FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Artificial intelligence for automated headache diagnosis through self-reported data

Álvaro Francisco Barbosa Miranda

DISSERTATION

Master's Degree in Informatics Engineering and Computation

Supervisor: João Reis

Second Supervisor: Gil Manuel Gonçalves

July 28, 2023

Artificial intelligence for automated headache diagnosis through self-reported data

Álvaro Francisco Barbosa Miranda

Master's Degree in Informatics Engineering and Computation

July 28, 2023

Abstract

Headaches are extremely frequent and highly disabling disorders that affect almost anyone. Although, from a societal point of view, it is still regarded as harmless and innocuous. This couldn't be further from the truth. In the world, thousands of years of quality life are lost due to this disability, which leads to an estimated annual cost of hundreds of millions of euros.[1]

However, not all headaches are the same, and consequently, the treatments differ too. The problem with the process of diagnosing is that when a patient with headaches finally decides to go on a medical appointment, a great amount of information before that point can already be lost. This happens because the patient didn't register any symptoms, their intensity, or when they began. This information will facilitate in the future correctly diagnosing headaches.

The objective of this project is to create a tool that first gathers all headache information from the patients, such as their symptoms and characteristics so that the neurologist has an easier task to follow the patient. Then by using machine learning methods, using the data provided by the patient, a preliminary diagnosis of which type of headache is provided to the neurologist. This diagnosis then must be confirmed by the neurologist, so no wrong treatments should be given to the patients. So the main objective is to complement the headache diagnosis and not replace the neurologist's job, and also help them to follow the patients more closely.

This project was done in collaboration with Serviço de Neurologia of Hospital Pedro Hispano.

Keywords: Headache Diagnosis, Self Reported Data, Machine Learning, Artificial Neural Network

Resumo

As dores de cabeça são extremamente frequentes e altamente incapacitantes, afetando quase qualquer pessoa. No entanto, do ponto de vista social, ainda são consideradas inofensivas e inócuas. Isso está longe da verdade. No mundo, milhares de anos de qualidade de vida são perdidos devido a esta incapacidade, o que resulta num custo anual estimado de centenas de milhões de euros[1]

No entanto, nem todas as dores de cabeça são iguais, e consequentemente os seus tratamentos também diferem. O problema com o processo de diagnóstico é que, quando um paciente com dores de cabeça finalmente decide marcar uma consulta médica, muita informação importante pode ter sido perdida. Isso acontece, porque o paciente não registrou nenhum sintoma, a sua intensidade ou quando é que eles começaram. Essas informações seriam úteis no futuro para o diagnóstico correto das dores de cabeça.

O objetivo deste projeto é criar uma ferramenta que, primeiramente, reúna todas as informações sobre a dor de cabeça dos pacientes, como sintomas e características, para que o neurologista tenha uma tarefa mais fácil ao acompanhar o paciente. Em seguida, usando métodos de aprendizagem de máquina e os dados fornecidos pelo paciente, é fornecido um diagnóstico preliminar sobre o tipo de dor de cabeça ao neurologista. Esse diagnóstico deve ser confirmado pelo neurologista, para evitar que tratamentos errados sejam dados aos pacientes. Portanto, o principal objetivo é complementar o diagnóstico de dor de cabeça e não substituir o trabalho do neurologista, além de ajudá-los a acompanhar os pacientes mais de perto.

Este projeto foi realizado em colaboração com o Serviço de Neurologia do Hospital Pedro Hispano.

Acknowledgements

I would like to take this moment to express my gratitude towards several individuals who played a crucial role not only in the completion of this step in my life but also throughout my journey as a college student, which is now reaching its end. First and foremost, I want to acknowledge the unwavering support of my supervisors, João Reis and Gil Gonçalves, as well as the valuable guidance Dr. Axel Ferreira and Dr. Sandra Moreira provided. Their mentorship, accessibility, and, most importantly, patience for not giving up on me and giving me time to fulfill this work.

Next, to the entire group of individuals who have been part of my life and who I can call friends, I wouldn't be the same today if not for you. While it's impossible to mention everyone individually, I think everyone important to me knows I have the utmost respect and gratitude for their influence in my life. I am deeply grateful for their enduring friendship over the years, for brightening my days when I needed it the most, and their presence has been invaluable on this journey.

Finally, I would like to express my heartfelt gratitude to my family. To my parents, their unwavering love, both emotionally and financially. Thank you to my brothers and sisters for putting up with me all these years.

Álvaro Francisco Barbosa Miranda

"Destiny is a funny thing. You never know how things are going to work out. But if you keep an open mind and an open heart, I promise you will find your own destiny someday."

Uncle Iroh

Contents

List of Figures

List of Tables

List of Equations

Abbreviations and Symbols

Mi	Migraine
TTH	Tension-Type Headache
CTTH	Chronic Tension-Type Headache
MwA	Migraine with Aura
MwoA	Migraine without Aura
MAwoH	Migraine Aura without Headache
CH	Cluster Headache
MOH	Medication Overuse Headache
TAC	Trigeminal Autonomic Cephalalgia
CTTH	Chronic Tension-Type Headache
EH	Epicranial Headache
TH	Thunderclap Headache
TAwM	Typical aura with migraine
TAwoM	Typical aura without migraine
FHM	Familial hemiplegic migraine
SHM	Sporadic hemiplegic migraine
BTA	Basilar-type aura
SH	Secondary Headache
TCH	Thunderclap Headache
ICHD	International Classification of Headache Disorders
CCIHS	Classification Committee of The International Headache Society
ANN	Artificial Neural Network
BN	Bayesian Network
C4.5	Decision Tree Algorithm
CART	Classification and Regression Tree
CDSM	Clinical Decision Support Mechanisms
CMF	Consistency measure Filter
DDN	Distributed Delay Network
DT	Decision Tree
FFN	Feed-Forward Network
GAW	Genetic algorithm wrapper
GENESIM	GENetic Extraction of a Single Interpretable Model
HRI	Headaches as a result of infection
HRICP	Headaches as a result of Intracranial pressure
HPSS	Headache Prediction Support System
KBS	Knowledge Based System
k-NN	K-Nearest Neighbors
LVQ	Learning Vector Quantization
LEVNN	Levenberg Neural Network
LFE	Learning from Examples
LNN	Linear Neural Network
LRM	Logistic Regression Model
LVQ	Learning Vector Quantization
MLP	Multi-Layer Perceptron
NB	Naive Bayes
PNN	Probabilistic Neural Network
RF	Random Forest
SVM	Support Vector Machine
XGB	Extreme Gradient Boost

Chapter 1

Introduction

This chapter introduces the dissertation and the work conducted on the project. Its motivation and scope are presented. The problem statement and the project's objectives and requirements are also described. Finally, a summary of its context environment and the document's structure are also detailed.

1.1 Motivation

Diagnosing a headache is a long and complicated process, as there are many different headaches. Firstly, there are two major groups: primary and secondary. The primary type has no known underlying cause, which means other diseases do not create it. That means they are considered the disease, whereas secondary headaches result from other diseases.

It is very time-consuming, as the process only begins when the patient goes for the first time to a doctor. After this first appointment, the patient will go home and register his symptoms, location, and intensity in a diary. This process can last six months until the correct diagnosis can be provided. And only after the diagnosis, the proper treatment can be provided. Thus, headaches can disturb the patient for an extended time before they can be correctly treated.

The proper treatment can only start after a successful diagnosis when the decision is not entirely sure. The neurologist can only provide the best treatment according to the current diagnosis. If it changes, the treatment can also change. Thus, proper follow-up is essential in headache treatment.

1.2 Context

This work was done with Serviço de Neurologia of Pedro Hispano Hospital in Matosinhos. A questionnaire will be used so that the neurology team at the referred hospital can more easily follow the patients.

1.3 Problem Definition

Can the headache diagnosis process be improved? Will the neurologist have his work reduced, or will it remain the same, as he will still have to double-check the diagnosis? Will the questionnaire improve the data collection? Will it allow the neurologist to notice if a severe problem happens to a patient? Will this improve the quality of life of those that suffer from headaches?

1.4 Objectives

When discussing people's health, if a method is not 100% effective, it cannot be used alone. So the main objective of this work is not to create a tool for the patients to use on their own without the supervision of the neurologist, but rather a tool for the neurologist to use in the diagnosis and an easy way to help him in the patient's follow-up and treatment.

After the first appointment, an initial diagnosis will be given. As time goes by, this diagnosis can change and become more accurate. Nonetheless, the neurologist will always have to check it. With this information, the best treatment can be provided.

The project's scope allows the patient to use it as a diary for their symptoms, intensity, and location so that the neurologist can maintain a close follow-up. With this, they can be alerted if, at some point, and more severe symptom or alert is discovered.

