food source ($x_i$) between [LB, UB]
6      $x_{if} \leftarrow f(x_i)$
7  end
8  Save global best information ($x_g, x_{gf}$)
   // Optimization process
9  repeat
10     for $i \leftarrow 0$ to $NP$ do
11         $x_i \leftarrow$ EmployedBees($x_i$, $x_j$)
12         $x_i \leftarrow$ OnlookerBees($xn_i$)
13         $x_g, x_{gf} \leftarrow$ Compare($x_i$, $x_g$)
14         $x_i \leftarrow$ ScoutBees(LIMIT)
15     end
16 until $FEs == NFE$
```

2.4.3 Particle Swarm Optimization

Introduced by Eberhart and Kennedy [52], the PSO is an SI algorithm based on the movement of flocking birds, the algorithm refers to the candidate solutions and the population as, respectively, particles and swarm, where each particle is composed by the following attributes: velocity (v); position (x); a fitness (x_f) value associated to the position; a local best position ($\hat{x}$); and a local best fitness ($\hat{x}_f$). At each generation t, the position is updated according to the movement rule (2.27),

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{2.27}$$

where the i-th particle of the swarm is moved according to the new velocity given by (2.28)

$$v_i^{t+1} = w \times v_i^t + \phi_1 \times c_1 \times (\hat{x}_i{}^t - x_i^t) + \phi_2 \times c_2 \times (x_g^t - x_i^t), \tag{2.28}$$

where w, c_1 and c_2 are control parameters to adjust the particle movement of distinct components of the formula, while ϕ_1 and ϕ_2 are random numbers sampled from a uniform distribution between range $[0, 1]$. After moving a particle, the fitness value is calculated according to an objective function. If the new position x_i^{t+1} has a better fitness value when compared to x_i^t, the local best position is replaced and it is compared with the position of the global best particle in the swarm x_g.

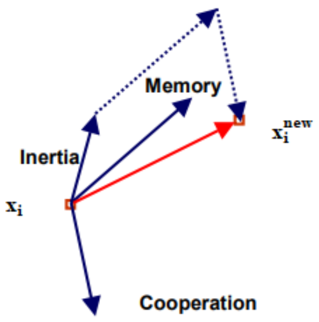


Figure 2.11: Depiction of PSO movement

The movement equation of PSO is divided into three terms: inertia; memory; and cooperation. An example of PSO movement is presented in Figure 2.11, where the inertia term is the capability of the particle to keep moving in the same direction, the memory is a movement towards the best trajectory ever visited by the particle, and lastly, the cooperation guides the particle to the best particle direction. The pseudocode of PSO is shown in Algorithm 2 on the following page.

2.4.4 Evolutionary Particle Swarm Optimization

Along the years, PSO has been criticized regarding its optimization mechanism. The works developed by Miranda and Fonseca [58], and Zeng and Cui [59], show that, at each iteration, when the i-th particle is the current global best, the last term weighted by c_2 would be excluded, resulting in a movement performed only with the first and second terms. Also, when a particle replaces its local best at generation t, *t+1* generation would have its second term excluded, since the current position is the same as the local best position. Besides the criticisms related to the movement rule, there is a deep discussion about the correct values for each control parameter, requiring a parameter tuning for each distinct application. As an approach to circumvent issues found on classic PSO, Miranda and Fonseca [58] developed a novel algorithm named Evolutionary PSO (EPSO), which combines strategies found on the AI literature to enhance the algorithm optimization process.

The merge of evolution strategies with PSO introduces: 4 dynamic weights w_i for the particle movement; replicas; and reproduction components, into the basic algorithm. In EPSO, the dynamic weights replace the static weights, e.g. movement formula control parameters, found in PSO and also each particle has its own collection of weights, which differs from PSO that has weights for the whole population. Regarding the optimization process, EPSO starts by generating replicas, each replica will copy the particle information and their weights are mutated according

Algorithm 2: PSO pseudocode

```
Input: Objective function f(x), D, LB, UB, NP, NFE, $c_1$, $c_2$, $w$
Output: Best solution found $P_g = \{x_1, x_2, x_3, ..., x_D\}$
// After each function evaluation:
// 1) Increment FEs counter;
// 2) Check for a possible new global best;
// 3) Check the stopping criteria.
1  $FEs \leftarrow 0$
2  $MaxV \leftarrow abs(UB - LB)$
3  $MinV \leftarrow MaxV * -1$
   // Initialization phase
4  for $i \leftarrow 0$ to to NP do
5      Initialize particle ($x_i$) between [LB, UB]
6      Initialize velocity ($v_i$) between [MinV, MaxV]
7      $x_{if} \leftarrow f(x_i)$
8  end
9  Save all local best information $(\hat{x}_i, \hat{x}_{if}, \hat{x}_{i\mu}, \hat{x}_{i\sigma})$
10 Save global best information $(x_g, x_{gf}, x_{g\mu}, x_{g\sigma})$
   // Optimization process
11 repeat
12     for $i \leftarrow 0$ to to NP do
13         $v_i \leftarrow$ MoveParticle($x_i$, $v_i$, $\hat{x}_i$, $x_g$, $c_1$, $c_2$, $w$)
14         $x_i \leftarrow$ UpdatePosition($x_i$, $v_i$)
15         $\hat{x}_i, \hat{x}_{if} \leftarrow$ Compare($x_i$, $\hat{x}_i$)
16         $x_g, x_{gf} \leftarrow$ Compare($\hat{x}_i$, $x_g$)
17     end
18 until $FEs == NFE$
```

to Equation (2.29),

$$w_{rk}^t = w_{ik}^t + \tau N(0,1), \tag{2.29}$$

where τ is the mutation rate and $N(0,1)$ is a sample drawn from a Gaussian distribution of mean 0 and standard deviation and w_{ik}^t is the k-th weight of the i-th particle. With this new collection of weights, each of the *NR* replicas and the original particle are moved with the new movement formula (2.30),

$$v_i^{t+1} = w_{i1}^t \times v_i^t + w_{i2}^t \times (\hat{x}_i^t - x_i^t) + w_{i3}^t \times P[(x_{g^*}^t - x_i^t)], \tag{2.30}$$

where P stands for a communication factor, e.g. a binary mask filled with ones with probability *cp*. Furthermore, Equation (2.30) introduces a perturbation in the global best as x_{g^*}:

$$x_{g^*} = x_g \times (1 + w_{i4} \times N(0,1)). \tag{2.31}$$

In the end, a tournament between all replicas and the original particle is performed. The winner is assigned as the new i-th particle and, in case of the winner is a replica, the weights are also replaced. Finally, the current particle is compared with the local and global best particles as it happens in PSO. The EPSO pseudocode is presented in Algorithm 3.

Algorithm 3: EPSO pseudocode

```
Input:  Objective function f(x), D, LB, UB, NP, NFE, $\tau$, CP, NR, MLL
Output: Best solution found $P_g = \{x_1, x_2, x_3, ..., x_D\}$
// After each function evaluation:
// 1) Increment FEs counter;
// 2) Check for a possible new global best;
// 3) Check the stopping criteria.

1  $FEs \leftarrow 0$
2  $MaxV \leftarrow abs(UB - LB)$
3  $MinV \leftarrow MaxV * -1$
   // Initialization phase
4  for $i \leftarrow 0$ to to NP do
5      Initialize particle ($x_i$) between [LB, UB]
6      Initialize velocity ($v_i$) between [MinV, MaxV]
7      Initialize strategic weights ($w^*_{i1}$, $w^*_{i2}$, $w^*_{i3}$, $w^*_{i4}$) between [0, 1]
8      $x_{if} \leftarrow f(x_i)$
9  end
10 Save all local best information $(\hat{x}_i, \hat{x}_{if}, \hat{x}_{i\mu}, \hat{x}_{i\sigma})$
11 Save global best information $(x_g, x_{gf}, x_{g\mu}, x_{g\sigma})$
   // Optimization process
12 repeat
13     for $i \leftarrow 0$ to to NP do
14         $best_{replica} \leftarrow$ GenerateReplicas$(NR, w^*_i, \tau)$
15         $best_{replicaf} \leftarrow f(best_{replica})$
16         $x_{new} \leftarrow$ MoveParticle$(x_i, v_i, \hat{x}_i, x_g, CP)$
17         $x_{newf} \leftarrow f(x_{new})$
18         $x_i, x_{if}, v_i, w^*_i \leftarrow Compare(x_{new}, best_{replica})$
19         $\hat{x}_i, \hat{x}_{if} \leftarrow$ Compare($x_i$, $\hat{x}_i$)
20         $x_g, x_{gf} \leftarrow$ Compare($\hat{x}_i$, $x_g$)
21     end
22 until $FEs == NFE$
```

Chapter 3

Data Transformation and Feature Selection for Machine Learning

This Chapter describes the state-of-the-art techniques used for the core ideas found in the proposed approach: 1) data transformation, where the data domain is changed; followed by 2) a Feature Selection (FS) phase based on image similarity to fit the Convolutional Neural Network (CNN). Gathering these main concepts, developed works that are mentioned in this Chapter met at least one of the following criteria:

- the proposal used a pipeline of data transformation or FS to deal with tabular datasets problems and further solve them by using Machine Learning (ML) algorithms;
- the proposal compares the technique with other transformation or feature selection strategies applied at public tabular datasets;
- the work is a survey or discussion regarding the state-of-the-art algorithms for FS, data transformation or ML.

To summarize the collected information, a brief overview of each contribution will be detailed in subsequent sections, where the next section points out the review of data transformation for ML, while Section 3.2 on page 41 outlines the review of FS for ML.

3.1 Data Transformation for Machine Learning

Data transformation methods can be broadly divided into three categories: input encoding; reshaping; or embedding methods. These methods are agnostic to the data domain and can be applied to: all features found in data or to the target variable with the purpose to adjust the statistical distribution or prepare the data for specific algorithms. The next subsection shows recently and widely used methods to achieve this goal.

3.1.1 Input/Output Encoding

Since most ML and Deep Learning (DL) approaches require numerical data as input, categorical data must be transformed to numbers, where Input Encoding (IE) is an area that applies several methods to overcome this problem. Among popular strategies it is possible to mention: 1) Ordinal/Integer Encoding; 2) One-Hot-Encoding; 3) Learned Embedding.

Ordinal Encoding: is a straightforward yet efficient strategy to transform categorical data into binary. Using a group of colors as data example of categories: Red; Blue; Yellow; Green, this encoding strategy will set a numerical value for each class, encoding each subsequent data instance into the same format [60]. The group of colors will be transformed like the following: Red will be assigned as the number 0; Blue will be identified as 1; Yellow will be used as 2; and Green will be transformed to 3. The numerical sequence increases according to the number of distinct classes found at the feature.

One-Hot-Encoding: this strategy was detailed in the previous Chapter (Table 2.1 on page 9). As a reminder: it assumes a flat label space creating binary vectors with size equal to the number of distinct classes.

Learned Embedding: in this strategy, categories are mapped into distinct vectors that are adapted at the training phase of an Artificial Neural Networks (ANN). The strategy is a mix of both ordinal and one-hot encoding, where it still allows to understand relationships learned from data and uses a vector representation [61]. This strategy is part of Embedding methods, where, according to Brownlee [61], it is a technique developed to provide a distributed representation for words in the text data context.

3.1.2 Data Reshaping

Tabular datasets are represented by rows and columns (instances and features) as shown in the previous Chapter. An ML model collects the dataset and applies its learning mechanism to acquire knowledge from it. Data reshaping is a process applied at the pre-processing phase that can make ease the learning process. According to Wickham et al. [62], reshaping the data will create or reduce the number of features, allowing to visualize the data from a new perspective. According to the same authors, in the reshaping scenario, the data is divided into two categories:

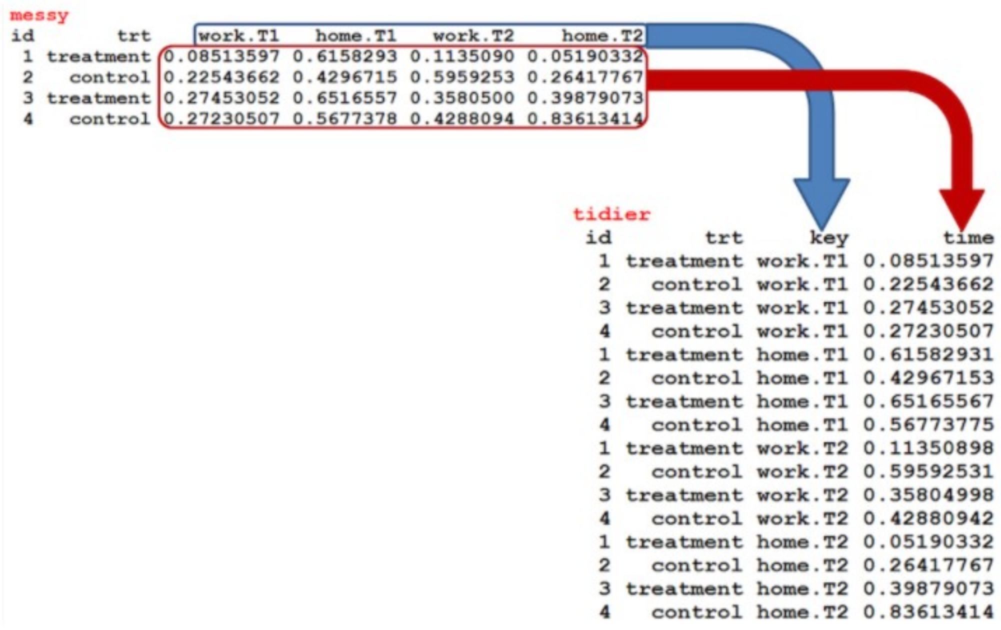
1. **Identifier:** which is related to the *ID* feature that identifies the measured unit;
2. **Measured variables:** features used to measure the target variable (input variables).

where the identifier acts as a pivot to allow these transformations for each measured variable. Several approaches can be applied in order to reshape the dataset, where widely used approaches at the data pre-processing domain are presented in the next topics of this subsection.

3.1.2.1 Gather and Spread

Gather in the most general term stands as collecting or assembling distinct entities into a group. In the data domain, gather represents the action of collecting information from a spread-out or scattered state and group them into a dataset. As a pre-processing strategy, Gather aims to transform data from a wide format into a long, e.g. reduce the number of features inserting it into groups.

In the documentation of scientific programming languages (R, Python, MATLAB, etc.), Gather can be found in reshape packages, where the function will group multiple columns and collapse them into a key. The transformation process using Gather can be visualized at Figure 3.1. In the Figure, the dataset has 6 features: id (the identifier); and 5 measured variables - *trt*, *work.T1*, *work.T2*, *home.T1*, *home.T2*. Each *home* and *work* variables are transformed into a new key, while its values are set into a new feature named *time*.



```
messy
id       trt    work.T1   home.T1   work.T2    home.T2
 1 treatment 0.08513597 0.6158293 0.1135090 0.05190332
 2   control 0.22543662 0.4296715 0.5959253 0.26417767
 3 treatment 0.27453052 0.6516557 0.3580500 0.39879073
 4   control 0.27230507 0.5677378 0.4288094 0.83613414
```

```
tidier
 id       trt     key       time
  1 treatment work.T1 0.08513597
  2   control work.T1 0.22543662
  3 treatment work.T1 0.27453052
  4   control work.T1 0.27230507
  1 treatment home.T1 0.61582931
  2   control home.T1 0.42967153
  3 treatment home.T1 0.65165567
  4   control home.T1 0.56773775
  1 treatment work.T2 0.11350898
  2   control work.T2 0.59592531
  3 treatment work.T2 0.35804998
  4   control work.T2 0.42880942
  1 treatment home.T2 0.05190332
  2   control home.T2 0.26417767
  3 treatment home.T2 0.39879073
  4   control home.T2 0.83613414
```

Figure 3.1: Gather example as a function for data pre-processing.

Source: Bradley [63]

The Spread concept in a general term stands as extending an entity, the meaning of the word may vary according to the area. In the data pre-processing domain, Spread acts as the process of splitting a variable into one or more variables. As a pre-processing strategy, Spread aim to capture parts of the variable and split them into several others, transforming a long dataset into a wide dataset, the opposite of Gather.

Regarding the transformation process, Spread is performed in the opposite direction of Gather. Using the same Figure 3.1 as example, the values found in the *key* feature are transformed in unique features with its respective value, e.g. it will have several features - *work.T1*, *work.T2*, *home.T1*, *home.T2* - instead of *key*. Spread is usually used when the data represents time, where the programmer aims to split the date in quarters, bimonthly, etc.

3.1.2.2 Separate and Unite

Separate and Unite are straightforward concepts, one opposite to the other similar to Gather and Spread. Separate indicates the process to split an entity in multiple entities, while Unite represents the joint of multiple entities into one. In the data pre-processing domain, these approaches are performed for both categorical and numerical variables, aiming to remove irrelevant information that can be found in the original feature or group of features.

The dataset found at Table 3.1 has 3 features: the identifier; a Year-Month and City-State measured variables. The values found at the Year-Month feature show 2 information, the year and the month separated by the hyphen, while City-State show the City with the State inside parenthesis. Separate function will split Year-Month into Year and Month, where the month might be transformed using ordinal encoding or transformed into month related information, e.g. bimonthly, quarters, semesters. While City-State is transformed into City and State, however, since the values found in City are unique, the feature can be removed, while the state information can be maintained.

Table 3.1: Example of features that can be transformed by Separate

ID	**Year-Month**	**City-State**
1	2006-Jan	Grand Forks (ND)
2	2006-Feb	Fargo (ND)
3	2006-May	Rochester (MN)
4	2006-Jun	Dubuque (IA)
5	2007-Jan	Ft. Collins (CO)
6	2007-Feb	Lake City (MN)
7	2007-May	Rushford (MN)
8	2007-Jun	Unknown

Source: Adapted from Bradley [63]

3.1.3 Tabular Learning Methods

A few works have been investigating how to take advantage of deep learning methods such as CNN, mostly used successfully for image data, applied to tabular data. Arik and Pfister [64] propose an end-to-end model, named Attentive Interpretable Tabular Learning (TabNet), where tabular data is fed directly to a CNN, which is responsible for feature extraction and classification. Various datasets are tested and results outperform classical learning methods such as boosting and gradient-based as well as AutoML.

Not limited to CNN, Huang et al. [65] proposed layers named Transformer layers that create modifications of the dataset embedding of categorical features into a robust contextual embedding. The robust embedding is sent to a traditional ANN in a multi layer format that performs the learning task. The method baptized TabTransformer outperformed several ML baseline algorithms while matches the performance of a tree based algorithm in distinct supervised classification tasks. A different work proposed by Somepalli et al. [66], also presented an embedding layer. The method named Self-Attention and Intersample Attention (SAINT) creates a projection of categorical and numerical variables in a feature space. The projection is sent to an embedding layer that applies two tasks named Self-Attention and Intersample Attention, which focus in learning the features representation based on samples instead of looking towards the complete feature. Similar to TabTransformer, SAINT also propagates the learned content to an ANN for the final learning task.

Another brand new pipeline devised to handle tabular data, named Net-Disjunctive Normal Form (Net-DNF) was devised by Katzir et al. [6]. The pipeline learn based on a network named Disjunctive Normal Form Networks, which is based on three components. The main component is a block of layers based on disjunctive normal form, which create features and can be trained. The other layers are feature selection and location, where the FS is performed through a greedy hierarchical FS, while the localization, according to the author, encourages each unit in a Net-Disjunctive Normal Form (Net-DNF) ensemble to specialize in some focused proximity of the input domain. The work provides an end to end pipeline that can handle tabular datasets, however, the results indicate that the model is outperformed by state-of-the-art ML algorithms.

Criticizing the classic CNN and transformer methods that require several parameters to be learned and also several input data to acquire knowledge, the work developed by Kossen et al. [67] introduce another end to end pipeline that added flexibility to use training data directly when making predictions, named Non-Parametric Transformers (NPTs). A mechanism named self-attention captures the relationship between data points found in the training set, test set and between them. The results are compared and the model was able to outperform TabNet and standard ML algorithms.

Part of the pipelines, like TabNet, received several attention from researchers. Gorishniy et al. [7], the authors compare several DL strategies in a few datasets. The authors mention that these DL learning pipelines for tabular datasets are not capable to outperform a Residual Neural Network (ResNet) or gradient boosting Decision Tree (DT) algorithms. Also, Kadra et al. [68] compare TabNet and other pipelines against an empirical study of the regularization impact in the standard ANN algorithm, where pipelines were outperformed by the method.

3.2 Feature Selection for Machine Learning

Focusing on classification supervised learning problems, among published works regarding FS, it is possible to notice that algorithms and measurements are the focus, where distinct evaluation criteria are used to verify the quality of the FS performed by the algorithm. Between analyzed works, it was possible to check that Filter has a distinct way to evaluate it, while Wrapper and Embedded share the same evaluation criteria since it follow the same basic idea.

3.2.1 Filter Feature Selection

Filters were criticized based on the assumption that this category ignores the data structure, considering that features are independent of each other [14]. Nevertheless, this category is still used in some applications and as comparison for the development of novel strategies. Table 3.2 present a variety of popular filter methods that can be used for classification problems, where each is identified according to the strategy criteria, e.g. if it is based on statistical information or information gain [69].

Table 3.2: Popular filter methods of FS for classification problems

Name	**Filter Strategy**
ANOVA	Statistical
Correlation	Statistical
Fisher Score	Information
Gain Ratio	Information
Mutual Information	Statistical
Relief/ReliefF	Information

Source: Adapted from Jović et al. [69]

Methods presented at Table 3.2 are used to solve diversified problems, including benchmarks and real-world scenarios. Several works can be found at the literature pointing out where they can be efficient in order to provide the best feature subset. To summarize the application of each method, next subsections present a brief discussion of the approach while will also present distinct cases of success.

3.2.1.1 Analysis of Variance - ANOVA

ANOVA or Analysis of Variance is a statistical test that measure the dispersion in data points, comparing the differences between two or more groups [70]. The outcome of the test is a *p*-value used to verify the statistical significance between these groups. This significance is measured by evaluating one of the following hypothesis:

- ***H0***: All features have equal variance;
- ***H1***: At least one feature is different.

The accept or reject of the hypothesis is based on a confidence level, which is usually 0.05. The rejection of the null hypothesis (*H0*) will indicate if the feature subset should be included in the training set.

In the literature, ANOVA has been widely used for FS. In the study described by Dey and Rahman [71], ANOVA is combined with a recursive feature selection approach to detect anomaly in networks. The test was used as comparison among multiple filter algorithms for large scale datasets, where ANOVA presented a poor performance for the benchmark datasets [72]. Not limited to the standard approach, a recent work use ANOVA as a validation process of a Random Forest (RF) ensemble, where the test combined with a One vs Rest algorithm was used to select the best group of features for a dataset of Alzheimer's disease and mild cognitive impairment [73].

