Federal University of São Carlos (UFSCAR)
Exact Sciences and Technology Center (CCET)
Computer Science Department (DC)
Computer Science Graduate Program (PPGCC)

Bruna Zamith Santos

Climate Variables Forecasting and Forest Fire Risk Rate Classification in the Brazilian Pantanal

São Carlos
2023

Bruna Zamith Santos

Climate Variables Forecasting and Forest Fire Risk Rate Classification in the Brazilian Pantanal

Dissertation submitted to the Computer Science Graduate Program of the Exact Sciences and Technology Center at the Federal University of São Carlos, as a requirement for obtaining the title of Master in Computer Science.

Field of Study: Machine Learning and Natural Language Processing

Supervisor: Ricardo Cerri
Co-Supervisors: Marcelo Narciso and Balbina Soriano

São Carlos
2023

Abstract

The Pantanal biome is of inestimable ecological importance, mainly due to its fauna and flora. However, this biome is constantly threatened by the occurrence and recurrence of forest fires, which are strongly associated with the regions climatic conditions. Thus, methods for forecasting and classifying forest fire risk are essential for forest fire prevention and firefighting planning in the Brazilian Pantanal region. The main fire risk indexes known in the literature have limitations, such as (1) not adjusting to the characteristics of each biome; (2) being limited to specific climatic variables; (3) not being able to predict forest fire risk for a given number of days. This last aspect, in particular, is of utmost relevance. Addressing it allows for coordinated planning and action by environmental authorities with adequate anticipation.

Aiming to solve this problem, this study developed a software capable of: (1) Climatic variables forecasting for a given number of days; and (2) Forest fire risk classification in the Brazilian Pantanal. For the first objective, different time series prediction algorithms based on Machine Learning (ML) were tested for climactic variables forecasting. This prediction is used as input for the second objective, for which different classification algorithms, also based on ML, were tested. Such software was then improved from a exhaustive hyperparameters search approach to two different Genetic Algorithms (GAs) approaches: Traditional and NSGA-II.

Results for both software versions were evaluated based on the average correlation between forest fire risk classes and hotspots' observation. The exhaustive search version demonstrated that the software can outperform the main statistical forest fire indexes regarding "Null", "Low", "High" and "Very High" classes. When it comes to the GA version, the software was competitive to the forest fire indexes, still with the advantage of being able to predict the forest fire risk for a given number of days.

Keywords: Supervised machine learning; classification; time series forecasting; genetic algorithms; forest fire risk.

Resumo *(Portuguese Abstract)*

O bioma Pantanal tem uma importância ecológica inestimável, principalmente devido à sua fauna e flora. No entanto, este bioma está constantemente ameaçado pela ocorrência e recorrência de incêndios florestais, os quais estão fortemente associados às condições climáticas da região. Deste modo, métodos para previsão e classificação do risco de incêndio florestal mostram-se essenciais na prevenção e no combate aos incêndios na região do Pantanal brasileiro. Os principais índices de risco de incêndio utilizados atualmente possuem limitações, como (1) não se ajustarem às características de cada bioma; (2) serem limitados a variáveis climáticas específicas; e (3) não serem capazes de prever o risco de incêndio florestal para um determinado número de dias. Este último aspecto, em especial, é de suma relevância, uma vez que seu endereçamento permite um planejamento e uma ação coordenada das autoridades ambientais com devida antecedência.

Visando solucionar este problema, este estudo desenvolveu um *software* capaz de: (1) Prever variáveis climáticas para um determinado número de dias; e (2) Classificar risco de incêndio florestal no Pantanal brasileiro. Para o primeiro objetivo, diferentes algoritmos de séries temporais, baseados em Aprendizado de Máquina (AM), foram testados para previsão de variáveis climáticas. A previsão é utilizada como entrada para o segundo objetivo, para o qual foram testados diferentes algoritmos de classificação, também de AM. Tal *software* foi então evoluído de uma abordagem de busca exaustiva por hiperparâmetros para duas diferentes abordagens de Algoritmos Genéticos (AGs): Tradicional e NSGA-II.

Resultados para ambas as versões do *software* foram avaliadas com base na correlação média entre as classes de risco de incêndio florestal e a observação de focos de calor. A versão de busca exaustiva superou os principais índices de risco de incêndio florestais para classes “Nulo”, “Baixo”, “Alto” e “Muito Alto”. Já para a versão com base em AGs, o *software* foi competitivo com os índices de risco de incêndio florestais, e ainda com a vantagem de prever o risco para um determinado número de dias no futuro.

Palavras-chave: Aprendizado de máquina supervisionado; classificação; previsão de séries temporais; algoritmos genéticos; risco de incêndio florestal

List of Figures

List of Tables

List of Acronyms

ADASYN Adaptive Synthetic

AI Artificial Intelligence

ANN Artificial Neural Network

ARIMA Autoregressive Integrated Moving Average

AUC Area Under the ROC Curve

BPTT Backpropagation Through Time

CNN Convolutional Neural Network

DS Data Science

EFB Exclusive Feature Bundling

ENN Edited Nearest Neighbours

EVAP/P Cumulative Evaporation by Precipitation Index

FMA Monte Alegre Formula (from Portuguese, "*Fórmula de Monte Alegre*")

FMA$^+$ Modified Monte Alegre Formula (from Portuguese, "*Fórmula de Monte Alegre Modificada*")

FWI Fire Weather Index

GA Genetic Algorithm

GOSS Gradient-Based One-Side Sampling

GRU Gated Recurrent Unit

INPE National Institute for Aerospace Research (from Portuguese, "*Instituto Nacional de Pesquisas Espaciais*")

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory

MAE Mean Absolute Error

ML Machine Learning

MLP Multi-Layer Perceptron

MSLE Mean Squared Logarithmic Error

MOEA Multi-Objective Evolutionary Algorithm

NSGA-II Nondominated Sorting Genetic Algorithm II

P-EVAP Cumulative Precipitation-Evaporation Index

RNN Recurrent Neural Network

SARIMA Seasonal Autoregressive Integrated Moving Average

SMOTE Synthetic Minority Oversampling Technique

SVM Support Vector Machine

TS Time Series

UNESCO United Nations Educational, Scientific and Cultural Organization

List of Algorithms

Contents

Chapter 1

Introduction

1.1 Contextualization and Motivation

The Pantanal biome is a wetland area covering approximately 150,355 km^2, with the majority of its territory belonging to Brazil but also extending into Bolivia and Paraguay. The ecological importance of the Pantanal is recognized by United Nations Educational, Scientific and Cultural Organization (UNESCO), which designates it as a Biosphere Reserve. Despite still maintaining approximately 80% of its vegetation cover (ALHO et al., 2019), the ecosystems of this biome are constantly threatened by anthropogenic activities as well as the complex seasonal dynamics of its climate.

One of the major threats is the occurrence of forest fires, which is strongly associated with weather conditions. From June to September, there is a period with very little rainfall in the Pantanal. This often results in an accumulation of vegetative material, which serves as fuel for the occurrence and spread of forest fires in the region.

By using meteorological data, it is possible to classify the risk of forest fire occurrence and, thus, contribute to its prevention and control. In (SORIANO; DANIEL; SANTOS, 2015), the efficiencies of five different forest fire risk indexes are compared for the Brazilian Pantanal: Monte Alegre Formula (from Portuguese, "*Fórmula de Monte Alegre*") (FMA), Modified Monte Alegre Formula (from Portuguese, "*Fórmula de Monte Alegre Modificada*") (FMA$^+$), Telicyn, Angström, and Nesterov. Furthermore, in the study by (TORRES; RIBEIRO, 2008), the indexes FMA, Telicyn, Nesterov, Cumulative Precipitation-Evaporation Index (P-EVAP), and Cumulative Evaporation by Precipitation Index (EVAP/P) were applied and compared to each other for forest fire prediction in Juiz de Fora (MG), Brazil. Among all these indexes, one of the most commonly used in Brazil is the FMA index, which was developed based on studies in the Araucária region

(SOARES, 1972), in Brazil. The FMA index is cumulative and its formula includes only two variables: relative humidity and precipitation. It has been applied in different regions of Brazil to classify forest fire risk (TETTO et al., 2010; NUNES et al., 2010; ALVARES et al., 2014), including the Pantanal (SORIANO; DANIEL; SANTOS, 2015; ONIGEMO, 2007).

There is also the SARIPAN system (NARCISO; SORIANO, 2019), which is being used by environmental authorities to combat forest fire in the Brazilian Pantanal region. SARIPAN makes use of FMA, FMA$^+$, Nesterov, Telicyn and Angström indexes to identify forest fire risk.

However, the application of such indexes has limitations, including:

1. Not being adjusted according to the characteristics of each biome;

2. Being limited to specific climatic variables;

3. Not being able to predict the risk of forest fire for a specific number of days.

For the classification of forest fire risk, the use of Machine Learning (ML)-based models can also be employed. These models have the advantage of considering the characteristics of the specific region under study, as well as not having limitations on the quantity or type of variables that can be used to describe the region.

Recent studies have shown that the utilization of algorithms based on Artificial Neural Network (ANN) can yield good results for forest fire detection (LUO et al., 2018; AL-ZEBDA et al., 2021; YANG; LUPASCU; MEEL, 2021; GAO; LIN; HU, 2023), including in the Brazilian Pantanal (VIGANÓ et al., 2017). The study (RAKSHIT et al., 2021) evaluated different ML algorithms for classifying particular areas as highly prone, moderately prone, low prone and no fire prone for forest fire. They evaluated Decision Trees, K-Nearest Neighbors (KNN)s, Support Vector Machine (SVM)s and Naive Bayes classifiers, demonstrating better performance when using Decision Trees. Another recent study, (RUBÍ; CARVALHO; GONDIM, 2023), evaluated different ML models for prediction of both spread and behavior of wildfires in the Brazilian Federal District region (Cerrado biome). The data included as features the climatic variables, satellite data, and topographic, hydrographic, and anthropogenic information. Regarding the wildfire spread prediction, the study found out that AdaBoost model outperforms all others models (such as Random Forest, ANN and SVM) with 91% accuracy.

However, none of these studies have presented a model capable of predicting the risk of forest fire for a specific number of days, which would further benefit planning with sufficient advance notice for coordinated action by the appropriate authorities.

In this regard, it is possible to utilize the knowledge associated with the field of Time Series (TS) forecasting. A TS consists of data related to the observation of a phenomenon over time, such as climatic variables. To forecast TS, statistical methods

based on autocorrelations present in the data can be applied, such as Autoregressive Integrated Moving Average (ARIMA) and Seasonal Autoregressive Integrated Moving Average (SARIMA) models (DIMRI; AHMAD; SHARIF, 2020; MURAT et al., 2018; MUKADI; GONZÁLEZ-GARCÍA, 2021). However, in addition to these, it is also possible to make use of ML-based models, such as Long Short-Term Memory (LSTM) networks (ABBES; MAGAGI; GOITA, 2019). In fact, (LIN et al., 2023) applied LSTM for forest fire prediction in Chongli, China. However, it did not leverage forecasting for more than one day in the future.

So, it is clear that it is possible to leverage ML models for climatic variables forecasting and forest fire risk classification. Nevertheless, given the multitude of algorithms and approaches available, navigating the vast hyperparameter space and identifying the optimal solution can be challenging. In that sense, Genetic Algorithm (GA)s offer a computationally efficient alternative that has the potential to yield superior results. In fact, (GANAPATHY, 2020) proposed using GAs as an automatic tuning method for neural networks. The GA outperformed a random search of hyperparameters in a task of machine translation from Japanese to English. In the study of (ALIBRAHIM; LUDWIG, 2021), the authors compared the use of GAs against Grid Search and Bayesian Optimization for hyperparameter search, applied to a customer transaction prediction dataset. Again, the GA outperformed the other methods - it achieved 0.826 Area Under the ROC Curve (AUC) against 0.792 for Grid Search and 0.789 for Bayesian Optimization.

GA-based models were also proposed in climate and environmental applications. (AMOL, 2020) presented a GA to predict forest fire propagation, based on four parameters: drought factor, temperature, relative humidity and wind. The model was able to identify the ranges of values for input parameters that contribute towards the initiation of fire. Based on the such ranges, the model is accurate in predicting the spread category of the affected area. The study of (MATOS et al., 2022) applies GA to a forest firefighting resource scheduling problem. The goal is to obtain the best ordered sequence of actions to be taken by a firefighting resource in combating forest fire. The data was collected from Braga, in Portugal, and the results demonstrated the usefulness and validity of the proposed approach.

Still, for the best of our knowledge, there is no study that applied Genetic Algorithms for selecting the best hyperparameters for climatic variables forecasting and forest fire risk classification.

1.2 Hypothesis and Objectives

The hypothesis of this Master's project can be presented as follows:

- ❑ It is possible to classify the risk of forest fire occurrence in the Brazilian Pantanal by using historical climatic variables from the region, with a certain number of days

in advance.

The general objective of this work is to develop software containing ML models that are capable of:

1. Forecasting climatic variables for a specific number of days; and
2. Classifying the risk of forest fire occurrence in the Brazilian Pantanal.

To achieve the first objective, different time series forecasting algorithms were implemented and evaluated. The output of these algorithms is then used as input for supervised classification algorithms, which allow for the accomplishment of the second objective.

During this study development, there were incremental experiments that led to two different software versions:

- ❏ Exhaustive Search version (v1): The software hyperparameters are searched in an exhaustive manner;
- ❏ GA version (v2): The software hyperparameters are chosen based on GA. In this version, it was tested both a Traditional GA, as well as the Nondominated Sorting Genetic Algorithm II (NSGA-II) (DEB et al., 2002) – both are detailed in Chapter 5.

Both software versions, presented in details on Chapter 8, proved to be better or competitive with the main forest fire risk indexes, with the advantage of predicting fire risk for a given number of days in the future.

1.3 Document Structure

The remainder of this document is organized as follows:

- ❏ Chapter 2 - Data Pre-Processing: Introduces the main data pre-processing methods, focusing on scaling and sampling. The described methods are those that were implemented as part of this project.
- ❏ Chapter 3 - Machine Learning: Presents basic concepts of ML for data classification, as well as some of the main paradigms used in the literature and their respective algorithms. The described algorithms are those that were implemented as part of this project.
- ❏ Chapter 4 - Time Series Forecasting: Introduces the concepts of TS, as well as the main algorithms used in the literature for TS forecasting. The described algorithms are those that were implemented as part of this project.

- Chapter 5 - Genetic Algorithms: Presents the concept of GA and their different implementations. The described algorithms are those that were implemented as part of this project.
- Chapter 6 - Forest Fire Risk Indexes: Introduces the main forest fire risk indexes known in the literature, as well as their applications and limitations.
- Chapter 7 - Materials and Methods: Details the data used and the methods applied to build the software as proposed.
- Chapter 8 - Software: Details the software pipelines (training and prediction), the differences between the two versions (Exhaustive Search and GA) and how both were implemented.
- Chapter 9 - Experiments and Results: This chapter details the experiments we run for the developed software, along with their results. Those were compared to the main forest fire risk indexes used in the literature.
- Chapter 10 - Conclusion: Discuss the main findings of the proposal, experiments, and presented results. As well as suggesting possible next steps and future work.

Chapter 2

Data Pre-Processing

In different Data Science (DS) problems, one of the first and main steps is the Data Pre-Processing. This step refers to applying different techniques to the adapt and transform the data, so that it can be later used to feed Machine Learning models (as will be detailed in Chapter 3).

As exposed by (XIANG-WEI; YIAN-FANG, 2012), data pre-processing is a critical step to different DS use cases. A well pre-processed data input has the potential not only to increase the accuracy of the ML models, but also raise their computational efficiency.

In general, the data pre-processing step refers to techniques for data cleaning, data integration, data transition, data reduction, etc. In our study, we apply mainly two different data pre-processing techniques: Scaling and Sampling, as detailed below.

2.1 Scaling

Data – or more specifically, "feature" – scaling refers to the data pre-processing method that updates the range of independent variables of the data. It is usually misunderstood as meaning the same as "Normalization". However, normalization is just one of the different scaling forms.

Some ML algorithms are very sensitive to the data range, specially if there are variables with very different ranges between each other. So scaling methods play an important role on the ML models training and data fitting.

The study by (AMBARWARI; ADRIAN; HERDIYENI, 2020) demonstrated that data scaling methods provide significant improvements on ML algorithms performance, such as KNN, Naive Bayes, ANN, and SVM. The results revealed that the SVM with scaling outperformed other algorithms' performance. (AHSAN et al., 2021) also investigated

the effects of different scaling methods on ML models performance. Its results show that algorithm performance varies with different scaling methods. It was not possible to identify a single scaling method that can be ranked as being the best one among all of all others, though.

As will be detailed later in Chapter 8, there were investigated the following scaling methods in this project:

1. Min-Max: Scales each feature such that their values are between 0 and 1;

2. Max-Abs: Scales each feature such that the maximal absolute value of each feature is 1;

3. Robust: Scales features by removing the median and scaling them according to the interquartile range;

4. Standard: Scales features by removing the mean and scaling to unit variance.

2.2 Sampling

Unbalanced data is a common occurrence in real-world applications, characterized by a substantial disparity in proportions between different data classes. For example, when studying a rare disease and attempting to predict its occurrence, one may observe a considerably higher number of instances with the negative label (no disease) compared to the positive label (disease).

Addressing this challenge becomes crucial when developing ML models since they might struggle to learn the patterns associated with the less frequent class in comparison to the more prevalent class. In fact, an insufficiently trained model might exhibit a tendency to consistently classify instances into the more prevalent class, even when it shouldn't.

Given this context, it becomes imperative to employ sampling methods that aim to mitigate the disparities in class proportions within the data. Sampling methods can be classified into three different categories: Oversampling, undersampling and the combination between these two (hybrid).

(BUDA; MAKI; MAZUROWSKI, 2018) showed that a model performance may degrade as the level of class imbalance increases. The study of (JOHNSON; KHOSHGOFTAAR, 2020) evaluated the use of sampling methods when training ANNs with highly unbalanced data, using AUC as the evaluation measure. The results demonstrated that oversampling techniques performed significantly better than all undersampling and baseline methods. The hybrid methods ended up performing as good as the oversampling methods.

The different sampling techniques are explained in details in the following sections, as well as their main algorithms, as applied in this study. To simplify the explanations and align with the study's context, we will focus on binary label domains when discussing the methods.

2.2.1 Oversampling

Oversampling encompasses various sampling techniques designed to address disparities in class proportions by increasing the number of instances from the less frequent class while preserving the integrity of the most prevalent class. As a result, the updated dataset will contain a greater number of examples compared to the original dataset. Figure 1 illustrates the oversampling approach.

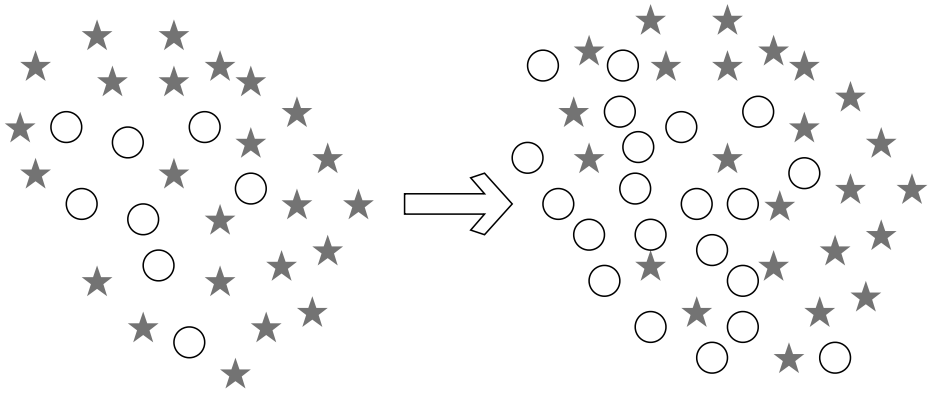

Figure 1 – Illustration of Oversampling approach. Adapted from (ALI et al., 2019).

In the literature, the main oversampling methods are:

1. Random Over Sampler: This method increases the number of instances from the least frequent class by randomly copying samples from such class (JOHNSON; KHOSHGOFTAAR, 2020). It is a very simple and computational efficient algorithm;

2. Synthetic Minority Oversampling Technique (SMOTE): Proposed by (CHAWLA et al., 2002), this algorithm works by first drawing an instance for the least frequent class. Then, it finds the k nearest neighbours from the selected instance. Finally, it creates synthetic instances by using the line segments from the initially selected instance and its neighbors;

3. Adaptive Synthetic (ADASYN): Proposed by (HE et al., 2008), it is an extension of the SMOTE algorithm. It focuses on generating synthetic instances that are harder to learn. To do so, it considers the distribution density of their k nearest neighbors, so that it can place more emphasis on the regions that are difficult to classify correctly;

4. Borderline SMOTE: Also an extension of the SMOTE algorithm. It focuses on generating synthetic instances from the leas frequent class instances that are near the decision boundary. To do so, it first identifies the borderline instances and then applies the SMOTE algorithm only to those instances (HAN; WANG; MAO, 2005);

5. SVM-SMOTE: This algorithm combines the concepts of SVMs and SMOTE. It works by first applying an SVM classifier to identify the support vectors (the SVM algorithm will be detailed later in Section 3.4.3), which are the instances near the decision boundary. Finally, it applies the SMOTE algorithm to generate the synthetic instances (HAN; WANG; MAO, 2005).

2.2.2 Undersampling

In contrast to the oversampling approach, undersampling aims to decrease the number of instances from the most prevalent class while preserving the integrity of the less frequent class. Consequently, the updated dataset will contain a reduced number of examples compared to the original dataset. Figure 2 illustrates the undersampling approach.

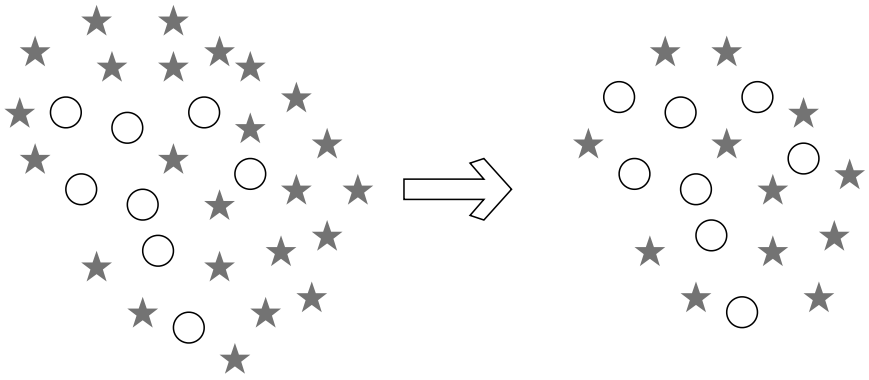

Figure 2 – Illustration of Undersampling approach. Adapted from (ALI et al., 2019).

In the literature, the main undersampling methods are:

1. Random Under Sampler: This method decreases the number of instances from the most frequent class by randomly discarding samples from such class (JOHNSON; KHOSHGOFTAAR, 2020). It is a very simple and computational efficient algorithm;

2. Cluster Centroids: This method works by first identifying clusters of the most frequent class making use the K-Means algorithm. Then, it replaces the samples of the most frequent class with synthetic instances that are representative of each cluster's centroid (LEMAITRE; NOGUEIRA; ARIDAS, 2016);

3. Near Miss: Proposed by (ZHANG; MANI, 2003), it refers to a collection of different methods (NearMiss-1, NearMiss-2 and NearMiss-3) that select the instances to be

discarded based on the distance of most frequent class instances to least frequent class instances;

4. Edited Nearest Neighbours (ENN): Based on (WILSON, 1972), this method applies the KNN algorithm to each instance, checking if the most frequent class from the instances's K-nearest neighbor is the same as the instances's class or not. If they differ, the instance and its K-nearest neighbor are discarded;

5. Repeated ENN: Extension of the ENN method. As the name suggests, it runs the ENN multiple times, in order to improve the performance of the undersampling method;

6. All KNN: Also an extension of the ENN method. It runs the ENN multiple times, but varying the number of nearest neighbours (LEMAITRE; NOGUEIRA; ARIDAS, 2016);

7. One Sided Selection: Based on (KUBAT; MATWIN, 1997), it combiness Tomek Links technique and the Condensed Nearest Neighbor rule. The Tomek Links is a data cleaning technique, which works by identifying pairs of instances, one from the most frequent class and one from the least frequent class, that are the nearest neighbors of each other. Then it removes the instances from the most frequent class in these pairs, hence improving the class distribution.