Only some headaches will be addressed in this work, so if the headache being diagnosed is outside the project's scope, its diagnostic cannot be very reliable.

1.5 Structure of the document

Beyond the Introduction, this thesis contains four chapters. In Chapter 2, we start by understanding what we are trying to classify in this project, the headaches. It also gives an introduction to the area of machine learning, including several concepts belonging to it. Also, in this chapter, some technologies that will be used in the project are expanded. Finally, it also explores the current similar estate of headache diagnosis using intelligent systems and machine learning, giving an insight into some related works.

In Chapter 3, everything related to the data used in this project is demonstrated, including how it was collected, where it was collected, and future work to improve their quality and gathering.

In Chapter 4, the previously obtained data is used to train the different models, and after applying the trained models to the test data, the results will be shown once divided.

In Chapter 5, the results previously obtained will be analyzed so that conclusions can be retrieved from the experiments done before. It will also be diagnosed if any methods can achieve high efficiency in headache diagnosis.

Chapter 2

State of the Art

This chapter provides everything necessary to understand this project. First, it explains health-related concepts like what a headache is, what the various types of them exist, and what their symptoms are. The types mentioned in this project will be explored more so it's easier to understand their diagnosis process. Other types will also be displayed to see which classes of similar projects classify their headaches.

Then it introduces the technological concepts needed to understand the work described in later chapters. It will explore the field of Machine Learning, but as it is so vast, it will focus only on the part the scope of the project belongs to.

Finally, it provides a comprehensive overview of headache diagnosis using different methods. It covers intelligent diagnostic systems' methodologies, algorithms, and technologies. It contains reviews of relevant literature, presenting case studies and research findings that demonstrate the effectiveness of machine learning models in diagnosing various types of headaches. Additionally, it addresses crucial considerations like data collection, preprocessing, feature extraction, and model evaluation.

2.1 Medical Background

2.1.1 Headache Definition

A headache is a discomfort or pain in the head, face, or neck. They may vary in location, intensity, and how often they occur. The brain doesn't feel pain because its tissue doesn't have pain-sensitive nerve fibers. So other parts of the head must be responsible for sensing the headache, like the face or neck muscles, nerves from the mouth, throat, or the top of the head, and blood vessels from the head. The pain their parts feel will help identify what kind of headache it is.

2.1.2 Classification

The classification of headaches is defined in the International Classification of Headache Disorders (ICHD) [2] made by the International Headache Society (IHS). Its latest version, the third, was

released in 2018 and serves as a dictionary for all kinds of headaches. It is an extensive document and is referred by its authors believe that knowing every detail of the paper is unnecessary, as even they admit they don't know all of it. It serves to be consulted to help identify and diagnose the kind of headache.

Every single kind of headache is attributed to an ICHD-3 code. This code serves as an identifier and shows the level of detail of the headache, as this code creates a hierarchy. For example, code 1 refers to migraines, whereas code 1.2 refers to Migraines with Aura, a sub-type of migraines. It can go to 5 levels of detail; for example, 1.2.3.1.2, called Familial hemiplegic migraine type 2, is a sub-type of Migraine, and also a sub-type of Migraine with aura, and so go on, a sub-type of the 1.2.3 headache (Hemiplegic Migraine) and 1.2.3.1(Familial hemiplegic Migraine).

Three groups need to be clarified before entering into more detail about each type of headache. First, there are primary and secondary headaches. Primary headaches are not caused by any underlying medical condition and are considered the medical condition itself. Muscle tension, stress, migraines, or hormonal changes can trigger them. On the other hand, secondary headaches are symptoms of another underlying medical condition, such as infections, vascular problems, brain tumors, or head injuries. In these cases, the headache is a warning sign that something is wrong in the body and requires investigation and appropriate treatment of the underlying cause. There is still a final group called painful cranial neuropathies and other facial pain that the IHS also included in ICHD. Primary headaches use the ICHD-3 codes 1 to 4, secondary codes 5 to 12, and lesions and pains are ICHD-3 codes 13 and 14.

This work focuses on diagnosing primary types. However, it also includes some secondary classes and cranial lesions, as requested by Dr.Axel Ferreira. This range and variety of headache types allow for a broader perspective of what algorithms can be used for a classification problem.

Before entering into detail about each headache, explaining how a diagnosis is attributed is necessary. For each headache type, there are some criteria that the patient must meet so that the ICHD-3 diagnosis can be accounted for. Some are mandatory, and others can be optional, as seen below.

2.1.2.1 Migraine

Migraine is a frequently encountered primary headache disorder that can cause significant disability. According to the Global Burden of Disease Study, it is the third of the most widespread conditions globally and the third-highest cause of disability in the whole globe[3].

It can be classified into two main types: Migraine without aura [1.1], where the headache displays specific features and associated symptoms described below, and Migraine with aura [1.2], which is primarily characterized by temporary focal neurological symptoms that usually precede or sometimes occur alongside the headache.

Each of these two types can still be divided into two kinds: Episodic or chronic if the amount of times the person suffers from the headache per month doesn't surpass 14 times for the first type and more or equal to 15 for the second type.

- **Migraine without aura**

  Its criteria are the following:

  (A) A minimum of five episodes that meet the criteria described in B to D

  (B) Headaches range from 4 hours to 72 hours, either when left untreated or when treatment attempts are unsuccessful.

  (C) The headache exhibits at least two out of the following four characteristics:

  (i) unilateral localization (pain only felt in one side of the face)

  (ii) pulsating sensation of pain (like the heartbeat)

  (iii) moderate to intense levels of pain

  (iv) pain gets worse doing regular physical activities like tying shoes or walking

  (D) During the headache, there is the presence of at least one of the following:

  (i) nausea and/or vomiting

  (ii) increased sensitivity to light (photophobia) and sound (phonophobia)

- **Migraine with aura**

  Its criteria are the following:

  (A) A minimum of two episodes that meet the criteria described in B and C

  (B) The presence of one or more fully reversible aura symptoms from the following list:

  (i) visual

  (ii) sensory

  (iii) impairment or disturbances in speech and/or language abilities

  (iv) motor

  (v) brainstem (connection of the brain to the spinal cord)

  (vi) retinal

  (C) A minimum of three out of the following six characteristics:

  (i) the gradual spread of at least one aura symptom over 5 minutes or longer

  (ii) the occurrence of two or more aura symptoms happening in succession

  (iii) each distinct aura symptom has a duration lasting between 5 and 60 minutes

  (iv) at least one aura symptom is unilateral

  (v) at least one aura symptom is positive(perception of shimmering or tingling sensation)

  (vi) The aura is accompanied or is followed by a headache within 60 minutes

  Regarding migraine without aura in children and teenagers under 18, it is very common on both sides of the head, while it is less frequent in adults. A small percentage, less than 10% of women, experience migraine attacks in correlation with most of their menstrual cycles.

Concerning the migraine with aura, there are some difficulties for the patient. Some common errors include inaccurate descriptions of the location of symptoms, such as reporting them on one side instead of both sides, mistakenly describing the onset of symptoms as sudden rather than gradual, and perceiving visual disturbances in one eye instead of affecting both eyes. Another mistake is misjudging the duration of aura and confusing sensory loss with weakness.

2.1.2.2 Tension-Type Headache

Tension-type headache (TTH) is the most prevalent form of primary headache. Depending on the study, its occurrence in the general population spans from 30% to 78% over a person's lifetime. Despite having the most significant socio-economic impact among primary headache disorders, it remains the least investigated [4].

- **Infrequent episodic tension-type headache**

  Its criteria are the following:

  (A) At least ten headache episodes that happen in less than one day per month on average and meet the criteria outlined in B to D.

  (B) Headaches that persist for a duration of 30 minutes to 7 days

  (C) At least two of the following four characteristics:

  (i) bilateral localization

  (ii) the sensation of pressure or tightening

  (iii) mild or moderate levels of pain

  (iv) not aggravated by regular physical activities like tying shoes or climbing stairs

  (D) Both of the following:

  (i) no nausea or vomiting

  (ii) no more than one of increased sensitivity to light or (photophobia) and sound (phonophobia)

- **Frequent episodic tension-type headache**

  Its criteria are like the ones referred to in Infrequent episodic tension-type headache, with only criteria A changing:

  (A) A minimum of 10 headache episodes between 1 and 14 days per month on average and meet the criteria outlined in B to D from Infrequent episodic tension-type headache.

- **Chronic tension-type headache**

  Its criteria are the following:

(A) A minimum of 10 headache episodes that happen 15 times or more per month on average and meet the criteria outlined in B to D.