3.2.1.2 Correlation

The correlation criteria was a widely used filter strategy in the past decade, however, as pointed out by Li et al. [14], data may be presented in graphs, trees and other structures, hindering the process to acquire relevant information when using a correlation criteria. This leads to the fact that using this approach for a feature selection may lead to a poor performance as shown in Neto [74], where the Pearson correlation had the worst overall performance when compared against 10 other FS wrapper and embedded algorithms.

In another perspective, the correlation can be combined with other approaches to enhance the canonical idea assisting ML models to increase its predictive capability, which is the case presented at Gopika and A [75], where other 3 variations of correlation based strategies where used to assist a DT and a Logistic Regression (LR) for a private dataset regarding internet of things. Also, the approach performed by Saqlain et al. [76] combines the Fisher score with the Matthews correlation coefficient, where it is used to capture the subset with higher correlation among subsets selected by Fisher score. Both approaches improve the accuracy when compared to the standard correlation as a FS strategy.

3.2.1.3 Fisher Score

The Fisher score is based on the Fisher information, which measure the amount of information that a variable carries regarding another. According to Gu et al. [77], the Fisher score as a feature selection strategy lead to an sub-optimal feature subset, however, even with limitations the strategy is constantly used as a FS criteria.

The strategy is constantly used for real-world problems, for instance, the work developed by [78], applied Fisher score to select the feature genes of hepatocellular carcinoma and to identify hub genes with the Maximal Clique Centrality algorithm, where the approach achieved a statistical significance when compared against others. In Hasanloei et al. [79], Fisher was combined with

Laplacian score, this combined strategy was applied to detect Quantitative structure–activity relationships regarding drug design using compounds with known and unknown activities under a semi-supervised environment, which show that the Fisher score assisted to achieve better results for the FS phase. The score was also used to detect intrusions, identifying binary real-world data regarding if the group of features represent or not a DDoS attack [80]. Fisher score selected a subset of features which was propagated to a k-Nearest Neighbors (KNN), DT and a Support Vector Machine (SVM).

3.2.1.4 Gain Ratio

Gain ratio or information gain is a entropy-based feature selection strategy [81]. The Entropy is used to measure how much information a feature gives regarding another variable. Following the same procedure of other filter methods, this strategy will create the feature rank, where features are inserted in ascending order according to the information acquired by entropy.

In the work presented by Win and Kham [81], this strategy searched for the best subset of features and subsequently propagated to train a RF based model. The training phase was performed in 2 public available datasets where the approach achieved the best score for both datasets. In Rodriguez-Galiano et al. [82], the Gain ratio was used to predict modelling of groundwater nitrate pollution, the strategy was compared against other wrapper and filter feature selection, where it had the same performance of a mutual information strategy. Nevertheless, the approach achieved higher error when compared to a wrapper SVM.

3.2.1.5 Mutual Information

The mutual information applied to the FS problem was used in several cases in distinct knowledge areas, however, it suffer with the same problem that the correlation criteria does, where the data may hinder the mutual information to acquire relevant information. The approach was also included in the comparison made at Neto [74], where the mutual information was the second with worst performance when applied to four benchmarks and a real-world problem.

Following the idea found at the correlation criteria, since mutual information achieved a poor general performance for FS, the strategy was inserted or modified in several algorithms in order to achieve greater results. The work developed by Gao et al. [83] measured a mutual information criteria for each class found in the dataset instead of generalizing, where the approach was applied to 20 datasets and compared against several methods. In Sharmin et al. [84], the mutual information is applied to discretize and point out the relevance of each feature, the algorithm created by the authors is compared at 30 benchmark datasets against several methods using mutual information.

3.2.1.6 Relief

Relief is a simple yet efficient FS algorithm constantly presented at the literature. The strategy consist into estimating the quality of attributes based on *near-hit* or *near-miss* instances, e.g. if these features are close to the feature vector. Relief strategy was created to handle binary problems, while the version devised by Kononenko [85], named ReliefF can be applied to multiclass problems.

The strategy was used to solve diversified problems in distinct knowledge areas, for instance, the work developed by Reddy et al. [86] developed a model based on Relief FS to identify the gender of persons that gave a written review for a popular tourism algorithm. At Suresha and Parthasarathi [87], ReliefF was used to select the best features to assist a SVM to classify Alzheimer disease based on three distinct types of classification: normal. Alzheimer disease; and Mild Cognitive Impairment. The work presented by Zhang et al. [88], a ReliefF is combined to the Information gain to acquire distinct information of the data, where the best subset is propagated to a RF to detect network intrusion. For all these approaches the experiment show that reliefF was useful to capture a great subset of features that led the algorithm to increase the standard model performance.

3.2.2 Embedded and Wrapper Feature Selection

Embedded and Wrapper methods follows the same concept of acquiring knowledge based on the baseline model, however, the means to reach the most important features are performed in distinct ways. The Embedded process, as described in previous Chapter, acquire the feature importance after a training section of the model, therefore, the state-of-the-art of this strategy vary among the models, for instance: ANN, KNN, LR, RF. Not limited to these models, it worth mentioning other two models that are based on gradient boosting: Extreme Gradient Boosting (XGB) [89] and the Light Gradient Boosting Machine (LGBM) [90]. These algorithms were also used to solve diversified problems and are widely used on competitions to solve real-world problems [91].

Regarding Wrapper methods, this strategy has several other sub-strategies that can be applied according to a algorithm criteria. Among popular strategies, it is possible to mention:

- **Forward Selection**: select relevant features through a significance level acquired by fitting regression models [92];
- **Step-wise Selection**: similar to the forward selection, however, it keeps verifying the significance level of already selected features and can remove then according to the new significance level [93];
- **Recursive Feature Elimination (RFE)**: fit the model with a complete dataset and keep removing features according to a significance level and a criteria which can be a percentage of features [94].

Between the mentioned approaches, RFE received more attention due to its ability to analyze all the features and create distinct subsets while fitting multiple models. RFE was combined with a SVM to assist on the diagnostic of Alzheimer's disease [95] and Diabetic Retinopathy [96]. Was also combined with a RF for enhanced agricultural crop. Not limited to these approaches, a high number of applications can be found at Google Scholar from the last decade until the present.

As mentioned earlier, wrapper strategy can have other sub-strategies (RFE, forward, etc). General purpose optimization algorithms, like evolutionary and swarm intelligence, are also used to solve the FS problem acting as wrapper, however, acquiring a feasible subset by applying its own optimization process. These approaches have a solution encoding based on a binary array, where each variable indicates whether or not a feature from data will be included in the subset. These algorithms are evaluated through a ML metric score acquired from the training phase of the model using the selected subset. The process is repeated until a stopping criteria is met, where the most used is a number of iterations.

3.2.3 Filter vs Embedded vs Wrapper

As presented in this section and previous Chapter, FS problem can be solved with multiple strategies. It is not straightforward to define which strategy will be useful, being data dependent. Dependencies are related to the data type or the model capability to acquire the knowledge from it or algorithm limitations in order to search for the best subset of features. Not limited to problems related to data, these strategies can be compared in 2 distinct scenarios: time complexity and predictive accuracy.

Time Complexity: By looking through how these strategies perform, it is clear that Filter strategies have the best performance. Part of these algorithms are commonly used in the data pre-processing phase, not requiring a model to be fit, therefore, being faster when compared to Embedded and Wrapper. Embedded and Wrapper strategies have to fit a model. However, the feature importance acquired at the embedded approach require the baseline model to be fit once, which is different in Wrapper, that uses a sub-strategy that may fit the model several times to acquire the feature rank.

Predictive Accuracy: papers mentioned at the Filter section have made comparisons against other wrapper and embedded strategies, where it is clear that wrapper methods have outperformed embedded and filters. Its performance is justified by reapplying a model to learn several combinations of features in order to achieve the best possible score. Embedded and Filter may have a similar performance. Nevertheless, since it rely on data, filter strategies may not be able to acquire the features relation and may lead the model for a state of over or underfit.

Chapter 4

Map-Optimize-Learn

Map-Optimize-Learn (MOL) is a three stages strategy based on data transformation, Feature Selection (FS) and Deep Learning (DL) with the purpose to transform a 2D tabular dataset into a 3D dataset, e.g. 1D samples into 2D instances, aiming to construct relevant information to feed a Convolutional Neural Network (CNN)'s to improve the prediction quality when compared to standard Machine Learning (ML) algorithms applied for tabular datasets.

The transformation process of 1D to 2D instances provides a novel representation for the instance, where an image format will be created according to the best solution found in the optimization process. The map and optimize stages of MOL are shown in Figure 4.1, where an arbitrary process is applied to the dataset found in Figure 2.1 on page 7. Subsequent sections will detail the process of each phase and point out which algorithms can be applied to each.

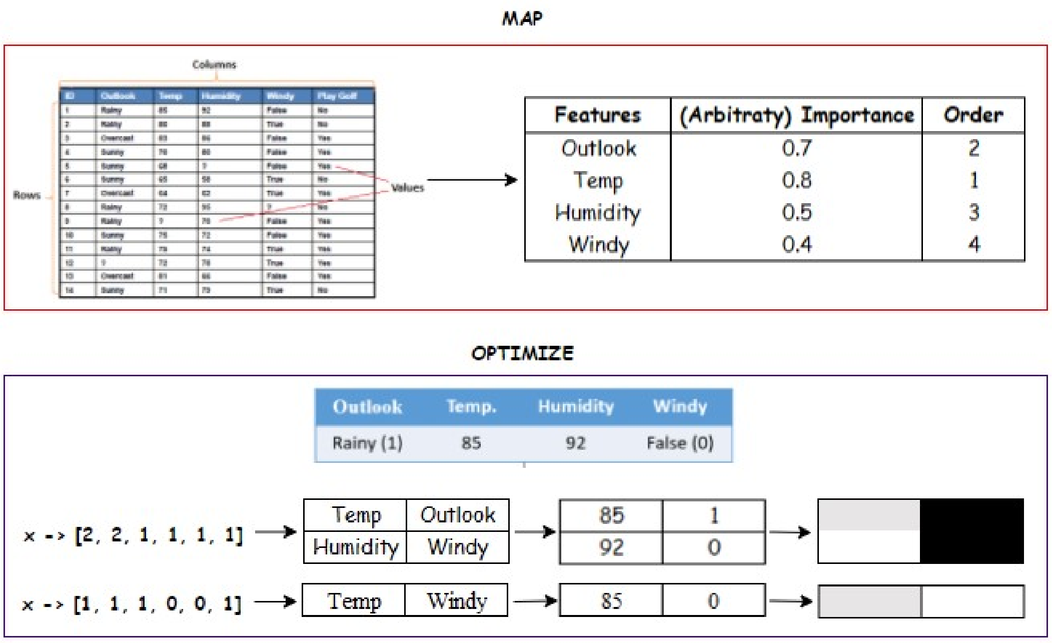


Figure 4.1: Map and optimize phases of MOL

4.1 MOL: Map

The Map stage mirrors the feature importance analysis conducted through various Filter FS strategies. The collection of algorithms detailed in the Filter FS section of Chapter 3 on page 37 can effectively determine feature importance, typically arranged in descending order of relevance. Each feature receives a rank based on its importance within the sorted array. This rank then dictates the reordering of features, shaping the row-wise image format during transformation.

For demonstrative purposes, the procedure is depicted in Figure 4.1 on the previous page, highlighting the map stage within a red box. Within the figure, the sample dataset showcases its feature importance arbitrarily assigned and arranged accordingly. Based on the obtained sequence, Temperature emerges as the most important feature, whereas Windy holds the least significance among them.

A second instance is exemplified in Table 4.1, employing Pearson Correlation as the mapping method. This dataset comprises 4 features and 4 instances. Adhering to the Map phase guidelines, the pairwise correlation between each feature and the Target Variable yields the computed correlations depicted in Table 4.2. These correlations highlight x1 as the most pivotal feature, whereas x3 emerges as less influential.

The results obtained in the example remain unchanged until the subsequent phase of MOL. The correlation table is sent to the Optimize phase in order to be used in the ongoing process. At this stage of the pipeline, no modifications are made to the original dataset.

Index	x1	x2	x3	x4	Target Variable
1	12	5	8	21	1
2	6	14	3	9	0
3	17	4	11	7	1
4	10	2	19	15	0

Table 4.1: Generated Dataset to present the Map Phase

Features	Target Variable	Order
x1	0.82	1
x2	-0.38	2
x3	-0.12	4
x4	0.18	3

Table 4.2: Pairwise Pearson Correlation for the features found in the previous dataset

In the preceding chapter, it was demonstrated that algorithms for determining feature importance differ based on various criteria such as statistical information, distance, entropy, and more. Given the diverse nature of these algorithms, each will assign unique importance values to features, resulting in varying final ranks depending on the specific algorithm employed.

Consequently, identifying the most suitable algorithm for optimal performance at this stage is not straightforward.

The nature of the data types present within the dataset can aid in selecting a method, aligning with the algorithm foundational principles. This correlation between the data types and the algorithm underlying principles can guide the choice of an appropriate approach for determining feature importance.

Images can be produced by employing the standard available format from the dataset, typically using the features in their existing order. However, employing an importance-based method is anticipated to aid in grouping similar features within specific regions of the image. This process helps a Convolutional Neural Network (CNN) in capturing pertinent information during the final stage of the strategy, particularly when learning from images.

While determining the absolute best method might not be definitive, the anticipated effect of employing the optimal strategy is to generate images that closely resemble one another during the optimization phase. This closeness in resemblance among images is a key expectation of the most effective strategy.

4.2 MOL: Optimize

The objective of the optimize stage is to generate images for each instance in the dataset, aiming to streamline the learning process of the CNN algorithm. This stage operates on the premise that computer vision techniques, widely considered state-of-the-art, can efficiently identify correlations within images compared to conventional learning algorithms handling tabular data characterized by various variables (e.g., categorical and numerical).

To create images that preserve similarity among similar instances while effectively distinguishing dissimilar instances, we formulate the following model as an optimization problem:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{N}\sum_{i=1}^{N-1}\frac{1}{SM}\sum_{j=1}^{SM}(X_{i,j}-X_{i+1,j})^2 \\
\text{subject to} \quad & 2 \leq \hat{x_1}\hat{x_2} \leq M, \\
& \hat{x_1}\hat{x_2} = \sum_{i=3}^{M}\hat{x_i}. \\
& \hat{x} \in \mathbb{N}^0
\end{aligned} \tag{4.1}$$

this formulation seeks a vector ($\hat{x}$) of dimension $2+M$ (where M represents the number of features in the dataset) using a specific encoding. The initial two components of $\hat{x}$ are integers denoting the image's dimensions, while the remaining components signify the feature subset through binary values, indicating whether each feature should be included in the image creation step.

The objective function aims to minimize the mean squared error while imposing constraints ensuring that the number of selected features aligns with the shape of the generated image. The features are arranged in a row-wise format according to their computation order during the map stage.

Formulation (4.1) poses a nondifferentiable integer problem. It is recognized that gradient-based methods struggle to find solutions in discontinuous, non-separable, and nonlinear landscapes [97], similar to the characteristics of the problem described in (4.1).

Conversely, while SI algorithms can offer a feasible solution to (4.1), they do not guarantee global optimality. In addition, we illustrate that solely SI algorithms can provide a solution to (4.1). Theoretically, if we reframe (4.1) as a convex problem by relaxing the equality constraint, transforming the search space into $\mathbb{R}$, and introducing a non-negativity constraint, upon computing the first-order derivative of the objective function with respect to $X_{i,j}$, we arrive at the following formulation:

$$\begin{aligned} \text{minimize} \quad & -\frac{1}{N}\sum_{i=1}^{N-1}\frac{2}{SM}\sum_{j=1}^{SM}(X_{i,j} - X_{i+1,j}) \\ \text{subject to} \quad & 2 \leq \hat{x_1}\hat{x_2} \leq M, \\ & \hat{x_1}\hat{x_2} \geq \sum_{i=3}^{M}\hat{x_i}. \\ & \hat{x} \geq 0 \\ & \hat{x} \in \mathbb{R} \end{aligned} \tag{4.2}$$

although (4.2) is differentiable, gradient-based algorithms still struggle to yield viable solutions due to two significant issues within (4.2): rounding of the components and handling the infeasibility of the incumbent solution.

Given that the components of an optimal solution $\hat{\mathbf{x}}^\star$ from (4.2) are non-integers, an additional 'repair' step becomes necessary to properly round the components of $\hat{\mathbf{x}}^\star$ and rectify any violations of the equality constraint. Assuming this repair step utilizes a branch-and-bound algorithm as a subroutine search, in the best-case scenario, the time complexity for resolving the equality constraint would be linear, while in the average case, it could be exponential. Consequently, projecting $\hat{\mathbf{x}}^\star$ from the space of (4.1) into (4.2) for solution purposes leads to the loss of proof of optimality.

Hence, solving (4.2) demands considerably more computational resources compared to obtaining a local optimal solution from a derivative-free algorithm, such as those within the family of SI algorithms, for solving (4.1).

The optimization phase will reshape the instances found in the dataset, generating images based on criteria, defined at the map stage, and the best solution found at the optimization problem. It is expected that similar images are created, allowing the CNN to detect patterns from this format leading to an improvement to the classification task. The generated images are sent to the learning phase of MOL, where the training, validation, and testing are performed.

Returning to Figure 4.1 on page 47, two examples of solutions (x) are shown, where the first will use a 2x2 image format including all features, while the second will use a 1x1 image format with the first and last feature. The features are inserted into an image format using the feature importance order acquired in the map phase. In the Figure, an example instance found in the data is used to illustrate the values filling the image matrix, where later these values are replaced by colors according to their magnitude. The new image feature is created for each instance, e.g. an image is created for each instance.

4.3 MOL: Learn

The learning phase in MOL serves to understand and derive insights from the generated images. In this phase, the CNN learns the format and characteristics of the images rather than a specific set of features. The choice of CNN as the model was based on its ability to detect local spatial coherence by employing convolution on adjacent pixel patches. The resulting feature maps, obtained from the sliding window over the image, help capture deep relationships derived from the patterns generated in the map and optimization stages. The image dataset is divided into training and test sets, with a cross-validation strategy used during training to evaluate model performance. The trained model is then utilized to predict outcomes for the test set images

4.4 MOL as a Tabular Learning Method

MOL is a Tabular-based Learning Method (TBM) that can be compared to various methods discussed in Chapter 3 on page 37. While similar pipelines start with standard tabular datasets and aim to learn from the data points, MOL introduces several components that diverge in their approach to achieving this goal.

MOL strategy is designed to enhance prediction quality given a preprocessed dataset; however, it does not handle missing data. The map phase necessitates a preprocessed dataset to capture data information and determine feature importance, enabling the rearrangement of present features within the dataset.

In contrast to other TBM strategies employing transformers or embedding layers to generate new information and feed it to ML or DL models, MOL restructures features based on insights gained from statistical methods and optimization, creating images from non-image data. Previous works discussed in Chapter 3 on page 37 indicate that ML models tend to perform better when identifying essential features within the dataset. Therefore, MOL's optimization phase focuses on feature selection to minimize the complexity of propagating comprehensive information to a CNN model. Instead, the strategy emphasizes the CNN learning key characteristics from an image set with fewer details.

Chapter 5

Experiments

This chapter provides a comprehensive overview of the experiment conducted, including the algorithms, techniques, hyperparameters, and settings used to compute the results. It is divided into two subsections, with the first section outlining the experimental settings, and the second section detailing the dataset structure, such as the goal and features information.

5.1 Methodology

The methodology employed in the experiments, which are composed of two phases: data preprocessing and statistical comparison of the Map-Optimize-Learn (MOL) approach against state-of-the-art Machine Learning (ML) and Feature Selection (FS) techniques. The experiments were implemented using Python, with the SciPy, Keras and TensorFlow libraries for the models and the NumPy library for handling numerical floating-point precision.

The MOL approach employs several strategies to sort the dataset at the map stage, including ANOVA F-test, Pearson Correlation, Euclidean distance, Fisher score, Information Gain/Gain ratio, and Mutual Information. For the optimization task, we used three popular swarm intelligence algorithms: Particle Swarm Optimization (PSO), Evolutionary Particle Swarm Optimization (EPSO), and Artificial Bee Colony (ABC), which were selected for their demonstrated capabilities in solving diversified problems, as discussed in Chapter 3.

The MOL approach was compared against seven state-of-the-art ML algorithms commonly applied to tabular datasets: Artificial Neural Networks (ANN), Logistic Regression (LR), k-Nearest Neighbors (KNN), Random Forest (RF), Extreme Gradient Boosting (XGB), Light Gradient Boosting Machine (LGBM), and TabNet. To further investigate the performance of these algorithms, both Wrapper and Embedded strategies were explored, along with their Baseline versions, as follows:

- Wrapper: Utilizing a Recursive Feature Elimination (RFE) strategy, 10% of the features were dropped at each iteration.

- Embedded: Initially, a Baseline run was conducted to capture feature importance. Features with importance above the average were retained for a second dataset used to evaluate the Embedded approach. For the ANN model, Permutation Importance was employed to capture feature importance values.

For the purpose of addressing the challenge of unbalanced datasets and ensure accurate model evaluation, the training phase employed stratified 10-fold cross-validation, with balanced accuracy serving as the evaluation metric. For two real-world experiments, comparison against results from the literature was conducted using the same accuracy metric employed in those studies. Hyperparameters for each model were tuned using a Grid Search (GS) algorithm, and results were compared based on balanced accuracy and the Wilcoxon signed-rank test to assess statistical significance.

Each algorithm was tested using both embedded and wrapper FS strategies, as well as without feature selection. To handle unbalanced datasets and evaluate the performance of each model correctly, the training phase was performed under stratified 10-fold cross-validation, using balanced accuracy as the evaluation metric. Two of the real-world experiments are compared against results found in the literature, therefore the same accuracy metric used in the literature is used in those experiments. The hyperparameters of each model were tuned using a GS algorithm, and the results were compared based on the balanced accuracy metric and the Wilcoxon signed-rank test to measure statistical significance.

The FS techniques employed the following parameters: the Convolutional Neural Network (CNN) used one convolutional layer with 32 filters and one max-pooling layer; the ANN had a learning rate of 0.001; the LR was trained using the Limited-memory Broyden–Fletcher–Goldfarb –Shanno (LBFGS) algorithm with an l2 penalty; the KNN used a KD-tree to compute the neighbors and a Euclidean distance metric; the RF used 100 estimators based on decision trees and performing bootstrap; and XGB and LGBM used a learning rate of 0.1 with gradient boosting decision tree algorithms. The network-based models used a single hidden layer with neurons equal to the mean of the input from the FS and the output, and they were trained using the Adaptive Moment estimation (ADAM) algorithm for a maximum of 100 iterations. These parameters are commonly found in multiple libraries and have been shown to achieve good performance on various problems.