8. Neighbourhood Cleaning Rule: Proposed by (LAURIKKALA, 2001), it applies the ENN to remove instances from the most frequent class. For each instance, it finds three nearest neighbors. Then if checks whether the instance belongs to the most frequent class and if the nearest neighbors is different from the selected class instance. If so and the selected class instance belongs to the least frequent class, then the nearest neighbors that belong to the most frequent class are removed;

9. Instance Hardness Threshold: Based on (SMITH; MARTINEZ; GIRAUD-CARRIER, 2014), the main idea of this method is to identify instances that are likely to be misclassified and remove them if they belong to the most frequent class. Theses instances are identified based on the instance hardness, typically computed using a measure such as the confidence score.

2.2.3 Hybrid

The hybrid methods apply both oversampling and undersampling techniques, in order to improve the data balancing. In the literature, the main hybrid methods are:

1. SMOTE-ENN: Combines the SMOTE (oversampling) and ENN (undersampling) methods, as presented by (BATISTA; PRATI; MONARD, 2004);

2. SMOTE-Tomek: As proposed by (BATISTA; BAZZAN; MONARD, 2003), this method applies both the SMOTE (oversampling) and the Tomek Links technique.

Chapter 3

Machine Learning (ML)

Machine Learning (ML), a field of Artificial Intelligence (AI), studies algorithms that allow computer programs to automatically improve through experience (MITCHELL, 1997). Therefore, ML aims to replace or assist the practice of programming imposed by conventional computing (RUSSELL; NORVIG, 2010; ALPAYDIN, 2014). In conventional computing, solving a particular problem involves: (1) Acquiring knowledge about the problem domain; (2) Writing algorithms that incorporate business rules, functions, and/or mathematical models to solve the problem. The limitations involved in this process range from low adaptability to changes in the problem domain to time and human resource expenditure. With the use of ML, on the other hand, the steps can be summarized as follows: (1) Collection and cleaning of data that describe the problem domain; (2) Training of algorithms such as classifiers and regressors. The learning task is therefore delegated to the machine (computer). There are different ways to approach a ML problem, which is typically determined by the data domain. Figure 3 presents the main approaches.

Among the various ML approaches, there is the Supervised Learning. This learning approach relies on labeled datasets, where the labels can be a set of classes $y \subset Y$, with Y representing a complete class domain (classification), or a continuous numerical output (regression).

In the present study, we have a classification problem. The data consists of climatic variables, and each example is labeled with one of two classes: Fire occurrence (1) or no fire occurrence (0) (further details about the dataset used will be presented in Chapter 7). Thus, we aim to find a function f capable of mapping the attribute set x to a set of classes y. This model is built using a training dataset, where the classes are known for all n examples. The model can be represented in various forms, such as a tree, an ANN, or even a probability table. The model captures the relationships between the attribute

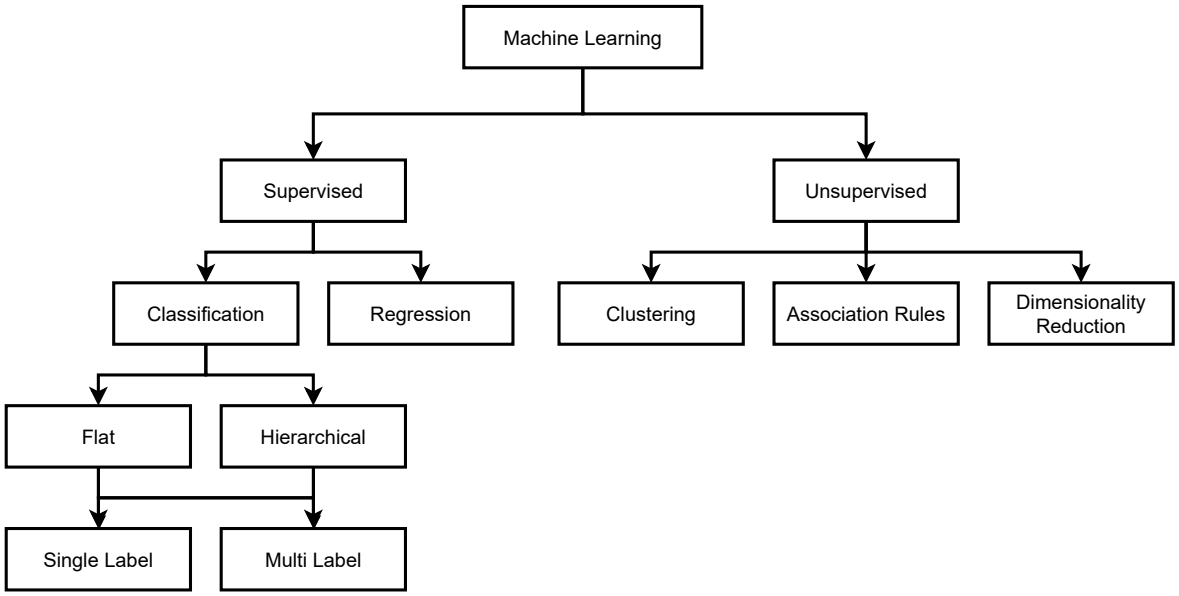

Figure 3 – Machine Learning approaches.

space and the class space, and it is considered capable of correctly classifying an object (x, y) if $f(x) = y$ (TAN; STEINBACH; KUMAR, 2005).

Since $|Y| = 2$, there is a binary domain. If we had $|Y| > 2$, it would be a multi-class problem. Multi-class problems with $|y| \geq 2$ are referred to as multi-label, meaning that an object can be assigned to more than one class simultaneously.

In the field of ML literature, conventional classification problems are typically solved using flat (non-hierarchical) classification methods. This type of classification disregards the hierarchical relationships between classes and assumes independence among them.

In more complex classification problems, classes are often organized in a hierarchy, where classes can be divided into subclasses and grouped into superclasses. In such cases, classifiers take into account the most relevant relationships among the training data for classification. The classifier f must respect the constraints of the hierarchical taxonomy. This means that when a class is predicted, all its superclasses should also be predicted. Figure 4 illustrates the difference between flat and hierarchical classification problems.

In the following sections of this chapter, some of the main paradigms of ML and their respective algorithms that can handle classification problems will be presented. Although many of these algorithms also perform well as regressors, these applications will not be detailed as they are not within the scope of this study.

3.1 Symbolic Paradigm

The Symbolic Paradigm is based on symbols that represent concepts from the real world to create “rules”. These “rules” allow the concrete manipulation of these symbols and the generalization of knowledge, and they can typically be interpreted in natural

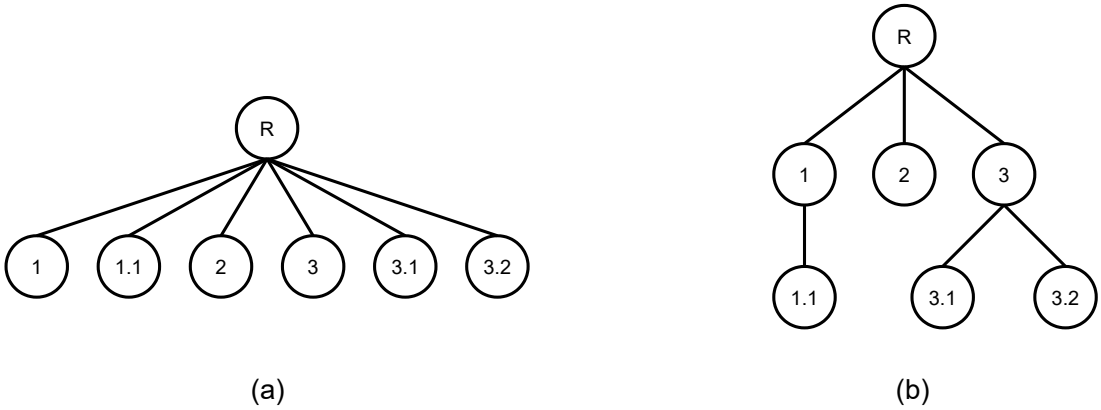

Figure 4 – Difference between classification problems: (a) Flat; (b) Hierarchical.

language. Examples of algorithms that apply this paradigm are Decision Trees and more complex algorithms that are based on these trees, such as Random Forests.

3.1.1 Decision Tree

Decision Trees are a category of supervised machine learning algorithms that stand out in various applications due to their simplicity and interpretability. In these algorithms, the model is represented as a tree where each node represents a test on an attribute, and the leaf nodes represent classes. Objects are classified one by one by traversing the tree from the root to a leaf node. In this process, at each level of the tree, a test is applied to some attribute to decide which next node to follow. The leaf node represents the final classification found for that object after traversing the tree. An illustrative example of a Decision Tree is shown in Figure 5, where the data consists of weather aspects with attributes such as humidity, temperature, and wind speed. The expected classification in this example is whether it will rain ("Yes" or "No").

For example, consider an object with the following attribute values: Humidity = High, Wind = Weak, Temperature = High, Sky = Cloudy. By traversing the Decision Tree in Figure 5 according to the attribute values of the object, the class "No" is obtained, indicating that it will not rain on that day.

There are different algorithms based on Decision Trees proposed in the literature, each with different methods for model construction and different heuristics. The most well-known algorithms in the literature are ID3 (QUINLAN, 1986) and its successor, C4.5 (QUINLAN, 1993).

In most examples, the Decision Tree is constructed using a top-down procedure, starting with the root and descending level by level. The first question to be asked is, "Which attribute should be tested at the root of the tree?" (MITCHELL, 1997). To answer this question, heuristics are applied to each attribute, and the one that yields the best result is chosen. The data is then partitioned, with objects that satisfy the heuristic on one side

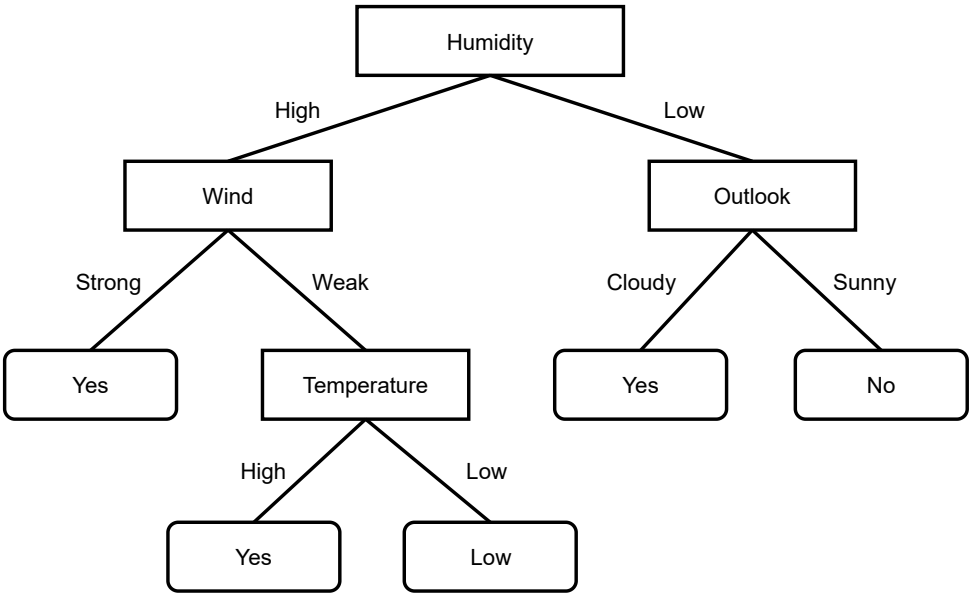

Figure 5 – Illustration of Decision Tree. Adapted from (MITCHELL, 1997).

of the node and objects that do not satisfy it on the other side. This is based on binary splits. It is worth noting that for non-binary splits, the procedure is similar, except that each node can have more than two children. The procedure is then repeated recursively until a pre-defined stopping criterion is reached, resulting in a leaf node. The final classification given by that leaf node considers the majority class of the objects that reached that node.

One of the main advantages of applying Decision Trees to classification problems is their interpretability. By constructing and visualizing the tree, it is possible to identify the most relevant attributes for determining classes at different levels, for example. The computational cost of Decision Trees is directly related to the heuristics and parameters chosen, such as the maximum tree depth. Additionally, the presence of a larger number of continuous attributes in the data increases the complexity of attribute selection and node splitting calculations. It is worth noting that Decision Trees are highly sensitive to imbalanced data, and in such scenarios, it is common for the constructed tree to be biased towards the dominant class.

3.1.2 Random Forest

The Random Forest is an ensemble algorithm that combines multiple Decision Trees (Section 3.1.1). Ensemble methods construct multiple individual models that are heterogeneous in nature to obtain a single final model. This final model is based on the combination of these individual models, aiming for better generalization and reduced risk of overfitting (MOYANO et al., 2018). Overfitting in machine learning occurs when the model fits the training data extremely well but fails to generalize well to unseen test data,

which is an undesirable behavior.

In a Random Forest, each tree contributes a unit vote for the final class. By combining these results, the classification outcome is determined based on the majority vote. The representation of a Random Forest is shown in Figure 6.

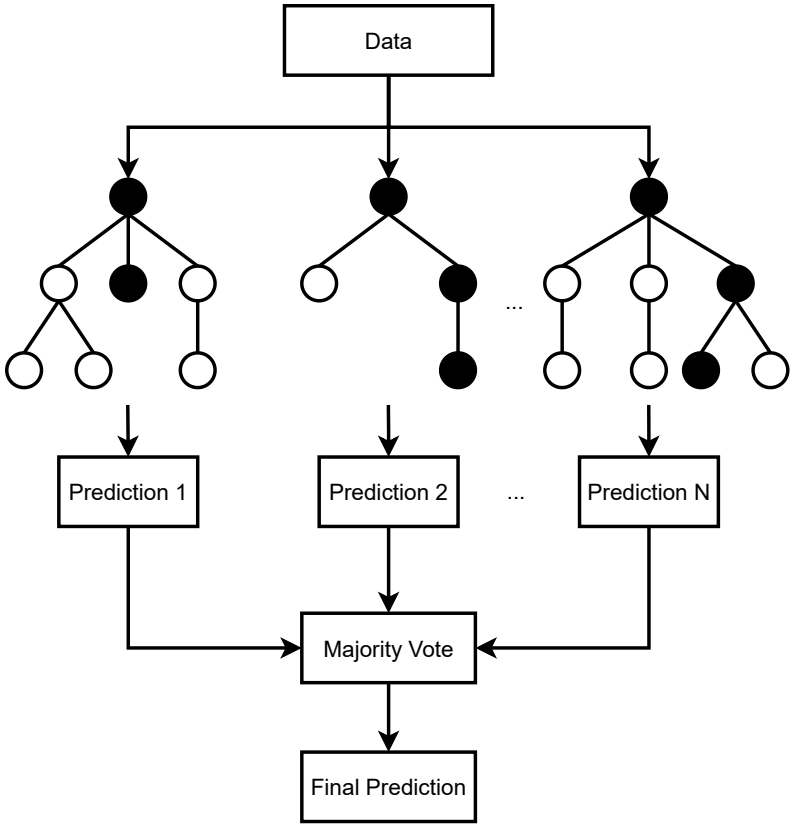

Figure 6 – Illustration of Random Forest.

In (DIETTERICH, 2000), some reasons are presented as to why an ensemble classifier can be better than a single classifier, such as: (1) Choosing a single classifier may lead to a poor choice; (2) A particular algorithm may not always be able to find the optimal solution, so running the same algorithm multiple times (with different parameters) and combining the responses can lead to a better approximation of the optimal solution.

However, compared to the Decision Tree algorithm, the Random Forest algorithm is more computationally expensive in terms of training time, especially for large datasets. Additionally, the interpretability associated with the Decision Tree algorithm is lost.

3.1.3 XGBoost

The XGBoost algorithm is an ensemble algorithm of Decision Trees (Section 3.1.1) that utilizes a technique called boosting. It was first introduced by (CHEN; GUESTRIN, 2016). The boosting method aims to construct a "strong" classifier from multiple "weak" classifiers combined in series. Some algorithms that apply boosting are ADABoost (SCHAPIRE,

2013), LightGBM (KE et al., 2017) (Section 3.1.4), CatBoost (PROKHORENKOVA et al., 2018) (Section 3.1.5), and XGBoost itself.

In XGBoost, a specific boosting method called "gradient boosting" is applied. The objective of this method is to find a function (f) that best describes the data, as well as to find the optimal parameters (p) of this function. To find f, gradient boosting combines different simple functions. Each function takes as input the gradient of the error with respect to the predictions of the previous functions. As a result, the functions learn to "correct" the errors, allowing the combination of simple functions to compensate for each other's weaknesses in order to better fit the data.

Thus, considering the different classifiers in series in XGBoost, at each iteration, the residual errors from the previous classifier are included in the training of the next classifier. The final classification is given by the weighted sum of all the classifications of the "weak" classifiers. An illustration of an XGBoost model is shown in Figure 7.

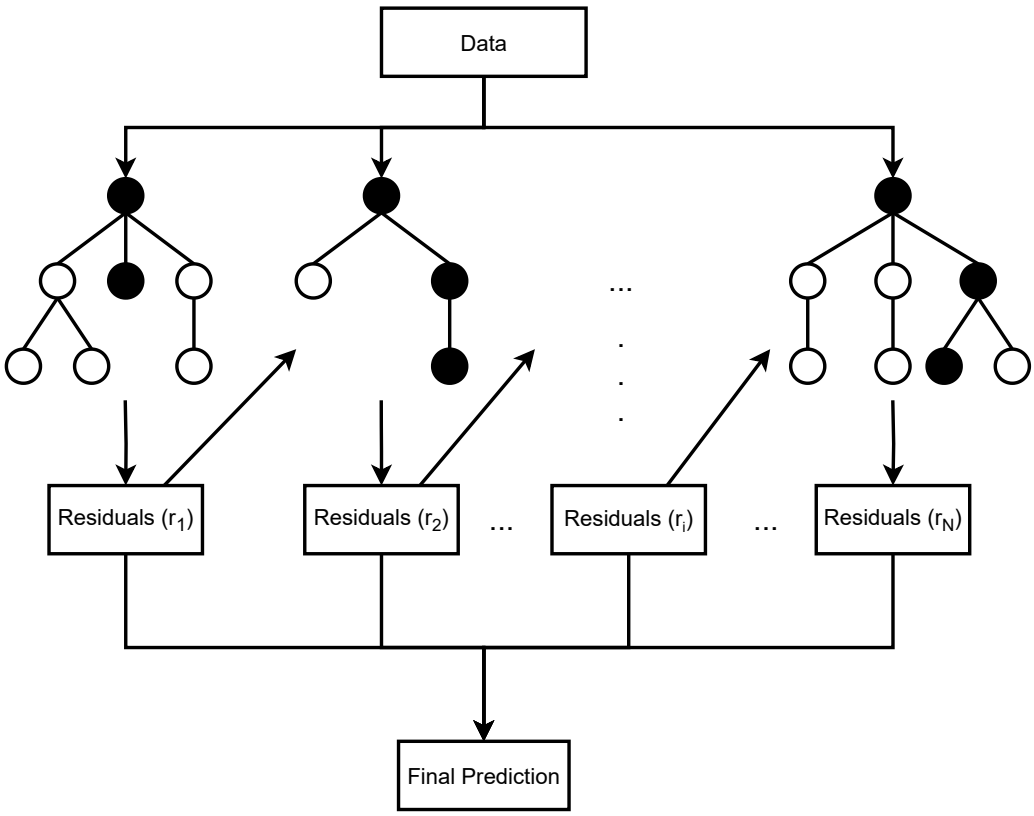

Figure 7 – Illustration of XGBoost.

In XGBoost, the trees are built in parallel, and the computational power of this algorithm has been shown to be superior to many others used in the literature. The algorithm performs well on both small and large datasets, except in cases where the data is very sparse or imbalanced. Similar to Random Forest (Section 3.1.2), interpretability associated with the Decision Tree algorithm is lost when using XGBoost.

3.1.4 LightGBM

LightGBM, like XGBoost (Section 3.1.3), applies gradient boosting. However, two new methods are included in the algorithm: (1) Gradient-Based One-Side Sampling (GOSS); and (2) Exclusive Feature Bundling (EFB). Both methods allow for a significant reduction in the computational complexity of the model compared to XGBoost (KE et al., 2017).

GOSS is a subsampling method based on the hypothesis that instances with small error gradients are already well trained. During training, GOSS retains instances with large gradient errors and randomly samples instances with small errors. By doing so, GOSS focuses on instances that are not well adjusted (high gradient errors).

On the other hand, EFB aims to reduce the feature space of the problem. This method is based on the hypothesis that highly sparse data indicates the existence of mutually exclusive features, i.e., features that never take different values from zero at the same time. These attributes are combined into "bundles" and treated as a new and unique attribute.

LightGBM has the same advantages as XGBoost but improves its computational performance while maintaining its predictive performance.

3.1.5 CatBoost

CatBoost, proposed by (PROKHORENKOVA et al., 2018), is also a gradient boosting method. It focuses on improving the model's performance when dealing with categorical features, which are discrete attributes that are not necessarily comparable (examples include attributes like “name” or “color”). Additionally, the algorithm proposes a new scheme for calculating leaf node values, which allows for the creation of more balanced trees and helps reduce overfitting.

For attributes with a small number of distinct categories, CatBoost applies the method known as One-Hot Encoding. This creates a new binary attribute for each distinct category. For other categorical columns, CatBoost uses an efficient encoding method. It is similar to Target Encoding but with an additional random permutation mechanism aimed at reducing overfitting. In Target Encoding, the average target value in the training set of all observations with a particular category is used to encode that category (PARGENT et al., 2022).

In the study by Prokhorenkova et al. (PROKHORENKOVA et al., 2018), it was demonstrated that CatBoost outperforms XGBoost and LightGBM in predictive performance. However, achieving such performance requires careful optimization of its hyperparameters, which can impact the computational training cost in most cases.

3.2 Example-Based Paradigm

The Example-Based Paradigm is based on the assumption that similar examples belong to the same class. Thus, new examples are compared to the known examples, usually one by one. For this purpose, the known examples need to be stored in memory during the execution of the classifier, and therefore classifiers of this paradigm are also commonly referred to as Memory-Based Classifiers (CUNNINGHAM; DELANY, 2020).

3.2.1 K-Nearest Neighbors (KNN)

The operation of the KNN algorithm is intuitive. Simplistically, during the training phase, the input examples are mapped into a d-dimensional space, where d is the number of attributes that describe each example. As mentioned earlier, the training examples are stored in memory to be used during the execution of the classifier. To determine the class of a test example, it is also mapped into the space constructed during training, and its class is defined based on the majority class among the k nearest neighbors in that space.

Figure 8 illustrates the operation of a KNN classifier with a 2-dimensional space and k equal to 3. The class space Y is X, O. In the figure, a test example $q1$ is presented to the classifier and mapped into the space. Then, the three closest examples to $q1$ are identified: all three belong to class O. Therefore, $q1$ is also classified as O. As for $q2$, among the three nearest neighbors, two belong to class X and one belongs to class O. Thus, $q2$ is classified into the majority class, which is X.

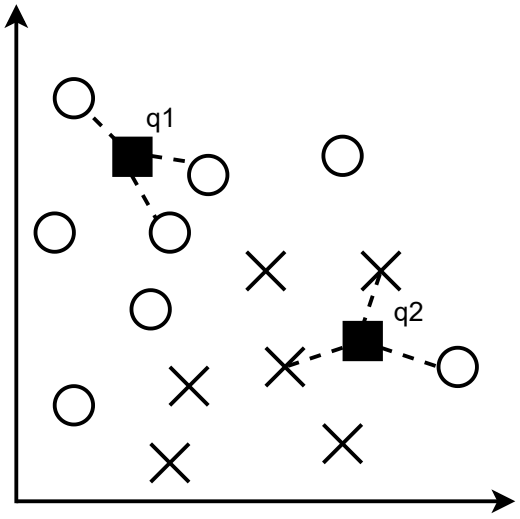

Figure 8 – Illustration of a 2-dimensional KNN classifier with 2 classes and k equal to 3.

Thus, it can be said that the classification process in KNN consists of two phases: The first phase involves finding the k nearest neighbors, and the second phase involves identifying the majority class among these neighbors.