(B) Headaches that persist for hours to unremitting

(C) At least two of the following four characteristics:

  (i) bilateral localization

  (ii) the sensation of pressure or tightening

  (iii) mild or moderate levels of pain

  (iv) not aggravated by regular physical activities like tying shoes or climbing stairs

(D) Both of the following:

  (i) neither moderate nor severe levels of nausea or vomiting are present

  (ii) No more than one of the following is experienced: sensitivity to light, sensitivity to sound, or mild nausea

2.1.2.3 Cluster headaches

Cluster headaches, a sub-type of trigeminal autonomic cephalalgias (TACs), usually occur in a series that may last weeks or months. It is the most painful type of primary headache and is often called the "suicide headache" because of the extreme suffering it causes. The intensity of the pain is so severe that it can lead individuals to contemplate suicide due to the fear of experiencing another cluster attack [5].

The typical age range for the onset of migraine is between 20 and 40 years. Interestingly, men can be up to three times more susceptible to experiencing migraines than women, although the underlying reasons for these differences are still unknown.

Its criteria are the following:

(A) A minimum of five episodes that meet the criteria described in B to D

(B) Intense or extremely intense pain in the orbital (eye), supraorbital (above the eye), and/or temporal (side of the head) regions, lasting 15 to 180 minutes when left untreated.

(C) Either one or both of the following::

  (i) at least one of the following symptoms or signs on the same side as the headache:

    (a) redness of the eye and/or excessive tearing

    (b) nasal congestion and/or excessive nasal discharge

    (c) swelling of the eyelids

    (d) sweating on the forehead and face

    (e) constriction of the pupil and/or drooping of the eyelid

  (ii) sensation of being restless, unsettled, or agitated

(D) Occurring at a rate ranging from once every two days to as frequently as eight times per day

- **Episodic cluster headache**

  Its criteria are the following:

  (A) Headache attacks that meet the criteria for cluster headache and occur in distinct episodes known as cluster periods

  (B) Experiencing a minimum of two cluster periods that endure for a duration of seven days to one year (in the absence of treatment), with pain-free intervals of at least three months occurring between each cluster period

- **Chronic cluster headache**

  Its criteria are the following:

  (A) Headache attacks that meet the criteria for cluster headache

  (B) Manifesting without any significant periods of relief, or with remission periods lasting less than three months, persisting for a minimum duration of one year

2.1.2.4 Headache attributed to a substance or its withdrawal

The issue of medication overuse, which leads to medication overuse headaches (MOH), is a rapidly growing problem that is still not fully recognized globally. Several recent studies on epidemiology indicate that a considerable proportion, up to 4%, of the general population in Europe, North America, and Asia engage in excessive use of analgesics and other medications for the management of pain conditions like migraine [6].

- **Medication-overuse headache**

  Its criteria are the following:

  (A) headache on 15 or more days within a month in an individual who already has an existing headache disorder

  (B) exceeding one or more medications listed in C, for a duration surpassing three months

  (C) At least one of the following:

    (i) at least ten days of consumption of ergotamine per month

    (ii) at least ten days of consumption of any combined or specific analgesic medication for your headache(acetaminophen with caffeine, migretil, triptans)

    (iii) at least 15 days of consumption of simple analgesic medication (paracetamol, ibuprofen, diclofenac) per month

2.1.2.5 Painfull lesions of the cranial nerves and other facial pain

- **Trigeminal neuralgia**

  Trigeminal neuralgia (ICHD-3 code 13.1.1) is a rare condition characterized by recurring episodes of unilateral facial pain that resemble electric shocks. A gentle touch typically triggers it and can initially be misinterpreted as a dental issue due to its manifestation in the lower branches of the trigeminal nerve [7].

  Its criteria are the following:

  (A) Repeated episodes of intense facial pain on one side of the face, limited to specific regions corresponding to one or more divisions of the trigeminal nerve. The pain does not spread beyond these areas

  (B) The pain exhibits all of the following attributes::

  (i) ranging from a fraction of a second to a maximum of two minutes.

  (ii) severe intensity

  (iii) sensation resembling an electric shock, shooting, stabbing

  (C) triggered by harmless stimulus occurring within the affected regions associated with the trigeminal nerve

2.1.2.6 Other headache disorders

Many other headache types and sub-types can be consulted in ICHD-3. In this work, if a headache cannot be classified as one of the above, it will rank as Another headache.

2.2 Technological Background

2.2.1 Machine Learning

Although humans acquire knowledge through experience, computers can also do so. This is made possible through a field called machine learning. Machine learning is a methodology that enables computer systems to learn from experience using computational techniques. In computer systems, knowledge is represented in the form of data, and the primary objective of machine learning is to create algorithms that can construct models based on this data. By providing the learning algorithm with experiential data, we can generate a model to make predictions and draw insights[8].

Figure 2.1 shows a simplified guide to Machine Learning. First, it has two major fields: supervised and unsupervised learning.

- **Supervised Learning**

  It consists in teaching a machine learning algorithm using labeled examples, enabling it to understand the underlying patterns and make accurate predictions based on new, unseen data

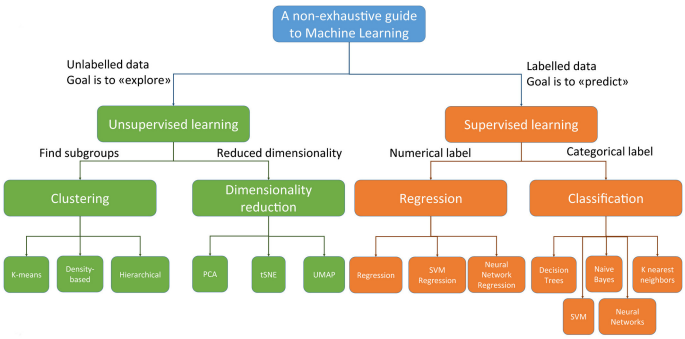

Figure 2.1: Machine Learning Guide

[9]. The two types of supervised learning are regression which predicts continuous values, and classification, which classifies in categorical labels.

- **Unsupervised Learning**

  Instead of being guided by labeled examples, unsupervised learning algorithms focus on finding similarities or groupings within the data. They aim to identify clusters of similar data points or reduce the dimensionality of the data by capturing its essential features. By doing so, unsupervised learning algorithms can provide insights, detect anomalies, or simplify complex datasets for further analysis [10].

Following this introduction is time to go through the steps of machine learning and understand what they are and what their function is.

2.2.1.1 Data Preparation

The initial step of any machine learning process is to gather and process data to be ready for training and testing for a model. Normally three different steps are composing the data preparation [11]:

- **Data collection** involves gathering the necessary data from various sources. The data collected should be relevant and representative of the problem at hand. This step aims to acquire a dataset containing the required information for effectively training a machine learning model.

- **Data preprocessing** is a crucial step where the collected data is cleaned and transformed to make it suitable for analysis. It addresses issues like missing values, outliers, inconsistencies, and noise in the data. Data preprocessing techniques are applied to improve the quality and reliability of the data, ensuring that it is in a format that can be readily used for subsequent analysis.

- **Feature selection** is creating or selecting features from the dataset that can enhance the model's ability to make accurate predictions. It involves applying domain knowledge and

statistical techniques to transform raw data into meaningful features. The goal is to provide the model with informative and relevant features that capture the underlying patterns in the data, improving its predictive performance. Some feature selection methods are Recursive Feature Elimination which works by recursively removing the least significant features and building models with the remaining features until a predefined number of features is reached or a specific performance criterion is met [12], Least absolute shrinkage and selection operator (LASSO), which works by adding a penalty term to the linear regression cost function, encouraging the model to minimize the absolute values of the coefficients. This has the effect of setting some coefficients to precisely zero, effectively eliminating the corresponding features from the model [13], or genetic algorithm wrapper, which works by representing each candidate feature subset as a chromosome in the genetic algorithm, and then evolves and improves these subsets through genetic operations such as selection, crossover, and mutation[12].

2.2.1.2 Classification Models

Next, it is going to be explored other models which are used in this work so, in the end, the optimal solution can be achieved.

- **Logistic Regression**

  Logistic regression is a statistical model used for binary classification tasks but also can be extended to handle classification problems with more than two classes. When this happens is called multinomial logistic regression.

  While the logistic function (also known as the sigmoid function) is to model the relationship between the input variables and the probability of the binary outcome, the multinomial logistic regression uses the softmax function instead of a single logistic function, which generalizes the logistic function for multiple classes.

  Logistic regression estimates one or multiple sets of coefficients, depending on the number of classes involved. It predicts each class's probabilities individually, considering each category's corresponding set of coefficients.

  During prediction, the model calculates the probabilities of each class individually using their respective set of coefficients. It assigns a probability to each class, representing the likelihood of that class being the correct prediction. The class with the highest chance is then selected as the predicted class[14].