Table 5.1 shows the grid of hyperparameters used for each algorithm, which includes critical parameters that typically increase the performance of the models. Some parameters were not optimized, as several studies have shown that the standard parameters achieve better performance in multiple problems, such as the learning parameters of the ADAM algorithm and the CNN architecture parameters, given that the data is generated through MOL.

Concerning the swarm intelligence algorithms, they shared a population of 50 solutions and 500 function evaluations (10 iterations), applying the following parameters: the PSO had w= 0.8; $c1$= 1.8; $c2$= 1.8; for the EPSO τ is equal to 0.8 and a communication probability of 0.9; Finally, for the ABC, the maximum limit value is the average value between number of features

Table 5.1: List of parameters that compose the grid of each algorithm at the hyperparameter tuning phase with a GS

Algorithm	Parameters	Values
ANN	Training algorithm	[LBFGS, ADAM]
	Learning rate	[from 10^{-1} to 10^{-10}]
	Hidden layer neurons	[from 5 to 50 with a step of 5]
KNN	Number of neighbors	[from 2 to 10]
	Distance metric	[Euclidean, Manhattan]
	Weight metric	[Uniform, Distance]
LR	Penalty function	[L1, L2]
	Gamma	[Log space of 20 values from -4 to 4]
RF	Max. features in the best split	[1, 3, 10]
	Min. number of splits	[2, 3, 10]
	Min. samples to be in a leaf	[1, 3, 10]
	Number of estimators	[100, 300, 500]
XGB	Gamma	[0.5 to 3.0 with a 0.5 step]
	Sub samples	[0.6, 0.8, 1.0]
	Samples by tree	[0.6, 0.8, 1.0]
	Maximum depth	[2 to 5]
LGBM	Maximum number of leaves	[31, 127]
	Min. data in a leaf	[30, 50, 100, 300, 400]
	L1 and L2 regularization	[0.1, 1, 1.5]
CNN	Learning rate	[from 10^{-1} to 10^{-4}]

and number of solutions. The parameters were selected based in historical performance of the algorithms presented in previous works across multiple domain areas as shown in Miranda and Fonseca [58], Zhou et al. [98], Neto et al. [99] and others.

5.2 Datasets

This section present the selected datasets to perform the complete experiment. The datasets were selected based on distinct characteristics, where it vary among number of features, problem type, number of instances and number of outputs (binary or multiclass) and also based on distinct data types, varying among binary, continuous and nominal.

The problem type divides the experiment section in two distinct branches, where the first branch performs the evaluation of the strategy on well established benchmarks found evaluating machine learning algorithms. On the other hand, the second branch will focus on the application of the algorithms and strategies to a series of real-world problems, comparing the obtained results against other found in the literature. Table 5.2 presents the general dataset structure.

Table 5.2: Structure of each dataset used in the experiment. Attributes show the number of binary (b), continuous (c), and nominal (n) features in the dataset. The problem type presented with the letter R stands by Real-World, while with the letter B indicates a Benchmark

Dataset	Attributes				Problem	# Classes	# Instances
	b	c	n	Total			
Cancer	0	9	0	9	B	2	699
Card	40	6	5	51	B	2	690
Diabetes	0	8	0	8	B	2	768
Gene	120	0	0	120	B	3	3,175
Glass	0	9	0	9	B	6	214
Heart	18	6	11	35	B	2	920
Horse	25	14	19	58	B	3	364
Soybean	46	9	27	82	B	2	683
Thyroid	9	6	6	21	B	19	7,200
KDD	4	34	4	42	B	2	494,020
Cardiac Pathology	6	3	4	18	R	2	9,484
Forest Cover Type	44	10	0	54	R	7	581,012
Poker Hand	0	0	11	11	R	10	1,025,010

5.2.1 Benchmark Datasets

The following subsections details the classification task of each Benchmark dataset, pointing out general information, origin, special properties, etc. The benchmark experiment also follow guidelines specified by Prechelt [100] which indicate how to correctly compare part of the datasets.

5.2.1.1 Cancer

The Breast Cancer dataset is a classification task that aims to detect tumors as either benign or malignant based on cell descriptions obtained through microscopic examination. The dataset contains only continuous input variables. The target class is represented by two outcomes, with 65.5% of cases being benign and the remaining being malignant. The original dataset can be found at the UCI repository of machine learning datasets, which was originally obtained from the University of Wisconsin Hospitals.

5.2.1.2 Card

The Credit Card dataset is a classification task aimed at determining whether a customer's credit card application should be approved or not. The dataset consists of real credit card applications, and the output indicates whether the application was granted or not. However, the attribute information has been modified for confidentiality reasons.

5.2.1.3 Diabetes

The Diabetes dataset is a classification task aimed at diagnosing diabetes in Pima Indians. The data includes variables such as age, blood pressure, and body mass index. The dataset consists of 8 inputs and 2 outputs, where 65.1% of the target variable represents examples with a negative diagnosis for diabetes. The dataset was created by Prechelt [100] who added noise and made modifications to the original Pima Indians dataset available at the UCI repository. These modifications were made for preparation and optimization purposes.

5.2.1.4 Gene

The Gene Dataset is a classification task which aims to detect Intro/Exon boundaries in nucleotide sequences. The input is collected from a a window of 60 DNA sequence elements, where the algorithm has to detect if the 120 inputs is an intron/exon boundary, exon/intron or none of these. According to Prechelt [100], the number of attributes and features vary since each nucleotide, which is a four-valued nominal attribute, is encoded binary by two binary inputs (The input values used are 1 and 1, therefore the inputs are not declared as boolean.

5.2.1.5 Glass

The Glass dataset represents a classification task to identify glass types. According to [100], the results of a chemical analysis of glass splinters (percent content of 8 different elements) plus the refractive index are used to classify the sample to be either oat processed or non oat processed building windows, vehicle windows, containers, tableware, or head lamps.

5.2.1.6 Heart

The Heart dataset has two distinct versions, where the first named as 'Heart' and the second named as 'Heartc', where 'Heartc' is a smaller dataset which contains data from the Cleveland Clinic Foundation, while 'Heart' is a combination of Cleveland Clinic Foundation, Hungarian Institute of Cardiology, V.A. Medical Center Long Beach, and University Hospital Zuric.

The classification task of the dataset consists into detecting whether one of four vessels is reduced in diameter. The collected data is related with different patients and medical analysis, such as blood pressure, age, pain descriptions, etc.

5.2.1.7 Horse

The classification task of the Horse Dataset aim to predict if a horse that has colic will survive, faint or will be euthanized. The examples of the dataset contains 62% cases where a horse survived, 24% fainted and 14% it was euthanized.

5.2.1.8 Soybean

The Soybean dataset is a classification task with the objective to recognize 19 different diseases of soybeans. The inputs are related with the bean size and the plant description. In the literature, it is possible to find several results for the Soybean dataset, however, the Soybean version created by Prechelt [100] use the Soybean Large dataset from UCI repository with some modifications. In the literature, the original dataset has a limited number of instances for a few classes, where in the majority of cases, only 15 of 19 classes are used. In this version, more instances can be found for these classes, allowing the algorithm to learn all the 19 target classes.

5.2.1.9 Thyroid

The Thyroid dataset is a classification task with the goal to diagnose hypo-function or hyper-function based on patient query and exam data. The dataset contains several missing values that were adjusted by Prechelt [100]. There are three different outputs: thyroyd has over function; normal function; or under function, the class probabilities are respectively, 5.1%, 92.6% and 2.3%. Among the benchmark experiment, the Thyroid set is the most unbalanced dataset.

5.2.1.10 KDD

The KDD is a classification task used to detect anomaly systems. The dataset was proposed by Stolfo et al. [101], the data was captured using an Intrusion Detect System which recorded 7 weeks of network traffic. The goal is to detect wether it is a normal connection or an attack. The types of attack may vary between four categories:

1. Denial of Service Attack (DoS)
2. User to Root Attack (U2R)
3. Remote to local Attack (R2L)
4. Probin Attack

Among the features, it is possible to find several information related with attributes that can be extracted from TCP/IP connections and network traffic based on same host or same service connections. Comparing KDD with the other datasets in the benchmark experiment, it is clearly the set with the most number of instances and continuous features, however, it is not the one with most features, classes or number of features.

5.2.2 Real World Datasets

The following subsections details the classification tasks of each selected Real-World dataset, describing features, statistical information and relevant data description.

5.2.2.1 Cardiac Pathology

The dataset was collected at a hospital specialized in cardiovascular system located at the northeastern part of Brazil. The data is pseudonymised, where patients personal information is modified by an artificial identifier which can be one way to comply both with the European Union's and Brazilian's new General Data Protection Regulation demands for secure data storage of personal information. Features considered important to the clinical evaluation approaches are included (heart murmur and heart sound) as well as general information about the patient (Age, Height, Weight).

Data was analyzed through bivariate and multivariate analysis, where preprocessing was performed in order to follow medical domain standards (e.g. age ranges, pressure ranges, etc) applying: data transformation; cleaning; normalization; removal of irrelevant features, such as ID and features with more than 95% of missing values. It was detected that the *S2* feature had a strong relation with the patient history and the target variable which indicates the presence or absence of a cardiac pathology. This information led us to create a feature named History Based Emergency Level (HEL), which based on the knowledge that the patient already visited the hospital once, may indicate an emergency level, from best to worst: green, yellow or red, regarding the patient with CP.

The final dataset, with the inclusion of the feature engineering and exclusion of irrelevant data, has a population of 9,484 (53% of the original dataset) and 13 features: Weight; Height; Body-Mass-Index (BMI); Age; Wrist state (WS); Blood Pressure (PPA); Second heart sound (B2); Heart murmur (HM); Cardiac frequency (CF); Disease History (DH); Gender; Visit Reason (VR); and History Emergency Level (HEL). It is worth mentioning that 6,144 individuals (64.96%) were healthy and 3,340 (35.31%) had CP, which maintains the characteristic of an unbalanced dataset.

5.2.2.2 Forest Cover Type

The Forest Cover Type dataset, owned by Dua et al. [102], comprises various tree observations from four regions within the Roosevelt National Forest in Colorado, United States of America. These observations are derived from cartographic variables measured in 30x30 meter sections of the forest. The primary objective of this dataset is to predict the forest cover type based on the provided information. The dataset is publicly available at the UCI Machine Learning Repository.

5.2.2.3 Poker Hand

According to Dua et al. [102], in each instance of the Poker Hand dataset, a hand consisting of five playing cards randomly drawn from a standard deck of 52 cards. To describe each card, two attributes are used: suit and rank, resulting in a total of 10 predictive attributes for each hand. The dataset includes a Class attribute that specifies the "Poker Hand" associated with each combination of cards.

It's crucial to note that the order of the cards within a hand is significant, leading to variations in possible hands. For instance, due to this ordering factor, there are 480 possible Royal Flush hands, one for each combination of suits, compared to the 4 variations if the order were not considered.

Chapter 6

Results

This chapter follows a structure similar to the Experiment chapter, featuring distinct sections for benchmark and real-world problems. Each dataset is accompanied by a table presenting statistical results, including a significance test and a brief discussion. The concluding section offers a summarized performance overview for each approach.

In every experiment, be it benchmark or real-world, a comprehensive table includes the following columns:

- Algorithm: the model used to train the data;
- FS Str: the main strategy, options include
  - Baseline (B): a baseline approach without feature selection;
  - Embedded Feature Selection (FS) (E);
  - Wrapper FS (W);
  - Map-Optimize-Learn (MOL);
- Optimization Strategy (Opt Str): The swarm intelligence optimization algorithm employed in the Optimize phase of MOL;
- Train: The accuracy achieved on the training set using the best model selected through grid search.
- Test: the accuracy achieved on the test set;
- # Feat: the number of features selected;

Highlighted results in each table signify the best combination of algorithm, feature selection strategy, and other parameters. Furthermore, images generated for the best combination are depicted using a color scale, where lighter shades correspond to lower values and darker shades to higher values. In terms of feature selection with TabNet, the algorithm reports feature importance calculated during its process. However, it does not explicitly identify the features being utilized, likely due to the inherent nature of its feature selection methodology. While it is possible to estimate the number and identity of selected features using a threshold, this approach may not provide precise results, therefore, the number of features has been intentionally avoided.

6.1 Benchmark Problems

This Section present the obtained results for the Benchmark experiment detailed in the previous chapter.

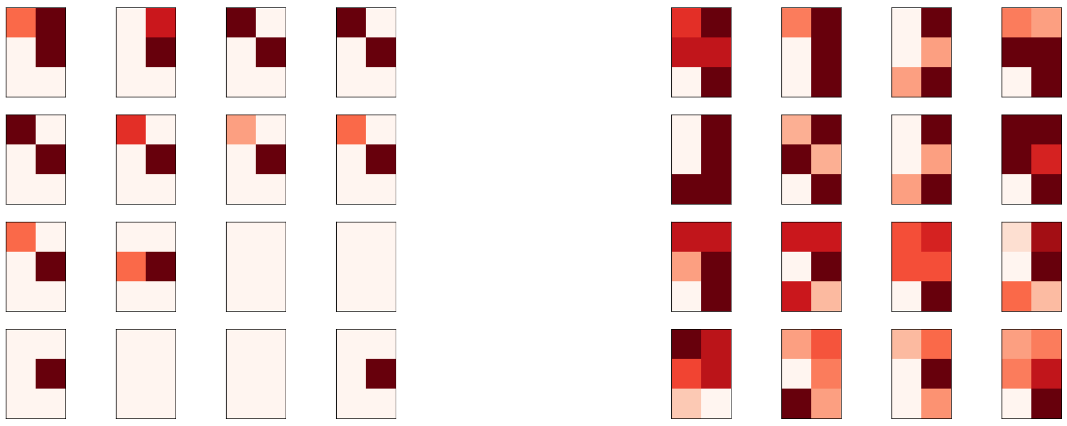
6.1.1 Cancer

The results obtained by each approach combination for the Cancer dataset are presented in Table 6.1 on page 64.

Analyzing the results of Map-Optimize-Learn (MOL), the MOL-Distance-EPSO demonstrated the best performance for this dataset, achieving a 97,70% accuracy on the test set. However, during cross-validation, the algorithm exhibited a slightly lower score of 95.42%, suggesting a potential underfitting scenario. Similar behavior is observed across other MOL approaches, where cross-validation scores are consistently lower than the respective test set predictions. These results should be examined in conjunction with the generated images showcased in 6.1 on the next page. In this provided subset, a distinct pattern is discernible, distinguishing outcomes 0 and 1. While this pattern is evident in the subset, its presence in the entire dataset is not guaranteed, and the existence of similar images in both outcomes may pose challenges for the model, preventing it from achieving higher accuracy.

In subsequent experiments, within the Embedded approaches, k-Nearest Neighbors (KNN) demonstrated superior performance compared to other algorithms, while in the Wrapper experiment, Random Forest (RF) exhibited the best results, and in the baseline result, Light Gradient Boosting Machine (LGBM) outperformed others. Interestingly, KNN-E and LGBM-B showcased a similar behavior observed in MOL approaches, with higher test scores hinting at a potential underfit scenario. Conversely, RF-W correctly classified every set during cross-validation but achieved a lower score in the test set compared to KNN-E. Despite these results, all approaches fell short when compared to TabNet, which achieved 99.08% accuracy in the test set.

Analyzing the cross-validation results and performing a statistical significance test, TabNet displayed statistical significance compared to MOL-Distance-EPSO, LGBM-B, KNN-E, and RF-W ($p < 0.05$). Conversely, other algorithms did not exhibit statistical significance in cross-validation when compared to each other ($p >= 0.05$). The number of selected features ranged from 6 to 8, with the majority of the experiments opting for 8 features, while the best MOL combination selected only 6. Although the exact number of features selected by TabNet is unclear, considering a feature importance threshold of 0.05, the model identified 8 significant features.



(a) Outcome 0 (b) Outcome 1

Figure 6.1: Input generated by MOL-Distance-EPSO for the Cancer dataset.

6.1.2 Card

Table 6.2 on page 66 presents the outcomes for each approach combination on the Card dataset. The results reveal that the majority of MOL combinations achieved relatively similar performances, with MOL-GainRatio-EPSO emerging as the top performer in the test set. Similar to the Cancer dataset, there is a tendency towards a slight underfit scenario in the cross-validation results. Furthermore, the analysis uncovers that most MOL approaches utilized 50 features, while MOL-Correlation-PSO deviated by using only 20 features, resulting in the poorest performance among all combinations.

Examining the outcomes generated by MOL-GainRatio-EPSO, depicted in 6.2 on page 67, can be challenging for the human eye due to the dataset's large number of features. Consequently, part of this challenge is managed by the CNN properties to identify patterns in the generated images. Different kernel sizes may yield varied performances, suggesting that a more comprehensive evaluation of the images could enhance accuracy in this dataset.

Among the remaining strategies, RF emerged as the best Embedded approach, while RF achieved the highest score within the Wrapper methods, and Artificial Neural Networks (ANN) demonstrated the best baseline performance in this dataset. Ranking these results by the score obtained in the test set, MOL-GainRatio-EPSO would be positioned in the third position after RF-W and RF-B. Upon analyzing the cross-validation scores, RF-B exhibited statistical significance ($p < 0.05$) when compared against these methods, while RF-W showed no significant improvements compared to the best MOL approach ($p >= 0.05$). The number of selected features

Table 6.1: Statistical comparison of the obtained balanced accuracy at the training and testing for the Cancer dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9504 ± 0.0259	0.9713	8
CNN	MOL	Correlation	ABC	0.9581 ± 0.0223	0.9655	8
CNN	MOL	Distance	ABC	0.9542 ± 0.026	0.9713	8
CNN	MOL	Fisher	ABC	0.9466 ± 0.0295	0.9713	8
CNN	**MOL**	**Gain Ratio**	**ABC**	**0.9618 ± 0.0258**	**0.9713**	**8**
CNN	MOL	Mutual Info.	ABC	0.9599 ± 0.0278	0.9713	8
CNN	MOL	ANOVA	PSO	0.933 ± 0.0887	0.9713	8
CNN	**MOL**	**Correlation**	**PSO**	**0.9618 ± 0.0244**	**0.9713**	**8**
CNN	MOL	Distance	PSO	0.9561 ± 0.0173	0.9713	8
CNN	MOL	Fisher	PSO	0.9561 ± 0.0244	0.9713	8
CNN	MOL	Gain Ratio	PSO	0.9259 ± 0.0922	0.9713	8
CNN	MOL	Mutual Info.	PSO	0.9599 ± 0.0263	0.9713	8
CNN	MOL	ANOVA	EPSO	0.9619 ± 0.0257	0.9713	8
CNN	MOL	Correlation	EPSO	0.9619 ± 0.0257	0.9655	8
CNN	**MOL**	**Distance**	**EPSO**	**0.9542 ± 0.0334**	**0.9770**	**6**
CNN	MOL	Fisher	EPSO	0.9543 ± 0.0286	0.9713	8
CNN	MOL	Gain Ratio	EPSO	0.9581 ± 0.0239	0.9713	8
CNN	MOL	Mutual Info.	EPSO	0.9580 ± 0.0224	0.9713	8
KNN	**E**	-	-	**0.9530 ± 0.0239**	**0.9800**	8
LGBM	E	-	-	0.9561 ± 0.0256	0.9723	8
LR	E	-	-	0.9556 ± 0.0201	0.9615	8
ANN	E	-	-	0.9628 ± 0.0217	0.9692	8
RF	E	-	-	0.9685 ± 0.0248	0.9769	8
XGB	E	-	-	0.9614 ± 0.0213	0.9493	8
KNN	W	-	-	0.9673 ± 0.0072	0.9769	8
LGBM	W	-	-	1.0000 ± 0.0000	0.9546	8
LR	W	-	-	0.9557 ± 0.005	0.9538	8
ANN	W	-	-	0.9546 ± 0.0064	0.9692	8
RF	**W**	-	-	**1.0000 ± 0.000**	**0.9769**	**8**
XGB	W	-	-	1.0000 ± 0.0000	0.9646	8
KNN	B	-	-	0.9626 ± 0.0286	0.9769	-
LGBM	**B**	-	-	**0.9514 ± 0.0253**	**0.9846**	-
LR	B	-	-	0.9486 ± 0.0295	0.9538	-
ANN	B	-	-	0.9498 ± 0.0334	0.9692	-
RF	B	-	-	0.9614 ± 0.0196	0.9769	-
XGB	B	-	-	0.9528 ± 0.0253	0.9646	-
TabNet	-	-	-	**1.0000 ± 0.0000**	**0.9908**	-

across the experiment varied, with algorithms achieving higher accuracy selecting 17 features out of 51. However, selecting this number of features would not be possible in any MOL strategy due to the squared image constraint devised in the optimization phase of the strategy.

6.1.3 Diabetes

Table 6.3 on page 68 show the results for each combination on the Diabetes dataset. It is possible to notice that most of the algorithms had a poor performance on this set. The performance can be related with the limited number of iterations used in the experiments or data quality since there are few examples of the problem. Nevertheless, MOL-Distance-ABC achieved the best performance among the available MOL combinations in this experiment, achieving a 76,04% balanced accuracy, at least 2% better compared to the other approaches, in the test set where the generated images uses all the features found in the dataset.

In the other feature selection experiments, ANN emerged as the most effective Wrapper combination, RF secured the highest performance among the baseline experiments, and LGBM-E presented the best Embedded approach. It is worth noting that KNN demonstrated results similar to LGBM but with lower accuracy in cross-validation. Consequently, we consider LGBM superior based on this criterion. When comparing results with MOL combinations, an interesting observation surfaces. The statistical test indicates that ANN showed significance when compared against MOL-Distance-EPSO ($p < 0.05$). This is attributed to the fact that MOL exhibited poor performance in cross-validation; however, the model used in the test phase achieved better results compared to others.

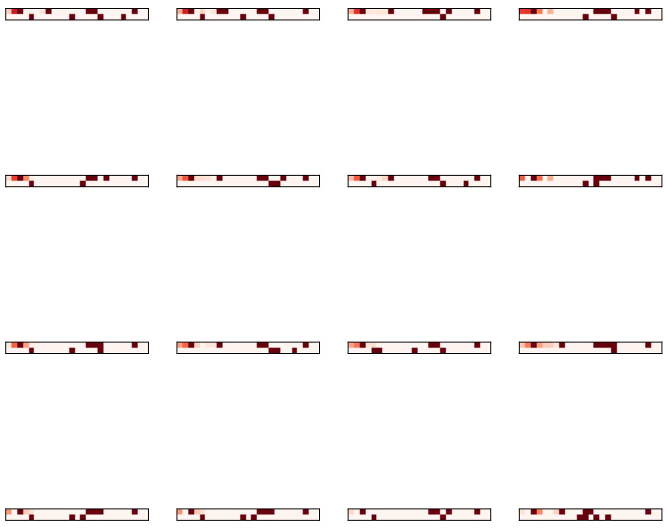
Upon observing the generated images by MOL-Distance-ABC in Figure 6.3 on page 69, distinctive characteristics are evident in the sampled subset, aiding in distinguishing instances representing outcome 0 from those indicating outcome 1. It is crucial to note that the presence of the same pattern throughout the entire dataset isn't guaranteed, potentially impacting the model's ability to achieve peak accuracy. Hypothetically assuming uniformity in the dataset images, a visual comparison of outcome 0 images in rows 3 and 4 with outcome 1 images in the same rows reveals similarities in color for a few points, indicating analogous numerical values. This similarity might pose a challenge for the model, depending on the parameters employed in the Convolutional Neural Network (CNN). Since this experiment serves as a general evaluation of model performance, the prospect of tailoring the model and adjusting settings specifically for this experiment could be pertinent for optimal results.