For the first phase, different distance calculations can be used, such as Euclidean, Hamming, Manhattan, and Minkowski distances. The most common one is the Euclidean distance, described by Equation 1.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}, \tag{1}$$

where n is the number of dimensions from the examples.

For the second phase, different heuristics can be applied. One of them involves assigning a weight to each "vote," with the weight being inversely proportional to the distance of that neighbor. In the case of continuous outputs, instead of seeking the majority class, the average of the output values of the k neighbors can be calculated.

One of the main problems of KNN is related to its computational complexity, both in terms of execution and memory. KNN is considered a Memory-Based Classifier, and as the number of training examples increases, the space complexity also increases (linear complexity). As for the execution complexity, it is mainly affected by the distance calculations. In order to find the k nearest neighbors, it is necessary to calculate all possible distances in the problem, i.e., the distances from the test example to each of the training examples mapped in the feature space. Thus, in Big-O notation, the complexity is equal to $O(d \cdot n)$, where d is the number of attributes describing the example (dimensions) and n is the number of training examples.

Given this challenge, it is important to consider the necessary amount of training examples for KNN and to work with dimensionality reduction techniques in problems with high dimensions.

3.3 Connectionist Paradigm

The Connectionist Paradigm is strongly inspired by the structure of the human brain, which consists of interconnected units that exchange signals (synapses) with each other. It is from this paradigm that ANNs are proposed.

3.3.1 Multi-Layer Perceptron (MLP)

The perceptron is one of the simplest models of an artificial neuron, first described in Rosenblatt's seminal work in 1957 (ROSENBLATT, 1957). It is illustrated in Figure 9.

The basic training of a perceptron consists of the following steps:

1. Randomly initialize a weight vector ($\mathbf{w} = [w_0, ..., w_N]$), where N is the number of inputs to the perceptron;

2. Calculate the estimated output $u_j = \mathbf{w} \cdot \mathbf{x}j$ for a training example $\mathbf{x}j = [x_{j0}, ..., x_{jN}]$;

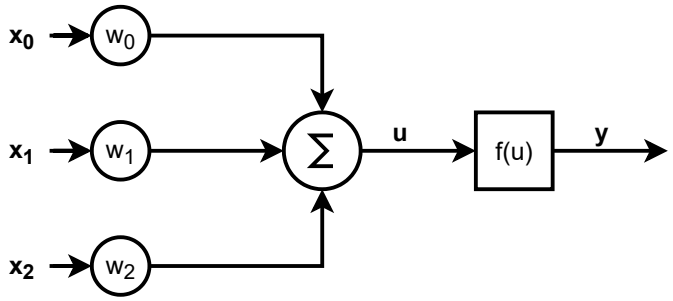

Figure 9 – Illustration of Perceptron.

3. Update the output value based on an activation function f, typically the sigmoid or sign function: $y_j = f(u_j)$;

4. Calculate the error between the predicted output (y_j) and the expected output (d_j): $e_j = d_j - y_j$;

5. Update the weight vector $\mathbf{w}$ according to the formula: $\Delta\mathbf{w} = \eta \cdot e \cdot \mathbf{x}$, where η is a predefined learning rate;

6. Repeat steps 2 to 5 until a specified number of epochs is reached or a termination criterion is satisfied.

The main limitation of the perceptron is its inability to handle non-linearly separable class domains. This significantly restricts its range of applications. To overcome this limitation, ANNs were proposed, which are ordered collections of artificial neurons.

One of the most widely used ANN algorithms in the literature is the Multi-Layer Perceptron (MLP). It consists of multiple artificial neurons (perceptrons) organized in different layers, which are interconnected. Signals are transmitted unidirectionally through the network, from the input layer to the output layer. This architecture is known as a "feedforward" network and is depicted in Figure 10.

The MLP is trained using the backpropagation algorithm, which was proposed by (BRYSON; HO, 1969). It focuses on minimizing the error between the network's output(s) and the expected output(s). This involves propagating the error signal backward through the network and updating the weights accordingly (POPESCU et al., 2009).

MLP models are known for their ability to handle complex nonlinear problems and perform well on both small and large datasets. Additionally, MLPs have a straightforward architecture compared to other types of artificial neural networks. However, there are some drawbacks to consider. One is the low interpretability of the model, meaning it may be difficult to understand the reasoning behind its predictions. Another drawback is the high computational cost required for training an MLP, which can be a time-consuming process.

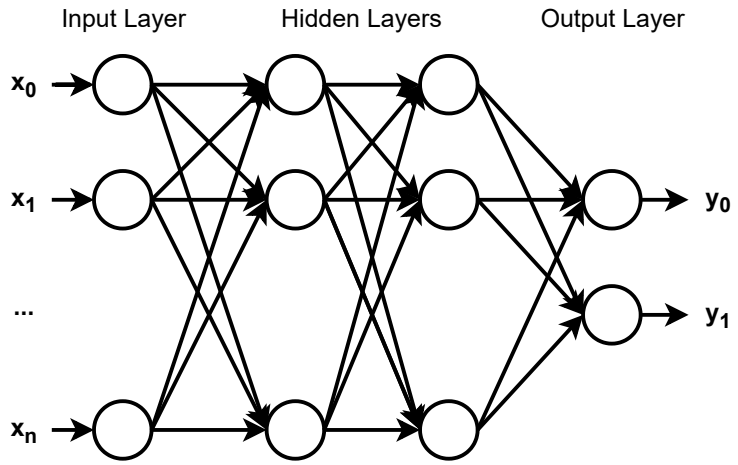

Figure 10 – Illustration of MLP.

3.4 Statistical Paradigm

In the Statistical Paradigm, the goal is to find a statistical model that provides a good approximation of the concept to be induced. The general idea of algorithms based on this paradigm is to explore the functional dependencies within a dataset in order to find a classifier capable of separating the data into specific classes (CERRI, 2010).

3.4.1 Naive Bayes

Naive Bayes is a probabilistic algorithm widely applied to classification tasks, mainly due to its simplicity and low computational cost. The algorithm is based on Bayes' theorem and assumes independence among the attributes, meaning that there is no correlation between the attributes describing the problem (LEWIS, 1998).

Consider Equation 2, which defines the Conditional Probability of class $y_j \in Y$ given a domain of attributes X.

$$P(y_j|X) = \frac{P(y_j) \cdot P(X|y_j)}{P(X)}, \tag{2}$$

where $P(y_j)$ indicates the probability of class y_j being present in the training data domain; $P(X|y_j)$ is the probabilistic distribution of X in the space of classes; and $P(X)$ is the sum of the observation probabilities of each attribute.

In practice, the calculation of $P(X|y_j)$ cannot be computed directly (ZHANG; GAO, 2011). This is where the algorithm becomes "naive". Independence between attributes is assumed, so the calculation of $P(X|y_j)$ can be simplified using Equation 3.

$$P(X|y_j) = \prod_i P(x_i|y_j) \tag{3}$$

So, the new formula $P(y_j|X)$ is calculated according to Equation 4.

$$P(y_j|X) = \frac{P(y_j) \cdot \prod P(x_i|y_j)}{P(X)} \tag{4}$$

The final class of an attribute vector $x_i \in X$ is then the one with the highest probability, as in Equation 5.

$$f(x_i) = \arg\max_j P(y_j|xi) \tag{5}$$

Despite the assumption that the attributes are independent, which is rarely true, the algorithm performs surprisingly well in different applications (ZHANG; GAO, 2011; WEI; VISWESWARAN; COOPER, 2011; CHEN et al., 2021). Additionally, it is easy to parallelize and has low computational cost in terms of execution time and storage space. However, its performance can degrade when the class space Y is very large and the data is imbalanced. Moreover, it is not ideal for domains where it is known that the attributes may have significant correlation among them.

3.4.2 Logistic Regression

The Logistic Regression algorithm, similarly to the Linear Regression, explores the relation between the (independent) variables and the outcome (dependent) variable. In that sense, the final model is able to describe the outcome variable as a sum of products, each product formed by multiplying the value and coefficient of the independent variable (PARK, 2013).

The Logistic Regression, in a binary data class domain, may apply the standard logistic function (also refered as sigmoid function). The logistic regression model can then be described as in Equation 6.

$$f_{\mathbf{w},b}(x) = \frac{1}{1 + e^{-\mathbf{w}x+b}}, \tag{6}$$

So, $f(\mathbf{x})$ may be interpreted as the probability of the y_i being positive (BURKOV, 2019). To estimate the coefficients (or weights) in the $\mathbf{w}x + b$ term, the logistic regression algorithm maximizes the likelihood function, that is, it applies the maximum likelihood optimization criterion, as defined in Equation 7.

$$L_{\mathbf{w},b} = \prod_{i=1...N} f_{\mathbf{w},b}(\mathbf{x_i})^{y_i}(1 - f_{\mathbf{w},b}(\mathbf{x_i}))^{(1-y_i)} \tag{7}$$

Due to the exponential function used in the model, it is required an adaptation in order to avoid numeric overflow. So, it is convenient to maximize the log-likelihood instead (Equation 8) (BURKOV, 2019).

$$ln(L_{\mathbf{w},b}) = \sum_{i=1}^{N} [y_i \cdot ln(f_{\mathbf{w},b}(\mathbf{x_i})) + (1 - y_i) \cdot ln(1 - f_{\mathbf{w},b}(\mathbf{x_i}))] \tag{8}$$

In contrast to the Linear Regression, in Logistic Regression there is not a single solution for the above optimization. It is typically applied the Gradient Descent method, but others such as the Newton's Method and the Adaptive Moment Estimation algorithm can also be tested.

The main advantages of the Logistic Regression algorithm is that it can be quite efficient, even in high-dimensional feature spaces. Also, it provides interpretability, as the coefficients provide insights into the influence of each feature on the predicted outcome. On the other hand, the Logistic Regression requires that the independent variables are linearly related to the log odds - if not, the model may not be able to generalize based on the training data. Besides that, it assumes that the Independence between the variables. If the features are strongly correlated, it can lead to unreliable coefficient estimates.

3.4.3 Support Vector Machine (SVM)

The SVM is also built upon the statistical paradigm. Support vector-based classifiers (CORTES; VAPNIK, 1995; CERRI, 2010) construct hyperplanes that aim to achieve an appropriate separation of classes in a high-dimensional attribute space (CRISTIANINI; SHAWE-TAYLOR, 2000). The goal of an SVM classifier is to identify a hyperplane that separates data from different classes while maximizing the margin of separation. In a binary classification problem, data can be separated with just one hyperplane. In problems with $|Y|$ classes, where $|Y| > 2$, the solution of binary classifiers can be utilized. One approach is to train $|Y|$ binary classifiers, one for each class.

Equation 9 represents a hyperplane and contains two parameters: $\mathbf{w}$ is a vector of real values with the same dimensionality as the attribute vector $\mathbf{x}$, and b is a real number (BURKOV, 2019).

$$\mathbf{w} \cdot \mathbf{x} - b = 0, \tag{9}$$

where $\mathbf{w} \cdot \mathbf{x}$ is the dot product of the two vectors ($\mathbf{w}$ and $\mathbf{x}$). Using this, we can predict the class of an example i based on Equation 10.

$$y_i = sign(\mathbf{w} \cdot x_i - b) \tag{10}$$

Thus, to achieve the objective of identifying a hyperplane that separates data from different classes while maximizing the margin, the algorithm seeks the optimal values of $\mathbf{w}$ and b. This is done through an optimization algorithm that aims to minimize the Euclidean norm $||\mathbf{w}||$.

In this context, support vectors are the data points that are close to the separation hyperplane and therefore define the position of the hyperplane in the space. They are the critical examples in the dataset. The larger the separation margin, the greater the model's generalization capability.

Figure 11 illustrates a hyperplane with maximum margin and its support vectors.

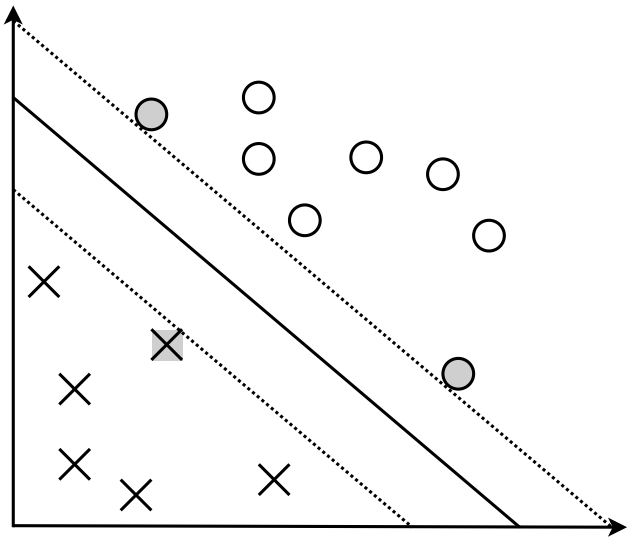

Figure 11 – Illustration of a hyperplane separating two classes and its support vectors highlighted, according to an SVM algorithm. Adapted from (CRISTIANINI; SHAWE-TAYLOR, 2000).

The definitions presented above consider a linear SVM, where the decision boundary is a hyperplane. Other more advanced SVM algorithms can incorporate kernels, which allow for the solution of non-linear problems as well (BURKOV, 2019).

SVMs perform well in domains where classes have a clear margin of separation. They are also efficient in handling high-dimensional spaces, even when the number of dimensions is larger than the number of training examples. However, SVM training has a high computational cost, and the algorithm is extremely sensitive to noise. Therefore, data cleaning and selection become even more important when dealing with SVMs.

Chapter 4

Time Series (TS) Forecasting

A TS consists of data related to the observation of a phenomenon over time. Algorithms for time series forecasting can operate in different ways, but the main ones are capable of identifying patterns and learning from historical data. Once trained, they can predict data for various future intervals (PARMEZAN; SOUZA; BATISTA, 2019). Therefore, a time series algorithm should be able to forecast data for a certain number of days in the future ("forecast horizon"), based on a number of days in the past ("observation window").

TS forecasting algorithms range from conventional statistical models, such as ARIMA and SARIMA, to more complex models based on machine learning, such as the algorithms described in the following sections.

4.1 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN), unlike conventional neural networks (Section 3.3.1), is capable of retaining information. Therefore, it is said that these networks have "memory" because they use information from previous inputs to influence the current input and output. This is achieved through loops in their neurons, as illustrated in Figure 12, and hidden states. The hidden states capture historical information from the first sequences up to the current moment of the network.

RNNs apply an adapted version of backpropagation called Backpropagation Through Time (BPTT). One of the problems with traditional RNNs is their inability to retain information from a long time ago. This is because, during BPTT, they suffer from the problem of vanishing gradients, where gradients diminish over time and have less and less

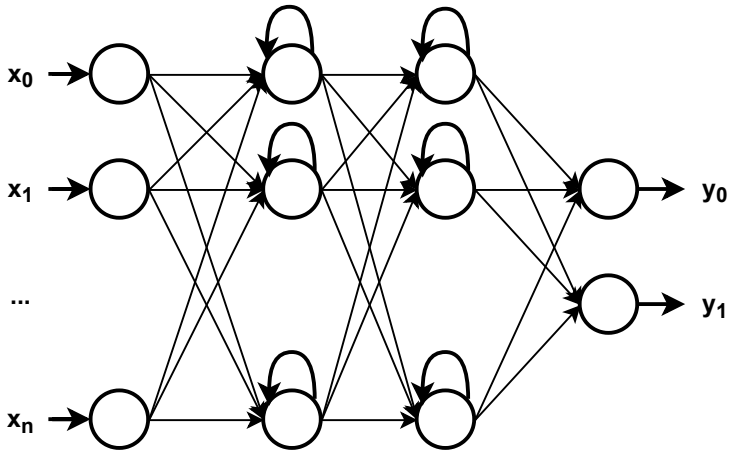

Figure 12 – Illustration of RNN.

influence on learning. LSTM and Gated Recurrent Unit (GRU) networks address this issue.

4.1.1 Long-Short Term Memory (LSTM)

LSTMs introduce the concept of "gates", which are internal mechanisms that regulate the flow of information through the network. These gates maintain the state of a particular cell in the network and control when information is added to or removed from these cells. The gates can use either the sigmoid or hyperbolic tangent ($tanh$) function to control the flow. The sigmoid function returns a value between 0 and 1, while $tanh$ is used to sustain the gradient for a longer time, reducing the impact of vanishing gradients.

A simplified LSTM architecture (Figure 13) consists of three parts: the forget gate, the input gate, and the output gate. The forget gate determines how much of the previous information should be retained and how much should be forgotten. If the sigmoid function output is 0, everything is forgotten. If it is 1, nothing is forgotten. The input gate is used to quantify the importance of the new information presented by the input. Finally, the output gate determines the value of the next hidden state.

Since they were proposed in 1997, LSTMs have been applied in various fields to solve a wide range of problems. In (ABBES; MAGAGI; GOITA, 2019), an LSTM was used to estimate soil moisture. In the work of (AKTER; LEE; KIM, 2021), LSTMs were studied in the context of energy consumption forecasting. Fourteen years of hourly energy consumption data were analyzed. LSTMs have also been applied to wind speed prediction. In (P.; VANITHA; R, 2019), the LSTM model was built using seasonal wind speed data from four different regions in India. The results demonstrated that seasonal wind speed prediction using the LSTM model can reduce errors compared to conventional weather forecasting. Deviation rates were calculated from a wind turbine.

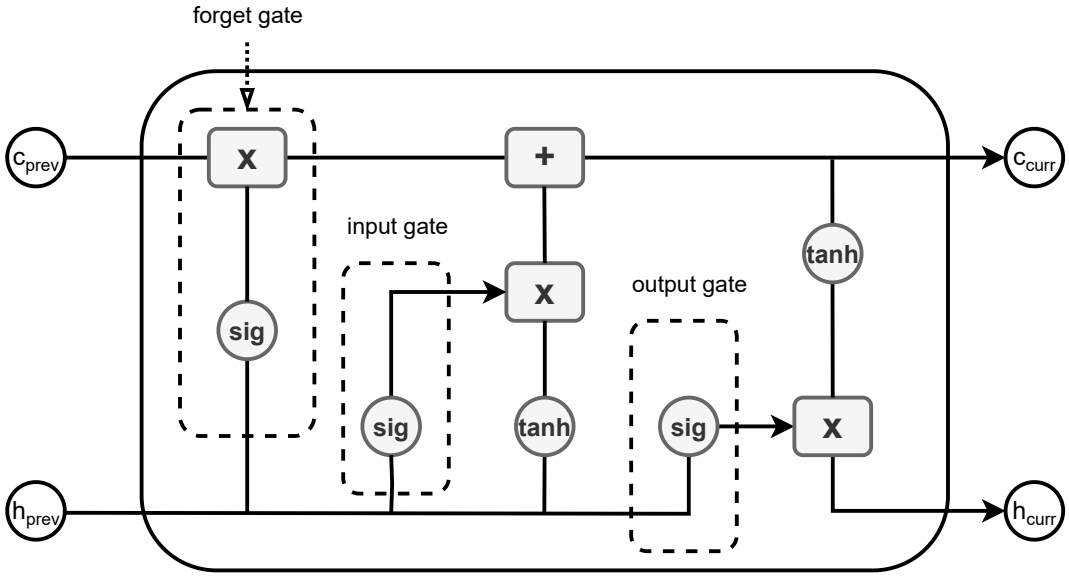

Figure 13 – Illustration of an LSTM. c_{prev} represents the previous cell state, h_{prev} represents the previous hidden state, c_{curr} represents the current cell state, and h_{curr} represents the current hidden state. Adapted from (ZHAO et al., 2020).

4.1.2 Gated Recurrent Unit (GRU)

The GRU, similar to the LSTM (Section 4.1.1), addresses the issue of gradient vanishing in conventional RNNs. It solves this problem by using two gates: the update gate and the reset gate. The update gate of the GRU determines how much of the new information should be stored and how much should be ignored. The reset gate, on the other hand, determines how much of the previous information should be retained and how much should be forgotten. Figure 14 illustrates the architecture of the GRU.

The GRU has fewer operators and gates compared to LSTM and, therefore, can be considered faster. However, LSTM often performs better in situations where the input sequence is extensive.

There are several recent studies in the literature that apply GRU for time series forecasting. In (XU et al., 2021), a GRU combined with a factorization machine (FM-GRU) was applied for water quality prediction over time. Another study applied GRU for machinery failure prediction and achieved an accuracy of 87% (ZAINUDDIN; A.; H., 2021).

4.2 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is an ANN algorithm, and its basic architecture consists of three types of layers: (1) Convolutional layer, (2) pooling layer, and (3) fully connected layer. Each layer plays a different role in the CNN, as described below:

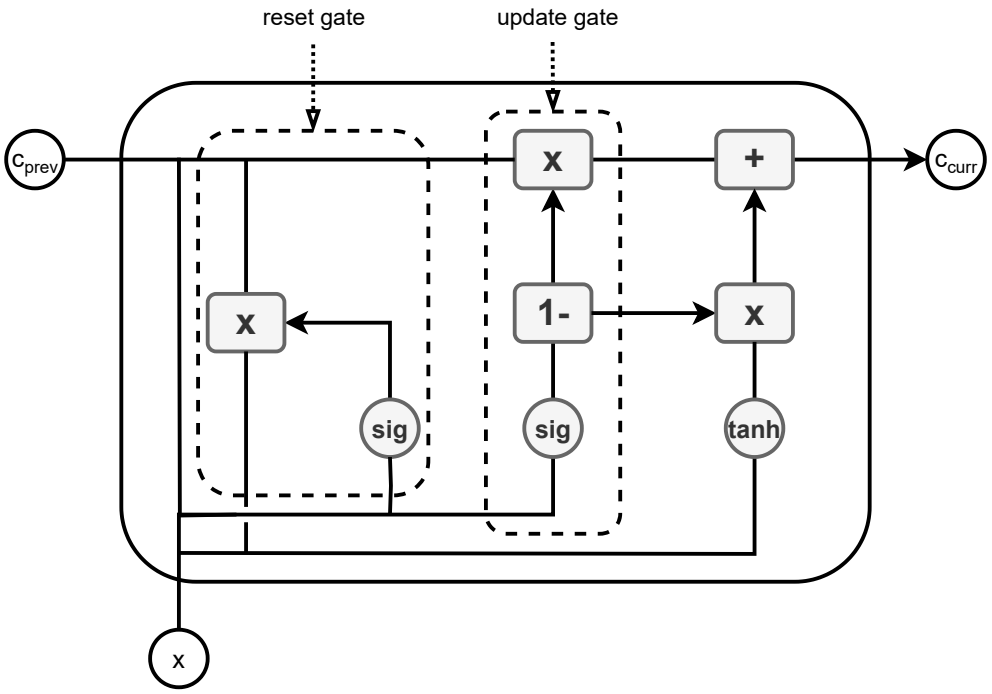

Figure 14 – Illustration of GRU. c_{prev} represents the previous state of the cell, c_{curr} represents the current state of the cell, and x represents the input. Adapted from (ZHAO et al., 2020).

- Convolutional layer: Responsible for feature extraction, it involves linear operations (convolution) and non-linear operations (activation function) (YAMASHITA et al., 2018).
  - Convolution: Involves applying a kernel, or filter, to an input sequence, also known as a "tensor". The kernel transforms the tensor into an output called a "feature map". A single convolutional layer can apply multiple kernels to the tensor, generating more than one feature map.
  - Activation function: The outputs from convolution pass through a non-linear activation function, such as sigmoid or hyperbolic tangent.
- Pooling layer: This layer reduces the dimensionality of the feature maps. Different types of pooling operations exist, such as "max pooling" (selecting the attribute with the highest value from the feature map) and "average pooling" (calculating the average value of the feature map).
- Fully connected layer: In this layer, the inputs are connected to the outputs of the convolutional and pooling layers. Classification or regression tasks are performed using an activation function (such as ReLU).

Although CNNs are primarily used for image classification and natural language processing tasks, they have also been applied to time series forecasting. (MEHTAB; SEN;

DASGUPTA, 2020) applied CNNs to forecast stock prices, and the results showed better performance compared to LSTMs. In (KOPRINSKA; WU; WANG, 2018), CNNs were used for forecasting electricity and solar time series. The results were compared with LSTMs and MLPs, and the CNNs were competitive with MLPs while outperforming LSTMs.