- **Decision Tree**

  A decision tree is a tree-like structure where each internal node represents a feature or attribute, each branch represents a decision based on that feature, and each leaf node represents a class label or a numerical value.

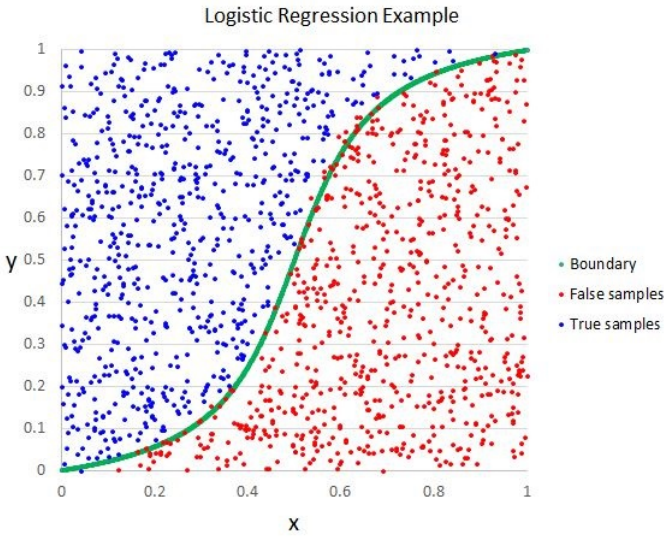

Figure 2.2: Logistic Regression Example

It looks for the best feature that can separate the data into groups that are similar in terms of the outcome that should be predicted. It keeps splitting the data until it creates groups that are as pure as possible, meaning they have similar results [15].

Once the tree is built, it can be used to make predictions by traversing the tree from the root to a leaf node based on the feature values of the input instance. With this, they are easy to interpret, as the tree structure visually provides clear decision paths.

However, decision trees are prone to overfitting if not adequately controlled, which can be mitigated using techniques like pruning or ensemble methods.

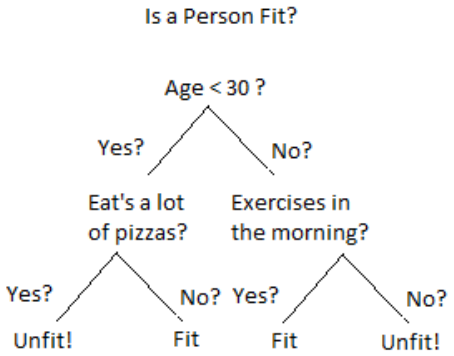

Figure 2.3: Decision Tree Example

- **K-nearest neighbors**

  For classification, the algorithm looks at the "k" nearest neighbors to a new data point and assigns the most common class label among those neighbors as the prediction for the new

point[16].

The choice of "k" affects the balance between flexibility and sensitivity to noise. A smaller "k" makes the algorithm more sensitive to individual data points, potentially leading to overfitting. A larger "k" makes the algorithm more noise-friendly but may overlook essential patterns.

K-NN is a flexible algorithm that doesn't assume anything specific about the data distribution. It's easy to understand and implement. However, it can be slow with large datasets because it calculates distances between points.

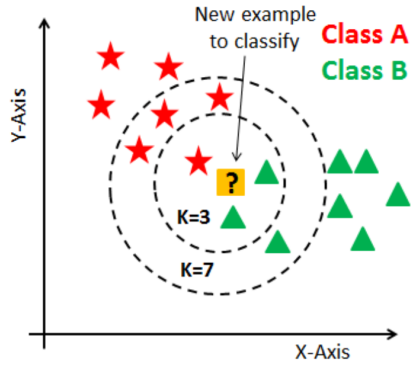

Figure 2.4: K-Nearest Neighbors Example

- **Random Forest**

  Random Forest is a combination of decision trees to make predictions. It gets its name because it creates a "forest" of decision trees, each being trained on a random subset of the data and considering only a random subset of the features. [17].

  The process starts by building multiple decision trees, each using a different subset of the training data. These trees make predictions independently based on the randomly selected features. When making predictions for a new data point, each tree in the forest gives its prediction, and the final prediction is determined by taking the majority vote or averaging the predictions from all the trees.

  Random Forest handles complex relationships between variables effectively and is less likely to overfit the data than a single decision tree. Combining the predictions of multiple trees reduces the impact of outliers and noise, leading to more accurate predictions.

  Another benefit of Random Forest is that it can provide information about the importance of different features. This helps identify which features influence the predictions most and can provide insights into the underlying patterns in the data.

- **Naive Bayes**

  It assumes that all features are independent, a simplifying assumption known as "naive."

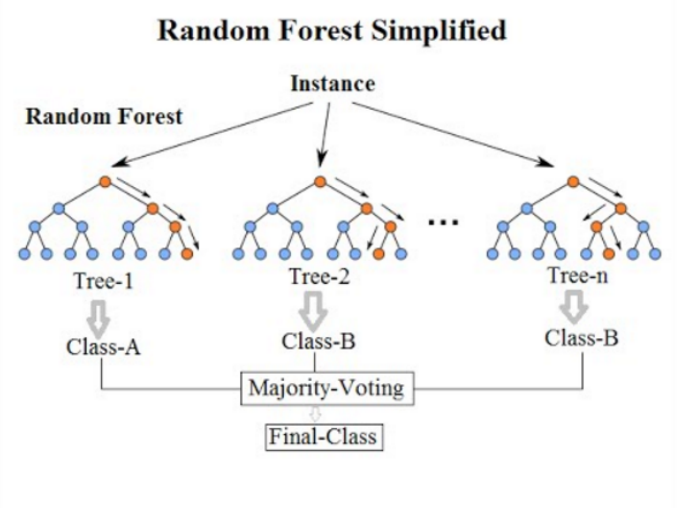

Figure 2.5: K-Nearest Neighbors Example

During training, it calculates probabilities based on feature occurrences in the data. To make predictions, it uses Bayes' theorem to calculate the probability of each class given the observed features. The class with the highest chance is assigned as the prediction. Naive Bayes is known for its simplicity, efficiency, and ability to handle large datasets. However, the independence assumption may not always hold, affecting prediction accuracy [18].

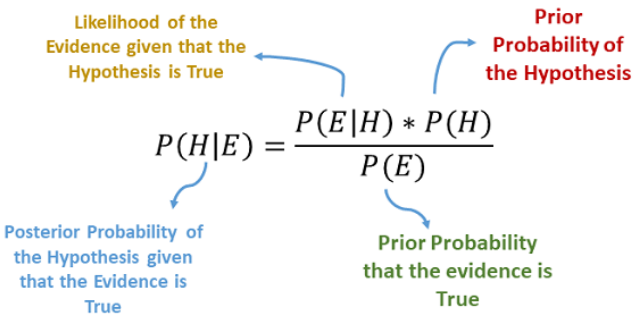

Figure 2.6: Naive Bayes Algorithm

- **Support Vector Machine**

  Support Vector Machines (SVM) is an algorithm that finds a hyperplane to separate data points of different classes. It tries to maximize the margin, the distance between the hyperplane, and the closest data points from each category.

  It can handle both linearly separable and non-linearly separable data. It achieves this by employing the kernel trick, which transforms the data into a higher-dimensional space where a hyperplane can effectively separate it. This technique allows SVM to capture intricate patterns and complex relationships between variables, providing a more flexible and accurate classification [19].

In the training phase, SVM tunes the position and direction of the hyperplane by minimizing a loss function. This loss function considers both maximizing the margin and minimizing misclassification errors. Through an optimization process, SVM finds the best hyperplane that optimally separates the classes.

Once trained, SVM can classify new data points based on their position relative to the decision boundary.

SVM is known for its ability to handle datasets with many features and perform well in various domains. However, SVM's performance can be highly influenced by the selection of parameters, making it essential to tune them carefully to achieve the best possible results.

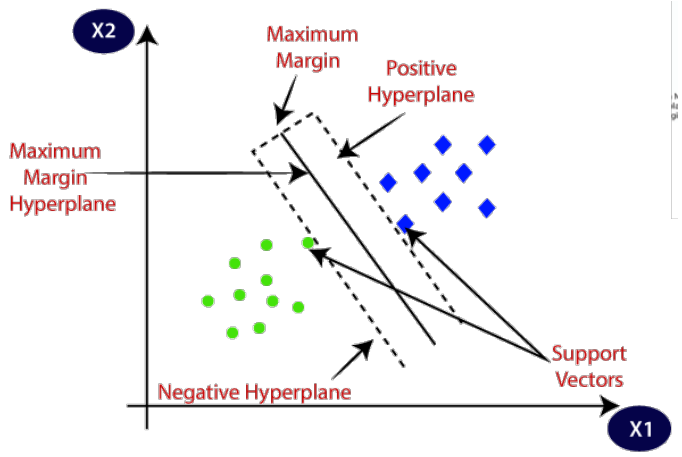

Figure 2.7: Support Vector Machine Example

- **Neural Network**

  Inspired by the human brain, the neural network uses interconnected nodes called artificial neurons to process and analyze data. These neurons are organized into layers and work together to transform the input data and make predictions [20].