6.1.4 Gene

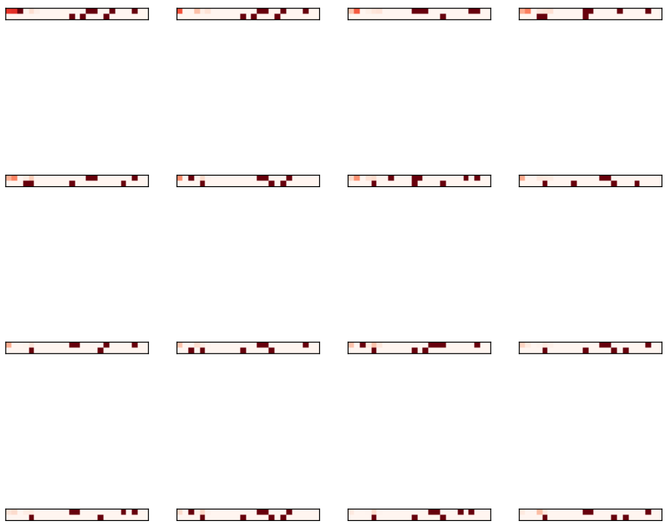
Table 6.4 on page 70 showcases the outcomes for the Gene dataset. Despite its extensive feature set, the dataset, as per the creator's intent, is optimized for achieving high accuracy with minimal effort. While acknowledging this optimization, it remains valuable to assess the algorithms' performance in this dataset. Notably, among various MOL combinations, MOL-ANOVA-PSO

Table 6.2: Statistical comparison of the obtained balanced accuracy at the training and testing for the Card dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.8418 ± 0.0623	0.8779	50
CNN	MOL	Correlation	ABC	0.7492 ± 0.0346	0.7965	50
CNN	MOL	Distance	ABC	0.838 ± 0.0681	0.8721	50
CNN	MOL	Fisher	ABC	0.8437 ± 0.0566	0.8779	50
CNN	MOL	Gain Ratio	ABC	0.8456 ± 0.0583	0.8721	50
CNN	**MOL**	**Mutual Info.**	**ABC**	**0.8456 ± 0.055**	**0.8779**	**50**
CNN	MOL	ANOVA	PSO	0.8418 ± 0.0589	0.8663	50
CNN	MOL	Correlation	PSO	0.7492 ± 0.0531	0.6279	20
CNN	MOL	Distance	PSO	0.838 ± 0.0615	0.8547	50
CNN	MOL	Fisher	PSO	0.8437 ± 0.0605	0.8779	50
CNN	MOL	Gain Ratio	PSO	0.8456 ± 0.0602	0.8721	50
CNN	**MOL**	**Mutual Info.**	**PSO**	**0.8456 ± 0.0589**	**0.8779**	**50**
CNN	MOL	ANOVA	EPSO	0.8447 ± 0.0582	0.8779	50
CNN	MOL	Correlation	EPSO	0.7221 ± 0.0561	0.7791	50
CNN	MOL	Distance	EPSO	0.8399 ± 0.0605	0.8779	50
CNN	MOL	Fisher	EPSO	0.8427 ± 0.0613	0.8779	50
CNN	**MOL**	**Gain Ratio**	**EPSO**	**0.8508 ± 0.0542**	**0.8837**	**50**
CNN	MOL	Mutual Info.	EPSO	0.8399 ± 0.0599	0.8779	50
KNN	E	-	-	0.8460 ± 0.0575	0.8272	31
LGBM	E	-	-	0.8765 ± 0.0668	0.8840	26
LR	E	-	-	0.8697 ± 0.0611	0.8511	44
ANN	E	-	-	0.8584 ± 0.0507	0.8723	36
RF	**E**	**-**	**-**	**0.8711 ± 0.0729**	**0.8886**	**35**
XGB	E	-	-	0.8540 ± 0.0464	0.8364	36
KNN	W	-	-	0.8884 ± 0.0047	0.8652	17
LGBM	W	-	-	1.0000 ± 0.0000	0.8832	17
LR	W	-	-	0.8562 ± 0.0113	0.8647	12
ANN	W	-	-	0.8562 ± 0.0084	0.8647	12
RF	**W**	**-**	**-**	**1.0000 ± 0.000**	**0.8902**	**17**
XGB	W	-	-	0.8614 ± 0.0213	0.8493	10
KNN	B	-	-	0.8258 ± 0.0766	0.8168	-
LGBM	B	-	-	0.8472 ± 0.0545	0.8660	-
LR	B	-	-	0.8561 ± 0.0532	0.8606	-
ANN	B	-	-	0.8382 ± 0.0626	0.8785	-
RF	**B**	**-**	**-**	**0.8586 ± 0.0682**	**0.8957**	**-**
XGB	B	-	-	0.8421 ± 0.0587	0.8598	-
TabNet	-	-	-	**0.8487 ± 0.1059**	**0.8639**	**-**



(a) Outcome 0

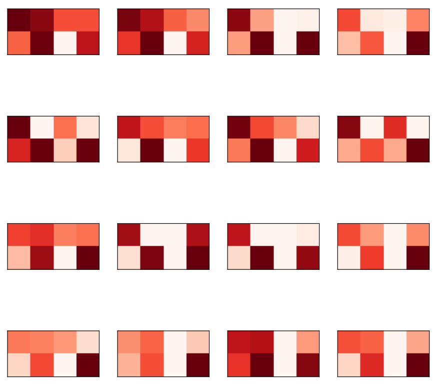


(b) Outcome 1

Figure 6.2: Input generated by MOL-GainRatio-EPSO for the Card dataset.

Table 6.3: Statistical comparison of the obtained balanced accuracy at the training and testing for the Diabetes dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.712 ± 0.0609	0.6979	8
CNN	MOL	Correlation	ABC	0.7467 ± 0.0562	0.7240	8
CNN	**MOL**	**Distance**	**ABC**	**0.7237 ± 0.0596**	**0.7604**	**8**
CNN	MOL	Fisher	ABC	0.7432 ± 0.0684	0.7500	8
CNN	MOL	Gain Ratio	ABC	0.7261 ± 0.0667	0.7344	8
CNN	MOL	Mutual Info.	ABC	0.7138 ± 0.068	0.7188	8
CNN	MOL	ANOVA	PSO	0.712 ± 0.0543	0.7083	8
CNN	MOL	Correlation	PSO	0.7467 ± 0.0612	0.7188	8
CNN	MOL	Distance	PSO	0.7237 ± 0.0644	0.6354	8
CNN	**MOL**	**Fisher**	**PSO**	**0.7432 ± 0.0666**	**0.7396**	**8**
CNN	MOL	Gain Ratio	PSO	0.7261 ± 0.0695	0.7396	8
CNN	MOL	Mutual Info.	PSO	0.7138 ± 0.0632	0.7240	8
CNN	MOL	ANOVA	EPSO	0.7058 ± 0.0591	0.7188	8
CNN	MOL	Correlation	EPSO	0.7372 ± 0.0627	0.6354	8
CNN	MOL	Distance	EPSO	0.7151 ± 0.0641	0.6354	8
CNN	**MOL**	**Fisher**	**EPSO**	**0.7492 ± 0.0694**	**0.7292**	**8**
CNN	MOL	Gain Ratio	EPSO	0.7346 ± 0.0712	0.7031	8
CNN	MOL	Mutual Info.	EPSO	0.7102 ± 0.0663	0.6354	8
KNN	E	-	-	0.7121 ± 0.0519	0.7425	7
LGBM	**E**	-	-	**0.7230 ± 0.0431**	**0.7426**	**7**
LR	E	-	-	0.7318 ± 0.0593	0.7008	7
ANN	E	-	-	0.7442 ± 0.0600	0.6916	5
RF	E	-	-	0.7142 ± 0.0337	0.6824	7
XGB	E	-	-	0.7222 ± 0.0412	0.7092	5
KNN	W	-	-	0.7990 ± 0.2381	0.6896	7
LGBM	W	-	-	1.0000 ± 0.0000	0.7201	4
LR	W	-	-	0.7747 ± 0.1028	0.6897	4
ANN	**W**	-	-	**0.7768 ± 0.3602**	**0.7376**	**6**
RF	W	-	-	1.0000 ± 0.0000	0.7039	6
XGB	W	-	-	0.7614 ± 0.0213	0.7293	6
KNN	B	-	-	0.6968 ± 0.0739	0.6896	-
LGBM	B	-	-	0.7002 ± 0.0612	0.7109	-
LR	B	-	-	0.6986 ± 0.0626	0.6703	-
ANN	B	-	-	0.7170 ± 0.0586	0.6804	-
RF	B	-	-	**0.7031 ± 0.0394**	**0.7437**	-
XGB	B	-	-	0.6656 ± 0.0330	0.6751	-
TabNet	-	-	-	**0.8668 ± 0.1190**	**0.7477**	**-**


(a) Outcome 0

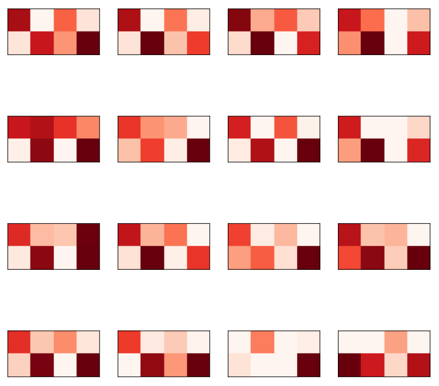

(b) Outcome 1

Figure 6.3: Input generated by MOL-Distance-ABC for the Diabetes dataset.

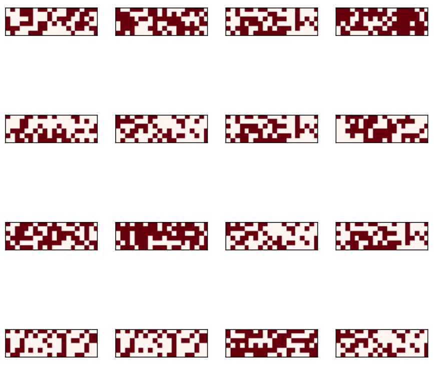
demonstrated the best result, achieving a balanced accuracy of 93.06% in the test set. With the exception of MOL-MutualInfo-PSO and MOL-Correlation-PSO, every other MOL combination achieved over 90% accuracy. Interestingly, all MOL strategies employed the complete set of features in an image format, suggesting convergence toward a common representation. However, even with the entire feature set, the algorithms selected distinct image sizes. For instance, MOL-Distance-ABC indicated the optimal format as a 30x4 image, while MOL-Distance-EPSO presented results with a 60x2 image shape.

The remainder experiments presented TabNet with 97,23% score in the test set and higher scores were also presented in the cross-validation. The strategy also presented statistical significance ($p < 0.05$) against Extreme Gradient Boosting (XGB)-E, MOL-ANOVA-PSO and XGB-B which were the best results obtained in the Embeded, MOL and Baseline experiments respectively.

The images presented in Figure 6.4 on page 71 pose a challenge for human visual identification due to the extensive number of features incorporated into the model's solution. However, upon closer inspection, it becomes apparent that images corresponding to outcome 0, located in row 1 column 4 and row 3 column 2, exhibit numerous darker points in comparison to others. This pattern is not exclusive to outcome 0 but is also observed in outcome 1 images. This visual complexity is interesting when considering the application of CNN. The intricate patterns might contain distinctive features that a CNN could evaluate for accurate predictions. On the other hand, the number of features and the variability in image sizes among different MOL strategies present potential challenges for a CNN, as the model may need to adapt to varying spatial configurations of information within these images.

Table 6.4: Statistical comparison of the obtained balanced accuracy at the training and testing for the Gene dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9131 ± 0.0122	0.9142	120
CNN	MOL	Correlation	ABC	0.8824 ± 0.0277	0.9016	120
CNN	MOL	Distance	ABC	0.9089 ± 0.0199	0.9193	120
CNN	MOL	Fisher	ABC	0.9131 ± 0.0224	0.9067	120
CNN	**MOL**	**Gain Ratio**	**ABC**	**0.9144 ± 0.0186**	**0.9281**	**120**
CNN	MOL	Mutual Info.	ABC	0.9051 ± 0.0121	0.9231	120
CNN	**MOL**	**ANOVA**	**PSO**	**0.9131 ± 0.0153**	**0.9306**	**120**
CNN	MOL	Correlation	PSO	0.8824 ± 0.0258	0.9092	120
CNN	MOL	Distance	PSO	0.9089 ± 0.0194	0.9180	120
CNN	MOL	Fisher	PSO	0.9131 ± 0.0192	0.9168	120
CNN	MOL	Gain Ratio	PSO	0.9144 ± 0.0188	0.9256	120
CNN	MOL	Mutual Info.	PSO	0.9051 ± 0.0223	0.8903	120
CNN	MOL	ANOVA	EPSO	0.9125 ± 0.0158	0.9142	120
CNN	MOL	Correlation	EPSO	0.8881 ± 0.023	0.9117	120
CNN	**MOL**	**Distance**	**EPSO**	**0.9072 ± 0.0197**	**0.9218**	**120**
CNN	MOL	Fisher	EPSO	0.9114 ± 0.0186	0.9105	120
CNN	MOL	Gain Ratio	EPSO	0.9148 ± 0.0184	0.9155	120
CNN	MOL	Mutual Info.	EPSO	0.8967 ± 0.0208	0.9079	120
KNN	E	-	-	0.7985 ± 0.0161	0.7961	108
LGBM	E	-	-	0.9635 ± 0.0130	0.9443	119
LR	E	-	-	0.9053 ± 0.0179	0.9144	116
ANN	E	-	-	0.9081 ± 0.0262	0.9071	16
RF	E	-	-	0.9297 ± 0.0180	0.9217	118
XGB	**E**	-	-	**0.9624 ± 0.0105**	**0.9446**	**117**
KNN	W	-	-	0.8132 ± 0.1483	0.7616	40
LGBM	W	-	-	**0.9994 ± 0.0003**	**0.9445**	**116**
LR	W	-	-	0.9195 ± 0.0574	0.9223	40
ANN	W	-	-	0.9997 ± 0.0008	0.9333	40
RF	W	-	-	0.9997 ± 0.0001	0.9321	40
XGB	W	-	-	0.9614 ± 0.0013	0.9293	40
KNN	B	-	-	0.7669 ± 0.0174	0.7640	-
LGBM	B	-	-	0.9545 ± 0.0150	0.9045	-
LR	B	-	-	0.8994 ± 0.0206	0.9061	-
ANN	B	-	-	0.9012 ± 0.0196	0.8980	-
RF	B	-	-	0.9203 ± 0.0186	0.8960	-
XGB	**B**	-	-	**0.9593 ± 0.0148**	**0.9071**	-
TabNet	-	-	-	**0.9979 ± 0.0053**	**0.9723**	**-**


(a) Outcome 0

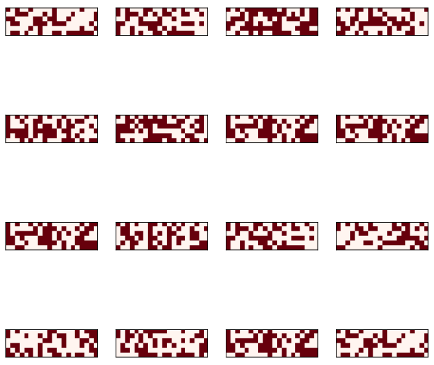

(b) Outcome 1

Figure 6.4: Input generated by MOL-ANOVA-PSO for the Gene dataset.

6.1.5 Glass

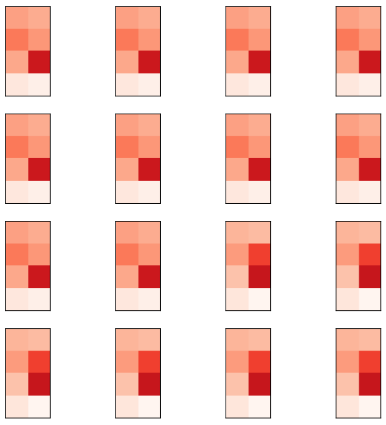
Table 6.5 on the following page presents the results obtained by each combination for the Glass dataset. Due to the number of iterations, instances, and features, this dataset can be challenging, and any wasted iteration can lead to poor performance, which is evident for every combination tested. The results obtained by MOL indicates that these combinations performed extremely poorly for this problem, and statistical significance is not required to demonstrate that they were the worst results and have no significance when compared to the others. The best obtained result was achieved by MOL-GainRatio-ABC with 58,60% balanced accuracy in the test set.

Experiment performed with the other strategies presented TabNet with the best outcome for this dataset, where the algorithm had close to 1% improvements compared against XGB-E. XGB also had the best performance among the Wrapper algorithms and the RF achieved the best performance using the Baseline approach.

Figure 6.5 on page 73 showcases the images generated by MOL-Distance-ABC. These images exhibit a few blocks with higher intensity, which are somewhat similar to instances found in both outcomes. This similarity could potentially challenge the performance of the CNN. However, it's worth noting that while these images are relatively interpretable to the human eye, the values they represent are associated with the minimum and maximum values found in the dataset. Consequently, the numerical scale of the image in different pixels differs from what the human eye perceives. This incongruity may pose a challenge for the CNN in interpreting these images accurately.

Table 6.5: Statistical comparison of the obtained balanced accuracy at the training and testing for the Glass dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.3978 ± 0.0586	0.4340	8
CNN	MOL	Correlation	ABC	0.354 ± 0.0685	0.2642	8
CNN	MOL	Distance	ABC	0.3721 ± 0.0497	0.2830	8
CNN	MOL	Fisher	ABC	0.4037 ± 0.0567	0.4340	8
CNN	**MOL**	**Gain Ratio**	**ABC**	**0.4224 ± 0.0729**	**0.5660**	**8**
CNN	MOL	Mutual Info.	ABC	0.3912 ± 0.0831	0.5660	8
CNN	MOL	ANOVA	PSO	0.3978 ± 0.0678	0.4340	8
CNN	MOL	Correlation	PSO	0.354 ± 0.0603	0.4340	8
CNN	MOL	Distance	PSO	0.3721 ± 0.0492	0.2830	8
CNN	MOL	Fisher	PSO	0.4037 ± 0.0829	0.4340	8
CNN	**MOL**	**Gain Ratio**	**PSO**	**0.4224 ± 0.0669**	**0.4340**	**8**
CNN	MOL	Mutual Info.	PSO	0.3912 ± 0.0841	0.4340	8
CNN	MOL	ANOVA	EPSO	0.3943 ± 0.0671	0.2642	8
CNN	MOL	Correlation	EPSO	0.3542 ± 0.0649	0.2642	8
CNN	**MOL**	**Distance**	**EPSO**	**0.3787 ± 0.0651**	**0.4340**	8
CNN	MOL	Fisher	EPSO	0.3847 ± 0.0764	0.2830	8
CNN	MOL	Gain Ratio	EPSO	0.3851 ± 0.0743	0.2642	8
CNN	MOL	Mutual Info.	EPSO	0.3938 ± 0.0773	0.2642	8
KNN	E	-	-	0.6372 ± 0.1116	0.5663	8
LGBM	E	-	-	0.6384 ± 0.0918	0.6184	8
LR	E	-	-	0.5928 ± 0.1087	0.6420	8
ANN	E	-	-	0.6839 ± 0.1175	0.6566	8
RF	E	-	-	0.7691 ± 0.1272	0.6409	8
XGB	**E**	-	-	**0.7052 ± 0.1655**	**0.6909**	**8**
KNN	W	-	-	0.7816 ± 0.2102	0.4393	6
LGBM	W	-	-	1.0000 ± 0.0000	0.6383	8
LR	W	-	-	0.6156 ± 0.499	0.3722	6
ANN	W	-	-	0.7069 ± 0.1846	0.3897	8
RF	W	-	-	1.0000 ± 0.0000	0.6209	8
XGB	**W**	-	-	**0.7330 ± 0.2497**	**0.6496**	**6**
KNN	B	-	-	0.5104 ± 0.0928	0.5163	-
LGBM	B	-	-	0.6646 ± 0.1681	0.6383	-
LR	B	-	-	0.4011 ± 0.0620	0.3523	-
ANN	B	-	-	0.4373 ± 0.0535	0.3723	-
RF	**B**	-	-	**0.7614 ± 0.1435**	**0.6822**	-
XGB	B	-	-	0.7030 ± 0.1497	0.6296	-
TabNet	-	-	-	**0.9386 ± 0.0612**	**0.7310**	**-**


(a) Outcome 0

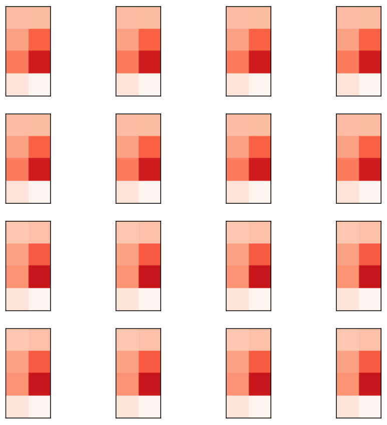

(b) Outcome 1

Figure 6.5: Input generated by MOL-GainRatio-ABC for the Glass dataset.

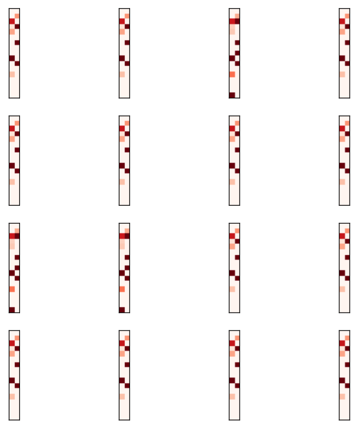
6.1.6 Heart

Table 6.6 on the following page provides an overview of the outcomes obtained for the Heart dataset across various combinations. Among the MOL strategies, results are notably consistent in terms of the test score, with the most favorable outcome achieved by MOL-MutualInfo-ABC, boasting a 79.13% balanced accuracy in the test set. Combinations employing Mutual Information (Mutual Information (MI)) as a map strategy, particularly with both EPSO and ABC algorithms, yielded superior results, indicating their effectiveness in generating image formats. All MOL strategies opted for selecting 34 features to generate images. When comparing RF-E, which had the second best performance in the test set, and MOL-MutualInfo-ABC, no statistical significance was found in the test results ($p >= 0.05$).