The CNN implemented in this study is known as CNN-LSTM (Figure 15), as it combines the architecture of a CNN with the architecture of an LSTM (Section 4.1.1). The conventional CNN layer is responsible for feature extraction from the input data, while the LSTM layer supports the prediction of time sequences.

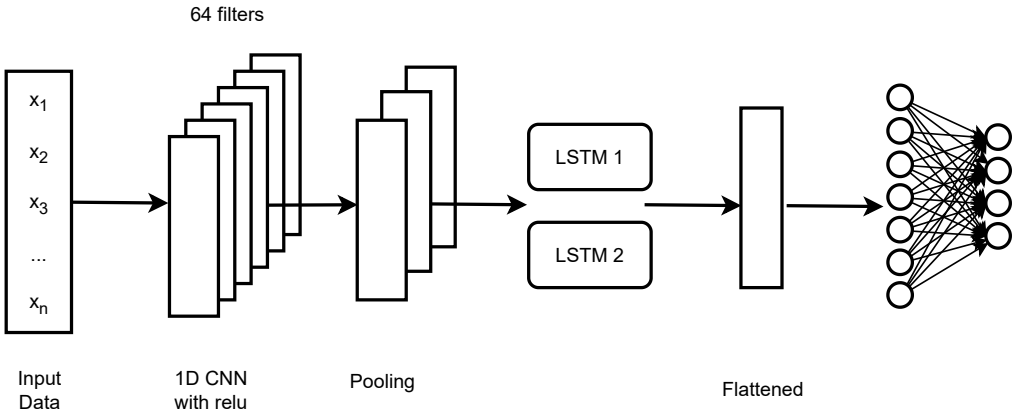

Figure 15 – The CNN-LSTM architecture implemented in this study. Adapted from (HAMAD et al., 2020).

Chapter 5

Genetic Algorithm (GA)

GA is a subfield of AI and was introduced in (HOLLAND, 1975), when its theoretical foundations and initial applications were first presented. The proposal is based upon the Darwin's natural selection process. That is, the individuals that fit best to a given environment survive, while the ones that fit less don't. Over time, the surviving individuals reproduce, generating an even stronger population (GANAPATHY, 2020).

In that sense, GAs are often used as a type of population-based metaheuristic algorithm, in which the goal is to train and evaluate multiple candidate solutions (individuals) in order to find the solution that optimizes a given (or a set of) complex problem(s) (KATOCH; CHAUHAN; KUMAR, 2021).

The GA field is evolving fast and hence there are different algorithms proposed. The Traditional GA is the main one, based on the initial proposal. Another algorithm is the NSGA-II, proposed by (DEB et al., 2002). It has been outperforming in different multi-objective tasks. Both the Traditional and NSGA-II algorithms are presented in the following sections.

5.1 Traditional

As discussed in the preceding section, GAs draw inspiration from Darwin's natural selection concept. To implement a Traditional GA algorithm successfully, a series of predetermined parameters and specifications must be set. The initial requirement entails establishing the makeup of an individual, essentially defining its attributes and how its quality is assessed. For instance, if we consider employing a genetic algorithm to tackle a scheduling problem, each individual within the population might represent a distinct schedule. In this context, an individual's attributes could encompass task sequencing,

resource allocation, and overall task completion time. The fitness (evaluation) function then gauges the schedule's merit based on specific criteria, such as minimizing completion time, maximizing resource utilization, or meeting designated deadlines.

Another important requirement is defining the population size, which determines how many individuals are present in each generation of the algorithm. The population size can significantly impact the algorithm's performance, as a larger population may explore a broader search space but can also require more computational resources.

Moreover, it is imperative to stipulate the genetic operations utilized, including selection, crossover (recombination) and mutation:

1. Selection: The mechanism by which individuals from the current generation are chosen to become parents for the next generation. It plays a pivotal role in shaping the genetic algorithm's evolutionary dynamics by favoring individuals with higher fitness values to pass their genetic information to the next generation;

    - ❑ A selection ratio must be defined. That is, the ratio or percentage of the best ranked individuals from the current population that will be selected and kept for the next generation;

2. Crossover: Entails combining attributes from two parental individuals to generate one or more offspring - a new individual or solution that is created through the process of reproduction;

3. Mutation: Introduces minor random alterations to an individual's attributes. These operations govern the transmission of genetic information from one generation to the subsequent one.

    - ❑ A mutation ratio must be defined. That is, the ratio or probability to which a mutation will or not occur.

Lastly, we must establish the termination criteria for the algorithm, which could include a maximum number of generations, a specific fitness threshold, or a time limit. These criteria help determine when the genetic algorithm should stop searching for a solution.

In summary, a Traditional GA requires: (1) defining the individual representation, (2) the fitness (evaluation) function, (3) population size, (4) genetic operators, and (5) termination criteria as fundamental requirements to guide the evolutionary process toward finding optimal or near-optimal solutions to a given problem. Algorithm 1 presents the Traditional GA pseudo-code.

Literature has shown that GAs offer a computationally efficient algorithm for solutions which include lots of hypeparameters. As mentioned in Chapter 1, (GANAPATHY, 2020) proposed using GAs as an automatic tuning method for neural networks. The GA

Algorithm 1 Traditional GA algorithm.

```
1: procedure TRADITIONAL_GA(population_size, nb_epochs, selection_ratio, mutation_ratio)
2:     population ← generate_random_initial_population(population_size)
3:     ranking ← evaluate_fitness(population)
4:
5:     for epoch in nb_epochs do
6:         parents ← selection(population, ranking, selection_ratio)
7:         offspring ← crossover(parents)
8:         offspring ← mutation(offspring, mutation_ratio)
9:         population ← parents + offspring
10:        ranking ← evaluate_fitness(population)
11:    end for
12:
13:    return best_individual(population)
14: end procedure
```

outperformed a random search of hyperparameters in a task of machine translation from Japanese to English. GA-based approaches have also found application in the fields of climate and environmental studies. In a study by (AMOL, 2020), a GA was employed to forecast the propagation of forest fires. The model utilized four key parameters: drought factor, temperature, relative humidity, and wind. It successfully identified specific parameter ranges that were indicative of fire initiation, allowing the model to accurately predict the spread category of affected areas. In a separate investigation conducted by (MATOS et al., 2022), GA was applied to address the resource scheduling problem in forest firefighting. The objective was to determine the optimal sequence of actions for firefighting resources when combating forest fires. Data for this study was collected from the Braga region in Portugal, and the results demonstrated the effectiveness and validity of this approach.

5.2 Nondominated Sorting Genetic Algorithm II (NSGA-II)

The NSGA-II algorithm, proposed by (DEB et al., 2002), is a type of Multi-Objective Evolutionary Algorithm (MOEA). One of the main differences between the Traditional GA and NSGA-II is that the last one is able to optimize for multiple objectives at once. The presence of multiple objectives that one wants optimize for may generate a set of optimal solutions - also known as Pareto-optimal solutions. A Pareto optimal solution represents a state in which no individual objective can be improved without at least one other objective. In other words, it's a solution that achieves the best possible trade-off among conflicting objectives without making any one objective better off at the expense of another, as illustrated in Figure 16.

Before NSGA-II, (SRINIVAS; DEB, 1995) presented the NSGA algorithm. Although it demonstrated good results, it was widely criticized due to its high computational com-

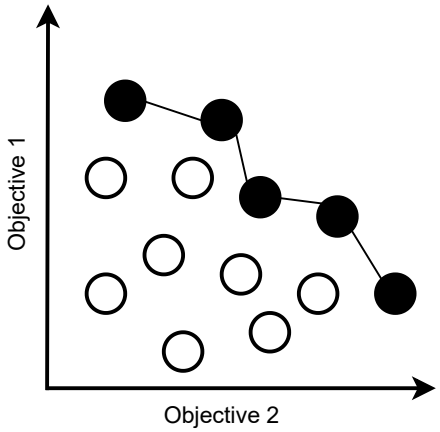

Figure 16 – Illustration of a set of Pareto optimal solutions (highlighted in black).

plexity and the lack of elitism. Elitism refers to a strategy where the best individuals from one generation are directly preserved and carried over to the next generation without any modification. In other words, the top-performing individuals are not subject to genetic operations.

Hence, NSGA-II comes as an improved version of the NSGA algorithm. NSGA-II introduces a more efficient and faster sorting algorithm. It uses a technique called "fast non-dominated sorting" that reduces the computational complexity of sorting individuals into fronts. The fast non-dominated sorting works as follows:

- ❏ Initially, for each solution p we calculate the number of solutions that dominates p. This is the domination count n_p;
- ❏ We also calculate S_p, which refers to the set of solutions that p dominates;
- ❏ In the first non-dominated front, are placed all solutions that a domination count of zero ($n_p = 0$);
- ❏ Subsequently, for each solution within this front, we conduct a "tournament" process. In this tournament, we randomly select a subset of solutions from S_p and decrement the domination count by one for each selected member.
- ❏ During this process, if the domination count of any member reaches zero, we categorize it into a distinct list. These members now constitute the second non-dominated front.
- ❏ We then repeat this procedure for every member in the new list, identifying the third non-dominated front in the process. This iterative sequence continues until all the fronts have been identified.

Notice that the "tournament" step is a critical part of NSGA-II's selection process, as it helps maintain diversity among solutions in each front. It does so by considering only a subset of solutions from S_p rather than the entire set, which prevents overly dominant solutions from dominating the entire front. This, in turn, leads to a more balanced and diverse set of solutions in the final Pareto front.

Another algorithm implemented by NSGA-II is the crowding-distance sorting, which is a critical component that promotes diversity within the population by assessing how well solutions are spread out across the Pareto front. The crowding distance measures the extent to which a solution is surrounded by other solutions in the objective space. To compute the crowding distance for each solution, NSGA-II considers each of the multiple objectives separately. First, it sorts the solutions within each front based on a particular objective. Then, for each solution, it calculates the difference between the objective values of the nearest neighbors. This difference represents the crowding distance for that solution in that specific objective dimension. The crowding distances are then summed across all objectives to obtain an overall crowding distance value for each solution. Solutions with larger crowding distances are given higher priority during selection, as they are considered to be more diverse and have not yet been surrounded by other solutions in the objective space.

Figure 17 represents the overall NSGA-II procedure, where P stands for population, Q stands for the generated offspring, and F_n is the n'th front. Algorithm 2 presents the NSGA-II pseudo-code.

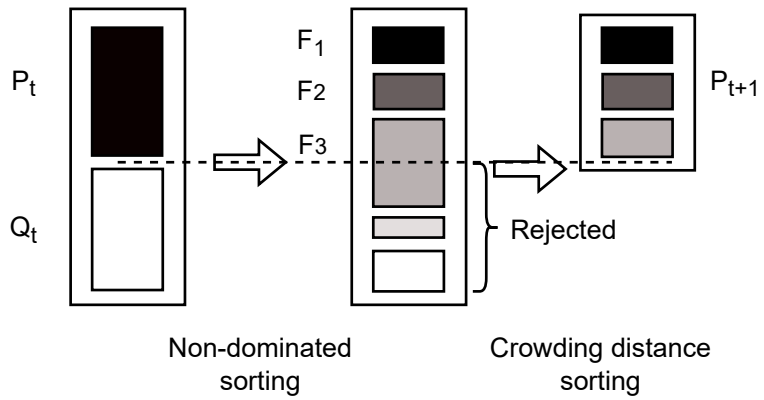

Figure 17 – Representation of the NSGA-II procedure. Adapted from (DEB et al., 2002).

Since its initial proposal, NSGA-II has been widely applied in different applications. In (YAHUI et al., 2020), NSGA-II algorithm was used for multi-objective flexible workshop scheduling, where the objective functions are the processing cycle, the advance/delay penalty and the processing cost. When compared to the Traditional GA and Particle Swarm algorithms, NSGA-II presented the best performance in terms of execution time, at the same time it was able to find the Pareto optimal solutions.

Algorithm 2 NSGA-II algorithm.

```
1: procedure NSGA_II(population_size, nb_epochs, selection_ratio, mutation_ratio)
2:     population ← generate_random_initial_population(population_size)
3:
4:     for epoch in nb_epochs do
5:         fronts ← fast_non_dominated_sorting(population)
6:         new_population ← null
7:         i ← 0
8:         while |new_population| + |fronts[i]| <= population_size do
9:             crowding_distances ← crowding_distance_assignment(fronts[i])
10:            new_population ← new_population + fronts[i]
11:            i ← i + 1
12:        end while
13:        fronts[i] ← crowding_distance_sorting(fronts[i], crowding_distances)
14:        new_population ← new_population + fronts[i][population_size −|new_population|]
15:        parents ← selection(new_population, selection_ratio)
16:        offspring ← crossover(offspring)
17:        offspring ← mutation(offspring, mutation_ratio)
18:        population ← parents + offspring
19:    end for
20:
21:    return best_individual(population)
22: end procedure
```

The study of (LI, 2022) also applied NSGA-II, but for maintenance decision-making of tunnel structures. It considered the maintenance cost and the structure condition as decision-making objectives. The NSGA-II was fused with a reverse chaotic map initialization, an adaptive crossover and an adaptive mutation – the authors named this as an NSGA-II improved version. It used the maximum spread as evaluation metric for representing the population diversity, and also the generational distance for representing the algorithm convergence. With a 0.963 maximum spread and a 0.047 generational distance, the NSGA-II improved version was able to effectively solve the Pareto frontier of the multi-objective maintenance problem.

In the context of environmental sciences, the NSGA-II has been applied as well. (GAO et al., 2020) presented an theoretical and experimental comparison of NSGA-II algorithm versions for sustainable land-use optimization. The study made use of different datasets regarding the city of Lhasa, Tibet. The three objectives were to maximize the ecological benefits, the economic benefits, and the spatial compactness. The different NSGA-II algorithms provided distinct solutions and the study discusses the trade-offs between them. (LIBERATI; RITTENHOUSE; VOKOUN, 2019) applied the NSGA-II for evaluating outcomes for US wildlife refuge expansion. The ecological objectives included maximizing total protected habitat, priority habitats, and connectivity between protected properties. It included economic and social objectives as well, such as minimizing acquisition cost, town character conflict and loss of areas under high development pressure. The algorithm identified solutions that met the 90% acquisition goals for the conservation focus areas.

Chapter 6

Forest Fire Risk Indexes

6.1 Monte Alegre Formula (FMA) and Modified Monte Alegre Formula (FMA$^+$)

The FMA is the main forest fire risk index developed based on the climatic conditions of Brazil. This index was initially proposed by (SOARES, 1972) and is based on the central region of Paraná, in Brazil, known as Monte Alegre. It includes variables such as relative humidity, which should be measured at 1 PM, and daily precipitation. It is an accumulative index, meaning it depends on the daily measurement of these climatic variables. The longer the sequence of days with low relative humidity and no rainfall, the higher the risk of forest fire. The FMA emphasizes the probability of ignition, that is, the likelihood of a forest fire starting. Its formula is described in Equation 11.

$$FMA = \sum_{i=1}^{n} \frac{100}{H_i}, \tag{11}$$

where n is the number of days without rainfall (precipitation less than 13 millimeters), and H is the relative humidity (%) measured at 1 PM.

The calculation is subject to adjustments based on the precipitation of the day, according to Table 1.

The value obtained through Equation 11 must then be converted to one of the five forest fire risk classes, which can be obtained from Table 2.

In (NUNES; SOARES; BATISTA, 2006), the FMA$^+$ was proposed, an updated version of FMA that includes the variable "wind speed" in the index calculation. The main objective was to create an index that also identifies the potential for the spread of forest fires, not just ignition. The FMA$^+$ is also an accumulative index, although the "wind

Daily Precipitation (mm)	Adjustment
≤ 2.4	None.
2.5 to 4.9	Subtract 30% from the FMA calculated the previous day and add (100/H) of the current day.
5.0 to 9.9	Subtract 60% from the FMA calculated the previous day and add (100/H) of the current day.
10.0 to 12.9	Subtract 80% from the FMA calculated the previous day and add (100/H) of the current day.
> 12.9	Stop the summation (FMA = 0) and start over the next day.

Table 1 – Adjustments in the calculation of FMA according to the precipitation. Adapted from (SOARES; BATISTA, 2007).

FMA	Forest Fire Risk
≤ 1.0	Null
1.1 to 3.0	Low
3.1 to 8.0	Medium
8.1 a 20.0	High
> 20.0	Very High

Table 2 – Forest Fire Risk classes according to the FMA. Adapted from (SOARES; BATISTA, 2007).

speed" variable is not – the wind speed value at 1 PM of each day is used. The index is subject to the same adjustments presented in Table 1 and can be calculated according to Equation 12.

$$FMA^{+} = \sum_{i=1}^{n} \left(\frac{100}{H_i} \right) \cdot e^{0,04 \cdot v_i}, \tag{12}$$

where n is the number of days without rain (precipitation less than 13 millimeters); H_i is the relative humidity (%) measured at 1 PM; and v is the wind speed (m/s) measured at 1 PM.

The value obtained from Equation 12 should then be converted into one of the five forest fire risk classes, which can be obtained from Table 3.

6.2 Telicyn

The Telicyn logarithmic index (I) was proposed in (TELICYN, 1970), and its calculation is described by Equation 13.

$$I = \sum_{i=1}^{n} log(T_i - r_i), \tag{13}$$

FMA$^+$	Forest Fire Risk
≤ 3.0	Null
3.1 to 8.0	Low
8.1 to 14.0	Medium
14.1 to 24.0	High
> 24.0	Very High

Table 3 – Forest Fire Risk classes according to the FMA$^+$. Adapted from (SOARES; BATISTA, 2007).

where n is the number of days without rain (precipitation less than 13 millimeters); T is the air temperature ($^\circ C$) at 1 PM; and r is the dew point temperature, which is the temperature at which the water vapor present in the air ($^\circ C$) changes to liquid state through condensation.

According to (SOARES; BATISTA, 2007), the dew point temperature (r) can be obtained from a table that correlates air temperature and relative humidity (Appendix A).

The calculation is subject to adjustments based on the occurrence of rainfall. These adjustments are described in Table 4.

Daily Precipitation (mm)	Adjustment
> 2.5	Stop the summation (I = 0) and start over the next day.

Table 4 – Adjustments in the calculation of Telicyn according to the precipitation. Adapted from (TORRES; RIBEIRO, 2008).

The value obtained through Equation 13 should then be converted to one of the four forest fire risk classes, which can be obtained from Table 5. Note that, unlike FMA and FMA$^+$ (Section 6.1), Telicyn does not include the "Very High" risk class.

I	Forest Fire Risk
≤ 2.0	Null
2.1 to 3.5	Low
3.6 to 5.0	Medium
> 5.0	High

Table 5 – Forest Fire Risk classes according to the Telicyn. Adapted from (ZICCARDI et al., 2020).

6.3 Angström

The fire danger index, or Angström formula (B), was developed in Sweden and is widely used in parts of Scandinavia (ANGSTROM, 1942). This index is calculated using Equation 14.

$$B = \left(\frac{H}{20}\right) + \left(\frac{T - 27}{10}\right), \tag{14}$$

where H represents the relative humidity (%) measured at 1 PM and T represents the air temperature ($°C$) at 1 PM.

This index does not require any specific adjustments and is not cumulative. The value obtained through Equation 14 should then be converted into one of the four forest fire risk classes, which can be obtained from Table 6.

B	**Forest Fire Risk**
< 3.5	Null
3.5 to 3.9	Low
4.0 to 4.2	Medium
4.3 to 4.5	High
> 4.5	Very High

Table 6 – Forest Fire Risk classes according to the Angström. Adapted from (CASAVECCHIA et al., 2019).

6.4 Nesterov

The Nesterov index was originally developed in the former Soviet Union (NESTEROV, 1949). It focuses on the concept of flammability, which provides an indication of the likelihood of forest fires. This index is based on the summation of "dangerous" days, i.e., days with a high saturation deficit. Equation 15 presents the calculation of the Nesterov index (G).

$$G = \sum_{i=1}^{n} d_i \cdot T_i, \tag{15}$$

where n represents the number of days without rainfall, d represents the saturation deficit (in millibars) at 1 PM, and T represents the air temperature (in degrees Celsius) at 1 PM.

The saturation deficit can be obtained using Equation 16.

$$d = E \cdot \left(1 - \frac{H}{100}\right), \tag{16}$$

where E represents the maximum vapor pressure (in millibars) and H represents the relative humidity of the air (%). According to (SOARES; BATISTA, 2007), the value of E can be obtained from the table correlating air temperature and maximum vapor pressure (see Appendix B).

The calculation is subject to adjustments based on precipitation, as it influences the flammability of vegetation. These adjustments are described in Table 7.

Daily Precipitation (mm)	**Adjusment**
≤ 2.0	None.
2.1 to 5.0	Subtract 25% from the G calculated the previous day and add $(d \cdot t)$ of the current day.
5.1 to 8.0	Subtract 50% from the G calculated the previous day and add $(d \cdot t)$ of the current day.
8.1 to 10.0	Discard the previous sum of G and start a new calculation. That is, $G = (d \cdot t)$ for the current day.
> 10.1	Stop the summation (G = 0) and start over the next day.

Table 7 – Adjustments in the calculation of Nesterov according to the precipitation. Adapted from (SOARES; BATISTA, 2007).

The value obtained through Equation 15 should then be converted into one of the five forest fire risk classes, which can be obtained from Table 8.

G	**Forest Fire Risk**
≤ 300	Null
301 a 500	Low
501 a 1000	Medium
1001 a 4000	High
> 4000	Very High

Table 8 – Forest Fire Risk classes according to Nesterov. Adapted from (SOARES; BATISTA, 2007).

6.5 Comparison of Indexes

The forest fire risk indexes presented earlier in this chapter are limited to one or more of the following four primary climatic variables: Precipitation (mm), relative humidity (%), air temperature ($^{\circ}C$), and wind speed (m/s). The relationship of these variables for each index is described in Table 9.

In addition to these indexes, there are others discussed and applied in the literature, but they may depend on other primary climatic variables. For example, the P-EVAP and

Index	Precipitation	Relative Humidity	Temperature	Wind Speed
FMA	X	X		
FMA$^+$	X	X		X
Telicyn	X	X	X	
Angström		X	X	
Nesterov	X	X	X	

Table 9 – Required climatic variables for each of the main Forest Fire Risk Indexes.

EVAP/P indexes can be mentioned (SAMPAIO, 1991). Both depend on precipitation (P) and evaporation (EVAP), in mm.

Another widely used index is the Fire Weather Index (FWI). It was developed in Canada in the 1970s and has been refined since then (WAGNER, 1987). It has different components, such as the moisture content of the organic layer and the drought index, which represents the soil moisture deficit.

In (TORRES; RIBEIRO, 2008), the FMA, FMA$^+$, P-EVAP, and EVAP/P indexes were applied to data from the Juiz de Fora region (Minas Gerais, Brazil). Data were collected at different times, and better results were observed when evaluating the relative humidity and temperature data measured at 3 PM. The indexes showed greater efficiency in predicting the absence of forest fires compared to predicting their occurrence. The index that performed best overall for the entire year was the EVAP/P index.

In (TORRES et al., 2017), the FMA, FMA$^+$, P-EVAP, EVAP/P, FWI, Nesterov, and Telicyn indexes were applied to data from the Viçosa region (Minas Gerais, Brazil). According to the results, the Telicyn index was the most efficient for the area in question, followed by the EVAP/P and P-EVAP indexes.

In the study by (TORRES; LIMA, 2019), the uses of the FMA, FMA$^+$, Nesterov, Telicyn, P-EVAP, EVAP/P, and FWI indexes were evaluated using data from the Serra do Brigadeiro State Park region (Minas Gerais, Brazil). The P-EVAP and FWI indexes were the most efficient in predicting forest fire occurrence in the studied region.

Regarding the Brazilian Pantanal region specifically, data from the Pantanal region in Mato Grosso do Sul were also used in a comparative study of different climatic indexes in (SORIANO; DANIEL; SANTOS, 2015). The FMA, FMA$^+$, Nesterov, Telicyn, and Angström indexes were evaluated. For detecting high-risk fire classes ("Very High" and "High") in data from 1999 to 2008, the Nesterov index was the most efficient, followed by the FMA index. Considering all classes, the FMA index had the highest accuracy.