  During training, it uses forward propagation to generate outputs based on input data and then uses backward propagation (backpropagation) to calculate gradients and update the weights and biases. This combination of forward and backward propagation is crucial for training neural networks and enabling them to learn from the data.

  Once trained, the neural network can predict new data by feeding it through the network. It applies the learned connections and computations to generate an output representing the expected result or class label.

  Neural networks are effective at learning complex patterns and relationships in data. However, they require careful design and tuning to ensure optimal performance and prevent overfitting to the training data.

2.2.1.3 Models Comparison

Figure 2.8 can represent the models discussed here. It is also visible that the most complex models are more efficient but harder to understand. For example, a decision tree is a white box model, which means it can be easily understood and explained. Otherwise, a neural network is a black-box model, which means its internal workings are not easily understandable or explainable. With this information, we can conclude that as we choose an increasingly more complex model, it can deal with more features and find intricate interactions between features. Still, at the same time, some of the transparency of the more simple methods is lost.

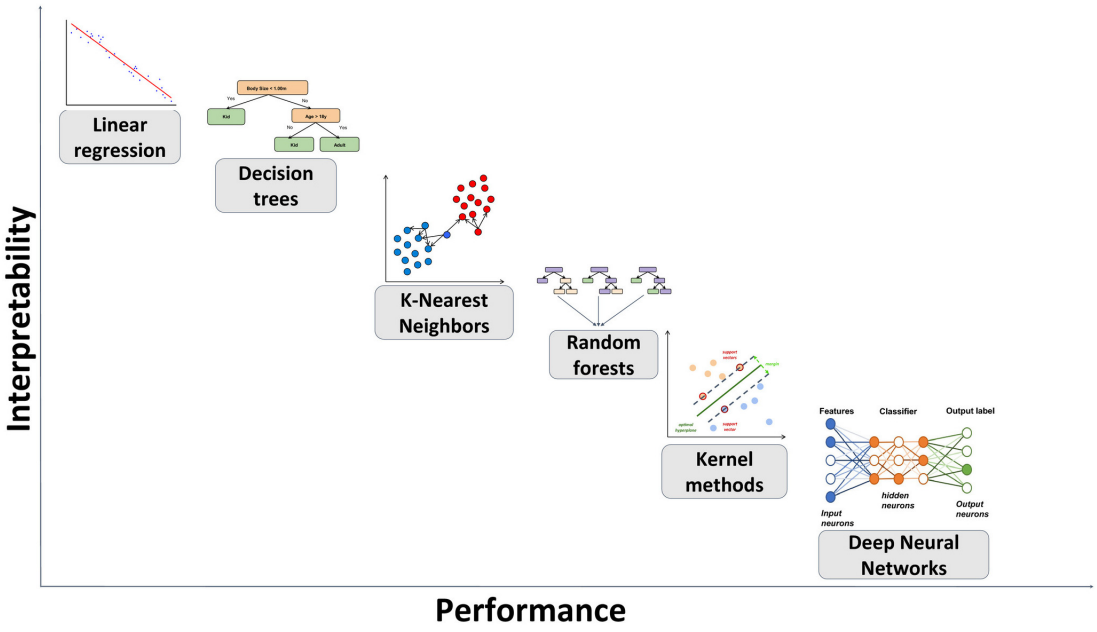

Figure 2.8: Interpretability vs. Performance of models

2.2.1.4 Artificial Neural Network

This section discusses the artificial neural network relevant to the project's scope. Also, what are their hyperparameters, and what is their function? First, we start with their types.

- **Feedforward Neural Networks or Multi-Layer Perceptron** FNNs or MLPs are neural networks that process data in a forward direction, from input to output. They are commonly used for tasks like classification and regression, where information flows through multiple hidden layers before reaching the output layer [21].
- **Recurrent Neural Networks** RNNs are designed for sequential data, such as time series or language. They have connections that allow information to persist over time, making them suitable for tasks with a temporal element. RNNs effectively capture dependencies

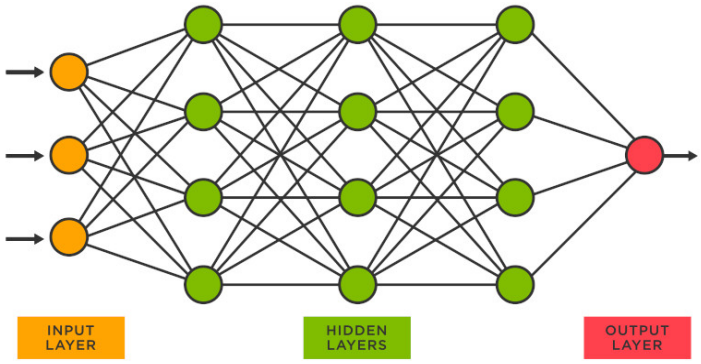

Figure 2.9: Multi-Layer Perceptron diagram

in sequential data and are commonly used in tasks like speech recognition and language translation [22].

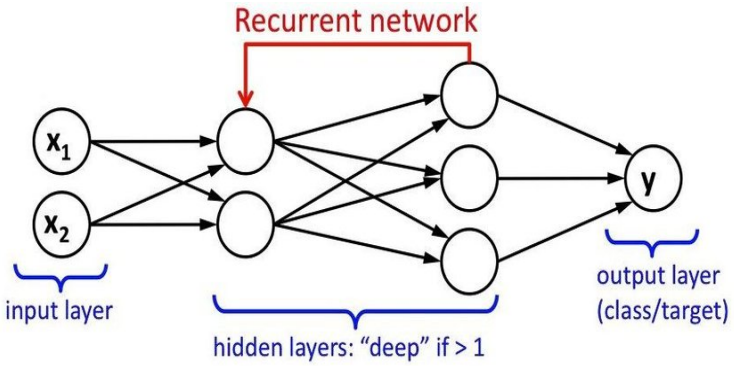

Figure 2.10: Recurrent Neural Network diagram

Training an artificial neural network can be challenging due to the presence of various hyperparameters, which are initial variables that define the structure and behavior of the network. These hyperparameters are set before the training begins and are specific to each problem. They encompass factors such as the activation functions used within each node of the network and the learning rate employed by the algorithm. The selection of appropriate hyperparameters depends on factors like the chosen network architecture, the size of the dataset, and its format. It is crucial to carefully tune these hyperparameters to ensure optimal performance and successful training of the neural network [23].

- **Number of Hidden Layers:** These are the intermediate layers between a neural network's input and output layers. Each layer comprises neurons that receive input from the previous layer and performs a linear transformation of the inputs using weights and biases. Then the neurons apply an activation function to the result to introduce non-linearity into the model. The choice of the number of hidden layers can impact the network's ability to learn complex patterns and relationships in the data. Having too few hidden layers in an artificial neural

network can lead to underfitting. On the other hand, using a larger number of hidden layers can yield better results at the cost of increased computational time needed for training and inference.

- **Number of Neurons in Hidden Layers:** Each neuron performs computations using weights and biases, and its objective is to generate an output. Works in the same way as the number of layers. Too few can lead to underfitting, and too many leads to increased computational requirements and the risk of overfitting.

- **Activation Function:** An activation function exists within each neural network node and is responsible for processing the inputs and associated weights to produce the node's output. The activation function introduces non-linearity to the network, enabling it to learn and model intricate relationships in the data.

  Commonly used activation functions include the sigmoid, ReLU (Rectified Linear Unit), and tanh (hyperbolic tangent) functions. The choice of activation function is an essential decision in the design and training of neural networks as it depends on the specific problem and the desired behavior of the network.

- **Learning Rate:** It determines how quickly a neural network updates its weights and biases during training. It governs the size of the steps taken in the optimization process. A high learning rate can speed up the training process, enabling faster convergence, but it may also miss the optimal solution or cause instability in the training process. Contrarily, a low learning rate ensures a more stable convergence but at the expense of longer training time. Finding an appropriate learning rate balances training efficiency and convergence quality.

- **Batch Size:** The batch size determines the number of samples processed by the model before updating the weights and biases during each iteration of the training process. Selecting an appropriate batch size is a trade-off between training efficiency and generalization performance. Smaller batch sizes introduce more randomness into the weight updates and may require more iterations (epochs) to converge, while larger batch sizes may lose some generalization performance.