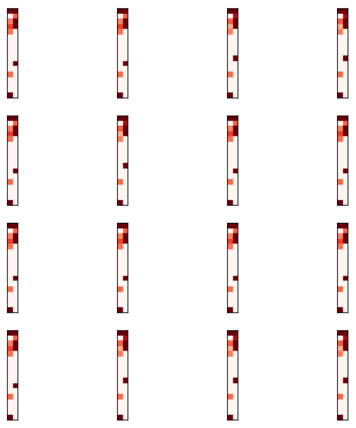
An examination of the images generated by MOL-MutualInfo-ABC in Figure 6.6 on page 75 reveals a distinct pattern in the image corresponding to outcome 1. Each sample in this category is identical and does not appear in the image representing outcome 0. While this pattern is evident in the sampled images, it is crucial to note that the entire dataset may not uniformly exhibit the same behavior. If the complete dataset show this pattern, it would be identifiable by the human eye, allowing for easy differentiation between positive and negative classes. However, it is essential to emphasize that this generalized experiment does not account for dataset-specific nuances. With a more nuanced construction of the CNN architecture and fine-tuning of other parameters, there is potential for enhanced accuracy, especially considering the observed image similarities in these samples.

Table 6.6: Statistical comparison of the obtained balanced accuracy at the training and testing for the Heart dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.8478 ± 0.0351	0.7435	34
CNN	MOL	Correlation	ABC	0.8507 ± 0.0449	0.7696	34
CNN	MOL	Distance	ABC	0.8362 ± 0.0421	0.7565	34
CNN	MOL	Fisher	ABC	0.8478 ± 0.0339	0.7870	34
CNN	MOL	Gain Ratio	ABC	0.8449 ± 0.0481	0.7870	34
CNN	**MOL**	**Mutual Info.**	**ABC**	**0.8566 ± 0.0352**	**0.7913**	**34**
CNN	MOL	ANOVA	PSO	0.8478 ± 0.0363	0.7609	34
CNN	**MOL**	**Correlation**	**PSO**	**0.8507 ± 0.0366**	**0.7739**	**34**
CNN	MOL	Distance	PSO	0.8362 ± 0.0471	0.7652	34
CNN	MOL	Fisher	PSO	0.8478 ± 0.0355	0.7478	34
CNN	MOL	Gain Ratio	PSO	0.8449 ± 0.0466	0.7609	34
CNN	MOL	Mutual Info.	PSO	0.8566 ± 0.0438	0.7696	34
CNN	MOL	ANOVA	EPSO	0.8478 ± 0.0374	0.7478	34
CNN	MOL	Correlation	EPSO	0.8515 ± 0.0378	0.7522	34
CNN	MOL	Distance	EPSO	0.8326 ± 0.0498	0.7652	34
CNN	MOL	Fisher	EPSO	0.8449 ± 0.0408	0.7783	34
CNN	MOL	Gain Ratio	EPSO	0.8478 ± 0.0467	0.7696	34
CNN	**MOL**	**Mutual Info.**	**EPSO**	**0.8508 ± 0.0459**	**0.7783**	**34**
KNN	E	-	-	0.8379 ± 0.0485	0.7721	25
LGBM	E	-	-	0.8447 ± 0.0406	0.7511	23
LR	E	-	-	0.8464 ± 0.0370	0.7646	31
ANN	E	-	-	0.8473 ± 0.0495	0.7598	29
RF	**E**	-	-	**0.8500 ± 0.0414**	**0.7895**	**29**
XGB	E	-	-	0.8406 ± 0.0341	0.7589	27
KNN	W	-	-	0.8750 ± 0.3694	0.7379	34
LGBM	W	-	-	1.0000 ± 0.0000	0.7421	24
LR	**W**	-	-	**0.8577 ± 0.1287**	**0.7733**	**34**
ANN	W	-	-	0.9230 ± 0.0355	0.7562	34
RF	W	-	-	1.0000 ± 0.0000	0.7592	34
XGB	W	-	-	1.0000 ± 0.0000	0.7392	34
KNN	B	-	-	0.8297 ± 0.0428	0.7379	-
LGBM	B	-	-	0.8278 ± 0.0413	0.7334	-
LR	B	-	-	0.8447 ± 0.0385	0.7733	-
ANN	B	-	-	0.8401 ± 0.0609	0.7559	-
RF	**B**	-	-	**0.8288 ± 0.0427**	**0.7769**	-
XGB	B	-	-	0.7759 ± 0.0427	0.7250	-
TabNet	-	-	-	**0.9712 ± 0.0341**	**0.7127**	-



(a) Outcome 0



(b) Outcome 1

Figure 6.6: Input generated by MOL-MutualInfo-ABC for the Heart dataset.

6.1.7 Horse

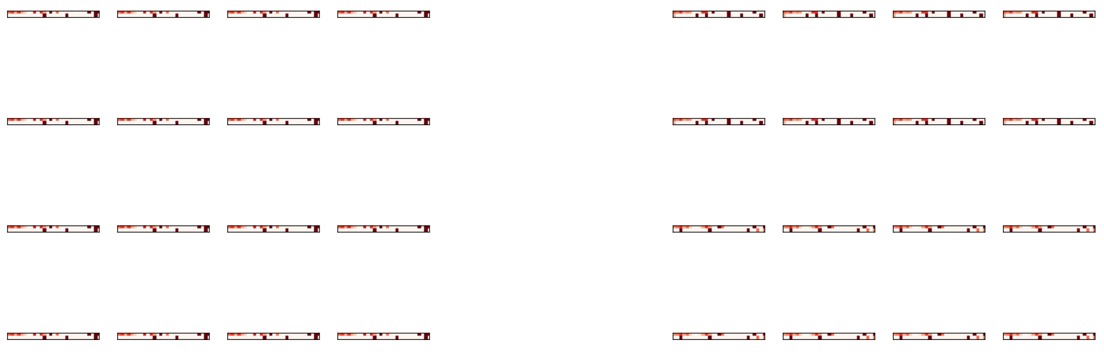
Table 6.6 on the facing page presents the performance metrics of various models and feature selection methods on the Horse dataset. Among MOL approaches, MOL-Correlation-PSO present the best performance. However, the training scores obtained in the cross-validation indicates a possible underfit scenario.

The best overall results were achieved by MOL-Correlation-PSO, followed by MOL-Distance-ABC and TabNet. TabNet performance in the cross-validation points out statistical significance when compared to both MOL approaches ($p < 0.05$), while the testing indicates a 3% superior performance with the predictions produced by the combination with the correlation map and Particle Swarm Optimization (PSO) algorithm.

Figure 6.7 on page 77 showcases the images generated by MOL-Correlation-PSO. Within this subset, images corresponding to outcome 0 exhibit a similar pattern, while images associated with outcome 1 display a more diverse range of patterns. It's important to note that this particular problem involves more than two potential outcomes, implying that the patterns observed in these groups might also appear in the third group. Due to the substantial number of features involved, distinguishing clear patterns with the naked eye is challenging. Moreover, the fact that the optimization algorithm did not reduce the number of selected features suggests that the algorithms struggled to simultaneously create smaller and more uniform images. Consequently, this issue might have trapped the solutions in local optima, impeding convergence.

Table 6.7: Statistical comparison of the obtained balanced accuracy at the training and testing for the Horse dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.6258 ± 0.0594	0.6923	58
CNN	MOL	Correlation	ABC	0.6483 ± 0.0499	0.7033	58
CNN	**MOL**	**Distance**	**ABC**	**0.6414 ± 0.0704**	**0.7473**	**58**
CNN	MOL	Fisher	ABC	0.6524 ± 0.0547	0.7143	58
CNN	MOL	Gain Ratio	ABC	0.6521 ± 0.0632	0.7033	58
CNN	MOL	Mutual Info.	ABC	0.6372 ± 0.0427	0.7363	58
CNN	MOL	ANOVA	PSO	0.6258 ± 0.0598	0.7143	28
CNN	**MOL**	**Correlation**	**PSO**	**0.6483 ± 0.0515**	**0.7692**	**58**
CNN	MOL	Distance	PSO	0.6414 ± 0.0653	0.7253	30
CNN	MOL	Fisher	PSO	0.6524 ± 0.056	0.6923	28
CNN	MOL	Gain Ratio	PSO	0.6521 ± 0.0542	0.6923	58
CNN	MOL	Mutual Info.	PSO	0.6372 ± 0.0498	0.7033	30
CNN	MOL	ANOVA	EPSO	0.6206 ± 0.0587	0.7253	32
CNN	MOL	Correlation	EPSO	0.6372 ± 0.0539	0.7253	32
CNN	MOL	Distance	EPSO	0.6468 ± 0.0617	0.7143	32
CNN	MOL	Fisher	EPSO	0.656 ± 0.0542	0.7033	32
CNN	MOL	Gain Ratio	EPSO	0.6447 ± 0.0554	0.7143	32
CNN	**MOL**	**Mutual Info.**	**EPSO**	**0.6427 ± 0.0524**	**0.7253**	**18**
KNN	E	-	-	0.4712 ± 0.0749	0.4921	53
LGBM	E	-	-	0.5352 ± 0.0958	0.6111	43
LR	**E**	-	-	**0.5378 ± 0.1089**	**0.7095**	**57**
ANN	E	-	-	0.5798 ± 0.0849	0.6071	57
RF	E	-	-	0.5059 ± 0.0815	0.6135	57
XGB	E	-	-	0.5846 ± 0.0772	0.6016	52
KNN	W	-	-	0.7360 ± 0.1988	0.3897	20
LGBM	W	-	-	0.9980 ± 0.2779	0.6548	57
LR	W	-	-	0.7520 ± 0.0751	0.6987	39
ANN	W	-	-	0.9768 ± 0.4888	0.6016	57
RF	W	-	-	0.9980 ± 0.0019	0.6238	39
XGB	**W**	-	-	**0.8380 ± 0.2779**	**0.7087**	**57**
KNN	B	-	-	0.7290 ± 0.0555	0.4643	-
LGBM	B	-	-	0.8910 ± 0.0847	0.6454	-
LR	B	-	-	0.6754 ± 0.0847	0.6754	-
ANN	B	-	-	0.7235 ± 0.0767	0.6010	-
RF	B	-	-	0.5579 ± 0.0662	0.5879	-
XGB	**B**	-	-	**0.7149 ± 0.0519**	**0.7053**	-
TabNet	-	-	-	**0.8502 ± 0.3121**	**0.7360**	**-**



(a) Outcome 0 (b) Outcome 1

Figure 6.7: Input generated by MOL-Correlation-PSO for the Horse dataset.

6.1.8 Soybean

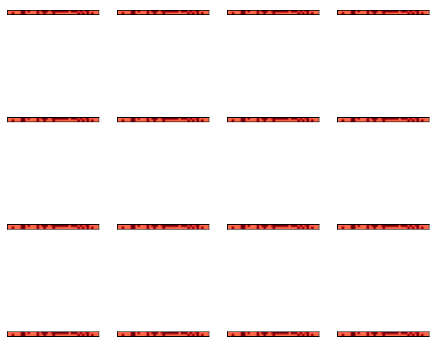
The table with identifier 6.8 on the following page displays the outcomes of each model-feature selection method combination for the Soybean dataset.

The test set results showed that LGBM-E had the best overall performance, achieving the same score as ANN-W. The MOL-Distance-ABC and MOL-Correlation-EPSO methods also had good results, but not as good as the two aforementioned approaches. On the validation set, ANN-W had the best k-folds results and was statistically significant when compared to the other methods ($p < 0.05$). Despite MOL-Distance-ABC's strong overall performance, it was not able to match the performance of ANN-W on the test set. Both methods used the same number of features, suggesting that transforming the problem into images may not lead to improved classification results.

Figure 6.8 on page 79 showcases the images generated by MOL-Distance-ABC for the Soybean dataset. Upon closer examination, it becomes evident that the subset contains images in Outcome 0 that are distinct from those in Outcome 1. In Outcome 1, the right side of the images displays three pixels that differentiate them, while the left part of the image exhibits a distinct wave-like pattern created by the darker pixels. Interestingly, this pattern is absent in the images associated with Outcome 0 within the presented subset. The model results suggest that not all images in the dataset are identical, but it has nonetheless achieved excellent performance on this dataset.

Table 6.8: Statistical comparison of the obtained balanced accuracy at the training and testing for the Soybean dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9752 ± 0.0671	0.9294	82
CNN	MOL	Correlation	ABC	0.9673 ± 0.0414	0.9235	82
CNN	**MOL**	**Distance**	**ABC**	**0.9772 ± 0.0619**	**0.9694**	**82**
CNN	MOL	Fisher	ABC	0.9788 ± 0.0568	0.9635	82
CNN	MOL	Gain Ratio	ABC	0.9456 ± 0.1014	0.9218	82
CNN	MOL	Mutual Info.	ABC	0.9032 ± 0.2018	0.8059	46
CNN	MOL	ANOVA	PSO	0.8745 ± 0.2278	0.8176	42
CNN	MOL	Correlation	PSO	0.8607 ± 0.2475	0.8647	40
CNN	MOL	Distance	PSO	0.8374 ± 0.218	0.7941	42
CNN	MOL	Fisher	PSO	0.8798 ± 0.1579	0.7000	38
CNN	MOL	Gain Ratio	PSO	0.8731 ± 0.2322	0.8412	40
CNN	MOL	Mutual Info.	PSO	**0.8635 ± 0.0554**	**0.9235**	82
CNN	MOL	ANOVA	EPSO	0.8006 ± 0.1822	0.8000	38
CNN	MOL	**Correlation**	**EPSO**	**0.8831 ± 0.109**	**0.8588**	38
CNN	MOL	Distance	EPSO	0.8898 ± 0.1683	0.8353	38
CNN	MOL	Fisher	EPSO	0.2711 ± 0.1911	0.1471	38
CNN	MOL	Gain Ratio	EPSO	0.8206 ± 0.177	0.7941	38
CNN	MOL	Mutual Info.	EPSO	0.2558 ± 0.1916	0.2647	26
KNN	E	-	-	0.9584 ± 0.0266	0.9143	34
LGBM	**E**	-	-	**0.9607 ± 0.0305**	**0.9142**	**52**
LR	E	-	-	0.9737 ± 0.0220	0.9125	79
ANN	E	-	-	0.9704 ± 0.0214	0.9125	61
RF	E	-	-	0.9807 ± 0.0178	0.9173	78
XGB	E	-	-	0.9694 ± 0.0208	0.9125	38
KNN	W	-	-	0.9742 ± 0.0075	0.9260	82
LGBM	W	-	-	1.0000 ± 0.0006	0.9534	55
LR	W	-	-	0.9852 ± 0.0090	0.9525	80
ANN	**W**	-	-	**0.9913 ± 0.0039**	**0.9642**	**80**
RF	W	-	-	1.0000 ± 0.0000	0.9514	55
XGB	W	-	-	1.0000 ± 0.0000	0.9334	45
KNN	B	-	-	0.9620 ± 0.0209	0.9260	-
LGBM	B	-	-	0.9695 ± 0.0250	0.9575	-
LR	B	-	-	0.9730 ± 0.0222	0.9525	-
ANN	**B**	-	-	**0.9723 ± 0.0243**	**0.9642**	-
RF	B	-	-	0.9764 ± 0.0211	0.9552	-
XGB	B	-	-	0.9579 ± 0.0220	0.9534	-
TabNet	-	-	-	**0.9937 ± 0.0085**	**0.9564**	-


(a) Outcome 0

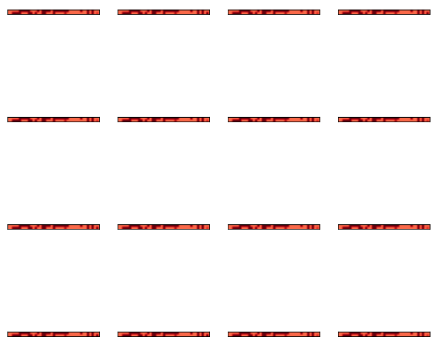

(b) Outcome 1

Figure 6.8: Input generated by MOL-Distance-ABC for the Soybean dataset.

6.1.9 Thyroid

The table labeled as 6.9 on the following page exhibits the results of each combination of model and feature selection method for the Thyroid dataset.

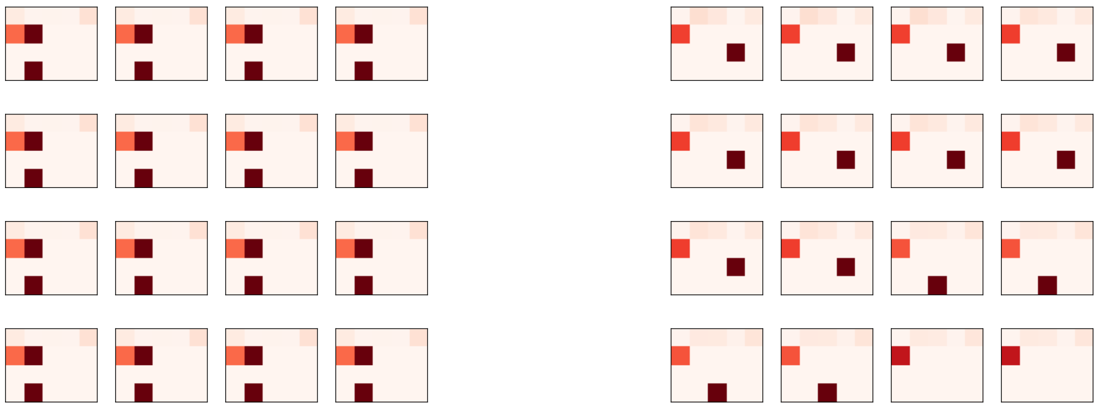
The test results revealed that XGB-E exhibited the most outstanding performance among all the experiments, outperforming the others. Following closely was MOL-GainRatio-ABC, which achieved a similar score to MOL-Distance-PSO. The remaining MOL combinations also demonstrated improved test scores compared to their respective baselines. However, upon analyzing the validation sets, it became evident that the results obtained by MOL seemed to exhibit slight underfitting, with the achieved results on the train set being less than 3%.

Statistical analysis indicated that XGB-E showed significant differences when compared to the MOL methods ($p < 0.05$). It is worth noting that while XGB-E had the best overall performance in the test set, MOL could still maintain its performance and achieve a near point to XBG-E performance for this dataset.

The images generated by MOL-GainRatio-ABC for the Thyroid dataset are depicted in Figure 6.9 on page 81. This dataset comprises 19 distinct classes, and the presented images correspond to outcomes 0 and 1. Upon close examination, it becomes evident that the pattern identified in the subset associated with outcome 0 does not repeat in outcome 1, making it easily discernible to the human eye. Given the dataset's numerous instances and classes, the generated images may exhibit variability, with some images from outcome 0 or 1 possibly appearing in other outcomes as well. However, the CNN inherent characteristics in image analysis may capture underlying patterns that contribute or hinder the strategy results.

Table 6.9: Statistical comparison of the obtained balanced accuracy at the training and testing for the Thyroid dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9285 ± 0.005	0.9552	20
CNN	MOL	Correlation	ABC	0.9263 ± 0.0018	0.9353	20
CNN	MOL	Distance	ABC	0.9344 ± 0.0062	0.9624	20
CNN	MOL	Fisher	ABC	0.9352 ± 0.0054	0.9668	20
CNN	**MOL**	**Gain Ratio**	**ABC**	**0.9313 ± 0.0052**	**0.9685**	**20**
CNN	MOL	Mutual Info.	ABC	0.9304 ± 0.0045	0.9413	20
CNN	MOL	ANOVA	PSO	0.9298 ± 0.0061	0.9618	14
CNN	MOL	Correlation	PSO	0.9257 ± 0.0015	0.9369	14
CNN	**MOL**	**Distance**	**PSO**	**0.9294 ± 0.0054**	**0.9673**	**12**
CNN	MOL	Fisher	PSO	0.9292 ± 0.0061	0.9380	12
CNN	MOL	Gain Ratio	PSO	0.9289 ± 0.0063	0.9424	14
CNN	MOL	Mutual Info.	PSO	0.9272 ± 0.0041	0.9353	12
CNN	MOL	ANOVA	EPSO	0.9254 ± 0.0008	0.9270	6
CNN	MOL	Correlation	EPSO	0.9254 ± 0.0008	0.9270	6
CNN	**MOL**	**Distance**	**EPSO**	**0.9278 ± 0.0048**	**0.9386**	**8**
CNN	MOL	Fisher	EPSO	0.9265 ± 0.0026	0.9347	6
CNN	MOL	Gain Ratio	EPSO	0.9254 ± 0.0008	0.9358	4
CNN	MOL	Mutual Info.	EPSO	0.9272 ± 0.0044	0.9341	6
KNN	E	-	-	0.6557 ± 0.0724	0.6637	17
LGBM	E	-	-	0.9972 ± 0.0039	0.9947	9
LR	E	-	-	0.7145 ± 0.0749	0.7330	18
ANN	E	-	-	0.9168 ± 0.0394	0.9172	16
RF	E	-	-	0.9981 ± 0.0034	0.9816	9
XGB	**E**	**-**	**-**	**0.9839 ± 0.0193**	**0.9848**	**10**
KNN	W	-	-	0.8042 ± 0.0042	0.8046	7
LGBM	**W**	**-**	**-**	**0.9421 ± 0.1523**	**0.9399**	**14**
LR	W	-	-	0.6431 ± 0.2803	0.4146	14
ANN	W	-	-	0.8900 ± 0.0311	0.7665	7
RF	W	-	-	0.9345 ± 0.2712	0.9352	7
XGB	B	-	-	0.9421 ± 0.0136	0.9364	-
KNN	B	-	-	0.5235 ± 0.0394	0.5234	-
LGBM	**B**	**-**	**-**	**0.9327 ± 0.0138**	**0.9299**	**-**
LR	B	-	-	0.3776 ± 0.0364	0.4228	-
ANN	B	-	-	0.7137 ± 0.0718	0.7586	-
RF	B	-	-	0.9249 ± 0.0233	0.9254	-
XGB	B	-	-	0.9215 ± 0.0136	0.9264	-
TabNet	**-**	**-**	**-**	**0.9984 ± 0.0069**	**0.9597**	-



(a) Outcome 0 (b) Outcome 1

Figure 6.9: Input generated by MOL-Gain Ratio-ABC for the Thyroid dataset.