Furthermore, still in the Brazilian Pantanal region there is a continuous effort of the environmental authorities to forecast and combat forest fires. An example is the SARIPAN system (NARCISO; SORIANO, 2019), which makes use of FMA, FMA$^+$, Nesterov, Telicyn and Angström indexes to identify forest fire risk. SARIPAN can only make predictions for the same day the climatic variables are collected - i.e., it is not able to forecast the forest fire risk for a number of days in the future. Also, it doesn't make use of ML,

meaning that it can't learn from historical data neither adapt to environmental changes in the climate.

The present study considers only the FMA, FMA$^+$, Nesterov, Telicyn, and Angström indexes. This is due to the fact that the measured climatic variables are limited to precipitation, relative humidity, air temperature, and wind speed – as described in Section 7.1.1. Therefore, the P-EVAP, EVAP/P, and FWI indexes were not considered for comparison with the results of the developed software.

6.6 Preventive Measures

In (SOARES; BATISTA, 2007), different preventive measures are proposed according to each forest fire risk class:

1. **Null**: It can be stated that there is no risk of forest fires occurrence. This period should be used for personnel training, activity planning, various maintenance tasks, and equipment review. Preventive surveillance can be demobilized. Command and surveillance towers do not need to be operational.

2. **Low**: There is a risk of forest fires, but this risk is low. This period should be used to intensify personnel training, activity planning, various maintenance tasks, and equipment review. Preventive surveillance can be reduced. Command and surveillance towers do not need to be operational.

3. **Medium**: The risk of forest fires is moderate. Control measures, such as firefighting teams and various equipment, should be ready and in a state of readiness. Vehicles and communication equipment should be turned on and tested daily. Command and surveillance towers begin operation.

4. **High**: There is a significant risk of forest fires. Control measures, such as firefighting teams and various equipment, should be ready and in a state of readiness. Agricultural and forestry operations that involve the use of fire should be monitored and restricted. Vehicles and communication equipment should be turned on and tested at least twice a day. Preventive surveillance should be intensified, which means extending the operating period of command and surveillance towers.

5. **Very High**: The risk of forest fires is extremely high. Control measures, such as firefighting teams and various equipment, should be ready and in a state of readiness. Agricultural and forestry operations that involve the use of fire should be suspended and prohibited. The population should be notified through communication channels. First response teams should be on standby for any emergencies. Preventive surveillance should be intensified, which means extending the operating period of command and surveillance towers.

Chapter 7

Materials and Methods

7.1 Datasets

As introduced in Chapter 1, the present study aims to classify the forest fire risk in the Brazilian Pantanal, using historical climatic variables from the region, and for a certain number of days in the future. To achieve this objective, two sets of data must be provided to the software: (1) Climatic Variables and (2) Hotspot Sources. These sets correspond to the inputs of time series forecasting and classification models, which will be discussed in detail in Section 8. The Climatic Variables dataset is the one that contains the climatic attributes for the region. The Hotspot dataset, on the other hand, is the one that contains the days on which hotspots were identified in the region.

7.1.1 Climatic Variables

The Climatic Variables dataset consists of a 23-year time series (1999-2022) of air temperature ($^{\circ}C$), relative air humidity (%), daily precipitation (mm), and wind speed (m/s) data. Air temperature, relative air humidity, and wind speed are measured daily at 1:00 PM (official Brasilia time, UTC-03). On the other hand, precipitation consists of the accumulated daily value. In total, the dataset contains 8,766 instances.

The data was collected at the main Climatological Station of Nhumirim (latitude 18°59'21"S, longitude 56°37'25"W, altitude 102 meters), located in the Nhecolândia sub-region of the southern Mato Grosso Pantanal (SORIANO; DANIEL; SANTOS, 2015). Table 10 summarizes the Climatic Variables dataset.

	Date	Temperature	Rel. Humidity	Precipitation	Wind Speed
Representation	Date	T	RH	P	WS
Measure	-	$°C$	%	mm	$m \cdot s^{-1}$
Data Type	Date	Numerical	Numerical	Numerical	Numerical
# Distinct Values	8766	274	112	401	14
Mean	-	31.11	57.70	2.97	3.37
Std	-	4.71	17.18	9.93	2.52
Min	01/01/1999	9.00	10.00	0.00	0.00
25%	-	28.90	45.25	0.00	2.00
50%	-	32.10	56.00	0.00	2.00
75%	-	34.30	69.00	0.00	4.00
Max	12/31/2022	42.00	100.00	203.60	20.00

Table 10 – Description of the Climatic Variables dataset.

7.1.2 Hotspot

The Hotspot dataset covers the period from January 1, 1999, to December 31, 2022 (23 years) - the same timeframe as the Climatic Variables dataset (Section 7.1.1). The data was collected from the database provided by the Image Processing Division of the Brazilian National Institute for Aerospace Research (from Portuguese, "*Instituto Nacional de Pesquisas Espaciais*") (INPE) (INPE, 2022). These data were generated by the NOAA12 satellite (up to 2002) and the AQUA_M-T satellite (from 2002).

Hotspots are detectable through spatial resolution elements (pixels) that highlight high temperatures. These present the lowest gray level values in the images of the thermal infrared region band 3 (3.7μm) of the Advanced Very High Resolution Radiometer (AVHRR) sensor. This band measures radiant energy emission from the Earth's surface, in which saturated pixels correspond to a temperature of at least $47°C$, normally associated with burning targets (SORIANO; DANIEL; SANTOS, 2015).

The data were filtered to eliminate the spots that were outside of the related coverage area, that is, a radius greater than 100 km and beyond the vertical line near the hills of the Nhumirim city station. Table 11 presents a snapshot of the mentioned data. From the hotspot dataset, there are 6059 days where no hotspot was identified (69.11%), against 2708 where it was (30.89%).

7.2 Evaluation Measure

The results were evaluated based on the correlation between the prediction of each forest fire risk class and the occurrence of heat sources. This correlation can be calculated according to Equation 17.

$$corr_C = \frac{|P_C \cap I|}{|P_C|}, \tag{17}$$

Date	Latitude	Longitude
01/22/1999	-19.1767	-55.9281
05/14/1999	-18.8581	-55.2281
...	...	...
08/14/2013	-19.357	-55.606
08/14/2013	-18.453	-56.482
...	...	...
12/29/2022	-19.009	-55.667
12/31/2022	-19.448	-56.616

Table 11 – Snapshot of the Hotspot dataset. Each row represents a date and location in which a hotspot was identified.

where $corr_C$ represents the correlation value of class C, ranging between 0 and 1; P_C corresponds to the set of days in which class C was predicted; and I denotes the set of days in which one or more heat sources were observed.

It is fair to assume that the lowest the correlation for classes “Null”, “Low” and “Medium”, the best. As well as it is desired to have the highest possible correlation for classes “High” and “Very High”.

The review of evaluation measures applied to forest fire risk (SORIANO; DANIEL; SANTOS, 2015; ZICCARDI et al., 2020; TORRES; RIBEIRO, 2008) shows there is no consensus on the use of a single evaluation measure. Also, it is not possible to apply the conventional ML evaluation measures, such as Accuracy and F-Measure, to evaluate our framework performance. This is because these measures rely on knowing the true values of the instances. In our study, however, the predictions are the forest fire risk classes, while the known true values are the occurrence of hotspots.

Given the described scenario and the context of this study, the correlation presented in Equation 17 was chosen as the evaluation measure, specially due to its simplicity and interpretability.

Chapter 8

Proposed Software

To achieve our goal of classifying the risk of forest fire occurrence in the Brazilian Pantanal, as stated earlier in Section 1.2, we propose the development of a software. This software may run different data pre-processing methods, as well as incorporate TS forecasting and classification layers.

The software consists of two main pipelines: Training and Prediction, both described in the following sections.

8.1 Training Pipeline

The software training pipeline can be described by Figure 18.

Each of these layers represent different software hyperparameters: (1) Scaling method; (2) Sampling method; (3) Forecasting algorithm; (4) Classification algorithm; (5) "Null" forest fire risk threshold (thr_{null}); (6) "Low" forest fire risk threshold (thr_{low}); (7) "Medium" forest fire risk threshold (thr_{medium}); and (8) "High" forest fire risk threshold (thr_{high}). There is no "Very High" threshold, because it simply corresponds to $1 - thr_{high}$, as is explained later in Section 8.1.1.

As presented in Chapters 2, 3 and 4, there are several methods and algorithms that can be applied to each layer, each with its own advantages and disadvantages. This wide range of possible software hyperparameters combinations makes it challenging to empirically and manually select a single combination of software hyperparameters. So, instead of manually picking a combination, we build the software so that it would choose the best combination itself. In that sense, there were incremental experiments that led to two different software versions:

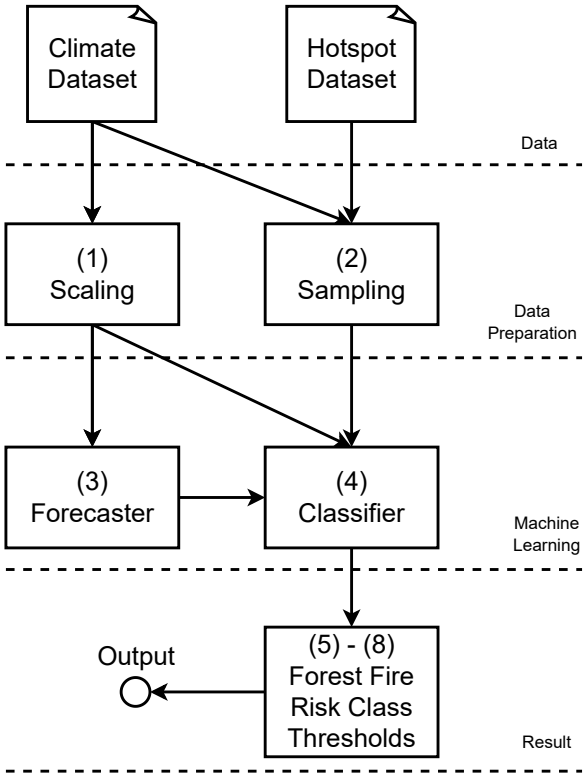

Figure 18 – Summary of the software training pipeline, including its different layers.

- ❏ Exhaustive Search version (v1): The software hyperparameters are searched in an exhaustive manner;

- ❏ GA version (v2): The software hyperparameters are chosen based on GA. In this version, it was tested both a Traditional GA, as well as the NSGA-II.

Table 12 summarizes the main differences between the versions.

Despite each version having its own implementation, they share common steps and approaches, as described in Section 8.1.1. Each version specificities are detailed in Sections 8.1.2 and 8.1.3.

8.1.1 Common Steps

The first common step between the two software versions refers to how the two datasets are transformed and merged into a single dataset. The combined dataset contains the climatic variables and a binary column that indicates whether a hotspot was identified (1) or not (0) in the corresponding day (Table 13).

Following common step is the data split into *training*, *validation* and *test*. To create such subsets, there are two sequential splits: 80% training plus validation (*training_validation*) and 20% *test*). The first is again splited into 80% *training* and 20% *validation*, which leads to a total split of 64%/16%/20% (Figure 19).

	Exhaustive Search (v1)	**Genetic Algorithm (v2)**
Software Hyperparameters	Scaling Method, Forecasting Algorithm, Classification Algorithm, Risk Class Thresholds	Same as v1, plus Sampling Method, Forecaster Batch Size and Forecaster Number of Units
Approaches for Software Hyperparameters Search	Exhaustive Search	GA and NSGA-II
Scaling Methods	Min-Max, Standard, Robust, Max-Abs and None	Min-Max and Max-Abs
Forecasting Algorithms	LSTM, GRU and CNN-LSTM	Same as v1
Sampling Methods	None	All presented in Section 2.2, and None
Classification Algorithms	MLP, SVM, KNN, Decision Tree, Random Forest, Naive Bayes, XGBoost, LightGBM and CatBoost	Same as v1, plus Logistic Regression

Table 12 – Main differences between the two software versions.

Date	**T**	**RH**	**P**	**WS**	**hotspot**
01/01/1999	29.7	79.0	2.0	0.0	0
01/02/1999	30.7	69.0	3.0	4.0	0
...	...	...	...	...	...
12/30/2022	33.5	61.0	2.0	0.0	0
12/31/2022	33.9	58.0	2.0	0.0	1

Table 13 – Snapshot of the combined dataset.

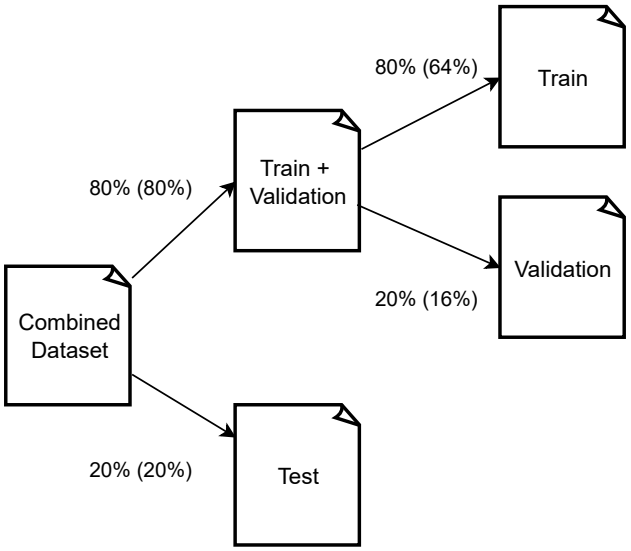

Figure 19 – Data split performed by the software.

During software hyperparameter selection, the models are trained upon the *training* subset, and the metrics are obtained based on the *validation* subset predictions. Once the best software hyperparameter combination is selected, the *training* and *validation* subsets are merged together (*training_validation*) to retrain the final model, and the results are reported based on the *test* subset predictions. The data is not shuffled, keeping the historical information ordered to be used by the forecaster later on.

Another common step is how the forest fire risk thresholds are defined. Since the classifiers outputs can be interpreted as the probability of occurring a hotspot, thresholds are needed to convert these outputs to one of the 5 forest fire risk classes. Upon applying the forest fire risk rate class thresholds, the probability for a given instance i is converted to a class C_i (Equation 18). Then, the correlation measure (Section 7.2) is calculated for each class ("Null", "Low", "Medium", "High" and "Very High"). This is performed for each day in the present or future ($0 \leq n \leq N$).

$$\begin{aligned} C_i = \{&\text{If } prob_i < thr_{null},\ i \in \text{Null or} \\ &thr_{null} \leq prob_i < thr_{low},\ i \in \text{Low or} \\ &thr_{low} \leq prob_i < thr_{medium},\ i \in \text{Medium or} \\ &thr_{medium} \leq prob_i < thr_{high},\ i \in \text{High else} \\ &i \in \text{Very High}\}, \end{aligned} \tag{18}$$

8.1.2 Software's Exhaustive Search Version (v1)

As mentioned in the beginning of this Chapter, the v1 chooses the best combination of software hyperparameters by exhaustive search.

For the scaling methods and forecasting algorithms, the software tests all possible

combinations, and selects the best one based on the lowest error obtained on *validation*. We started experimenting with Mean Absolute Error (MAE) because, when compared to other often-used error measures, is a more natural measure of average error and is unambiguous (WILLMOTT; MATSUURA, 2005). It is recommended for dimensioned evaluations and inter-comparisons of average model-performance errors, which is our case.

Nevertheless, when analysing preliminary results, we found that our Forecaster was not performing well in the "Precipitation" variable. This is due to the fact that this is a very sparse time series. In fact, as showed in Table 10, the p75% of the Precipitation value is equal to 0. In order to try to overcome such sparseness and improve our Forecaster, we decided to adopt the Mean Squared Logarithmic Error (MSLE) metric instead. The MSLE tends to penalize underestimates more than overestimates and treats small differences between small actual and predicted values as well as as big differences between large actual and predicted values (MASSMANN; HOLZMANN, 2012), which helps in the Precipitation sparseness.

After choosing the scaling method and forecasting algorithm, the classification algorithm is selected based on the highest F-Score.

The thresholds considered during the Forest Fire Risk Class Thresholds layer are based on a list of values. To select the best combination of thresholds, a score $S = \frac{S_1+S_2}{15}$ is calculated, with S_1 and S_2 given by Equations 19 and 20, where $corr_{n,c}$ is the correlation metric for a given class c in day n. The combination that provides the highest score S is chosen as the selected thresholds.

$$S_1 = \sum_{n=0}^{N} 5 \cdot (1 - corr_{n,null}) + 4 \cdot (1 - corr_{n,low}) + 3 \cdot (1 - corr_{n,medium}) \tag{19}$$

$$S_2 = \sum_{n=0}^{N} 2 \cdot corr_{n,high} + 1 \cdot corr_{n,very\_high} \tag{20}$$

As can be seen in Equation 19, we considered the difference $1 - corr_{n,class}$ for classes "Null", "Low" and "Medium". As explained in Section 7.2, $corr_C$ measures the correlation between the prediction of class C and the occurrence of a hotspot. We thus expect the lowest correlation possible, since there should be no detected hotposts when classes "Null", "Low" and "Medium" are predicted. The opposite is true for classes "High" and "Very High".

We can also see in Equation 19 that the correlations for "Null" and "Low" are multiplied by higher factors in the weighted average. Thus, they have a higher contribution to the final score S. This is because in our application, false negatives are a bigger concern compared to false positives. False negatives happen when the model predicts there will be no forest fire in a given day ("Null" and "Low"), but the forest fire occurs. False positives occur when the model predicts there will be forest fire in a given day ('High" and "Very High"), but the forest fire doesn't occur. It is thus clear that false negatives are a bigger

issue, since by predicting there will be no forest fire in a given day, the proper authorities may not be prepared to act upon the forest fire occurrence. This justifies our decision on weighting the correlations in a decreasing manner.

The general idea of the training pipeline for software Exhaustive Search version is summarized in Algorithms 3, 4, 5 and 6.

Algorithm 3 Software's Exhaustive Search version: Training pipeline.

```
procedure TRAIN(climate_data, hotspot_data)
    merged_data ← merge(climate_data, hotspot_data)
    training, validation, test ← split(merged_data)

    best_scaler, best_forecaster ← search_scaler_forecaster(training, validation)
    best_classifier ← search_classifier(best_scaler, training, validation)

    pred_present ← predict(best_scaler, best_classifier, validation)
    forescated_features ← predict(best_scaler, best_forecaster, validation)
    pred_future ← predict(best_scaler, best_classifier, forescated_features)

    best_thresholds ← search_thresholds(pred_present, pred_future)

    scaler ← refit(best_scaler, (training + validation))
    forecaster ← retrain(best_forecaster, (training + validation))
    classifier ← retrain(best_classificafier, (training + validation))

    final_pred_present ← predict(scaler, classifier, test)
    final_forescated_features ← predict(scaler, forecaster, test)
    final_pred_future ← predict(scaler, classifier, final_forescated_features)

    return apply(best_thresholds, final_pred_present, final_pred_future)
end procedure
```

Algorithm 4 Software's Exhaustive Search version: "Search Scaler and Forecaster" method.

```
procedure SEARCH_SCALER_FORECASTER(training, validation)
    best_msle ← +inf
    best_scaler ← NULL
    best_forecaster ← NULL
    for scaling_method in scaling_space do
        for forecasting_algorithm in forecasting_space do
            scaler ← fit(scaling_method, training)
            forecaster ← train(scaler, forecasting_algorithm, training)
            curr_msle ← predict(scaler, forecaster, validation)
            if curr_msle < best_msle then
                best_msle ← curr_msle
                best_scaler ← scaler
                best_forecaster ← forecaster
            end if
        end for
    end for
    return best_scaler, best_forecaster
end procedure
```

Algorithm 5 Software's Exhaustive Search version: "Search Classifier" method.

```
1: procedure SEARCH_CLASSIFIER(scaler, training, validation)
2:     best_fscore ← −inf
3:     best_classifier ← NULL
4:     for classification_algorithm in classification_space do
5:         classifier ← train(scaler, classification_algorithm, training)
6:         curr_fscore ← predict(scaler, classifier, validation)
7:         if curr_fscore > best_fscore then
8:             best_fscore ← curr_fscore
9:             best_classifier ← classifier
10:        end if
11:    end for
12:    return best_classifier
13: end procedure
```

Algorithm 6 Software's Exhaustive Search version: "Search Thresholds" method.

```
1: procedure SEARCH_THRESHOLDS(pred_present, pred_future)
2:     best_score ← −inf
3:     best_thresholds ← NULL
4:     for thresholds in thresholds_space do
5:         risk_rates ← apply(thresholds, pred_present, pred_future)
6:         correlations ← calculate_correlations(risk_rates)
7:         curr_score ← calculate_score(correlations)
8:         if curr_score > best_score then
9:             best_score ← curr_score
10:            best_thresholds ← thresholds
11:        end if
12:    end for
13:    return best_thresholds
14: end procedure
```

8.1.3 Software's Genetic Algorithm Version (v2)

The second version of the software applies GAs for searching the software hyperparameters space. It implements both the Traditional GA and the NSGA-II, so that the user can select which one to use. They are implemented as described in Algorithms 1 and 2, from Chapter 5. Additionally, the it adds the sampling layer, to handle the class imbalance.

Regarding the scaling method, for the software GA version we dropped the Standard and Robust methods. This is because later noticed that we need the features to be scaled between 0 and 1, so that the forecaster can make predictions (activation function) within this same range. This can only be achieved via Min-Max and Max-Abs scaling methods (Section 2.1).

The GA individual share the same hyperparameters between the Traditional GA and the NSGA-II, which are the same as presented earlier in this Section: (1) Scaling Method; (2) Forecasting Algorithm; (3) Sampling Method; (4) Classification Algorithm; (5) "Null" forest fire risk threshold (thr_{null}); (6) "Low" forest fire risk threshold (thr_{low}); (7) "Medium" forest fire risk threshold (thr_{medium}); and (8) "High" forest fire risk threshold (thr_{high}). Still, software GA version implements two additional hyperparameters,

related to the Forecaster architecture: (9) Batch Size and (10) Number of Units.

Also, while analyzing the software Exhaustive Search version preliminary results, we found that the scoring method initially implemented should be adapted for the software's GA version. This is because the score S, as presented in Equations 19 and 20, benefits low correlations for "Null" and "Low" classes, which is in fact desired. However, it ended up picking really low values for thr_{null} and thr_{low}. So low that there would be almost 0 instances predicted in such classes, leading to a correlation equal to 0 for both.

To overcome this, we updated the score S in the software GA version to also consider the number of instances predicted in each class c for a given day n ($count_{n,c}$). So, score becomes $S = S_1 + S_2 + S_3$, from Equations 21, 22 and 23.

$$S_1 = \sum_{n=0}^{N} 5 \cdot (1 - corr_{n,null}) \cdot (1 - e^{-1 \cdot count_{n,null}}) + 4 \cdot (1 - corr_{n,low}) \cdot (1 - e^{-1 \cdot count_{n,low}}) \quad (21)$$

$$S_2 = \sum_{n=0}^{N} 3 \cdot (1 - corr_{n,medium}) \cdot (1 - e^{-1 \cdot count_{n,medium}}) \quad (22)$$

$$S_3 = \sum_{n=0}^{N} 2 \cdot (1 - corr_{n,high}) \cdot (1 - e^{-1 \cdot count_{n,high}}) + 1 \cdot (1 - corr_{n,very\_high}) \cdot (1 - e^{-1 \cdot count_{n,very\_high}}) \quad (23)$$

As elucidated in Section 5.1, the Traditional GA algorithm requires a fitness function to enable the ranking of individuals. In software's GA version, this fitness function takes the form of the updated score S: A higher value of S means a superior evaluation of the individual's quality, yielding to a better ranking position.

For the NSGA-II in the other hand, there is no need to define a single fitness function (Section 5.2). In fact, it is implemented in way that can handle multi-objective tasks, such as this one. So, for the NSGA-II implementation, the goal is to keep the correlation value low for "Null", "Low" and "Medium" classes, while keeping the instances count assigned to each class as high as possible. This is taken into consideration to implement the Crowding Distance and Domination methods of the NSGA-II algorithm, as explained in Section 5.2. The representation of the GA individual is summarized in Table 14.

8.2 Prediction Pipeline

The prediction pipeline is common between the two different software versions and is described by Figure 20. It receives as input the climatic variables from the W previous days, with W being the observation window (Chapter 4).