- **Regularization Parameters:** Regularization is a technique that prevents overfitting in neural networks by adding a penalty term to the loss function. The regularization parameters control the strength of the regularization and the trade-off between fitting the training data and generalizing it to unseen data. Standard regularization techniques include L1 and L2 regularization, which add a penalty based on the magnitudes of the weights. Proper regularization can help prevent overfitting and improve the network's generalization of new data.

- **Optimization Algorithm:** are essential to training neural networks. They determine how the model's parameters (weights and biases) are updated during the learning process to minimize the loss function and improve the network's performance.

The choice of algorithm depends on factors such as the problem complexity, dataset size, and computational resources available. Experimentation and tuning are often required to determine the most suitable optimization algorithm for a specific neural network application. Stochastic Gradient Descent (SGD), Adam (Adaptive Moment Estimation), and AdaGrad are examples of optimization algorithms.

- **Loss Function:** The loss function measures the model's performance during training. It calculates the difference between the predicted values and the actual values. The specific loss function chosen depends on the problem type. For example, in classification tasks, categorical cross-entropy can be used for multiple classes or binary cross-entropy for two classes.

- **Number of Epochs:** The number of epochs defines the number of times the entire training dataset is passed forward and backward data shuffling the network during training. Increasing the number of iterations allows the model to learn more from the data, but too many iterations may lead to overfitting. It's crucial to find the right balance to achieve optimal model performance. One epoch is completed when all batches have been processed once. Therefore, the number of iterations within an epoch depends on the batch size.

- **Random State:** The random state is a starting point for the random number generator used in various model processes, such as weight initialization and shuffling of data. By setting a specific random state, you can reproduce the same random processes and obtain consistent results when running the model multiple times. This ensures reproducibility, allowing for fair comparisons and consistent testing.

There are more than the ones described here, but their concepts are not expanded in the scope of work due to the inability to modify them using the technologies used.

2.2.1.5 Model Evaluation

The model evaluation assesses how well a machine learning model performs on new, unseen data. To do this, the dataset is divided into training and test data, with values normally around 80% and 20% for their size. Then after training the model, it tries to predict the test data. Then, we can build the confusion matrix seen in figure 2.11 with the results. The table summarizes the performance by showing the number of correct and incorrect predictions made by the model, categorized by the actual and predicted class labels.

1. **True Positive (TP)**: The number of instances that are correctly predicted as positive (actual positive, predicted positive).

2. **False Positive (FP)**: The number of instances that are incorrectly predicted as positive (actual negative, predicted positive).

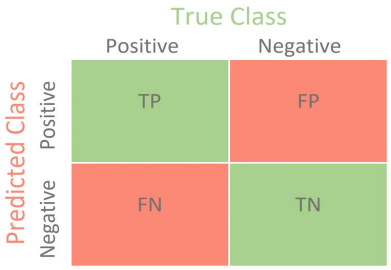

Figure 2.11: Confusion Matrix

3. **True Negative (TN)**: The number of instances that are correctly predicted as negative (actual negative, predicted negative).

4. **False Negative (FN)**: The number of instances that are incorrectly predicted as negative (actual positive, predicted negative).

With these values, other evaluation metrics can be calculated:

- **Precision:** It represents the proportion of true positive predictions out of all positive predictions.

$$Precision = \frac{TP}{TP+FP} \tag{2.1}$$

- **Recall or Sensitivity:** It measures the proportion of true positive predictions out of all actual positive instances

$$Sensitivity = \frac{TP}{TP+FN} \tag{2.2}$$

- **Accuracy:** measures the overall correctness of the model's predictions by comparing them to the true labels

$$Accuracy = \frac{TP+FN}{TP+FP+TN+FN} \tag{2.3}$$

- **F1 Score:** It combines precision and recalls into a single metric by taking their harmonic mean. It provides a balanced evaluation when both precision and recall are important.

$$F1Score = 2 * \frac{2*TP}{2*TP+FP+FN} \tag{2.4}$$

Cross Validation is a model evaluation technique that assesses a predictive model's performance and generalization ability.

It divides the dataset into several subsets/folds and trains the model multiple times. Each time, one fold is used as the validation set, and the rest are used for training. This allows us to simulate how the model would perform on unseen data. By repeating this process and averaging the performance metrics across all folds, we can get a more reliable estimate of the model's ability to generalize to new data.

Cross-validation helps assess the model's performance, detect overfitting, and helps in making good decisions about model selection and hyperparameter tuning.

The time each method requires in training and applying it to test data is also a good measure to see the efficiency vs. the resources used.

2.2.2 Usefull libraries

- **Scikit-learn:**, often referred to as sklearn, is a Python library that offers a collection of tools and algorithms for machine learning tasks. It provides a convenient and consistent interface for implementing various machine-learning techniques, making it easy to experiment with different algorithms and evaluate their performance. It offers a wide range of functionalities, including data preprocessing, feature selection, and model evaluation [24].

- **TensorFlow:** TensorFlow is an open-source library for numerical computation and machine learning developed and maintained by Google. It provides a flexible and efficient framework for building and training various machine learning models, including neural networks [25].

- **PyTorch:** PyTorch is a popular open-source library for machine learning and deep learning. It is primarily used for building and training neural networks. PyTorch provides a flexible and dynamic approach to deep learning, allowing researchers and developers to design and implement complex models easily [26].

- **Pandas:** is an open-source Python library that provides powerful and easy-to-use data manipulation and analysis tools. It is built on top of the NumPy library and is widely used in data science, machine learning, and data analysis projects. [27].

2.2.3 Literature review

Next, it explores some related projects and insights on their strong points and some of their bad characteristics. In table 2.1, it can be seen a summary of the projects found relevant.

- Automatic diagnosis of primary headaches by machine learning methods [12]: various model predictions are demonstrated, alongside a comprehensive comparison of multiple feature selection methods. It explores the outcomes generated by different models and assesses their performance while examining various techniques for selecting relevant features. It is important to note that specific questions and data are not provided, as the focus lies on

Table 2.1: Literature review summary

Ref.	Methods	Feature Sel.	Data	Classes	Best Results
[12]	NB, C4.5, SVM, Bagging, AdaBoost, RF	CMF, ReliefF, GAW	1022 people; 4 questionnaires	Mi, TTH, Others	RF with RelieF Greedy: 81.02±2.45
[13]	XGBoost, RF, SVM, k-NN	LASSO, SVM-RFE, mRMR-MIQ, mRMR-MID	2162 patients; 75 questions	Mi TTH, TAC, EH, TCH, SH,	XGBoost With Lasso: 0.8071
[28]	LNN, SVM, k-NN, DT, RF, LEVNN, LDA		836 patients; 65 features	TTH, CTTH, MwA, MwoA TAC	AUC of 0.985; sensitivity of 1; specificity of 0.958
[29]	23 models and HPSS	19->15 features using 9 attribute selection method	614(199m, 415f)	Mi, TTH, CH, SH	both primary and secondary with an overall accuracy of 93%
[30]	BN, ANN		2177 patients, 14 variables	MwoA, MwA, TTH, MOH, Others	accuracy at 75,75%
[31]	MLP, LRM, SVM, k-NN, CART	Cross-Validation	400 patients 23 in the beginning, 18 after pruning	TAwM, TAwoM, MwoA, FHM, SHM, BTA, Others	accuracy and precision levels >97%
[32]	DT(Gini Algorithm), DDN, PNN, FFN, LVQ		100 people and 535 people; 8 variables	Mi, Probable Migraine, No Migraine	DDN with 95.45% accuracy
[33]	K-Means Clustering		353 students; 20 variables	MwoA, Probable MwoA, No Migraine	accuracy for no migraine at 88.89 %
[34]	Fuzzy Expert System using LFE algorithm, MLP, SVM		190 patients; 12 variables	Mi, TTH, HRICP, HRI	SVM with accuracy +- 0.94
[35]	GENESIM, C4.5, RF, XGB, CART, LR, SVM, k-NN, NN	a genetic algorithm	849 patients; 10 variables	Mi, TTH, CH	Higher: GENESIM:0.983510
[36]	Neural Networks with different architectures		2,177 patients; 9 variables	MwA, MwoA, TTH, MOH, Others	around 0.95

showcasing the model predictions and evaluating the effectiveness of diverse feature selection methods.

- Machine learning-based automated classification of headache disorders using patient-reported questionnaires [13]: presents a proposed model for classification, specifically focusing on the XGBoost, a learning algorithm that combines the predictions of multiple decision trees to achieve high accuracy and predictive power, which falls under the category of boosting methods. While questions regarding the classification task are open to anyone, obtaining the necessary data requires proper permission. The study emphasizes the model's performance and explores its effectiveness in addressing the classification problem.