6.1.10 KDD

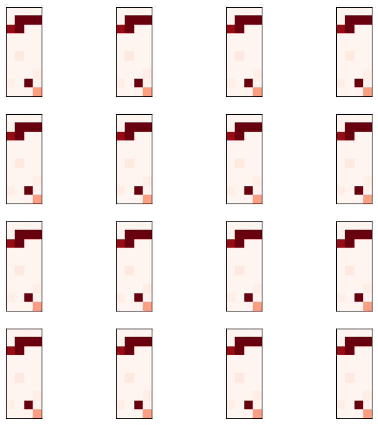
The results obtained by each combination of strategies and algorithms, for the KDD dataset, can be found in Table 6.10 on page 83.

The KDD dataset serves as a benchmark for which achieving high accuracy is relatively straightforward, as demonstrated in this experiment. In the MOL experiments, every combination achieved a test set accuracy exceeding 90%, and it is noteworthy that the number of selected features remained consistent across all combinations, with only two features being excluded from the final dataset. Among the MOL combinations, the best performance was observed in the MOL-Correlation-EPSO combination.

The remaining experiments also yielded excellent results when feature selection strategies were not applied. Baseline algorithms outperformed the same algorithms with FS strategies. The highest accuracy was achieved by the LGBM algorithm. Upon analyzing statistical significance across this experiment, no statistically significant differences ($p >= 0.05$) were found in the cross-validation when performing pairwise comparisons among the algorithms with the highest accuracy in the test set.

Figure 6.10 on the next page displays the images generated by MOL-Distance-ABC for the KDD dataset. It's important to note that the values in the KDD dataset vary from 0 to millions. To create illustrative images, the dataset used for generating these images was normalized between 0 and 1. While some recurring patterns are discernible, the generated images are distinguishable to the human eye. Upon closer inspection of this small sample of images, it becomes apparent that certain patterns appear to be exclusive to Outcome 1. This observation potentially aligns with the algorithm's overall performance in classifying this specific outcome.

It's worth emphasizing that the KDD dataset encompasses multiple classes, extending beyond two outcomes. Analyzing the similarities among the images provides insights into the algorithm's performance based on this visual representation. However, it's crucial to consider that the presence of other images sharing the same pattern across classes might pose challenges for the CNN's performance, particularly given the inherent complexity of the problem.


(a) Outcome 0

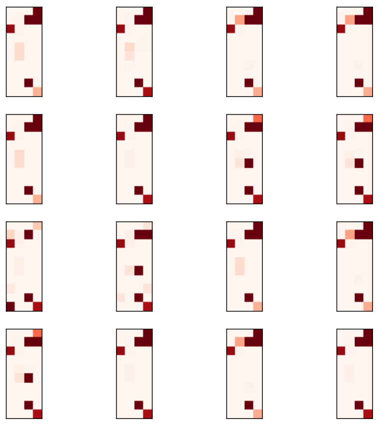

(b) Outcome 1

Figure 6.10: Input generated by MOL-Correlation-EPSO for the Gene dataset.

6.2 Real-World Problems

This section presents the results obtained from the Real-World experiment discussed in the previous chapter. It is essential to note that, in this analysis, only the test scores were evaluated. Due to comparisons against literature results and the unavailability of training scores, the focus is solely on the performance of the model on the test set.

6.2.1 Cardiac Pathology

Table 6.11 on page 85 presents the results obtained from the study, including the test score and the number of selected features by each algorithm under a specific strategy.

Among the optimization algorithms, the mapping strategy based on the Euclidean distance yielded the best scores when combined with all three algorithms. ABC was unable to generalize the data, achieving an accuracy of only 64.42% on the test set. PSO, on the other hand, achieved very good accuracy on the test set (92.25%). Finally, the EPSO algorithm achieved a training score of 93.60% and a test score of 93.92%.

Among the approaches that utilized feature selection, the ANN achieved the highest test score

Table 6.10: Statistical comparison of the obtained balanced accuracy at the training and testing for the KDD dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Train**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9388 ± 0.0526	0.8840	40
CNN	MOL	Correlation	ABC	0.9402 ± 0.0604	0.9330	40
CNN	MOL	Distance	ABC	0.9123 ± 0.0669	0.9199	40
CNN	MOL	Fisher	ABC	0.9824 ± 0.0078	0.9933	40
CNN	MOL	Gain Ratio	ABC	0.9872 ± 0.0077	0.9933	40
CNN	**MOL**	**Mutual Info.**	**ABC**	**0.9824 ± 0.0068**	**0.9933**	**40**
CNN	MOL	ANOVA	PSO	0.9812 ± 0.0079	0.9914	40
CNN	MOL	Correlation	PSO	0.9907 ± 0.0073	0.9865	40
CNN	**MOL**	**Distance**	**PSO**	**0.9812 ± 0.0083**	**0.9935**	**40**
CNN	MOL	Fisher	PSO	0.9306 ± 0.0493	0.9243	40
CNN	MOL	Gain Ratio	PSO	0.9848 ± 0.0097	0.9865	40
CNN	MOL	Mutual Info.	PSO	0.9807 ± 0.0075	0.9542	40
CNN	MOL	ANOVA	EPSO	0.9819 ± 0.0086	0.9895	40
CNN	**MOL**	**Correlation**	**EPSO**	**0.9892 ± 0.0081**	**0.9935**	**40**
CNN	MOL	Distance	EPSO	0.9851 ± 0.0095	0.9692	40
CNN	MOL	Fisher	EPSO	0.9845 ± 0.0082	0.9445	40
CNN	MOL	Gain Ratio	EPSO	0.9862 ± 0.0106	0.9923	40
CNN	MOL	Mutual Info.	EPSO	0.9871 ± 0.0091	0.9913	40
KNN	E	-	-	0.7509 ± 0.0493	0.7321	28
LGBM	E	-	-	0.8245 ± 0.0587	0.7962	34
LR	E	-	-	0.8132 ± 0.0771	0.7215	28
ANN	E	-	-	0.8734 ± 0.4374	0.7922	32
RF	E	-	-	0.7648 ± 0.0348	0.7456	22
XGB	**E**	**-**	**-**	**0.8031 ± 0.0404**	**0.7503**	**-**
KNN	W	-	-	0.9261 ± 0.4114	0.6919	40
LGBM	W	-	-	0.9012 ± 0.0327	0.8162	38
LR	W	-	-	0.8352 ± 0.2482	0.7878	28
ANN	W	-	-	0.8056 ± 0.0146	0.7476	15
RF	W	-	-	0.9998 ± 0.1458	0.7947	40
XGB	**W**	**-**	**-**	**0.9997 ± 0.0465**	**0.8756**	**41**
KNN	B	-	-	0.7009 ± 0.0657	0.6919	-
LGBM	**B**	-	-	**0.9803 ± 0.0429**	**0.9998**	-
LR	B	-	-	0.8002 ± 0.0271	0.7802	-
ANN	B	-	-	0.9845 ± 0.0049	0.9914	-
RF	B	-	-	0.8048 ± 0.0888	0.8303	-
XGB	B	-	-	0.9858 ± 0.0100	0.9997	-
TabNet	**-**	**-**	**-**	**0.9862 ± 0.0028**	**0.9761**	-

at 90.38% under an Embedded FS strategy, while the Wrapper strategy resulted in a 90.58% test score. The LGBM achieved a 90.59% balanced accuracy at the test set among models without any feature selection.

The images displayed in Figure 6.11 depict the generated patterns for the dataset. While some pixels in these images are discernible to the human eye, it's important to note that the values within these images are scaled to the maximum number. For instance, the first pixel is a binary variable. Consequently, these produced images may appear to have the same color, but the underlying values differ and are constrained to distinct scales.

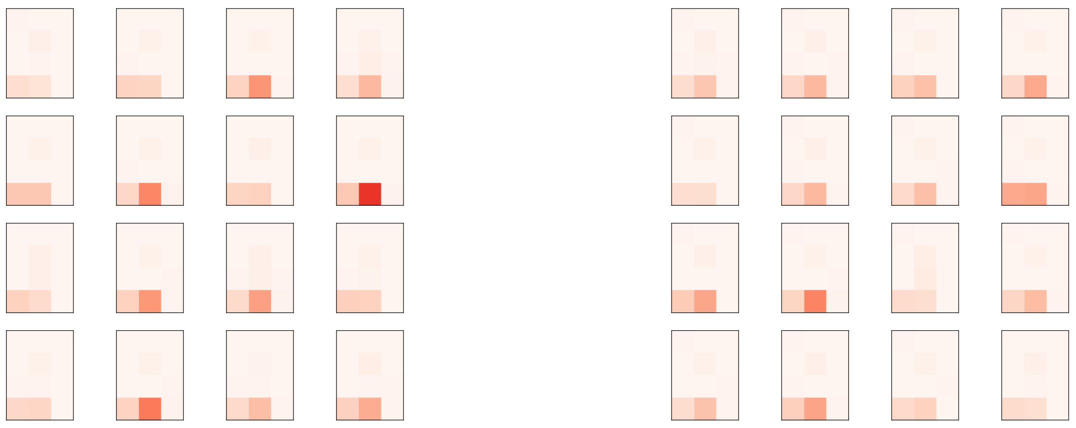

(a) Outcome 0 (b) Outcome 1

Figure 6.11: Input generated by MOL-Correlation-PSO for the Cardiac Pathology dataset.

6.2.2 Forest Cover Type

Table 6.12 on page 86 presents the results obtained from the study, showcasing the test scores and the number of selected features by each algorithm under a specific strategy.

In this dataset, we compared the results obtained by XGB, LGBM, CatBoost, AutoML Tables, and TabNet with the findings reported in published papers available in the literature. Among these algorithms, TabNet achieved the highest accuracy of 96.99%, securing the top position. MOL-Distance-EPSO came in second, recording the best score among all other MOL approaches with a respectable 95.08% accuracy.

Regarding the process of feature selection introduced in MOL, an interesting observation emerged as all the algorithms and combinations selected the same number of features. This uniform behavior indicates that the selected optimization algorithms might not have fully converged. There is a possibility that the optimization function used in the MOL optimization phase needs further refinement. Despite the good results achieved by the Distance-EPSO

Table 6.11: Statistical comparison of the obtained balanced accuracy at the testing for the Cardiac Pathology dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.9254	12
CNN	MOL	Correlation	ABC	0.9250	12
CNN	**MOL**	**Distance**	**ABC**	**0.9254**	**12**
CNN	MOL	Fisher	ABC	0.6559	12
CNN	MOL	Gain Ratio	ABC	0.6559	12
CNN	MOL	Mutual Info.	ABC	0.9245	12
CNN	MOL	ANOVA	PSO	0.9253	12
CNN	**MOL**	**Correlation**	**PSO**	**0.9263**	**12**
CNN	MOL	Distance	PSO	0.9254	12
CNN	MOL	Fisher	PSO	0.6241	12
CNN	MOL	Gain Ratio	PSO	0.7258	4
CNN	MOL	Mutual Info.	PSO	0.9254	12
CNN	MOL	ANOVA	EPSO	0.9232	4
CNN	MOL	Correlation	EPSO	0.8932	4
CNN	**MOL**	**Distance**	**EPSO**	**0.9250**	**4**
CNN	MOL	Fisher	EPSO	0.6442	4
CNN	MOL	Gain Ratio	EPSO	0.6442	4
CNN	MOL	Mutual Info.	EPSO	0.9209	4
ANN	E	-	-	**0.9038**	**6**
KNN	E	-	-	0.7310	6
LR	E	-	-	0.9030	9
RF	E	-	-	0.5178	1
XGB	E	-	-	0.8888	10
LGBM	E	-	-	0.9000	11
ANN	W	-	-	0.9038	5
KNN	W	-	-	0.9019	5
LR	W	-	-	**0.9058**	**5**
RF	W	-	-	0.8711	5
XGB	W	-	-	0.9042	5
LGBM	W	-	-	0.9012	12
ANN	S	-	-	**0.9050**	13
KNN	S	-	-	0.7457	13
LR	S	-	-	0.9030	13
RF	S	-	-	0.8990	13
XGB	S	-	-	0.8888	13
LGBM	S	-	-	**0.9059**	**13**
TabNet	-	-	-	**0.9147**	-

combination, this observation raises concerns that the current optimization phase might be hindering the overall process for this dataset.

Table 6.12: Statistical comparison of the obtained accuracy at the testing for the Forest Cover Type dataset

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**Test**	**# Feat**
CNN	MOL	ANOVA	ABC	0.8683	52
CNN	MOL	Correlation	ABC	0.6683	52
CNN	**MOL**	**Distance**	**ABC**	**0.8684**	**52**
CNN	MOL	Fisher	ABC	0.8156	52
CNN	MOL	Gain Ratio	ABC	0.8187	52
CNN	MOL	Mutual Info.	ABC	0.8348	52
CNN	MOL	ANOVA	EPSO	0.8765	52
CNN	MOL	Correlation	EPSO	0.6322	52
CNN	**MOL**	**Distance**	**EPSO**	**0.9508**	**52**
CNN	MOL	Fisher	EPSO	0.6442	52
CNN	MOL	Gain Ratio	EPSO	0.6442	52
CNN	MOL	Mutual Info.	EPSO	0.9209	52
CNN	MOL	ANOVA	PSO	0.8467	52
CNN	MOL	Correlation	PSO	0.7133	52
CNN	MOL	Distance	PSO	0.8133	52
CNN	MOL	Fisher	PSO	0.8789	52
CNN	MOL	Gain Ratio	PSO	0.7960	52
CNN	**MOL**	**Mutual Info.**	**PSO**	**0.9256**	**12**
XGB [5]	-	-	-	0.8934	-
LGBM [5]	-	-	-	0.8928	-
CatBoost [5]	-	-	-	0.8514	-
AutoML Tables [5]	-	-	-	0.9495	-
TabNet [5]	**-**	**-**	**-**	**0.9699**	-

Upon analyzing the samples of the generated images shown in Figure 6.12, several key points are evident in the images produced by MOL-Distance-EPSO. These key points are situated at the first and last pixels of the first row, and the last pixel of the last row. Notably, these points are easily visualized, and the colors observed range between shades of grey and white.

Similar to findings from previous datasets, the generated patterns exhibit different patterns in both outcomes, where the values are set into the same scale. In the other hand, the values that the CNN analyzes are in another scale, potentially impacting the learning process. The extent to which this mixing hinders learning depends on the prevalence of repeatable patterns within the dataset.

While the CNN may encounter challenges in this regard, it has the ability to search for and identify other types of patterns and numerical values associated with each black color pixel. This capability enables the CNN to detect hidden patterns that may not be immediately apparent to the human eye. As a result, the CNN's computational power allows it to excel in uncovering intricate patterns within the generated images, ultimately contributing to a more comprehensive understanding of the underlying data.

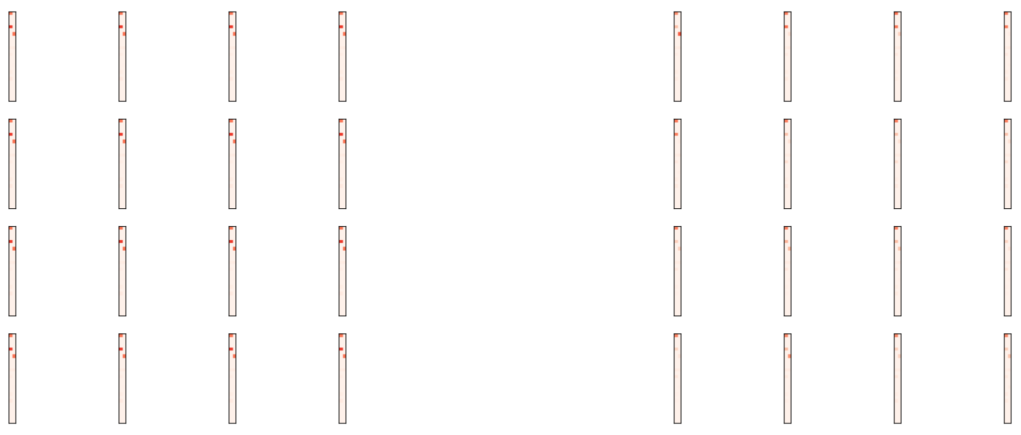

(a) Outcome 0 (b) Outcome 1

Figure 6.12: Input generated by MOL-Distance-EPSO for the Forest Cover dataset.

6.2.3 Poker Hand

Table 6.13 on the next page provides a comprehensive overview of the obtained results, presenting both the test scores and the number of selected features by each algorithm under a specific strategy.

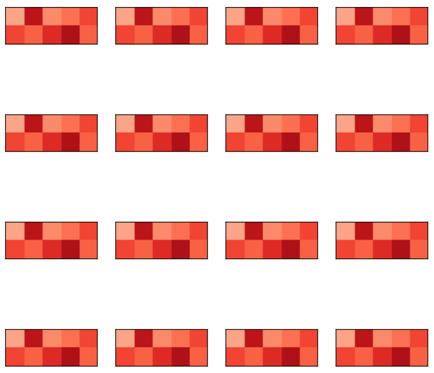
Similar to the previous dataset, the performance results for XGB, LGBM, CatBoost, and Deep Neural Decision Tree are sourced from published papers in the literature. Among the outcomes, TabNet exhibited the highest accuracy, reaching an impressive 99.20%. Following closely are MOL-ANOVA-EPSO with an accuracy score of 90.70% and MOL-GainRatio-EPSO with a score of 90.72%.

In terms of the feature selection process executed by MOL, it is evident that the optimization process encountered challenges in identifying a combination with fewer than 10 features. This outcome implies that at least one feature was consistently included in the selected combinations. However, the results reveal the significance of this particular combination, as demonstrated by CNN-MOL-ANOVA achieving a noteworthy accuracy improvement of around 30% accuracy score compared to other approaches like ABC. This emphasizes the pivotal role of selecting appropriate components to constitute the MOL pipeline, underlining the influential role this process plays in achieving optimal results.

Table 6.13: Statistical comparison of the obtained accuracy at the testing for the Poker Hand dataset

Algorithm	FS Str	Map Str	Opt Str	Test	# Feat
CNN	**MOL**	**ANOVA**	**ABC**	**0.8973**	**10**
CNN	MOL	Correlation	ABC	0.8610	10
CNN	MOL	Distance	ABC	0.7815	10
CNN	MOL	Fisher	ABC	0.8015	10
CNN	MOL	Gain Ratio	ABC	0.8509	10
CNN	MOL	Mutual Info.	ABC	0.6365	10
CNN	**MOL**	**ANOVA**	**EPSO**	**0.8821**	**10**
CNN	MOL	Distance	EPSO	0.6611	10
CNN	MOL	Correlation	EPSO	0.6344	10
CNN	MOL	Fisher	EPSO	0.6525	10
CNN	MOL	Gain Ratio	EPSO	0.6653	10
CNN	MOL	Mutual Info.	EPSO	0.8133	10
CNN	MOL	ANOVA	PSO	0.8745	10
CNN	MOL	Distance	PSO	0.8167	10
CNN	MOL	Correlation	PSO	0.8338	10
CNN	MOL	Fisher	PSO	0.7238	10
CNN	**MOL**	**Gain Ratio**	**PSO**	**0.9072**	**10**
CNN	MOL	Mutual Info.	PSO	0.8247	10
XGB [5]	-	-	-	0.7110	-
LGBM [5]	-	-	-	0.7000	-
CatBoost [5]	-	-	-	0.6660	-
Deep Neural DT [5]	-	-	-	0.6510	-
TabNet [5]	**-**	**-**	**-**	**0.9920**	-

The generated images corresponding to the Poker Hand dataset are available for observation in Figure 6.13 on the facing page. Once again, it's noticeable that the patterns vary in every pixel of the image, however, this is a multiclass problem, where the patterns can appear for distinct classes, potentially posing a challenge for the classification process. Despite this mixing, it's noteworthy that certain patterns consistently appear in samples belonging to outcome 1. This observation holds the promise of enhancing predictions for this specific outcome, given the regular presence of these distinctive patterns among the samples.


(a) Outcome 0

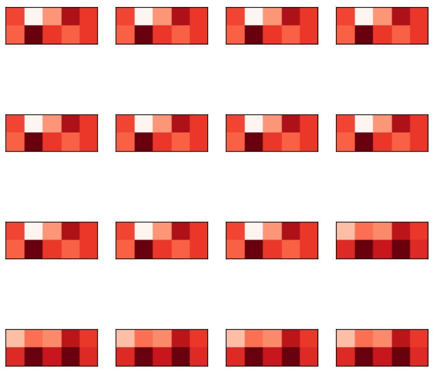

(b) Outcome 1

Figure 6.13: Input generated by MOL-GainRatio-PSO for the Poker Hand dataset.

6.3 Results Remarks

This section will be presented into three distinct subsections:

1. **Benchmark Comparison Overview**: This section will offer an encompassing view of the benchmark comparison, highlighting key insights and findings;
2. **Remarks on Real-World Scenarios**: The second section will dive into comments concerning real-world scenarios, providing a context-specific analysis of the results;
3. **Cardiac Pathology Scenario**: The third section will provide a comprehensive analysis of the results obtained from the Cardiac Pathology dataset, with a focus on their real-world interpretability;
4. **Exploration of MOL Strategies**: The final section will center around MOL strategies, denoting into their various aspects and their implications on the experimental outcomes.

6.3.1 Benchmark Overview

To facilitate the benchmark result comparison, Table 6.14 on the next page provides a ranked overview of each algorithm and combination for every dataset. The rank score is determined by averaging the positions achieved by each strategy across all Benchmark experiments. These positions are derived from the scores attained by all strategies for each dataset in the test set. Subsequently, the final ranking is established by sorting the obtained average positions, offering a consolidated average perspective on the performance of each strategy across datasets. It is worth mentioning that the table provides comparison only for the benchmark problems.