Notice that, differently from the training pipeline, there is no sampling layer, as the class unbalance should only be handled during the classifier training phase.

Common	
Hyperparameters	Scaling Method, Forecasting Algorithm, Sampling Method, Classification Algorithm, Forest Fire Risk Thresholds, Forecaster Batch Size, and Forecaster Number of Units
Traditional GA	
Fitness Function	$S = S_1 + S_2 + S_3$, from Equations 21, 22 and 23
NSGA-II	
Objectives	(1) Correlations and (2) Instances Count per class

Table 14 – Software's GA version: Representation of the GA individual.

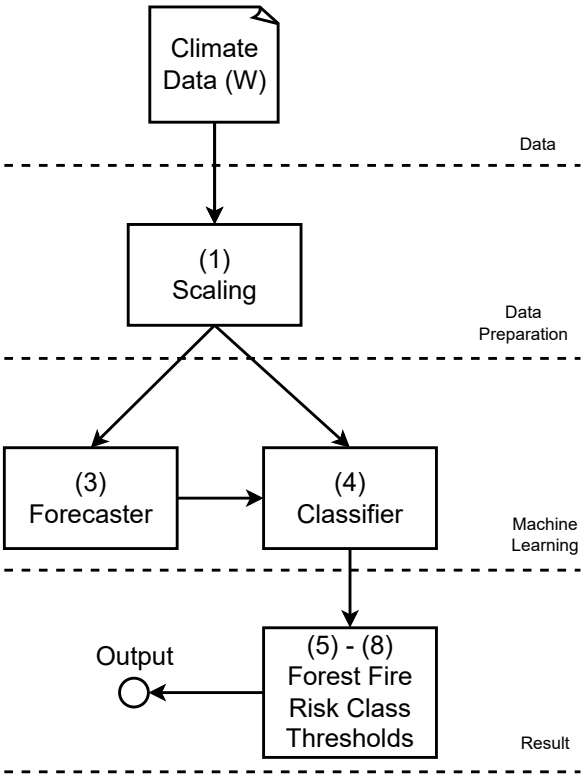

Figure 20 – Summary of the software prediction pipeline, including its different layers.

The output is the forecasted climate variables and predicted forest risk rate classes for the N, with N being the forecast horizon (Chapter 4).

8.3 Implementation

The software was developed using Python 3 programming language. Version control was managed using GitHub. Besides that, the software's GA version was developed so that it can be run inside a Docker container.

The main libraries used are detailed in Table 15. And the software Exhaustive Search version source code is available at <https://github.com/bzamith/IJCNN2023PantanalFireDetection>. The software GA version source code is not yet publicly available, as we are expecting to have it published by a journal or conference soon.

Library	**Application**	**Source**
Pandas	Data manipulation and analysis	<https://pandas.pydata.org/>
Imblearn	Data sampling	<https://imbalanced-learn.org/stable/>
Sktime	Time series prediction	<https://www.sktime.org/en/stable/>
Scikit-Learn	Supervised classification	<https://scikit-learn.org/stable/>
Tensorflow	Predictive analysis	<https://www.tensorflow.org/>

Table 15 – Main libraries used during the software implementation.

Chapter 9

Experiments and Results

This chapter details the set of experiments that were run and they respective results. For all experiments, we evaluated our software regarding its performance in predicting forest fire risk for up to 3 days in the future ($N = 3$). To do so, the forecasting algorithm is configured to make predictions based on a past 30 days window ($W = 30$). These numbers are arbitrarily chosen, and other values could be tested as well. Given that currently in Pantanal there are no methods to predict future forest fire risks, we consider that being able predict 3 days in the future is already an important innovation, at the same time allowing the proper authorities to become prepared and take actions when necessary.

Such experiments were executed in 3 different hardware configurations, summarized in Table 16.

	Machine 1	**Machine 2**	**Machine 3**
OS	Ubuntu	Windows 10	macOS Catalina
Processor	Intel Core i7 @ 2.80GHz	Intel Core i5 @ 1.70GHz	Intel Core i5 @ 2.30 GHz
RAM	16 Gb	16 Gb	8 Gb
GPU	NVIDIA GeForce GTX 1050	Intel UHD Graphics	Intel Iris Plus Graphics 640

Table 16 – Technical specifications of the machines used for the experiments.

9.1 Software's Exhaustive Search version

The first set of experiments were regarding the Exhaustive Search version (Section 8.1.2).

9.1.1 Scaling Methods

We considered a space of 5 scaling methods: Min-Max, Standard, Robust, Max-Abs, and None.

9.1.2 Forecasting Algorithms

We assessed 3 forecasting algorithms: LSTM, GRU, and CNN-LSTM. The implemented Long Short-Term Memory utilizes 4 LSTM layers with 100 units each and a single Dense layer employing the sigmoid activation function. Meanwhile, the implemented Gated Recurrent Unit architecture employs 4 GRU layers, also with 100 units each, and one Dense layer with the sigmoid activation function. The CNN-LSTM model, adapted from (HAMAD et al., 2020), incorporates a raw CNN layer for feature extraction (64 filters with ReLU activation) combined with LSTMs to facilitate sequence prediction.

All forecasting algorithms underwent training for a maximum of 250 epochs, with early stopping activated if no improvements in error were detected within a span of 5 consecutive epochs.

9.1.3 Sampling Methods

For Software's Exhaustive Search version, no sampling methods were implemented or subjected to testing.

9.1.4 Classification Algorithms

The classifiers under evaluation encompassed Decision Trees, Random Forests, XGBoost, LightGBM, CatBoost, Naive Bayes, KNN, MLP, and SVM. These classification algorithms underwent training employing a cross-validation approach with 3 folds to fine-tune classifier hyperparameters.

We opted for a modest fold count to ensure a substantial training dataset size. For the MLP and SVM algorithms, a maximum of 200 epochs was set as the training limit.

9.1.5 Forest Fire Risk Class Thresholds

In the quest for appropriate thresholds for each fire risk class, the software systematically assessed all conceivable combinations of values, spanning the range from 0 to 1 in increments of 0.05 (resulting in 21 distinct values). However, to consider

a combination of thresholds as valid, we required that the following inequality held: $thr_{null} < thr_{low} < thr_{medium} < thr_{high}$, thereby satisfying Equation 18.

9.1.6 Results

We ran the training pipeline 5 times, varying the seeds, in order to get the average results between executions, and their standard deviations.

A comprehensive execution entails the consideration of 5 scaling methods and 3 forecasting algorithms, resulting in a total of $5 \times 3 = 15$ combinations. Additionally, it involves 9 classification algorithms and the exploration of 194,481 distinct threshold combinations ($21 \times 21 \times 21 \times 21$). This culminates in a grand total of 194,505 iterations ($15+9+194,481$).

Table 17 shows the average correlations (Equation 17) across the executions, and in parallel, Table 18 shows the the count of examples predicted in each class and for each day, where "Angs." stands for "Angström" and "Nest." stands for "Nesterov". The best results are highlighted in bold face. Notice that the correlations for the present day for the Forest Risk Rate Indexes have a standard deviation equal to 0.0. This is expected since the results for such cases are deterministic - there is not forecasting and the Equations 11 to 15 are applied to the actual climate values.

	Software	**FMA**	**FMA$^+$**	**Telicyn**	**Angs.**	**Nest.**
			Present			
Null	**0.009 ± 0.021**	0.055 ± 0.000	0.096 ± 0.000	0.165 ± 0.000	0.288 ± 0.000	0.059 ± 0.000
Low	**0.036 ± 0.060**	0.144 ± 0.000	0.220 ± 0.000	0.258 ± 0.000	0.125 ± 0.000	0.196 ± 0.000
Medium	0.204 ± 0.114	0.234 ± 0.000	0.290 ± 0.000	0.348 ± 0.000	0.186 ± 0.000	**0.117 ± 0.000**
High	0.398 ± 0.112	0.292 ± 0.000	0.289 ± 0.000	**0.423 ± 0.000**	0.191 ± 0.000	0.294 ± 0.000
Very High	**0.528 ± 0.044**	0.444 ± 0.000	0.419 ± 0.000	-	0.379 ± 0.000	0.403 ± 0.000
			1 Day in the Future			
Null	**0.050 ± 0.112**	0.153 ± 0.090	0.108 ± 0.070	0.176 ± 0.099	0.200 ± 0.447	0.141 ± 0.085
Low	0.093 ± 0.142	0.113 ± 0.072	0.112 ± 0.069	0.260 ± 0.146	**0.060 ± 0.134**	0.093 ± 0.052
Medium	0.196 ± 0.130	0.136 ± 0.079	0.156 ± 0.092	0.253 ± 0.144	0.143 ± 0.086	**0.112 ± 0.074**
High	0.326 ± 0.191	0.238 ± 0.133	0.210 ± 0.120	**0.423 ± 0.057**	0.255 ± 0.042	0.179 ± 0.102
Very High	0.397 ± 0.224	**0.409 ± 0.061**	0.382 ± 0.042	-	0.314 ± 0.176	0.386 ± 0.039
			2 Days in the Future			
Null	**0.000 ± 0.000**	**0.000 ± 0.000**	0.018 ± 0.041	0.183 ± 0.106	**0.000 ± 0.000**	**0.000 ± 0.000**
Low	0.087 ± 0.140	0.160 ± 0.124	0.163 ± 0.100	0.223 ± 0.127	0.200 ± 0.447	**0.000 ± 0.000**
Medium	0.180 ± 0.165	0.160 ± 0.100	0.158 ± 0.104	0.239 ± 0.137	0.209 ± 0.129	**0.134 ± 0.101**
High	0.326 ± 0.191	0.190 ± 0.111	0.180 ± 0.107	**0.392 ± 0.044**	0.255 ± 0.041	0.162 ± 0.095
Very High	0.369 ± 0.259	**0.388 ± 0.041**	0.348 ± 0.026	-	0.334 ± 0.188	0.380 ± 0.035
			3 Days in the Future			
Null	**0.000 ± 0.000**	**0.000 ± 0.000**	0.075 ± 0.103	0.188 ± 0.108	0.200 ± 0.447	**0.000 ± 0.000**
Low	0.096 ± 0.143	0.097 ± 0.101	0.175 ± 0.104	0.264 ± 0.150	**0.050 ± 0.112**	0.067 ± 0.149
Medium	0.182 ± 0.167	0.166 ± 0.098	0.156 ± 0.093	0.291 ± 0.174	0.142 ± 0.131	**0.042 ± 0.094**
High	0.330 ± 0.201	0.226 ± 0.136	0.182 ± 0.121	**0.400 ± 0.052**	0.265 ± 0.040	0.176 ± 0.102
Very High	0.284 ± 0.179	**0.397 ± 0.052**	0.363 ± 0.045	-	0.341 ± 0.192	0.391 ± 0.045

Table 17 – Software's Exhaustive Search version: Correlations between each class and method.

To conduct a holistic assessment encompassing all days, we computed correlations by applying a weighted average across each class and aggregating results from all days, both

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
			Present			
Null	34.8 ± 77.8	128.0 ± 0.0	249.0 ± 0.0	538.0 ± 0.0	73.0 ± 0.0	185.0 ± 0.0
Low	80.8 ± 110.8	146.0 ± 0.0	214.0 ± 0.0	182.0 ± 0.0	56.0 ± 0.0	56.0 ± 0.0
Medium	773.4 ± 560.2	278.0 ± 0.0	193.0 ± 0.0	135.0 ± 0.0	86.0 ± 0.0	94.0 ± 0.0
High	452.2 ± 501.5	418.0 ± 0.0	187.0 ± 0.0	899.0 ± 0.0	340.0 ± 0.0	381.0 ± 0.0
Very High	412.8 ± 162.3	784.0 ± 0.0	911.0 ± 0.0	-	1199.0 ± 0.0	1038.0 ± 0.0
			1 Day in the Future			
Null	0.8 ± 1.8	16.0 ± 17.8	64.6 ± 54.6	640.4 ± 368.3	0.2 ± 0.4	55.6 ± 38.6
Low	400.6 ± 760.8	76.4 ± 50.8	157.6 ± 102.4	151.2 ± 84.1	3.6 ± 4.3	50.0 ± 29.9
Medium	743.4 ± 748.8	270.8 ± 163.8	149.6 ± 106.2	109.4 ± 61.7	46.8 ± 36.0	79.6 ± 49.2
High	516.6 ± 687.1	409.8 ± 236.3	143.2 ± 92.6	853.0 ± 513.1	939.6 ± 469.9	372.2 ± 225.2
Very High	92.6 ± 84.3	981.0 ± 461.5	1239.0 ± 352.5	-	763.8 ± 448.9	1196.6 ± 338.6
			2 Days in the Future			
Null	0.0 ± 0.0	0.0 ± 0.0	4.0 ± 4.2	618.6 ± 397.0	0.0 ± 0.0	0.2 ± 0.4
Low	390.4 ± 767.1	13.6 ± 10.7	130.2 ± 160.7	58.8 ± 34.7	0.2 ± 0.4	1.0 ± 1.2
Medium	810.4 ± 817.0	359.0 ± 273.2	106.4 ± 107.9	48.6 ± 27.4	48.2 ± 77.7	12.0 ± 9.0
High	511.6 ± 742.3	256.6 ± 152.2	79.4 ± 57.1	1028.0 ± 453.0	1059.2 ± 410.6	536.6 ± 346.1
Very High	41.6 ± 41.2	1124.8 ± 415.5	1434.0 ± 324.7	-	646.4 ± 403.4	1204.2 ± 351.7
			3 Days in the Future			
Null	0.0 ± 0.0	0.0 ± 0.0	6.6 ± 9.3	706.8 ± 448.2	0.2 ± 0.4	0.0 ± 0.0
Low	395.0 ± 765.7	19.2 ± 21.4	211.4 ± 253.9	57.8 ± 33.3	1.6 ± 3.6	0.8 ± 1.3
Medium	841.0 ± 836.0	458.2 ± 341.9	124.8 ± 94.7	39.0 ± 22.5	63.0 ± 129.2	7.8 ± 8.2
High	487.4 ± 750.1	231.0 ± 128.9	89.2 ± 62.1	950.4 ± 488.3	1127.2 ± 396.5	643.8 ± 422.0
Very High	30.6 ± 34.8	1045.6 ± 462.1	1322.0 ± 406.8	-	562.0 ± 392.5	1101.6 ± 425.4

Table 18 – Software's Exhaustive Search version: Number of predictions for each class and method.

present and future. Equation 24 elucidates the calculations, where $count_{n,C}$ represents the count of predictions for class C on the n'th day. The resulting weighted correlations for each class and method are summarized in Table 19.

$$weighted\_corr_C = \sum_{n=0}^{N} \frac{corr_{n,C} \cdot count_{n,C}}{count_C} \tag{24}$$

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
Null	**0.010 ± 0.023**	0.070 ± 0.019	0.106 ± 0.016	0.207 ± 0.031	0.291 ± 0.008	0.086 ± 0.020
Low	**0.087 ± 0.137**	0.144 ± 0.016	0.202 ± 0.029	0.287 ± 0.022	0.131 ± 0.024	0.161 ± 0.015
Medium	0.255 ± 0.103	0.208 ± 0.030	0.239 ± 0.027	0.334 ± 0.007	0.200 ± 0.027	**0.127 ± 0.020**
High	**0.428 ± 0.088**	0.280 ± 0.019	0.262 ± 0.021	0.405 ± 0.040	0.253 ± 0.039	0.240 ± 0.030
Very High	**0.518 ± 0.050**	0.405 ± 0.042	0.372 ± 0.031	-	0.395 ± 0.013	0.388 ± 0.031

Table 19 – Software's Exhaustive Search version: Weighted correlations for each class and method.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
Nulo	**0.010 ± 0.023**	0.070 ± 0.019	0.106 ± 0.016	0.207 ± 0.031	0.291 ± 0.008	0.086 ± 0.020
Baixo	**0.087 ± 0.137**	0.144 ± 0.016	0.202 ± 0.029	0.287 ± 0.022	0.131 ± 0.024	0.161 ± 0.015
Médio	0.255 ± 0.103	0.208 ± 0.030	0.239 ± 0.027	0.334 ± 0.007	0.200 ± 0.027	**0.127 ± 0.020**
Alto	**0.428 ± 0.088**	0.280 ± 0.019	0.262 ± 0.021	0.405 ± 0.040	0.253 ± 0.039	0.240 ± 0.030
Muito Alto	**0.518 ± 0.050**	0.405 ± 0.042	0.372 ± 0.031	-	0.395 ± 0.013	0.388 ± 0.031

As can be seen from Table 19, the Software outperformed all Forest Fire Risk Indexes for "Null", "Low", "High" and "Very High" classes. Still, the standard deviation for

"Low" and "Medium" classes are high (> 0.100) for the Software results, while it was more stable for the other 3 classes.

As explained in Section 8.1.2, this version conducts an exhaustive search across all possible combinations of hyperparameters and subsequently selects the optimal configuration based on specific metrics. Table 20 presents a summary of the chosen hyperparameter combinations resulting from the various executions.

Execution #	**Scaling Method**	**Forecasting Algorithm**	**Classification Algorithm**	thr_{null}	thr_{low}	thr_{medium}	thr_{high}
1	Max-Abs	CNN-LSTM	LightGBM	0.05	0.1	0.7	0.75
2	Max-Abs	CNN-LSTM	CatBoost	0.05	0.1	0.5	0.55
3	Max-Abs	CNN-LSTM	CatBoost	0.05	0.1	0.15	0.5
4	Max-Abs	CNN-LSTM	CatBoost	0.05	0.1	0.15	0.55
5	Max-Abs	CNN-LSTM	CatBoost	0.05	0.1	0.5	0.55

Table 20 – Software's Exhaustive Search version: Combination of hyperparameters selected by the software.

Notably, Max-Abs consistently emerged as the preferred scaling method in all 5 executions, alongside the CNN-LSTM for forecasting algorithm. In terms of classification algorithms, those based on the Symbolic Paradigm (as detailed in Section 3.1) were favored in all 5 executions. Specifically, CatBoost was the preferred choice in 4 executions, while LightGBM was selected once.

Finally, as forest fire risk rate thresholds, all executions selected the 0.05 as thr_{null} and 0.1 as thr_{low}, favoring the lowest thresholds possibles. As of the thr_{medium}, the results varied considerably, and for the thr_{high} is ranged between 0.5 and 0.75. We understand that this behaviour is due to the scoring S (Equations 19 and 20), which benefits low correlations for "Null" and "Low" classes. Although such low correlations are desired, the software ended up picking really low values for thr_{null} and thr_{low}. So low that there would be almost 0 instances predicted in such classes, leading to a correlation equal to 0 for both. This is validated from Table 18, in which for 2 and 3 days in the future, there were 0 instances predicted as "Null", for example. As mentioned in Section 8.1.3, we updated the score S for Software's GA version (Equations 21 to 23), looking forward to overcome this.

9.1.7 Computational Performance

Across all machines, the Training pipeline exhibited an average execution time of approximately 2 hours and 30 minutes, while the Prediction pipeline typically concluded within an average of 30 seconds. Ideally, the Training pipeline should be executed either once or whenever there is a need to retrain the model. On the other hand, the Prediction pipeline should be executed more frequently, as it is designed for regular usage.

9.2 Software's GA version - Traditional GA

The second set of experiments were regarding the Software's GA version, for the Traditional GA algorithm (Section 8.1.3).

9.2.1 Scaling Methods

We considered a space of 2 scaling methods: Min-Max and Max-Abs.

9.2.2 Forecasting Algorithms

We assessed 3 forecasting algorithms: LSTM, GRU, and CNN-LSTM. The implemented Long Short-Term Memory utilizes 4 LSTM layers with the number of units being set by the GA individual's hyperparameter, as well as the batch size. There is a single Dense layer employing the sigmoid activation function. Meanwhile, the implemented Gated Recurrent Unit architecture employs 4 GRU layers, also with the number of units being set by the GA individual's hyperparameter, as well as the batch size. There is one Dense layer with the sigmoid activation function. The CNN-LSTM model, adapted from (HAMAD et al., 2020), incorporates a raw CNN layer for feature extraction (64 filters with ReLU activation) combined with LSTMs to facilitate sequence prediction. The CNN-LSTM number of units and batch size are also determined by the GA individual's hyperparameters.

All forecasting algorithms underwent training for a maximum of 250 epochs, with early stopping activated if no improvements in error were detected within a span of 10 consecutive epochs.

9.2.3 Sampling Methods

There were 17 sampling methods considered, all explained in Section 2.2: Random Over Sampler, SMOTE, ADASYN, Borderline SMOTE, SVM-SMOTE, Random Under Sampler, Cluster Centroids, Near Miss, ENN, Repeated ENN, All KNN, One Sided Selection, Neighbourhood Cleaning Rule, Instance Hardness Threshold, SMOTE-ENN, SMOTE-Tomek and None.

9.2.4 Classification Algorithms

The classifiers under evaluation encompassed Decision Trees, Random Forests, XGBoost, LightGBM, CatBoost, Naive Bayes, KNN, MLP, SVM and Logistic Regression. Differently from Software's Exhaustive Search version, we did not apply cross-validation for the Classification Algorithms, to improve execution time.

9.2.5 Forest Fire Risk Class Thresholds

For the forest fire risk class thresholds, we randomly generate the thresholds for the initial population, and they are subject to mutation and crossover. Regarding the random generation, we set the following rules: (1) $0 < thr_{null} <= 0.2$; (2) $thr_{null} < thr_{low} <= 0.4$; (3) $thr_{low} < thr_{medium} <= 0.6$; (4) $thr_{medium} < thr_{high} <= 0.8$.

9.2.6 Forecaster Number of Units

There were 6 different values considered for the forecaster number of units: 20, 50, 100, 200 and 500.

9.2.7 Forecaster Batch Size

There were 6 different values considered for the forecaster number of units: 1, 4, 8, 32 and 64.

9.2.8 Results

To execute the GA, we set 20 epochs (generations), with a population size equal to 10 and an early stopping equal to 3. Besides that, we set both the mutation and selection ratio to 0.5. We ran the training pipeline 5 times, varying the seeds, in order to get the average results between executions, and their standard deviations.

Table 21 shows the average correlations (Equation 17) across the executions, and in parallel, Table 22 shows the the count of examples predicted in each class and for each day, where "Angs." stands for "Angström" and "Nest." stands for "Nesterov". The best results are highlighted in bold face. As mentioned in the previous section, it is expected that the correlations for the present day for the Forest Risk Rate Indexes have a standard deviation equal to 0.0.

As done for Software's Exhaustive Search version experiments (Section 9.1, we computed correlations by applying a weighted average across each class and aggregating results from all days, both present and future, as described in Equation 24. The resulting weighted correlations for each class and method are summarized in Table 23.