- An Intelligent Systems Approach to Primary Headache Diagnosis [28]: includes available questions for analysis, although the dataset required for the study is not accessible. Notably, the study demonstrates the visual representation of class distributions using t-Distributed Stochastic Neighbourhood Embedding (tSNE), a dimensionality reduction technique that maps high-dimensional data points into a lower-dimensional space while preserving local relationships and revealing underlying patterns, and Principal Component Analysis (PCA). This linear dimensionality reduction technique transforms data into a new coordinate system to maximize variance and identify the most critical components. Additionally, the study presents comprehensive and detailed metric results, highlighting the performance of the models employed in the study and providing good informative graphical information.

- A High-Performance System for the Diagnosis of Headache via Hybrid Machine Learning Model [29]: A web-based Headache Prediction Support System (HPSS) was developed to assist patients in obtaining a diagnosis for their headaches. It provides an interface for users to input relevant features associated with their symptoms. However, the dataset used for training and testing the models are not provided. Utilizing a total of 23 different, it evaluates the performance and effectiveness of each model.

- Diagnosis of Headaches Types Using artificial neural networks and Bayesian networks [30]: Despite the unavailability of the dataset, the study utilized the WEKA® software (Waikato Environment for Knowledge Analysis), a popular open-source software for machine learning and data mining, providing a comprehensive environment for knowledge analysis, for testing purposes. The evaluation process involved performing 10-fold cross-validation to assess the performance of the models. Additionally, a comparison was conducted between two different structural forms of the input to analyze their impact on the models' outcomes.

- Automatic migraine classification using artificial neural networks [31]: well-documented work for headache classification: the study has made all necessary resources available, including the dataset, code repository, and features used. It provides detailed explanations of the hyperparameters used for the Multilayer Perceptron (MLP) model and the parameters utilized for other models in the analysis. Cross-validation techniques were employed for the feature selection process and to ensure a reliable model performance. This approach

helped reduce the number of features utilized and facilitated the testing and evaluation of the model's effectiveness. Finally, the performance of the MLP model was evaluated by comparing it with other models to determine its accuracy and precision to alternative approaches.

- Migraine Diagnosis by Using Artificial Neural Networks and Decision Trees [32]: Uses a decision tree was constructed using the Gini algorithm in RapidMiner. To evaluate their performance, a comparison was made with various types of neural networks from previous studies. Additionally, a visual representation of the generated decision tree was provided to facilitate a better understanding of its structure and decision-making process.

- Headache diagnosis with K-Means algorithm [33]: The unsupervised learning technique of k-means clustering, a clustering technique that partitions data into k distinct clusters based on similarity, aiming to minimize the within-cluster variance, was employed to partition the data into three different clusters, which correspond to the three identified classes of headaches. This method allows for classifying data points based on their similarities and helps understand the underlying patterns and structures within the dataset.

- Diagnosis of Common Headaches Using Hybrid Expert-Based Systems [34]: A comparison was conducted between a simple method employing 123 rules of fuzzy if-then questions and more complex techniques such as Support Vector Machines and Multi-Layer Perceptron. The aim was to demonstrate that a complex system is not always necessary for accurate diagnosis, as the three methods exhibited similar levels of accuracy. This highlights the effectiveness of the simple approach in achieving comparable results to more sophisticated algorithms.

- A decision support system to follow up and diagnose primary headache patients using semantically enriched data [35]: The project involves creating a mobile application that enables the collection of essential patient data for improved diagnosis. It incorporates an automated diagnosis support module that leverages expert knowledge and semantic annotations on the data to generate a decision tree. This decision tree provides interpretable insights and assists neurologists in formulating precise and accurate diagnoses. Additionally, a web application is developed to efficiently interpret captured data and visualize the insights generated by the automated diagnosis support module.

- [36] Diagnosis of Headache using Artificial Neural Networks: accuracy of a neural network model across various treatment scenarios involving modifications to both the input and output data. Additionally, different hyperparameters are adjusted to examine their impact on the model's performance. The objective is to assess the effectiveness of different configurations and identify the optimal settings that yield the highest accuracy.

2.2.4 Knowledge-Based Systems

Knowledge-Based Systems use pre-defined knowledge in the form of rules, facts, and heuristics to solve problems and make decisions. In KBS, human expertise is captured in the form of a knowledge base, and the system uses that knowledge to reason, infer, and provide solutions to problems.

As seen in a study conducted by Aljaaf et al. [37], the first step involves defining the specific classes of headaches that will be considered in the diagnostic process using standardized terminology. Once the categories are identified, the researchers analyze the distinctive characteristics associated with each selected type of headache.

To facilitate the classification of primary headaches, the researchers develop procedural functions that aid in the diagnostic process. These functions are designed to automate the diagnosis, eliminating the need for extensive medical expertise. By following the proper execution of these functions, anyone can obtain a headache diagnosis with accuracy.

This approach simplifies the diagnostic procedure, making it accessible to a broader range of individuals and reducing the reliance on specialized medical professionals. The automated diagnostic system streamlines the process, providing users with a diagnosis based on the executed functions.

In [38], the authors build an application, using Delphi, that utilizes a rule-based expert system approach to diagnose headaches based on patient-reported symptoms.

The application provides a user-friendly interface where patients can input their symptoms into a diary. Using the information collected, the app applies a series of conditional statements or rules to determine the type of headache. For example, if a patient reports symptoms X and Y, the app may generate a diagnosis of "Migraine with Aura" based on the corresponding rule.

This approach allows individuals without medical expertise to receive a preliminary diagnosis by running the app and entering their symptoms accurately.

In the study by Mahajan et al. [39], a flowchart-based diagnostic approach was developed to diagnose various diseases, not necessarily limited to headaches. This flow chart follows a sequential process based on significant symptoms reported by the patient.

The diagnostic process begins by considering the significant symptoms presented by the patient. The flow chart then searches for diseases known to cause those specific symptoms. It continues through a cycle, considering additional symptoms, until only one disease remains the likely diagnosis.

It offers a simple approach that can be easily understood. However, developing the underlying logic for such systems can be challenging. Additionally, this approach lacks adaptability, as it always provides the same diagnosis unless the symptoms change, which limits the method.

2.3 Chapter summary

In this chapter, it was shown all the necessary concepts related to headaches needed to understand the scope of the project. It also explored the document containing the globally accepted headache classification. Also, for each type shown, their respective symptoms and diagnosis were explored. Then, the technological concepts needed to understand this project were also explained.

Finally, it presents a thorough literature review on knowledge-based systems and machine-learning methods for headache classification. The review encompassed a wide range of studies and research papers in the field. A simplified consideration was given for each one to quickly understand their relevance for this project and what strong points they offer can be adapted in this study.

Chapter 3

Data Collection

In this chapter, an introduction to the implementations is shown. It begins by outlining the questions to be presented to the patients via a questionnaire. Furthermore, the storage location to collect and store the patients' responses will also be explored. This aspect will be explained in detail, including how the system will function.

Next, the creation of the initial dataset utilized for training the models is shown. A description of this process, including the steps in gathering and organizing the data and the limitations associated with this dataset creation process. Some areas for potential improvement in future endeavors are explored too.

3.1 Questionnaire

Dr. Axel Ferreira and Dr. Sandra Moreira from Hospital Pedro Hispano have meticulously crafted the questionnaire used in this study. Drawing upon their extensive expertise and knowledge in diagnosing headaches, they have chosen a thorough set of questions considered vital for the precise diagnosis of the headaches outlined in Chapter 2.

The questionnaire, presented in Table 3.1, is composed of 40 questions and three sub-questions that only need to be answered if the previous questions are responded to Yes. It is important to note that none of the questions are mandatory. If a patient is uncertain about how to respond to a particular question, it is recommended that they refrain from answering. This approach acknowledges the possibility of missing values in the dataset. As shown next, the models will be trained using examples containing missing values. This aspect can be seen as both a limitation and an advantage.

On the one hand, the presence of missing values may potentially reduce the accuracy of the models. On the other hand, it enables the models to correctly diagnose cases even when the patients have not answered specific questions. This flexibility in handling missing data allows for a more robust and practical application of the models in real-world scenarios.