Table 6.14: Overall rank obtained by each strategy on the Benchmark experiment. The table contains: the Algorithms used in the Benchmark experiment; the Feature Selection strategy (FS Str); the Map and Optimization strategies used by MOL (Map Str and Opt Str); The amount of times the algorithm was, respectively, the best, second best and third best to solve a problem; The average position obtained among all the algorithms; and the final Rank obtained

Algorithm	**FS Str**	**Map Str**	**Opt Str**	**#1**	**#2**	**#3**	**Avg. Position**	**Rank**
CNN	MOL	ANOVA	ABC	0	0	0	17.9 ± 6.71	15
CNN	MOL	Correlation	ABC	0	0	0	22.3 ± 8.28	27
CNN	MOL	Distance	ABC	2	1	0	13.0 ± 9.63	5
CNN	MOL	Fisher	ABC	0	1	1	9.8 ± 7.78	1
CNN	MOL	GainRatio	ABC	0	0	0	11.6 ± 5.54	3
CNN	MOL	MutualInfo	ABC	1	0	1	12.1 ± 7.92	4
CNN	MOL	ANOVA	PSO	0	0	0	16.0 ± 7.04	10
CNN	MOL	Correlation	PSO	1	0	0	18.0 ± 9.43	16
CNN	MOL	Distance	PSO	0	0	1	18.9 ± 11.57	17
CNN	MOL	Fisher	PSO	0	0	0	19.6 ± 7.62	21
CNN	MOL	GainRatio	PSO	0	0	0	17.6 ± 6.22	13
CNN	MOL	MutualInfo	PSO	0	0	0	19.0 ± 6.08	18
CNN	MOL	ANOVA	EPSO	0	0	0	21.2 ± 8.66	24
CNN	MOL	Correlation	EPSO	0	0	0	24.8 ± 10.61	34
CNN	MOL	Distance	EPSO	0	0	0	17.8 ± 8.96	14
CNN	MOL	Fisher	EPSO	0	0	0	20.5 ± 8.77	23
CNN	MOL	GainRatio	EPSO	0	0	0	19.1 ± 9.75	19
CNN	MOL	MutualInfo	EPSO	0	0	0	22.3 ± 11.28	28
KNN	E	-	-	0	0	1	22.5 ± 12.51	29
LGBM	E	-	-	1	0	0	14.0 ± 10.35	6
LR	E	-	-	0	0	0	23.3 ± 9.13	33
ANN	E	-	-	0	0	0	23.0 ± 7.38	31
RF	E	-	-	0	1	2	14.5 ± 11.67	7
XGB	E	-	-	0	3	0	20.4 ± 12.94	22
KNN	W	-	-	0	0	0	26.2 ± 10.11	35
LGBM	W	-	-	0	0	1	16.8 ± 10.54	11
LR	W	-	-	0	0	0	23.1 ± 9.55	32
ANN	W	-	-	0	1	0	21.3 ± 10.9	25
RF	W	-	-	0	1	0	15.5 ± 8.57	9
XGB	W	-	-	0	0	0	19.1 ± 9.65	20
KNN	B	-	-	0	0	0	28.1 ± 10.09	37
LGBM	B	-	-	1	1	0	17.3 ± 11.58	12
LR	B	-	-	0	0	0	26.5 ± 8.75	36
ANN	B	-	-	0	0	1	22.8 ± 10.93	30
RF	B	-	-	1	0	1	15.0 ± 12.49	8
XGB	B	-	-	0	1	0	21.8 ± 11.15	26
TabNet	-	-	-	3	0	1	10.3 ± 11.64	2

In the benchmark experiment outlined in this thesis, the algorithms that consistently exhibited the highest average performance were:

1. MOL-Fisher-ABC;
2. TabNet;
3. MOL-GainRatio-ABC.

On the other hand, the algorithm that consistently outperformed others across multiple datasets was TabNet, with XGB-E delivering the second-best performance and RF-E ranking third in terms of performance.

When comparing the top three average performers, TabNet consistently ranked in four distinct problems, showcasing its robust performance. In contrast, MOL-Fisher-ABC claimed the top positions in two problems, while MOL-GainRatio-ABC, despite its competitive average performance, did not secure a top-three ranking in any problem. It's noteworthy that the worst positions for these algorithms were 28th, 37th, and 19th for different datasets, indicating occasional performance dips. This information, coupled with the standard deviation analysis, highlights the greater consistency of MOL performance based on average outcomes. In contrast, the other algorithms delivered commendable results across most datasets but occasionally faced challenges in specific scenarios.

When comparing the results achieved by various MOL strategies, it becomes evident that the selection of algorithms for inclusion in the strategy is not a straightforward process. For instance, when employing the ABC algorithm, the Fisher map yielded the best results, yet this wasn't the case when the PSO or EPSO algorithms were utilized. On average, ABC demonstrated superior performance compared to PSO and EPSO, which struggled to consistently produce optimal outcomes. The performance disparity of these two algorithms might be attributed to the fact that PSO makes modifications to the entire candidate solution, whereas ABC alters only one part of the solution in each iteration. To translate this into the context of the specific optimization problem encountered in MOL, it suggests that PSO variations attempt to simultaneously change the image format and the selected features, while ABC focuses on modifying a single dimension of the image or adding/removing a feature from the selected subset.

Upon closer examination of the performance of MOL with the ABC algorithm, it is apparent that the two top-performing combinations delivered great results across several datasets, including Cancer, Card, Gene, Soybean, Thyroid, and KDD. While other strategies also demonstrated strong performance in specific instances, this particular combination excelled when the datasets contained multiple features encapsulated within the same attribute. For example, the Cancer dataset includes 9 continuous features, and the Diabetes dataset contains 8 continuous features. It is worth noting that when the total number of features in a dataset is an odd number, MOL will inevitably eliminate at least one feature to create a squared image, as dictated by the constraints of the optimization problem. This could potentially pose a challenge if the excluded feature is crucial for the problem, and the selected map strategy fails to distinguish its significance.

In the case of the Glass dataset, it's worth noting that, apart from TabNet, all other algorithms struggled to deliver satisfactory results. This dataset comprises 214 instances, with 33% allocated to the test set, leaving just 164 instances for training. Additionally, the dataset features 9 attributes and encompasses 6 distinct outcomes. Machine learning algorithms trained with these characteristics often face challenges with overfitting, resulting in poor test set performance. On the other hand, deep learning models, such as CNN, typically require a more substantial dataset to perform well. In scenarios with a limited number of instances, pre-trained networks are commonly employed. However, it's important to highlight that pre-trained networks come with specific image size requirements, which the Glass dataset fails to meet.

6.3.2 Real-World Overview

The objective of the experiment was twofold: firstly, to facilitate a comparison against results found in existing literature; and secondly, to address challenges inherent to real-world data scenarios. The experiment offered a unique perspective on MOL strategies, with the combinations involving the ABC algorithm being notably surpassed by those involving the PSO and EPSO algorithms. This outcome contrasts with the direction taken in the benchmark experiment.

To be more specific, the outcomes from the cardiac pathology dataset underscore the strength of the MOL-Distance-EPSO combination. This positive trend extends to the Forest Cover Type dataset as well. However, interestingly, the same MOL-Distance-EPSO combination encounters challenges when dealing with the complexities of the Poker Hand dataset. A parallel behavior is observed with MOL-MI-EPSO, which showcases great overall performance for the first two datasets but encounters difficulties in handling the poker hand dataset.

Within the realm of the poker hand dataset, the MOL-ANOVA-PSO combination emerges as the most effective, managing to enhance the results for the Deep Neural Decision Tree and other state-of-the-art algorithms. Notably, these improvements only fall short of the performance achieved by TabNet.

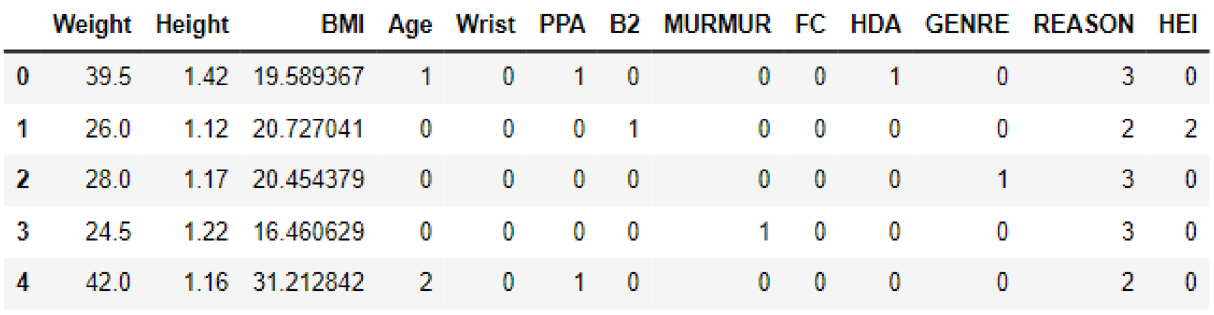
The observations indicates that no MOL strategies were able to maintain outcome quality across all the real-world experiments. MOL achieved the best result for the Cardiac Pathology and TabNet achieved for the remainder datasets, however, both MOL and TabNet were able to maintain its performance across the experiment and outperform other commonly used strategies.

6.3.3 Cardiac Pathology Scenario

As previously discussed in the preceding chapter, the Cardiac Pathology dataset was compiled from a hospital specializing in heart conditions among teenagers and children. This dataset encompasses 13 distinct features, which are as follows: Weight; Height; Body-Mass-Index (BMI);

Age; Wrist state (WS); Blood Pressure (PPA); Second heart sound (B2); Heart murmur (HM); Cardiac frequency (CF); Disease History (HDA); Gender; Visit Reason (VR); and History Emergency Level (HEL).

In this section, we will delve into the results derived from the combination of MOL organized according to the Correlation strategy and optimized using the PSO. This particular combination demonstrated the highest performance in the dataset, selecting a total of 12 features. The initial configuration of the dataset, without the target variable, can be seen in Table 6.14.



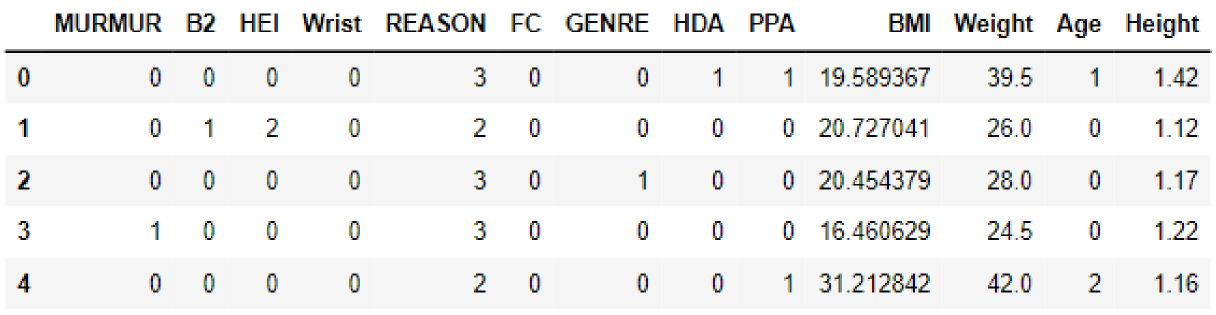
	Weight	Height	BMI	Age	Wrist	PPA	B2	MURMUR	FC	HDA	GENRE	REASON	HEI
0	39.5	1.42	19.589367	1	0	1	0	0	0	1	0	3	0
1	26.0	1.12	20.727041	0	0	0	1	0	0	0	0	2	2
2	28.0	1.17	20.454379	0	0	0	0	0	0	0	1	3	0
3	24.5	1.22	16.460629	0	0	0	0	1	0	0	0	3	0
4	42.0	1.16	31.212842	2	0	1	0	0	0	0	0	2	0

Figure 6.14: Initial version of the Cardiac Pathology dataset

The dataset is partitioned into two subsets: Train and Test. The Map process employs the Correlation strategy, where each feature is assessed based on its correlation with the target variable, resulting in the following initial list of features:

$$7 \quad 6 \quad 12 \quad 4 \quad 11 \quad 8 \quad 10 \quad 9 \quad 5 \quad 2 \quad 0 \quad 3 \quad 1$$

This list suggests that feature number 7, which represents the HM feature, should be the first feature, followed by B2, HEI, and subsequent features in accordance with their relevance. The final outcome of the Map phase presents a dataset that can be visualized in Table 6.15.



	MURMUR	B2	HEI	Wrist	REASON	FC	GENRE	HDA	PPA	BMI	Weight	Age	Height
0	0	0	0	0	3	0	0	1	1	19.589367	39.5	1	1.42
1	0	1	2	0	2	0	0	0	0	20.727041	26.0	0	1.12
2	0	0	0	0	3	0	1	0	0	20.454379	28.0	0	1.17
3	1	0	0	0	3	0	0	0	0	16.460629	24.5	0	1.22
4	0	0	0	0	2	0	0	0	1	31.212842	42.0	2	1.16

Figure 6.15: Sorted dataset based on the Correlation map

The sorted dataset is then submitted to the Optimization phase, where the PSO algorithm found the following solution:

$$4 \quad 3 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0$$

The provided solution complies with the constraints of the objective function, indicating that the image shape is 4 × 3. Furthermore, it points out the exclusion of the last feature, Height, from the dataset. This modified dataset, excluding the Height feature, serves as the foundation for image generation.

In the dataset, the BMI and Weight features are the only attributes represented by floating-point values, whereas the other variables are either binary or integer features scaled from 0 to a maximum of 7. However, when generating images using the actual feature values, these visualizations may not distinctly reveal patterns. For clarity and better visualization, the presented images in Figure 6.16 and Figure 6.17 on the next page have been scaled between 0 and 1 and represent, respectively, outcome 0 and outcome 1.

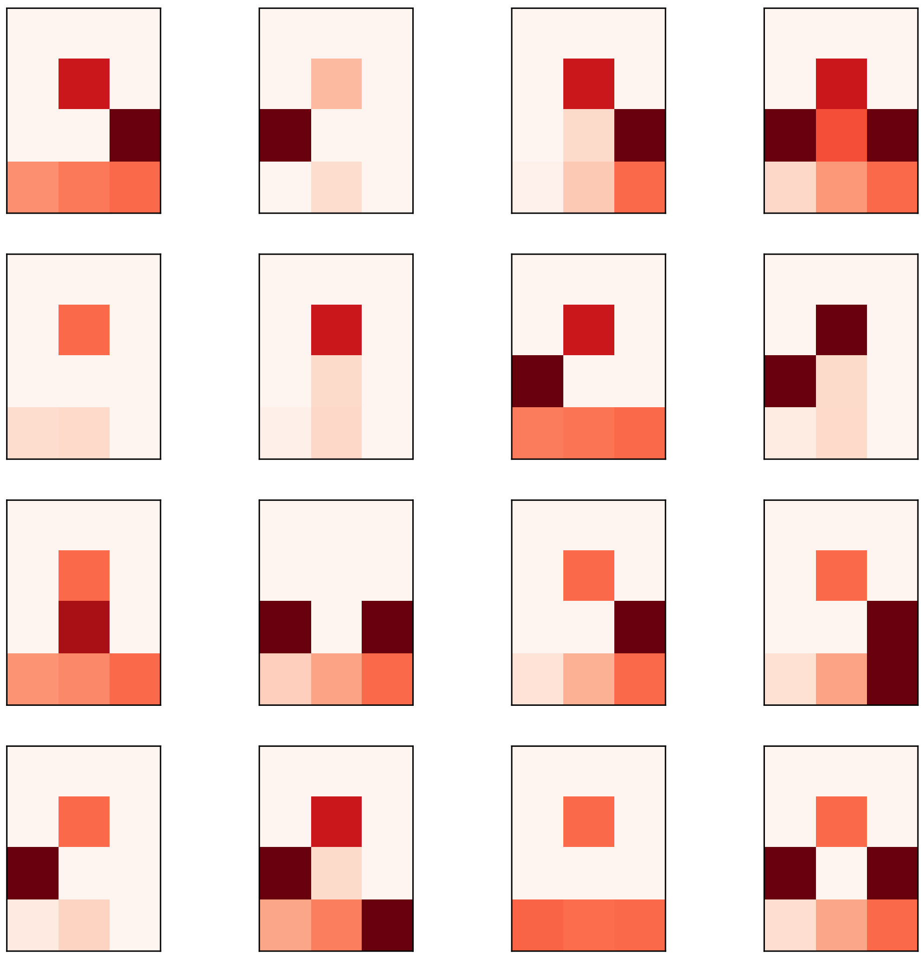


Figure 6.16: Test set samples of images for Outcome 0

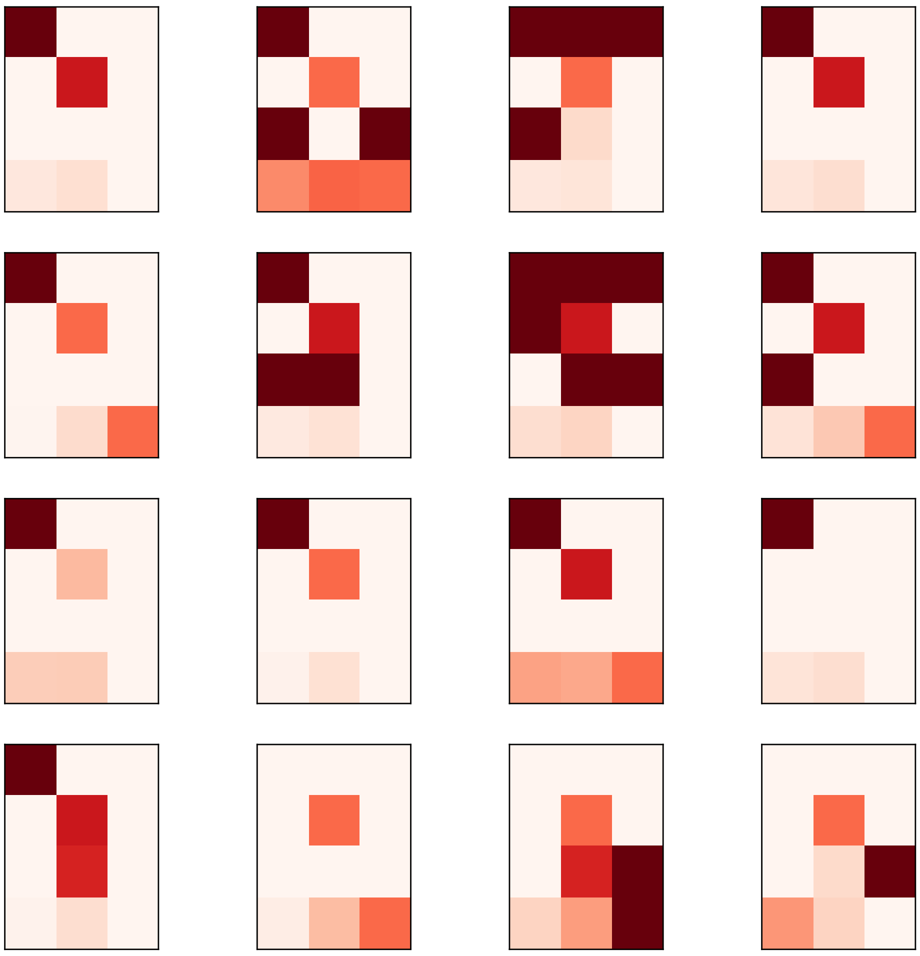


Figure 6.17: Test set samples of images for Outcome 1

It is worth to mention that this particular set of images is extracted from the test set, whereas the previous images, depicted in Figure 6.11 on page 84, originate from the training set of the dataset.

In the clinical domain related to cardiac pathology, the Body Mass Index (BMI) is generally not considered a significant indicator of pathology, whereas the presence of the second heart sound and murmur are recognized as highly relevant features Ferreira et al. [103]. The feature selection process undertaken by the combination highlights murmur, the second heart sound, and the history of emergency level as the top three most relevant features, based on the correlation criteria with the pathology.

The emergency level is an interesting inclusion in this context, as it is an engineered feature and have a significant correlation with both the visiting reason and the second heart sound concerning the target variable. However, it is worth noting that the visiting reason is ranked as the 5th most important feature.

Upon visual examination of the generated images, a discernible pattern can be noticed: in the first row, representing these three important variables, at least one of the three pixels exhibits a darker color. Conversely, the presented cases without pathology do not display dark pixels in the first row. However, an observation arises when analyzing the last three images in Outcome 1, which, like the images in Outcome 0, lack darker pixels in the first row. This phenomenon may indicate a false positive case upon visual inspection.

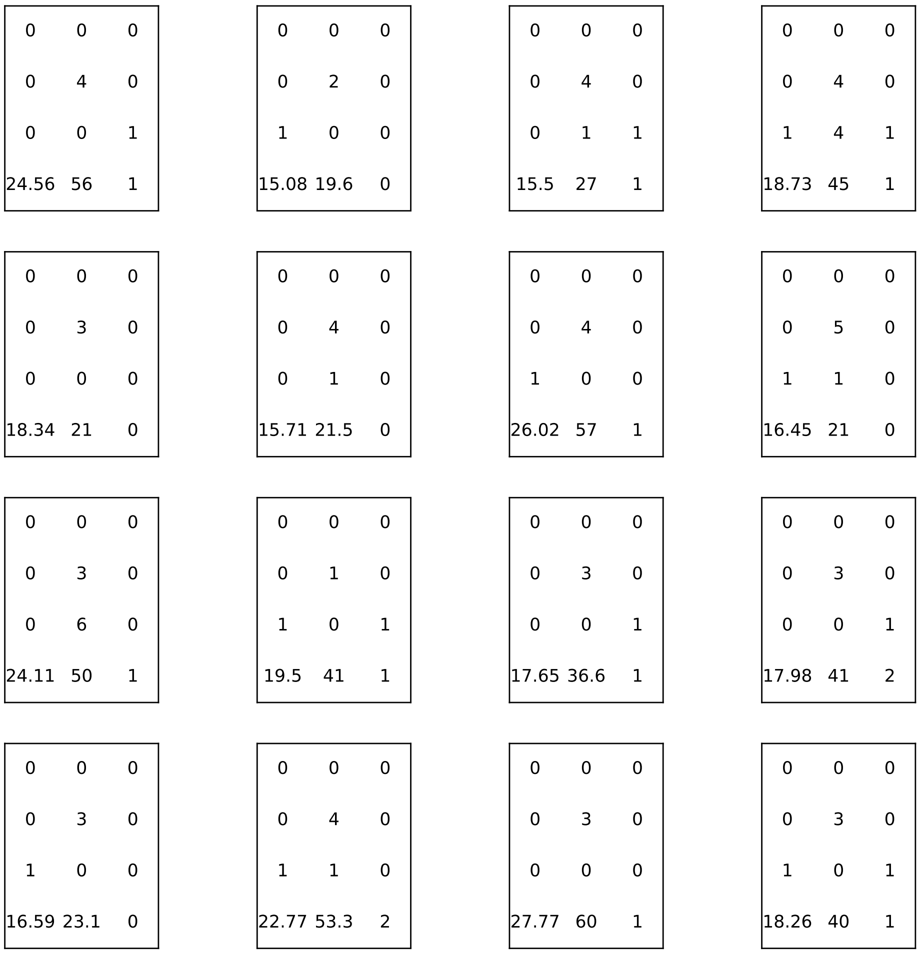
To delve deeper into the comparison of these image subsets, Figure 6.16 on page 94 and Figure 6.17 on the preceding page present, respectively, the original numerical values of each feature without minimum-maximum scaling.

Upon closely examining the numerical values found in the last three images previously discussed, it's notable that each case corresponds to a patient from a distinct age group, as follows: 0 signifies early childhood (birth to 5 years); 1 designates middle childhood (6 to 12 years); and 2 represents teenagers. Interestingly, for each of these cases, the BMI falls below the normal values, as per the official BMI standards in Brazil for young female children. It is important to highlight that these specific images do not indicate the presence of a murmur but rather suggest the presence of the pathology. In addition to this analysis, when the trained model is reapplied to predict these cases, the convolutional neural network (CNN) correctly classifies 13 out of 16 cases in Outcome 1, with misclassifications occurring specifically in the last three mentioned images.