As can be seen from Table 23, the Software outperformed all Forest Fire Risk Indexes for "Very High" class, while being competitive with Telicyn for the "High" class. With the updated score S for Software's GA version (Equations 21 to 23), the distribution of predictions per class for Software predictions got way more balanced, as can be seem from Table 22. However, it did not improve for the Forest Fire Risk Indexes. In fact, FMA has almost 0 predictions for 2 and 3 Days in the Future when it comes to "Null" class, as well as Angström, favoring their low correlations.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
			Present			
Null	0.087 ± 0.044	**0.055 ± 0.000**	0.096 ± 0.000	0.165 ± 0.000	0.288 ± 0.000	0.059 ± 0.000
Low	0.172 ± 0.067	0.144 ± 0.000	0.220 ± 0.000	0.258 ± 0.000	**0.125 ± 0.000**	0.196 ± 0.000
Medium	0.291 ± 0.062	0.234 ± 0.000	0.290 ± 0.000	0.348 ± 0.000	0.186 ± 0.000	**0.117 ± 0.000**
High	0.374 ± 0.055	0.292 ± 0.000	0.289 ± 0.000	**0.423 ± 0.000**	0.191 ± 0.000	0.294 ± 0.000
Very High	**0.505 ± 0.048**	0.444 ± 0.000	0.419 ± 0.000	-	0.379 ± 0.000	0.403 ± 0.000
			1 Day in the Future			
Null	0.115 ± 0.107	0.334 ± 0.388	0.164 ± 0.049	0.227 ± 0.021	**0.080 ± 0.179**	0.181 ± 0.025
Low	0.217 ± 0.148	0.129 ± 0.024	0.131 ± 0.056	0.344 ± 0.053	**0.107 ± 0.147**	0.127 ± 0.077
Medium	0.273 ± 0.119	0.186 ± 0.041	0.207 ± 0.062	0.347 ± 0.074	**0.079 ± 0.072**	0.130 ± 0.038
High	0.356 ± 0.139	0.294 ± 0.052	0.273 ± 0.082	**0.437 ± 0.016**	0.228 ± 0.026	0.225 ± 0.043
Very High	**0.509 ± 0.179**	0.430 ± 0.040	0.390 ± 0.039	-	0.386 ± 0.021	0.407 ± 0.026
			2 Days in the Future			
Null	0.136 ± 0.127	**0.000 ± 0.000**	0.103 ± 0.101	0.225 ± 0.035	**0.000 ± 0.000**	0.050 ± 0.112
Low	0.232 ± 0.166	0.107 ± 0.098	0.120 ± 0.100	0.296 ± 0.108	**0.022 ± 0.050**	0.059 ± 0.081
Medium	0.274 ± 0.133	0.171 ± 0.105	0.237 ± 0.137	0.300 ± 0.097	0.100 ± 0.092	**0.095 ± 0.096**
High	**0.443 ± 0.134**	0.228 ± 0.147	0.233 ± 0.157	0.409 ± 0.045	0.237 ± 0.046	0.185 ± 0.113
Very High	0.240 ± 0.225	**0.393 ± 0.046**	0.362 ± 0.046	-	0.392 ± 0.025	0.379 ± 0.034
			3 Days in the Future			
Null	0.149 ± 0.139	0.200 ± 0.447	0.227 ± 0.178	0.201 ± 0.116	**0.000 ± 0.000**	0.133 ± 0.298
Low	0.263 ± 0.165	0.129 ± 0.118	0.191 ± 0.110	0.349 ± 0.110	**0.000 ± 0.000**	0.256 ± 0.433
Medium	0.283 ± 0.126	0.189 ± 0.115	0.194 ± 0.140	0.257 ± 0.146	**0.063 ± 0.087**	0.191 ± 0.117
High	**0.467 ± 0.301**	0.236 ± 0.149	0.218 ± 0.146	0.409 ± 0.059	0.261 ± 0.048	0.199 ± 0.119
Very High	0.292 ± 0.185	**0.410 ± 0.068**	0.374 ± 0.057	-	0.398 ± 0.039	0.388 ± 0.046

Table 21 – Software's GA version, Traditional GA: Correlations between each class and method.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
			Present			
Null	335.6 ± 185.4	128.0 ± 0.0	249.0 ± 0.0	538.0 ± 0.0	73.0 ± 0.0	185.0 ± 0.0
Low	91.6 ± 69.0	146.0 ± 0.0	214.0 ± 0.0	182.0 ± 0.0	56.0 ± 0.0	56.0 ± 0.0
Medium	450.8 ± 357.1	278.0 ± 0.0	193.0 ± 0.0	135.0 ± 0.0	86.0 ± 0.0	94.0 ± 0.0
High	367.0 ± 268.2	418.0 ± 0.0	187.0 ± 0.0	899.0 ± 0.0	340.0 ± 0.0	381.0 ± 0.0
Very High	509.0 ± 263.9	784.0 ± 0.0	911.0 ± 0.0	-	1199.0 ± 0.0	1038.0 ± 0.0
			1 Day in the Future			
Null	354.0 ± 354.0	9.6 ± 9.3	58.6 ± 50.4	849.4 ± 213.7	1.8 ± 2.5	48.8 ± 27.3
Low	117.8 ± 76.2	93.6 ± 76.0	224.8 ± 160.5	162.2 ± 35.4	4.6 ± 7.1	56.6 ± 31.9
Medium	415.4 ± 392.2	379.2 ± 174.8	177.8 ± 87.4	121.2 ± 18.7	52.2 ± 68.0	93.2 ± 39.0
High	320.0 ± 410.3	489.4 ± 80.7	154.2 ± 68.6	621.2 ± 191.4	628.0 ± 149.7	538.4 ± 154.6
Very High	546.8 ± 715.4	782.2 ± 289.8	1138.6 ± 356.3	-	1067.4 ± 201.1	1017.0 ± 249.5
			2 Days in the Future			
Null	437.0 ± 450.3	0.2 ± 0.4	34.8 ± 48.0	739.0 ± 442.0	0.0 ± 0.0	1.2 ± 1.8
Low	126.6 ± 77.4	81.6 ± 103.7	249.4 ± 304.8	69.0 ± 18.7	2.4 ± 3.9	5.8 ± 7.1
Medium	389.4 ± 354.6	393.4 ± 316.0	92.0 ± 74.3	63.2 ± 16.7	67.2 ± 114.3	48.6 ± 53.9
High	274.0 ± 380.6	241.4 ± 143.2	68.2 ± 52.8	882.8 ± 455.3	705.4 ± 203.3	567.2 ± 392.0
Very High	527.0 ± 729.8	1037.4 ± 507.1	1309.6 ± 471.4	-	979.0 ± 301.7	1131.2 ± 444.6
			3 Days in the Future			
Null	545.8 ± 557.4	0.2 ± 0.4	80.2 ± 112.0	803.4 ± 540.8	0.0 ± 0.0	0.6 ± 1.3
Low	145.2 ± 90.6	145 ± 188.1	327.0 ± 344.7	49.6 ± 42.3	1.2 ± 2.7	4.0 ± 7.9
Medium	321.2 ± 282.4	465.6 ± 339.2	96.0 ± 62.2	43.4 ± 28.2	97.2 ± 176.2	87.0 ± 107.7
High	241.8 ± 328.9	180.2 ± 115.8	71.8 ± 48.0	857.6 ± 567.1	784.2 ± 239.1	651.4 ± 475.8
Very High	500.0 ± 715.1	963.0 ± 554.0	1179.0 ± 538.6	-	871.4 ± 376.3	1011.0 ± 548.0

Table 22 – Software's GA version, Traditional GA: Number of predictions for each class and method.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
Null	0.146 ± 0.091	**0.069 ± 0.016**	0.121 ± 0.021	0.219 ± 0.030	0.286 ± 0.008	0.085 ± 0.012
Low	0.250 ± 0.106	0.155 ± 0.023	0.203 ± 0.024	0.304 ± 0.040	**0.130 ± 0.007**	0.165 ± 0.026
Medium	0.287 ± 0.088	0.217 ± 0.036	0.263 ± 0.034	0.336 ± 0.039	**0.168 ± 0.010**	0.139 ± 0.017
High	0.380 ± 0.054	0.289 ± 0.045	0.282 ± 0.052	**0.413 ± 0.030**	0.237 ± 0.037	0.245 ± 0.035
Very High	**0.471 ± 0.089**	0.413 ± 0.040	0.380 ± 0.036	-	0.385 ± 0.016	0.391 ± 0.027

Table 23 – Software's GA version, Traditional GA: Weighted correlations for each class and method.

Table 24 presents a summary of the chosen hyperparameter combinations resulting from the various executions.

Execution #	**Scaling Method**	**Forecasting Algorithm**	**Sampling Method**	**Classification Algorithm**	thr_{null}
1	Min-Max	CNN-LSTM	Borderline SMOTE	CatBoost	0.051
2	Max-Abs	LSTM	ADASYN	MLP	0.162
3	Min-Max	CNN-LSTM	SMOTE-ENN	CatBoost	0.124
4	Max-Abs	CNN-LSTM	One-Sided Selection	LightGBM	0.070
5	Min-Max	GRU	None	CatBoost	0.136

Execution #	thr_{low}	thr_{medium}	thr_{high}	**Number of Units**	**Batch Size**
1	0.065	0.445	0.485	100	4
2	0.276	0.403	0.735	20	1
3	0.186	0.205	0.620	200	64
4	0.089	0.499	0.794	100	64
5	0.191	0.404	0.628	100	32

Table 24 – Software's GA version, Traditional GA: Combination of hyperparameters selected by the software.

Regarding the scaling method, there is not an outstanding method, which suggests that the choice between the Min-Max or Max-Abs methods does not play an important role in the overall results. The same applies to the sampling method, in which there was not a "winner" method. For the forecasting algorithm, the CNN-LSTM was favored in 3 of the 5 executions. The same is true for the CatBoost as classification algorithm.

It is not possible to identify an outstanding number of units or batch size. The selected hyperparameters for them varied considerably. Finally, as forest fire risk rate thresholds, the values varied a lot. One significant aspect is that, for Execution #1, there is a very low range between thr_{null} and thr_{low}, and the same for thr_{medium} and thr_{high}. This means that there were very low instances predicted as "Low" and "High", which explains the high standard deviations exposed for in Table 22.

9.2.9 Computational Performance

Across all machines, the Training pipeline exhibited an average execution time of approximately 16 hours, while the Prediction pipeline typically concluded within an average

of 1 minute. Ideally, the Training pipeline should be executed either once or whenever there is a need to retrain the model. On the other hand, the Prediction pipeline should be executed more frequently, as it is designed for regular usage.

9.3 Software's GA version - NSGA-II

The third set of experiments were regarding the Software's GA version, for the NSGA-II algorithm (Section 8.1.3).

The space of hyperparameters are the same as of the Traditional GA, detailed in Sections 9.2.1 to 9.2.7.

9.3.1 Results

As it was set for the Traditional GA (Section 9.2.8), the executions run 20 epochs (generations), with a population size equal to 10 and an early stopping equal to 3. Besides that, we set both the mutation and selection ratio to 0.5. Regarding the NSGA-II's tournament step (Section 5.2, we considered 5 participants.

We ran the training pipeline 5 times, varying the seeds, in order to get the average results between executions, and their standard deviations.

Table 25 shows the average correlations (Equation 17) across the executions, and in parallel, Table 26 shows the the count of examples predicted in each class and for each day, where "Angs." stands for "Angström" and "Nest." stands for "Nesterov". The best results are highlighted in bold face. As mentioned in the previous section, it is expected that the correlations for the present day for the Forest Risk Rate Indexes have a standard deviation equal to 0.0.

As done for the two previous experiments, we computed correlations by applying a weighted average across each class and aggregating results from all days, both present and future, as described in Equation 24. The resulting weighted correlations for each class and method are summarized in Table 27.

As can be seen from Table 27, the Software outperformed all Forest Fire Risk Indexes for "Very High" class, while being competitive with Telicyn for the "High" class, and with FMA for the "Null" class. Regarding the standard deviation between the executions, the NSGA-II algorithm demonstrated to be way more stable when compared to Traditional GA and Software's Exhaustive Search version.

Table 28 presents a summary of the chosen hyperparameter combinations resulting from the various executions.

Regarding the scaling method, there is not an outstanding method, which suggests that the choice between the Min-Max or Max-Abs methods does not play an important role in the overall results. The same applies to the sampling method, in which there was

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
			Present			
Null	0.099 ± 0.047	**0.055 ± 0.000**	0.096 ± 0.000	0.165 ± 0.000	0.288 ± 0.000	0.059 ± 0.000
Low	0.141 ± 0.071	0.144 ± 0.000	0.220 ± 0.000	0.258 ± 0.000	**0.125 ± 0.000**	0.196 ± 0.000
Medium	0.281 ± 0.044	0.234 ± 0.000	0.290 ± 0.000	0.348 ± 0.000	0.186 ± 0.000	**0.117 ± 0.000**
High	0.384 ± 0.033	0.292 ± 0.000	0.289 ± 0.000	**0.423 ± 0.000**	0.191 ± 0.000	0.294 ± 0.000
Very High	**0.528 ± 0.053**	0.444 ± 0.000	0.419 ± 0.000	-	0.379 ± 0.000	0.403 ± 0.000
			1 Day in the Future			
Null	0.063 ± 0.088	0.226 ± 0.189	0.111 ± 0.084	0.244 ± 0.043	**0.000 ± 0.000**	0.143 ± 0.085
Low	0.288 ± 0.403	0.112 ± 0.078	0.104 ± 0.062	0.260 ± 0.147	**0.083 ± 0.114**	0.088 ± 0.060
Medium	0.190 ± 0.118	0.188 ± 0.076	0.209 ± 0.065	0.269 ± 0.155	0.133 ± 0.085	**0.120 ± 0.073**
High	0.321 ± 0.076	0.238 ± 0.134	0.208 ± 0.120	**0.346 ± 0.194**	0.186 ± 0.105	0.248 ± 0.047
Very High	0.353 ± 0.208	0.338 ± 0.189	0.306 ± 0.171	-	**0.368 ± 0.027**	0.317 ± 0.177
			2 Days in the Future			
Null	0.020 ± 0.045	**0.000 ± 0.000**	0.027 ± 0.060	0.236 ± 0.050	**0.000 ± 0.000**	0.040 ± 0.089
Low	0.123 ± 0.119	0.027 ± 0.060	0.060 ± 0.083	0.256 ± 0.150	**0.000 ± 0.000**	0.025 ± 0.056
Medium	0.196 ± 0.126	0.196 ± 0.071	0.149 ± 0.129	0.238 ± 0.153	0.191 ± 0.155	**0.022 ± 0.050**
High	**0.337 ± 0.086**	0.188 ± 0.109	0.137 ± 0.127	0.317 ± 0.177	0.187 ± 0.105	0.208 ± 0.064
Very High	0.336 ± 0.197	0.309 ± 0.173	0.270 ± 0.151	-	**0.377 ± 0.032**	0.302 ± 0.169
			3 Days in the Future			
Null	**0.000 ± 0.000**	**0.000 ± 0.000**	0.025 ± 0.056	0.256 ± 0.037	**0.000 ± 0.000**	**0.000 ± 0.000**
Low	0.130 ± 0.121	0.104 ± 0.151	0.121 ± 0.079	0.241 ± 0.152	**0.000 ± 0.000**	**0.000 ± 0.000**
Medium	0.212 ± 0.139	0.206 ± 0.07	0.207 ± 0.082	0.275 ± 0.161	0.194 ± 0.229	**0.027 ± 0.061**
High	**0.353 ± 0.104**	0.224 ± 0.126	0.186 ± 0.138	0.316 ± 0.177	0.199 ± 0.112	0.243 ± 0.047
Very High	0.304 ± 0.175	0.312 ± 0.175	0.274 ± 0.154	-	**0.379 ± 0.036**	0.309 ± 0.173

Table 25 – Software's GA version, NSGA-II: Correlations between each class and method.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
			Present			
Null	190.6 ± 110.1	128.0 ± 0.0	249.0 ± 0.0	538.0 ± 0.0	73.0 ± 0.0	185.0 ± 0.0
Low	191.4 ± 168.7	146.0 ± 0.0	214.0 ± 0.0	182.0 ± 0.0	56.0 ± 0.0	56.0 ± 0.0
Medium	519.6 ± 376.3	278.0 ± 0.0	193.0 ± 0.0	135.0 ± 0.0	86.0 ± 0.0	94.0 ± 0.0
High	475.2 ± 302.7	418.0 ± 0.0	187.0 ± 0.0	899.0 ± 0.0	340.0 ± 0.0	381.0 ± 0.0
Very High	377.2 ± 165.9	784.0 ± 0.0	911.0 ± 0.0	-	1199.0 ± 0.0	1038.0 ± 0.0
			1 Day in the Future			
Null	12.8 ± 16.3	7.0 ± 4.7	34.2 ± 21.7	973.0 ± 441.5	0.8 ± 1.8	39.6 ± 22.9
Low	99.6 ± 122.1	54.0 ± 33.1	139.8 ± 77.0	149.8 ± 85.2	3.8 ± 6.1	43.6 ± 26.2
Medium	474.8 ± 483.4	608.0 ± 640.7	478.2 ± 709.9	117.2 ± 65.8	39.2 ± 25.9	77.0 ± 42.8
High	608.2 ± 697.2	455.4 ± 255.2	128.6 ± 72.1	514.0 ± 291.3	498.2 ± 281.7	742.8 ± 565.1
Very High	558.6 ± 609.2	629.6 ± 356.7	973.2 ± 544.4	-	1212.0 ± 305.3	851.0 ± 477.1
			2 Days in the Future			
Null	2.2 ± 4.4	0.0 ± 0.0	4.0 ± 6.2	877.0 ± 496.9	0.0 ± 0.0	1.0 ± 2.2
Low	105.0 ± 159.7	12.0 ± 18.6	41.2 ± 53.6	71.2 ± 43.4	0.6 ± 1.3	1.6 ± 3.6
Medium	542.6 ± 565.2	541.4 ± 685.5	394.0 ± 757.2	55.4 ± 34.2	35.6 ± 29.5	7.6 ± 10.9
High	586.2 ± 702.5	300.0 ± 176.3	33.8 ± 23.0	750.4 ± 426.3	595.2 ± 335.8	738.6 ± 569.1
Very High	518.0 ± 661.2	900.6 ± 508.7	1281.0 ± 721.2	-	1122.6 ± 355.3	1005.2 ± 566.0
			3 Days in the Future			
Null	0.8 ± 1.3	0.0 ± 0.0	4.0 ± 6.7	963.4 ± 443.3	0.0 ± 0.0	0.0 ± 0.0
Low	148.4 ± 242.9	11.6 ± 17.8	77.0 ± 79.0	62.0 ± 42.0	0.0 ± 0.0	0.0 ± 0.0
Medium	570.8 ± 623.0	656.2 ± 619.4	410.6 ± 747.9	51.6 ± 32.1	48.4 ± 46.8	5.2 ± 9.4
High	566.0 ± 698.3	273.2 ± 171.7	55.8 ± 31.9	677.0 ± 383.4	679.4 ± 382.2	907.4 ± 473.9
Very High	468.0 ± 664.8	813.0 ± 455.7	1206.6 ± 679.7	-	1026.2 ± 411.5	841.4 ± 471.9

Table 26 – Software's GA version, NSGA-II: Number of predictions for each class and method.

	Software	FMA	FMA$^+$	Telicyn	Angs.	Nest.
Null	0.099 ± 0.048	**0.065 ± 0.008**	0.100 ± 0.007	0.233 ± 0.041	0.285 ± 0.007	0.080 ± 0.012
Low	0.174 ± 0.034	0.141 ± 0.010	0.181 ± 0.018	0.290 ± 0.027	**0.129 ± 0.005**	0.162 ± 0.028
Medium	0.279 ± 0.019	0.210 ± 0.061	0.239 ± 0.049	0.338 ± 0.028	0.214 ± 0.071	**0.127 ± 0.016**
High	0.360 ± 0.035	0.281 ± 0.015	0.267 ± 0.031	**0.412 ± 0.006**	0.225 ± 0.021	0.246 ± 0.042
Very High	**0.456 ± 0.077**	0.414 ± 0.018	0.375 ± 0.026	-	0.374 ± 0.025	0.393 ± 0.007

Table 27 – Software's GA version, NSGA-II: Weighted correlations for each class and method.

Execution #	**Scaling Method**	**Forecasting Algorithm**	**Sampling Method**	**Classification Algorithm**	thr_{null}
1	Min-Max	CNN-LSTM	Random Under Sampler	SVM	0.090
2	Min-Max	CNN-LSTM	Random Over Sampler	Logistic Regression	0.162
3	Max-Abs	CNN-LSTM	ADASYN	Random Forest	0.040
4	Min-Max	CNN-LSTM	Boderline SMOTE	CatBoost	0.070
5	Max-Abs	LSTM	Random Over Sampler	CatBoost	0.083

Execution #	thr_{low}	thr_{medium}	thr_{high}	**Number of Units**	**Batch Size**
1	0.130	0.578	0.713	50	4
2	0.276	0.403	0.735	50	1
3	0.149	0.524	0.582	20	32
4	0.088	0.499	0.794	500	64
5	0.356	0.390	0.782	20	64

Table 28 – Software's GA version, NSGA-II: Combination of hyperparameters selected by the software.

not a "winner" method. For the forecasting algorithm, the CNN-LSTM was favored in 4 of the 5 executions. As of the classification algorithm, the tree-based methods (CatBoost and Random Forest) outperformed in 3 of the 5 executions.

It is not possible to identify an outstanding number of units or batch size. The selected hyperparameters for them varied considerably. Finally, as forest fire risk rate thresholds, the values varied a lot. Compared to the Traditional GA results, the ranges of the forest fire risk rate thresholds are better distributed, but there are still high standard deviations for the number of instances predicted in each class, as showed on Table 26.

9.3.2 Computational Performance

Across all machines, the Training pipeline exhibited an average execution time of approximately 23 hours and 30 minutes, while the Prediction pipeline typically concluded within an average of 1 minute. Ideally, the Training pipeline should be executed either once or whenever there is a need to retrain the model. On the other hand, the Prediction pipeline should be executed more frequently, as it is designed for regular usage.

9.4 Discussion

When it comes to the weighted correlations, the Software's Exhaustive Search version yield to the best results for "Null", "Low", "High" and "Very High" classes against all forest fire risk indexes, and also when compared with Software's GA version (both Traditional and NSGA-II). On the other hand, though, the Software's Exhaustive Search version had considerably higher standard deviations, meaning that it lacks stability across different executions.

The Software's GA version with the Traditional GA only outperforms the forest fire rate indexes for the "Very High" class, while being competitive for the "Null" and "High classes". The same is true for the NSGA-II. However, the Software's GA version with the NSGA-II improved in terms of stability. It has an average of 0.043 standard deviation across the five classes (Table 27), which is lower than the Software's Exhaustive Search version (0.080 average standard deviation - Table 19) and the Software's GA version with the Traditional GA (0.086 average standard deviation - Table 23). This makes sense given the NSGA-II algorithm, which is exploring the optimal hyperaparmeters based on the Pareto fronts (Section 5.2).

Although the Software GA version already requires 10 times more execution time for training than the Software Exhaustive Search version, our understanding is that the GA would benefit from more epochs and a even higher population size. In fact, the study of (MAN; TANG; KWONG, 1996) explains that undesired premature convergence of a GA may occur, which is called "Genetic Drift", and that this can easily happen when the GA is set with a small population size - which is our case, for the population size being equal to 10 individuals. The authors of (ULLAH; MASOOD, 2023) also explored genetic drift and its effects on the performance of GAs. It concludes that the negative effects of genetic drift are particularly perceptible in small populations, leading the GA to become slow or to provide a locally optimal solution. The same is true for small number of generations - which is our case, for the number of epochs being equal to 20.

In summary, the results seem promising as they meet the general objective of this work (Section 1.2). This is, to develop software containing ML models that are capable of: (1) Forecasting climatic variables for a specific number of days; and (2) Classifying the risk of forest fire occurrence in the Brazilian Pantanal. Both software versions also overcome the known limitations for the current forest fire risk indexes, which are: (1) Not adjusting to the characteristics of each biome; (2) Being limited to specific climatic variables; and (3) Not being able to predict forest fire risk for a given number of days.

Given its average performance, we recommend the use of the Software's Exhaustive Search version. At the same time, we understand that further studies are required for the GA version, as long as there are more computational resources available.

9.5 Other Achievements

Besides the software development itself, this study achieved another remarkable result: Being accepted and published by the IEEE's 2023 International Joint Conference on Neural Networks (IJCNN). The paper presented the initial results obtained from experiments with Software's Exhaustive Search version, but with some slightly different set up. The paper is entitled "A New Time Series Framework for Forest Fire Risk Forecasting and Classification" (SANTOS et al., 2023).

Moreover, there is an ongoing effort to include the Software's forecasting and predictions to the existing SARIPAN system (NARCISO; SORIANO, 2019). SARIPAN is currently being used by the environmental authorities and the Brazilian Agricultural Research Corporation (from Portuguese, "*Empresa Brasileira de Pesquisa Agropecuária*") (Embrapa) to detect and combat forest fires in the Brazilian Pantanal region. By integrating our Software with SARIPAN, there is the potential to improve its forest fire risk predictions, with the advantage of being able to forecast for a number of days in the future.

Chapter 10

Conclusion

This study was undertaken with the goal of developing Machine Learning (ML) models capable of achieving two primary objectives: (1) Forecasting climatic variables for a specific number of days; and (2) Classifying the risk of forest fire occurrence in the Brazilian Pantanal. This endeavor is of utmost urgency, given the Pantanal biome's immeasurable ecological significance, which is constantly threatened by the occurrence and recurrence of forest fires.

To create this software, climate and hotspot data spanning from 1999 to 2022 were preprocessed and utilized to train the ML models. Moreover, this studied extensively explored different methods for Scaling and Sampling, as well as algorithms for Classification, Forecasting and GAs. The exploration of a high-dimensional hyperparameter space was required. Consequently, two distinct software versions were developed: one employing an Exhaustive Search approach and the other integrating a Genetic Algorithm (GA). Both versions effectively overcame the recognized limitations of existing forest fire risk indexes, which include: (1) Not adjusting to the characteristics of each biome; (2) Being limited to specific climatic variables; and (3) Not being able to predict forest fire risk for a given number of days.

The performance of both software versions was assessed based on the average correlation between forest fire risk classifications and observed hotspots. Overall, the Software's Exhaustive Search version outperformed the current forest fire risk indexes for the "Null", "Low", "High" and "Very High" risk classes. On the other hand, the GA version of the software exhibited greater stability across multiple executions, although it did not achieve the same level of success as the Exhaustive Search version. This discrepancy is attributed to the software's tendency to converge to a local optimal solution due to genetic drift. The limitation in computational resources appears to be a contributing factor, and with

increased computational resources, this issue could likely be mitigated.

We highlight the significance of this practical scientific study for the Brazilian Pantanal, and it is worth noting that the software is already in the process of integration into the existing SARIPAN system (NARCISO; SORIANO, 2019). We anticipate that this software has the potential to empower environmental authorities to make informed and proactive decisions, ultimately benefiting both the environment and the populace at large. Furthermore, the software boasts adaptability and can readily be extended to various biomes, provided that historical climate and hotspot data are accessible.

As part of our forthcoming research, we intend to delve deeper into the GA methods, with the aim of surmounting the current computational limitations. One promising avenue to explore involves the application of surrogate-assisted evolutionary algorithms, which leverage meta-models to approximate the fitness function within the evolutionary algorithm. This approach holds the potential to address the genetic drift issue that we have identified.

Bibliography

ABBES, A. B.; MAGAGI, R.; GOITA, K. Soil moisture estimation from smap observations using Long Short-Term Memory (LSTM). 2019.

AHSAN, M. M. et al. Effect of data scaling methods on machine learning algorithms and model performance. **Technologies**, v. 9, n. 3, 2021. ISSN 2227-7080.

AKTER, R.; LEE, J. M.; KIM, D. S. Analysis and prediction of hourly energy consumption based on long short-term memory neural network. **International Conference on Information Networking (ICOIN)**, p. 732–734, 2021.

AL-ZEBDA, A. K. et al. Predicting forest fires using meteorological data: an ANN approach. In: . [S.l.: s.n.], 2021. v. 5, p. 51–57. ISSN 2643-9026.

ALHO, C. J. R. et al. Ameaças à biodiversidade do Pantanal Brasileiro pelo uso e ocupação da terra. **Ambiente & Sociedade**, v. 22, 2019.

ALI, H. et al. A review on data preprocessing methods for class imbalance problem. p. 390–397, 2019.

ALIBRAHIM, H.; LUDWIG, S. A. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In: **2021 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2021. p. 1551–1559.

ALPAYDIN, E. **Introduction to Machine Learning**. [S.l.]: Adaptive Computation and Machine Learning MIT Press, 2014.

ALVARES, C. A. et al. Perigo de incêndio florestal: aplicação da fórmula de monte alegre e avaliação do histórico para Piracicaba, SP. **Scientia Forestalis**, v. 42, n. 104, p. 521–532, 2014.

AMBARWARI, A.; ADRIAN, Q. J.; HERDIYENI, Y. Analysis of the effect of data scaling on the performance of the machine learning algorithm for plant identification. **Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)**, v. 4, n. 1, p. 117–122, 2020.

AMOL, D. Prediction of fire propagation in forest areas using genetic algorithm. **ITM Web of Conferences**, v. 32, 2020.

ANGSTROM, A. **Riskerna for skogsbrand och deras beroende av vader och klimat [“The risks for forest fires and their relation to weather and climate”]**. [S.l.]: Svenska Skogsvirdsforeningens Tidskrift, 1942. v. 40. 323–343 p.

BATISTA, G.; BAZZAN, A. L. C.; MONARD, M. C. Balancing training data for automated annotation of keywords: a case study. In: **WOB**. [S.l.: s.n.], 2003. p. 10–18.

BATISTA, G.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. **ACM Sigkdd Explorations Newsletter**, v. 6, n. 1, p. 20–29, 2004.

BRYSON, A. E.; HO, Y. C. **Applied Optimal Control**. [S.l.: s.n.], 1969.

BUDA, M.; MAKI, A.; MAZUROWSKI, M. A. A systematic study of the class imbalance problem in convolutional neural networks. **Neural Networks**, v. 106, p. 249–259, 2018. ISSN 0893-6080.

BURKOV, A. **The Hundred-Page Machine Learning Book**. [S.l.]: Andriy Burkov, 2019. ISBN 9781999579517.

CASAVECCHIA, B. H. et al. Índices de perigo de incêndios em uma área de transição cerrado-amazônia. **Revista de Ciências Agrárias**, v. 42, n. 3, 2019.

CERRI, R. **Técnicas de Classificação Hierárquica Multirrótulo**. Dissertação (Dissertação de Mestrado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC USP, 2010.

CHAWLA, N. V. et al. SMOTE: Synthetic minority over-sampling technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002.

CHEN, H. et al. Improved naive bayes classification algorithm for traffic risk management. **EURASIP Journal on Advances in Signal Processing**, v. 2021, n. 1, 2021.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. **ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 785—794, 2016.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

CRISTIANINI, N.; SHAWE-TAYLOR, J. **An Introduction to Support Vector Machines and Other Kernel-based Learning Methods**. [S.l.]: Cambridge University Press, 2000.

CUNNINGHAM, P.; DELANY, S. J. k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples). **Computer Vision and Pattern Recognition (CoRR)**, 2020.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.

DIETTERICH, T. G. Ensemble methods in machine learning. In: **Multiple Classifier Systems**. [S.l.]: Springer Berlin Heidelberg, 2000. p. 1–15.

DIMRI, T.; AHMAD, S.; SHARIF, M. Time series analysis of climate variables using seasonal arima approach. **Journal of Earth System Science**, v. 129, n. 1, p. 149, 2020.

GANAPATHY, K. A study of genetic algorithms for hyperparameter optimization of neural networks in machine translation. **Computer Vision and Pattern Recognition (CoRR)**, 2020.

GAO, C.; LIN, H.; HU, H. Forest-fire-risk prediction based on random forest and backpropagation neural network of heihe area in heilongjiang province, china. **Forests 2023**, v. 14, n. 2, 2023.

GAO, P. et al. Sustainable land-use optimization using NSGA-II: theoretical and experimental comparisons of improved algorithms. **Landscape Ecology**, Springer Science and Business Media LLC, v. 36, n. 7, p. 1877–1892, 2020.

HAMAD, R. A. et al. Joint learning of temporal models to handle imbalanced data for human activity recognition. **Applied Sciences**, v. 10, n. 15, 2020.

HAN, H.; WANG, W.; MAO, B. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: . [S.l.: s.n.], 2005. p. 878–887.

HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: . [S.l.: s.n.], 2008. p. 1322–1328.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. [S.l.]: University of Michigan Press, 1975.

INPE. **Queimadas: monitoramento de focos**. 2022. <https://queimadas.dgi.inpe.br/queimadas/bdqueimadas>. Acesso em: 14 de Abril de 2022.

JOHNSON, J.; KHOSHGOFTAAR, T. The effects of data sampling with deep learning and highly imbalanced big data. **Information Systems Frontiers**, v. 22, 2020.

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. **Multimedia Tools and Applications**, v. 80, n. 5, p. 8091–8126, 2021.

KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. **Advances in neural information processing systems**, v. 30, p. 3146–3154, 2017.

KOPRINSKA, I.; WU, D.; WANG, Z. Convolutional neural networks for energy time series forecasting. In: . [S.l.: s.n.], 2018. p. 1–8.

KUBAT, M.; MATWIN, S. Addressing the curse of imbalanced training sets: one-sided selection. In: **ICML**. [S.l.: s.n.], 1997. v. 97, p. 179–186.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In: **Springer Berlin Heidelberg**. [S.l.: s.n.], 2001.

LEMAITRE, G.; NOGUEIRA, F.; ARIDAS, C. K. **Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning**. 2016.

LEWIS, D. D. Naive (bayes) at forty: The independence assu mption in information retrieval. **European Conference on Machine Learning (ECML)**, p. 4–15, 1998.

LI, Z. Multi-objective maintenance decision-making of tunnel structure based on improved nsga-ii. In: **2022 2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)**. [S.l.: s.n.], 2022. p. 410–414.

LIBERATI, M. R.; RITTENHOUSE, C. D.; VOKOUN, J. C. Addressing ecological, economic, and social tradeoffs of refuge expansion in constrained landscapes. **Landscape Ecology**, v. 34, p. 627–647, 2019.

LIN, X. et al. Forest fire prediction based on long- and short-term time-series network. **Forests**, v. 14, n. 4, 2023.

LUO, Y. et al. Forest fire detection using spiking neural networks. In: . [S.l.: s.n.], 2018. p. 371—375.

MAN, K.; TANG, K.; KWONG, S. Genetic algorithms: concepts and applications [in engineering design]. **IEEE Transactions on Industrial Electronics**, v. 43, n. 5, p. 519–534, 1996.

MASSMANN, C.; HOLZMANN, H. Analysing goodness of fit measures using a sensitivity based approach. **General Assembly Conference Abstracts, pp. 12354**, p. 12354–, 04 2012.

MATOS, M. A. et al. A genetic algorithm for forest firefighting optimization. In: **Computational Science and Its Applications – ICCSA 2022 Workshops: Malaga, Spain, July 4–7, 2022, Proceedings, Part II**. [S.l.]: Springer-Verlag, 2022. p. 55—-67.

MEHTAB, S.; SEN, J.; DASGUPTA, S. Robust analysis of stock price time series using CNN and LSTM-based deep learning models. In: . [S.l.]: IEEE, 2020.

MITCHELL, T. M. **Machine Learning**. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

MOYANO, J. M. et al. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. **Information Fusion**, v. 44, p. 33–45, 2018. ISSN 1566-2535.

MUKADI, P. M.; GONZÁLEZ-GARCÍA, C. Time series analysis of climatic variables in peninsular spain. trends and forecasting models for data between 20th and 21st centuries. **Climate**, v. 9, n. 7, 2021. ISSN 2225-1154.

MURAT, M. et al. Forecasting daily meteorological time series using arima and regression models. **International Agrophysics**, v. 32, n. 2, p. 253–264, 2018. ISSN 0236-8722.

NARCISO, M.; SORIANO, B. Saripan - sistema de avaliação de risco de incêndio para o pantanal. **Congresso Brasileiro de Agroinformática**, 2019.

NESTEROV, V. G. Combustibility of the forest and methods for its determination (in russian). **USSR State Industry Press**, 1949.

NUNES, J. R. S. et al. Desempenho da fórmula de monte alegre (FMA) e da fórmula de monte alegre alterada (FMA+) no distrito florestal de monte alegre. **Revista Floresta**, 2010. ISSN 1982-4688.

NUNES, J. R. S.; SOARES, R. V.; BATISTA, A. C. **FMA+ - Um novo índice de perigo de incêndios florestais para o o estado do Paraná, Brasil**. [S.l.]: Revista Floresta, 2006. ISSN 1982-4688.

ONIGEMO, A. E. **Avaliação de índices de risco de incêndio em áreas com predominância de gramíneas cespitosas na sub-região da Necholândia, Pantanal, MS**. Tese (Doutorado em Ecologia) — Programa de Pós-graduação em Ecologia e Conservação, Universidade Federal do Mato Grosso do Sul, 2007.

P., P. P.; VANITHA, V.; R, R. Wind speed forecasting using long short term memory networks. **2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)**, p. 1310–1314, 2019.

PARGENT, F. et al. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. **Computational Statistics**, 2022.

PARK, H.-A. An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain. **Journal of Korean Academy of Nursing**, v. 43, p. 154–164, 04 2013.

PARMEZAN, A. R.; SOUZA, V. M. A.; BATISTA, G. E. A. P. A. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-art and the best conditions for the use of each model. **Information Sciences**, v. 484, p. 302–337, 2019.

POPESCU, M. et al. Multilayer perceptron and neural networks. **WSEAS Transactions on Circuits and Systems**, v. 8, 2009.

PROKHORENKOVA, L. et al. Catboost: unbiased boosting with categorical features. **Conference on Neural Information Processing Systems (NeurIPS)**, 2018.

QUINLAN, J. R. Induction of decision trees. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 1, n. 1, p. 81—106, 1986. ISSN 0885-6125.

_____. **C4.5: Programs for Machine Learning**. [S.l.]: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0.

RAKSHIT, P. et al. Prediction of forest fire using machine learning algorithms: The search for the better algorithm. In: **2021 6th International Conference on Innovative Technology in Intelligent System and Industrial Applications (CITISIA)**. [S.l.: s.n.], 2021. p. 1–6.

ROSENBLATT, F. **The Perceptron, a Perceiving and Recognizing Automaton**. [S.l.]: Cornell Aeronautical Laboratory, 1957.

RUBÍ, J. N.; CARVALHO, P. H. de; GONDIM, P. R. Application of machine learning models in the behavioral study of forest fires in the brazilian federal district region. **Engineering Applications of Artificial Intelligence**, v. 118, p. 105649, 2023.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Pearson, 2010.

SAMPAIO, O. B. **Estudo Comparativo de Índice, para Previsão de Incêndios Florestais, na Região de Coronel Fabriciano, Minas Gerais**. Dissertação (Dissertação de Mestrado) — Departamento de Engenharia Florestal. Universidade Federal de Vićosa, 1991.

SANTOS, B. Z. et al. A new time series framework for forest fire risk forecasting and classification. In: **2023 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2023.

SCHAPIRE, R. E. Explaining ADABoost. In: **Empirical inference**. [S.l.]: Springer, 2013. p. 37–52.

SMITH, M. R.; MARTINEZ, T.; GIRAUD-CARRIER, C. An instance level analysis of data complexity. **Machine Learning**, v. 95, n. 2, p. 225–256, 2014.

SOARES, R. V. **Determinação de um índice de perigo de incêndio para a região centro-paranaense, Brasil**. Dissertação (Dissertação de Mestrado) — Centro Tropical de Ensino e Investigação, Instituto Interamericano de Ciências Agrícolas, Costa Rica, 1972.

SOARES, R. V.; BATISTA, A. C. **Incêndios florestais: controle, efeitos e uso do fogo**. [S.l.: s.n.], 2007. ISBN 9788590435327.

SORIANO, B. M. A.; DANIEL, O.; SANTOS, S. A. Eficiência de índices de risco de incêndios para o pantanal sul-mato-grossense. **Ciência Florestal**, v. 25, n. 4, p. 809–816, 2015. ISSN 0103-9954.

SRINIVAS, N.; DEB, K. Multiobjective function optimization using nondominated sorting genetic algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221–248, 1995.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367.

TELICYN, G. P. Logarithmic index of fire weather danger for forests. **Lesnoe Khozyaistvo**, v. 11, n. 1, p. 1–58, 1970.

TETTO, A. F. et al. Comportamento e ajuste da fórmula de monte alegre na floresta nacional de Irati, estado do Paraná. **Scientia Forestalis**, v. 38, n. 87, p. 409–417, 2010.

TORRES, F. T. P.; LIMA, G. S. Conservation of nature forest fire hazard in the Serra do Brigadeiro State Park (MG). **Floresta e Ambiente (FLORAM)**, v. 26, n. 2, 2019.

TORRES, F. T. P. et al. Analysis of efficiency of fire danger indices in forest fire prediction. **Revista Árvore**, v. 41, n. 2, 2017.

TORRES, F. T. P.; RIBEIRO, G. A. índices de risco de incêndios florestais em Juiz de Fora/MG. **Floresta e Ambiente (FLORAM)**, v. 15, n. 2, p. 24–34, 2008.

ULLAH, S.; MASOOD, M. Genetic drift and its effects on the performance of genetic algorithm (ga). In: **2023 International Conference on Robotics and Automation in Industry (ICRAI)**. [S.l.: s.n.], 2023.

VIGANÓ, H. H. da G. et al. Redes neurais artificiais na previsão de queimadas e incêndios no Pantanal. **Revista Brasileira de Geografia Física**, 2017. ISSN 1984-2295.

WAGNER, C. V. Development and structure of the canadian forest fire weather index system. **Canadian Forest Service Publications**, v. 35, 1987.

WEI, W.; VISWESWARAN, S.; COOPER, G. G. The application of naive bayes model averaging to predict alzheimer's disease from genome-wide data. **Journal of the American Medical Informatics Association (JAMIA)**, v. 18, n. 4, p. 370—375, 2011.

WILLMOTT, C. J.; MATSUURA, K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. **Climate Research**, v. 30, n. 1, 2005.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-2, n. 3, p. 408–421, 1972.

XIANG-WEI, L.; YIAN-FANG, Q. A data preprocessing algorithm for classification model based on rough sets. **Physics Procedia**, v. 25, p. 2025–2029, 2012. ISSN 1875-3892. International Conference on Solid State Devices and Materials Science, April 1-2, 2012, Macao.

XU, J. et al. FM-GRU: A Time Series Prediction Method for Water Quality Based on seq2seq Framework. **Water**, v. 13, n. 8, 2021. ISSN 2073-4441.

YAHUI, W. et al. NSGA-II algorithm and application for multi-objective flexible workshop scheduling. **Journal of Algorithms & Computational Technology**, v. 14, 2020.

YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. **Insights into Imaging**, v. 9, n. 4, p. 611–629, 2018.

YANG, S.; LUPASCU, M.; MEEL, K. S. Predicting forest fire using remote sensing data and machine learning. **Computer Vision and Pattern Recognition (CoRR)**, 2021.

ZAINUDDIN, Z.; A., P. A. E.; H., H. M. Predicting machine failure using recurrent neural network-gated recurrent unit (RNN-GRU) through time series data. **Bulletin of Electrical Engineering and Informatics**, v. 10, n. 2, p. 870–878, 2021. ISSN 2302-9285.

ZHANG, J.; MANI, I. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In: **Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets**. [S.l.: s.n.], 2003.

ZHANG, W.; GAO, F. An improvement to naive bayes for text classification. **Procedia Engineering**, v. 15, p. 2160–2164, 2011.

ZHAO, J. et al. Traffic data imputation and prediction: An efficient realization of deep learning. **IEEE Access**, 2020.

ZICCARDI, L. G. et al. Forest-fire risk indices and zoning of hazardous areas in Sorocaba, São Paulo state, Brazil. **Journal of Forestry Research**, v. 31, n. 2, p. 581–590, 2020.

Annex

ANNEX A

Dew Point Temperature as a Function of Temperature and Relative Humidity

RH	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**U**
T	**Dew Point Temperature ($^{\circ}C$)**														**T**
6											0	1	2	3	**6**
7										0	1	2	3	4	**7**
8									0	1	2	3	4	5	**8**
9									0	2	3	4	5	6	**9**
10								0	1	3	4	5	6	7	**10**
11							0	1	2	4	5	6	7	8	**11**
12							0	2	3	5	6	7	8	9	**12**
13						0	1	3	4	5	7	8	9	10	**13**
14						1	2	4	5	6	8	9	10	11	**14**
15					0	2	3	5	6	7	8	10	11	12	**15**
16					1	2	4	6	7	8	9	11	12	13	**16**
17					1	3	5	7	8	9	10	12	13	14	**17**
18				0	2	4	5	7	9	10	11	13	14	15	**18**
19				1	3	5	7	8	10	11	12	13	15	16	**19**
20				2	4	6	8	9	11	12	13	14	15	16	**20**
21			0	3	5	7	9	10	12	13	14	15	16	17	**21**
22			1	4	6	8	10	11	13	14	15	16	17	18	**22**

23			2	5	7	9	10	12	14	15	16	17	18	19	**23**
24		0	3	5	8	10	11	13	14	16	17	18	19	20	**24**
25		1	4	6	9	11	12	14	15	17	18	19	20	21	**25**
26		1	5	7	9	11	13	15	16	18	19	20	21	22	**26**
27		2	6	8	10	12	14	16	17	19	20	21	22	23	**27**
28		3	6	9	11	13	15	17	18	20	21	22	23	24	**28**
29		4	7	10	12	14	16	18	19	20	22	23	24	25	**29**
30	0	5	8	11	13	15	17	18	20	21	23	24	25	26	**30**
31	0	5	9	11	14	16	18	19	21	22	24	25	26	27	**31**
32	1	6	10	12	15	17	19	20	22	23	25	26	27	28	**32**
33	2	7	11	13	16	18	20	21	23	24	26	27	28	29	**33**
34	3	8	11	14	16	19	20	22	24	25	27	28	29	30	**34**
35	4	9	12	15	17	19	21	23	25	26	27	29	30	31	**35**
36	5	10	13	16	18	20	22	24	25	27	28	30	31	32	**36**
37	6	10	14	17	19	21	23	25	27	28	29	31	32	33	**37**
38	7	11	15	17	20	22	24	26	27	29	30	32	33	34	**38**
39	8	12	15	18	21	23	25	27	28	30	31	33	34	35	**39**
40	9	13	16	19	22	24	26	28	29	31	32	34	35	36	**40**
41	9	14	17	20	23	25	27	29	30	32	33	34	36	37	**41**
42	10	14	18	21	23	26	28	29	32	33	34	35	37	38	**42**

Table 29 – Dew point temperature ($^{\circ}C$) as a function of air temperature (T, $^{\circ}C$) and relative humidity (RH, %). Adapted from (SOARES; BATISTA, 2007).

ANNEX B

Maximum Vapor Pressure of Water as a Function of Air Temperature

T ($^{\circ}C$)	**E (mb)**	**T** ($^{\circ}C$)	**E (mb)**	**T** ($^{\circ}C$)	**E (mb)**
0.0	6.1078	20.0	23.373	40.0	73.777
0.5	6.3333	20.5	24.107	40.5	75.767
1.0	5.5662	21.0	24.261	41.0	77.302
1.5	6.8086	21.5	25.635	41.5	79.885
2.0	7.0567	22.0	26.430	42.0	62.015
2.5	7.3109	22.5	27.247	42.5	84.194
3.0	7.5753	23.0	28.086	43.0	86.423
3.5	7.8480	23.5	28.947	43.5	88.703
4.0	8.1294	24.0	29.831	44.0	91.034
4.5	8.4198	24.5	30.739	44.5	93.418
5.0	8.7198	25.0	31.671	45.0	95.855
5.5	9.0280	25.5	32.637	45.5	98.347
6.0	9.3480	26.0	33.608	46.0	100.89
6.5	9.6743	26.5	34.615	46.5	103.50
7.0	10.013	27.0	35.649	47.0	106.16
7.5	10.362	27.5	37.709	47.5	108.88
8.0	10.722	28.0	37.796	48.0	111.66
8.5	11.092	28.5	38.911	48.5	114.50
9.0	11.474	29.0	40.055	49.0	117.40
9.5	11.867	29.5	41.228	49.5	120.37

10.0	12.272	30.0	42.430	50.0	123.40
10.5	12.690	30.5	43.663	50.5	126.49
11.0	13.119	31.0	44.927	51.0	129.65
11.5	13.562	31.5	46.223	51.5	132.88
12.0	14.017	32.0	47.551	52.0	136.17
12.5	14.486	32.5	48.912	52.5	139.51
13.0	14.969	33.0	50.307	53.0	142.98
13.5	15.466	33.5	51.736	53.5	146.49
14.0	15.977	34.0	53.200	54.0	150.07
14.5	16.503	34.5	54.700	54.5	153.73
15.0	17.644	35.0	56.236	55.0	157.46
15.5	17.800	35.5	57.810	55.5	161.27
16.0	18.173	36.0	59.422	56.0	165.16
16.5	18.762	36.5	61.072	56.5	169.13
17.0	19.367	37.0	62.762	57.0	173.18
17.5	19.990	37.5	64.493	57.5	177.31
18.0	20.630	38.0	66.264	58.0	181.53
18.5	21.288	38.5	68.078	58.5	185.83
19.0	21.964	39.0	69.937	59.0	190.22
19.5	22.659	39.5	71.833	59.5	199.26

Table 30 – Maximum Vapor Pressure of Water (E, mb) as a Function of Air Temperature (T, $^{\circ}C$). Adapted from (SOARES; BATISTA, 2007).