To address this issue, a more refined parameter tuning approach for the CNN model could be employed. Adjustments to filters and the type of pooling layer may yield different outcomes and merit further exploration. However, it is noteworthy that the combination algorithm successfully captured the essence of the problem and its relevant information before creating the images and feeding them into the model. In a hospital environment, these images could be presented alongside the numerical values. This integration could facilitate a closer inspection by a specialist, enhancing diagnostic quality. Specialists could analyze the obtained values more comprehensively and leverage the images to gain a different perspective, ultimately contributing to improved diagnostic

6.3.4 MOL Results Overview

The MOL strategy demonstrated a performance that fluctuated across various experiments, at times emerging as the most effective approach and at others, showing potential limitations depending on the specific scenario. Nevertheless, there remain several areas for refinement that can further enhance its overall performance. Summing up the acquired results, key points surface



0	0	0
0	4	0
0	0	1
24.56	56	1

0	0	0
0	2	0
1	0	0
15.08	19.6	0

0	0	0
0	4	0
0	1	1
15.5	27	1

0	0	0
0	4	0
1	4	1
18.73	45	1

0	0	0
0	3	0
0	0	0
18.34	21	0

0	0	0
0	4	0
0	1	0
15.71	21.5	0

0	0	0
0	4	0
1	0	0
26.02	57	1

0	0	0
0	5	0
1	1	0
16.45	21	0

0	0	0
0	3	0
0	6	0
24.11	50	1

0	0	0
0	1	0
1	0	1
19.5	41	1

0	0	0
0	3	0
0	0	1
17.65	36.6	1

0	0	0
0	3	0
0	0	1
17.98	41	2

0	0	0
0	3	0
1	0	0
16.59	23.1	0

0	0	0
0	4	0
1	1	0
22.77	53.3	2

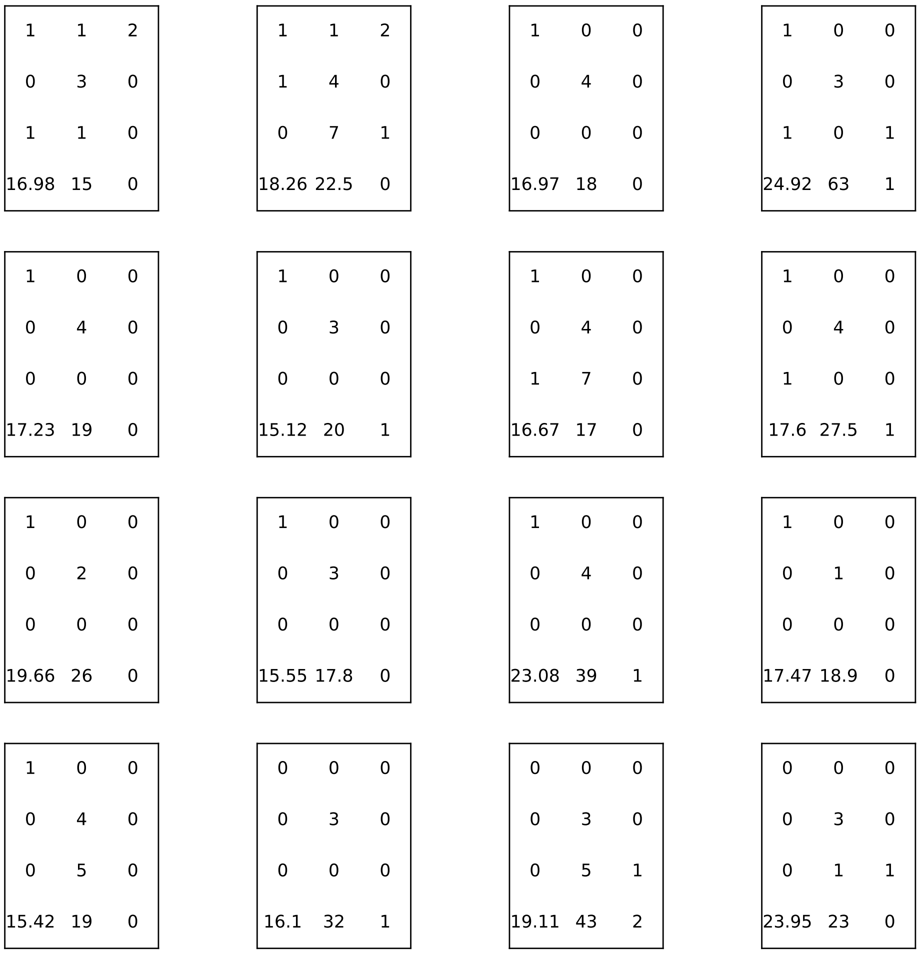
0	0	0
0	3	0
0	0	0
27.77	60	1

0	0	0
0	3	0
1	0	1
18.26	40	1

Figure 6.18: Numerical values of the test set samples for Outcome 0

that have proven advantageous for the MOL strategy:

- **Images**: visualizing the generated images can provide valuable insights into diverse issues. For instance, in the Cardiac Pathology experiment, these images could potentially offer a preliminary diagnostic indication of the pathology, prompting experts in the field to make necessary and informed decisions.
- **Attributes**: datasets where the majority of the features are within the same attribute is arguably the best scenario for MOL.
- **Learning Process**: in certain scenarios, MOL exhibited a performance surpassing that



1	1	2
0	3	0
1	1	0
16.98	15	0

1	1	2
1	4	0
0	7	1
18.26	22.5	0

1	0	0
0	4	0
0	0	0
16.97	18	0

1	0	0
0	3	0
1	0	1
24.92	63	1

1	0	0
0	4	0
0	0	0
17.23	19	0

1	0	0
0	3	0
0	0	0
15.12	20	1

1	0	0
0	4	0
1	7	0
16.67	17	0

1	0	0
0	4	0
1	0	0
17.6	27.5	1

1	0	0
0	2	0
0	0	0
19.66	26	0

1	0	0
0	3	0
0	0	0
15.55	17.8	0

1	0	0
0	4	0
0	0	0
23.08	39	1

1	0	0
0	1	0
0	0	0
17.47	18.9	0

1	0	0
0	4	0
0	5	0
15.42	19	0

0	0	0
0	3	0
0	0	0
16.1	32	1

0	0	0
0	3	0
0	5	1
19.11	43	2

0	0	0
0	3	0
0	1	1
23.95	23	0

Figure 6.19: Numerical values of the test samples for Outcome 1

of other frequently employed algorithms. This is attributed to the exceptional pattern recognition capabilities of CNN, enabling it to identify patterns that posed challenges for other algorithms, where part of these patterns were created by the Map and Optimize phases.

It is evident that during the Map and Optimization process of MOL, the image creation process often generates patterns that are recurrent across multiple classes. For instance, images generated for Outcome 0 may closely resemble those created for Outcome 1 and subsequent classes. The extent of similarity among these images can vary, depending on the dataset. However, in scenarios where the similarity between images across different target outcomes is pronounced, it can pose challenges for the classification process.

Upon a comprehensive analysis of the entire experiment, a few areas become apparent where room for improvement exists:

- **Pipeline**: choosing the components to comprise the entire pipeline can be a complex task, primarily due to the multiple techniques available for the Map and Optimize methods.
- **Map**: the experiments clearly demonstrate that a well-suited selection of the Map method can enhance the pipeline's performance. However, the process of correctly choosing the strategy is not straightforward, leaving room for an automated selection technique. This auto-selection approach could intelligently pick the Map method based on the characteristics of the data, optimizing the overall process.
- **Optimize**: similar to the Map phase, the optimization phase presents its own complexity in the quest to identify the most suitable algorithm to create the best combination with the input from the Map phase. Introducing a more intricate solution involving multiple algorithms that generate numerous outputs and conducting various tests with them could simplify the strategy selection process. However, with regard to the optimization process, there is room for discussion for potential modifications in the formulation. Modifying this process could result in inputs that are distinct across the classes, potentially opening the door for the application of other algorithms, such as gradient-based approaches. This, in fact, might offer more straightforward choices for feature selection strategies.
- **Learn**: variations of the CNN can be used to achieved an improved outcome. Not limited to new algorithms, the parameter tuning could be inserted into a self-adaptive scheme where based on previous experiments, a certain range of the parameters of the CNN could be used.

Chapter 7

Conclusion

This work presented Map-Optimize-Learn (MOL), a novel strategy focused on modeling data represented tabular datasets. It involved the utilization of various concepts and algorithms across three distinct phases: Map, Optimize and Learn. In the initial Map phase, the objective was to detect the relevance of each feature in the dataset. Subsequently, the insights obtained from this phase, regarding feature significance, were employed in the Optimize phase. In the second phase, multiple Swarm Intelligence (SI) algorithms were used to generate a set of images that maximized similarity based on the importance derived from the previous Map phase. Combining the information given in Map with the results obtained in Optimize, the last phase used the generated images to learn possible patterns.

The experiment conducted a comparative analysis between the MOL strategy and state-of-the-art Machine Learning (ML) algorithms. This study involved assessing the baseline algorithm both independently and in combination with various Feature Selection (FS) strategies to discern whether such combinations enhanced or impeded its performance. The evaluation encompassed three real-world problems and ten benchmark problems derived from existing literature, encompassing datasets with significant variations in terms of instance count, classes, number of features, and feature attributes.

The experiment identified a specific combination of MOL with the ABC algorithm that consistently outperformed others. Conversely, various combinations of MOL with different Particle Swarm Optimization (PSO) variants generally yielded less impressive results compared to alternative algorithms and strategies. However, in real-world scenarios, underperforming combinations displayed notable improvements, demonstrating the adaptability of MOL phases in addressing diverse challenges. Furthermore, aside from numerical outcomes, the input generated for the Convolutional Neural Network (CNN) during the Learn phase revealed discernible patterns, potentially aiding in the identification of relevant information in scenarios with a limited number of features.

Despite the performance, an important observation arose regarding the strategy's behavior. It was noted that the same combination might not consistently perform uniformly across various problems. This issue highlights a few challenges:

- Finding the correct combination in MOL might require significant time in real-world implementations.
- Compared to other algorithms that might need only a few attempts and parameter tuning to achieve the same objective, MOL typically requires several attempts to find the best possible combination for a specific problem.
- The complexity and processing time of MOL are amplified due to its multiple phases, each incorporating several algorithms.
- Identifying the optimal set of parameters that best fits all methods for every conceivable scenario is another challenging task.

Although the Map phase tends to operate swiftly in comparison to other stages, the Optimize phase often demands significant processing time due to the involvement of SI algorithms. Even in parallel programming environments, implementing these algorithms and the Optimize objective function remains a complex and detailed task, thereby amplifying the time required for real-world implementations.

Considering the drawbacks, there are a few aspects that can be highlighted to enhance the strategy's quality and diminish the time needed to build and process the pipeline:

- Enhancing the strategy by incorporating self-adaptive capabilities to automatically select the most suitable Map method for a given dataset and tailor the parameters for the Optimize algorithm.
- Revamping the Optimize objective function to enable the utilization of gradient-based algorithms, thereby reducing reliance on searching for an optimal solution using SI algorithms.
- Fine-tuning the CNN model utilized in the Learn phase by adapting its parameters and enabling other models to perform predictions on the generated images.

Bibliography

[1] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[2] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, San Francisco, CA, USA:, 2015.

[3] Mário Serra Neto, Marco Mollinetti, and Inês Dutra. Data domain change and feature selection to predict cardiac pathology with a 2d clinical dataset and convolutional neural networks (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(18):15883–15884, May 2021. doi:10.1609/aaai.v35i18.17938.

[4] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[5] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021. doi:10.1609/aaai.v35i8.16826.

[6] L. Katzir, G. Elidan, and Ran El-Yaniv. Net-DNF: Effective Deep Modeling of Tabular Data. In *ICLR*, 2021.

[7] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943. Curran Associates, Inc., 2021.

[8] Mario TR Serra Neto, Inês Dutra, and Marco Antonio F Mollinetti. Map-optimize-learn: Predicting cardiac pathology in children and teenagers with a deep learning based tabular learning method. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[9] Tom M Mitchell. *Machine learning*. Boston, MA: McGraw-Hill, 1997.

[10] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. In Yagang Zhang, editor, *New Advances in Machine Learning*, chapter 3. IntechOpen, Rijeka, 2010. doi:10.5772/9385.

[13] Saed Sayad. Data preparation. https://www.saedsayad.com/data_preparation.htm, 2010. Accessed: 09-08-20.

[14] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[15] John H Laub and Robert J Sampson. Integrating quantitative and qualitative data. *Methods of life course research: Qualitative and quantitative approaches*, pages 213–230, 1998.

[16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[17] Pau Rodríguez, Miguel A Bautista, Jordi Gonzalez, and Sergio Escalera. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 75:21–31, 2018.

[18] Ivan Nunes Da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, Silas Franco dos Reis Alves, Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. *Artificial neural network architectures and training processes*. Springer, 2017.

[19] Vjodor van Veen and Stefan Leijnen. The neural network zoo. https://www.asimovinstitute.org/neural-network-zoo/, 2019. Accessed: 09-12-20.

[20] P Sibi, S Allwyn Jones, and P Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, 47 (3):1264–1268, 2013.

[21] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN: 9781605589077.

[22] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[23] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[24] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, 1(4):580–585, 1985.

[25] Li-Yu Hu, Min-Wei Huang, Shih-Wen Ke, and Chih-Fong Tsai. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1):1304, 2016.

[26] William J Egan and Stephen L Morgan. Outlier detection in multivariate analytical chemical data. *Analytical chemistry*, 70(11):2372–2379, 1998.

[27] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[28] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[30] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R Sveinsson. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294–300, 2006.

[31] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[32] Yongheng Zhao and Yanxia Zhang. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, 41(12):1955–1959, 2008.

[33] Dave Raggett. Decision tree demo for chunk rules. https://www.w3.org/Data/demos/chunks/decision-tree/, 2020. Accessed: 2019-11-26.

[34] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[35] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[36] Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

[37] George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.

[38] Nenkat Venkatraman. The concept of fit in strategy research: Toward verbal and statistical correspondence. *Academy of management review*, 14(3):423–444, 1989.

[39] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[40] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2): 1153–1176, 2015.

[41] Aida Ali, Siti Mariyam Shamsuddin, Anca L Ralescu, et al. Classification with class imbalance problem: a review. *Int. J. Advance Soft Compu. Appl*, 7(3):176–204, 2015.

[42] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[43] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[45] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017.

[46] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[47] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[48] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization.* John Wiley & Sons, 2004.

[49] Marco Dorigo, Mauro Birattari, et al. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007.

[50] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

[51] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[52] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[53] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[54] Yang Wei and Li Qiqiang. Survey on particle swarm optimization algorithm [j]. *Engineering Science*, 5(5):87–94, 2004.

[55] David Martens, Bart Baesens, and Tom Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1–42, 2011.

[56] Valery Tereshko and Andreas Loengarov. Collective decision making in honey-bee foraging dynamics. *Computing and Information Systems*, 9(3):1, 2005.

[57] Dervis Karaboga and Bahriye Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, 8(1):687–697, 2008.

[58] Vladimiro Miranda and Nuno Fonseca. Epso-evolutionary particle swarm optimization, a new algorithm with applications in power systems. In *IEEE/PES Transmission and Distribution Conference and Exhibition*, volume 2, pages 745–750. IEEE, 2002.

[59] Jian-Chao Zeng and Zhi-Hua Cui. A guaranteed global convergence particle swarm optimizer. *Journal of computer research and development*, 8:1333–1338, 2004.

[60] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *European Conference on Machine Learning*, pages 145–156. Springer, 2001.

[61] Jason Brownlee. *Deep learning for natural language processing: develop deep learning models for your natural language problems*. Machine Learning Mastery, 2017.

[62] Hadley Wickham et al. Reshaping data with the reshape package. *Journal of statistical software*, 21(12):1–20, 2007.

[63] Boehmke Bradley. Reshaping your data with tidyr. https://uc-r.github.io/tidyr#separate, 2016. Accessed: 17-02-21.

[64] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, 2021.

[65] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[66] G. Somepalli, Micah Goldblum, Avi Schwarzschild, C. B. Bruss, and T. Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *ArXiv*, abs/2106.01342, 2021.

[67] Jannik Kossen, Neil Band, Clare Lyle, Aidan N. Gomez, Tom Rainforth, and Y. Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *ArXiv*, abs/2106.02584, 2021.

[68] Arlind Kadra, M. Lindauer, F. Hutter, and Josif Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. *ArXiv*, abs/2106.11189, 2021.

[69] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee, 2015.

[70] Soner Yigit and Mehmet Mendes. Which effect size measure is appropriate for one-way and two-way anova models? a monte carlo simulation study. *Revstat Statistical Journal*, 16:295–313, 2018.

[71] Samrat Kumar Dey and Md Mahbubur Rahman. Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method. In *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, pages 630–635. IEEE, 2018.

[72] Andrea Bommert, Xudong Sun, Bernd Bischl, Jörg Rahnenführer, and Michel Lang. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis*, 143:106839, 2020.

[73] J Ramírez, JM Górriz, A Ortiz, FJ Martínez-Murcia, F Segovia, D Salas-Gonzalez, D Castillo-Barnes, IA Illán, CG Puntonet, Alzheimer's Disease Neuroimaging Initiative, et al. Ensemble of random forests one vs. rest classifiers for mci and ad prediction using anova cortical and subcortical feature selection and partial least squares. *Journal of neuroscience methods*, 302:47–57, 2018.

[74] Mário Tasso Ribeiro Serra Neto. Feature selection with the ms-epso algorithm to predict cardiac pathology in children and teenagers. *M.Sc. Thesis University of Porto Open Repository*, 2020.

[75] N Gopika and Meena Kowshalaya A. Correlation based feature selection algorithm for machine learning. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 692–695. IEEE, 2018.

[76] Syed Muhammad Saqlain, Muhammad Sher, Faiz Ali Shah, Imran Khan, Muhammad Usman Ashraf, Muhammad Awais, and Anwar Ghani. Fisher score and matthews correlation coefficient-based feature subset selection for heart disease diagnosis using support vector machines. *Knowledge and Information Systems*, 58(1):139–167, 2019.

[77] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[78] Chengzhang Li and Jiucheng Xu. Feature selection with the fisher score followed by the maximal clique centrality algorithm can accurately identify the hub genes of hepatocellular carcinoma. *Scientific reports*, 9, 2019.

[79] Mohammad Amin Valizade Hasanloei, Razieh Sheikhpour, Mehdi Agha Sarram, Elnaz Sheikhpour, and Hamdollah Sharifi. A combined fisher and laplacian score for feature selection in qsar based drug design using compounds with known and unknown activities. *Journal of Computer-Aided Molecular Design*, 32(2):375–384, 2018.

[80] Doğukan Aksu, Serpil Üstebay, Muhammed Ali Aydin, and Tülin Atmaca. Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In *International Symposium on Computer and Information Sciences*, pages 141–149. Springer, 2018.

[81] Thee Zin Win and Nang Saing Moon Kham. International conference on computer applications. In *International Symposium on Computer and Information Sciences*. Seventeenth International Conference on Computer Applications (ICCA 2019), 2019.

[82] Victor F Rodriguez-Galiano, Juan Antonio Luque-Espinar, M Chica-Olmo, and María Paula Mendes. Feature selection approaches for predictive modelling of groundwater nitrate pollution: An evaluation of filters, embedded and wrapper methods. *Science of the total environment*, 624:661–672, 2018.

[83] Wanfu Gao, Liang Hu, and Ping Zhang. Class-specific mutual information variation for feature selection. *Pattern Recognition*, 79:328–339, 2018.

[84] Sadia Sharmin, Mohammad Shoyaib, Amin Ahsan Ali, Muhammad Asif Hossain Khan, and Oksam Chae. Simultaneous feature selection and discretization based on mutual information. *Pattern Recognition*, 91:162–174, 2019.

[85] Igor Kononenko. Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[86] T Raghunadha Reddy, B Vishnu Vardhan, M GopiChand, and K Karunakar. Gender prediction in author profiling using relieff feature selection algorithm. In *Intelligent Engineering Informatics*, pages 169–176. Springer, 2018.

[87] Halebedu Subbaraya Suresha and SS Parthasarathi. Relieff feature selection based alzheimer disease classification using hybrid features and support vector machine in magnetic resonance imaging. *International Journal of Computer Engineering and Technology*, 10(1):124–137, 2019.

[88] Yong Zhang, Xuezhen Ren, and Jie Zhang. Intrusion detection method based on information gain and relieff feature selection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. IEEE, 2019.

[89] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015.

[90] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[91] Kamil Belkhayat Abou Omar. Xgboost and lgbm for porto seguro's kaggle challenge: A comparison. *Preprint Semester Project*, 2018.

[92] F Guillaume Blanchet, Pierre Legendre, and Daniel Borcard. Forward selection of explanatory variables. *Ecology*, 89(9):2623–2632, 2008.

[93] Ewout W Steyerberg, Marinus JC Eijkemans, and J Dik F Habbema. Stepwise selection in small data sets: a simulation study of bias in logistic regression analysis. *Journal of clinical epidemiology*, 52(10):935–942, 1999.

[94] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[95] Bharat Richhariya, Muhammad Tanveer, AH Rashid, Alzheimer's Disease Neuroimaging Initiative, et al. Diagnosis of alzheimer's disease using universum support vector machine based recursive feature elimination (usvm-rfe). *Biomedical Signal Processing and Control*, 59:101903, 2020.

[96] Titin Siswantining, Devvi Sarwinda, Alhadi Bustamam, et al. Rfe and chi-square based feature selection approach for detection of diabetic retinopathy. In *International Joint Conference on Science and Engineering (IJCSE 2020)*, pages 380–386. Atlantis Press, 2020.

[97] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[98] Lin-cheng Zhou, Hui-zhong Yang, and Chun-bo Liu. Qpso-based hyper-parameters selection for ls-svm regression. In *2008 Fourth International Conference on Natural Computation*, volume 2, pages 130–133. IEEE, 2008.

[99] Mário Serra Neto, Marco Mollinetti, Vladimiro Miranda, and Leonel Carvalho. Maximum search limitations: Boosting evolutionary particle swarm optimization exploration. In *EPIA Conference on Artificial Intelligence*, pages 712–723. Springer, 2019.

[100] Lutz Prechelt. Some notes on neural learning algorithm benchmarking. *Neurocomputing*, 9 (3):343–347, 1995.

[101] Salvatore J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144. IEEE, 2000.

[102] Dheeru Dua, Casey Graff, et al. Uci machine learning repository. 2017.

[103] P. Ferreira, T. T. V. Vinhoza, A. Castro, F. Mourato, T. Tavares, S. Mattos, I. Dutra, and M. Coimbra. Knowledge on heart condition of children based on demographic and physiological features. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 314–319, June 2013. doi:10.1109/CBMS.2013.6627808.