Table 3.1: Questionnaire Questions

Id	Questions
1	Are your headaches always the same?
2	Where is your headache located/felt?
3	How intense is your headache?
4	How long does your headache usually last? (if you have multiple periods of pain during the day, choose the period with the longest duration and highest intensity)
4.1	How many times do you have a headache during the day?
5	Does your headache frequently occur at a specific time of day?
5.1	(if yes) - Do you wake up with a headache?
5.2	(if yes) - At what time of day do you usually feel the headache?
6	How frequently do you experience headaches in a month?
7	How long does it take for your headache to reach maximum intensity?
8	How would you describe your headache?
9	Does your headache worsen with physical activity?
10	Does stress increase the likelihood/severity of your headaches?
11	Does drinking alcohol increase the likelihood/severity of your headaches?
12	Do weather changes increase the likelihood/severity of your headaches?
13	Does taking a triptan medication (Zomig, Zolmitriptan, Imigran, Sumatriptan, Naramig) alleviate your headache?
14	Does lack of sleep increase the likelihood of having a headache?
15	Are your headaches accompanied by any of these symptoms, even if not always?
16	Is your headache associated with any of these characteristics, even if not always?
17	Does your headache usually coincide with your menstruation?
18	Does your headache improve with the consumption of coffee?
19	Before or during the onset of your headache, do you experience any of these symptoms?
20	Does your scalp become more sensitive when you have a headache?
21	How many cups of coffee do you usually drink per day?
22	How many days per month do you usually take simple analgesic medication (Paracetamol, Ibuprofen, Diclofenac)?
23	How many days per month do you usually take combined analgesic medication or medication specific to your headache (paracetamol with caffeine, Migretil, triptans - Zomig, Zolmitriptan, Imigran, Sumatriptan, Naramig)?
24	At what age did you start having headaches?
25	Select the symptoms that usually accompany your headache.
26	When you have a headache, do you prefer to stay still in a dark place?
27	When you have a headache, do you become restless and unable to keep still?
28	When you have a headache, do you feel exhausted?
29	Do you use oxygen when you have a headache?
30	Have you used or currently use Indomethacin for your headache, and did it work?
31	Does your headache also include facial pain?
32	Can touching/pressing a specific point provoke your headache/facial pain?
33	Is your headache/facial pain in an area where you usually experience tingling or numbness?
34	Do you usually have periods of at least three months, during the year, without a headache?
35	Weight (g)?
36	Height (cm)?
37	What is your education level?
38	What is your occupation?
39	Are you a smoker?
40	Do you have family members with headaches?

Some questions in Table 3.1 are incomplete without knowing the possible responses. To this information be completed, the ones that necessitate the answers are shown next with their possible responses.

- Question 2 (Single Choice)
    1. Unilateral (one side)
    2. Bilateral (both sides)
    3. Holocranial (entire head)
- Question 3 (Single Choice)
    1. Mild
    2. Moderate
    3. Severe
- Question 4.1 (Single Choice)
    1. Morning
    2. Afternoon
    3. Evening
- Question 15 (Multiple Choice)
    1. Nausea
    2. Vomiting
- Question 16 (Multiple Choice)
    1. Light sensitivity
    2. Sound sensitivity
    3. Smell sensitivity
- Question 25 (Multiple Choice)
    1. Tearing of the eyes
    2. Redness of the eye
    3. Redness of the face
    4. Drooping eyelid on one side
    5. Nasal congestion
    6. Runny nose
    7. Swollen eye

8. Reduced size of the pupil (black part of the eye)
9. Abdominal pain
10. Sweating from the forehead/face

- Question 37 (Single Choice)

1. Primary School
2. Middle School (2nd Cycle)
3. Middle School (3rd Cycle)
4. High School
5. College

3.2 Dataset Description

The patient's questionnaire responses will be saved in a Google Sheets file. There are six types of questions. For each one, the response to it is shown afterward. If a reply is not given for all kinds, the answer will be saved as a 0 in the dataset.

- **Yes or No Questions** - if the question is responded with no, it is saved as 1; the remaining possible responses are saved like:

1. No
2. Yes
3. Never had (applicable for specific questions like 11 and 13)

- **Single Choice Questions** - what is saved in the dataset is the number of the option chosen; for example, for question 3:

1. Unilateral (one side)
2. Bilateral (both sides)
3. Holocranial (entire head)

- **Multiple Choice Questions** - the number saved in the dataset corresponds to a vector containing whether any choices were selected. "1" indicates not selected, while "2" shows selected. The first digit from the left corresponds to the first option of the question, and the rest of the numbers correspond to the rest of the possibilities of the question:

  – For question 16, for example, a response saved as 221 shows that the first and second options were chosen while the third option wasn't.

- **Duration Questions**: The duration is saved in seconds, corresponding to the time given by the patient.

- **Counter Questions**: The amount number is saved in the dataset.
- **Open Questions**: The answers are saved as the text provided.

Each class of headache will also be saved as a numerical code in the dataset. We can see in Table 3.2 every headache class used and their corresponding code.

Table 3.2: Headaches Classes used in this study

Id	Headache Class
1	Episodic Migraine without Aura (EMwA)
2	Chronic Migraine without Aura (CMwA)
3	Episodic Migraine with Aura (EMwoA)
4	Chronic Migraine with Aura (CMwoA)
5	Infrequent episodic Tension-Type Headache (ITTH)
6	Frequent episodic Tension-Type Headache (FTTH)
7	Chronic Tension-Type Headache (CTTH)
8	Medication Overuse Headache (MOH)
9	Trigeminal neuralgia (TN)
10	Episodic Cluster Headache (ECH)
11	Chronic Cluster Headache (CCH)
12	Other Headaches Disorders

3.3 Dataset creation

A comprehensive dataset was generated following the established diagnostic criteria outlined in Chapter 2.

For instance, in the case of migraine without aura, the diagnosis requires meeting four specific criteria. Firstly, a minimum of five episodes must align with the defined criteria. Secondly, the duration of the headaches should fall within the range of 4 to 72 hours, either untreated or when treatment attempts are proven ineffective. Thirdly, the headache must exhibit at least two of four specified characteristics, including unilateral localization, pulsating pain sensation, moderate to intense pain levels, and aggravation or avoidance of routine physical activities such as tying shoes or walking. Lastly, during the headache, at least one of the following should be observed: either or both nausea and vomiting, or either or both heightened sensitivity to light (photophobia) and increased sensitivity to sound (phonophobia).

Combining the features that satisfy these criteria generated numerous examples to accurately represent positive predictions for the "Migraine without aura" class. It is worth noting that there might be additional factors not explicitly mentioned in the ICDH-3 criteria but can still be influential in the positive classification of this headache type. For example, many patients experiencing this type of headache prefer to be in a dark environment [40], leading them to respond "Yes" to Question 26. Although this preference is not directly addressed in the ICDH-3, it can still be considered a contributing factor for a positive diagnosis of migraine without aura.

With careful attention to these considerations, an extensive dataset was meticulously constructed, applying the prescribed methodologies for each of the 12 headache classes. We can see in the Annex A.1 a matrix that shows the possible features that each different type of headache can manifest or have. In constructing the dataset, some random values were also inserted as outliers.

The resulting dataset consists in Table 3.3. As can be seen, the number of samples for each class is very unbalanced, which can lead to problems when using it to train the models.

Table 3.3: Dataset Distribution

Id	Amount of Samples
1	175
2	175
3	415
4	369
5	36
6	36
7	72
8	12
9	39
10	515
11	515
12	22
Total	2381

3.4 Limitations

A "synthetic dataset" refers to a dataset that is artificially generated. Whether the data is generated through algorithmic processes or manually created, the primary criterion is that it emulates accurate data or adheres to predetermined patterns and rules. Therefore, if a dataset is made to simulate real data or to satisfy specific criteria, it can be classified as a synthetic dataset. This term is commonly used in scientific research to denote intentionally fabricated datasets to mimic authentic data distributions or to serve as controlled benchmarks for various analyses and modeling techniques [41].

Because of this, some limitations are present:

- **Lack of Real-World Variability**: Synthetic datasets may not capture real-world data's full complexity and variability, limiting their relevance.
- **Assumptions and Biases**: Synthetic datasets rely on assumptions, introducing potential biases and inaccuracies.
- **Limited Real Data Insights**: Synthetic datasets may miss important characteristics and relationships in real data.

- **Difficulty in Capturing Complex Dependencies**: Generating synthetic datasets that accurately represent complex dependencies is challenging.
- **Overfitting Risks** Synthetic datasets alone may be prone to overfitting and perform poorly on real data.

3.5 My Health Diary and Future Work

At first, the plan was to modify an existing application called My Health Diary, seen in Figure 3.1, to incorporate the questionnaire and then automatically give a preliminary diagnosis that the neurologist would concur or alter. This application was developed by a team of students of Faculdade de Engenharia da Universidade do Porto. Since then, it has been maintained by members of the same faculty, as its code is private.

This application was already used for patients of Hospital Pedro Hispano, which allowed them to use it as a diary for them to put their symptoms [42]. However, there have been some problems with its maintenance, and it has been down for some time. Due to this and due to technical difficulties and code compatibility issues, the questionnaire and diagnosis tool created could not be added to the application.

As an alternative, a Google Forms questionnaire was used to collect patient data instead. Although this method lacks the close patient monitoring that the modified application would have offered, it facilitated the headache questionnaire. The User Interface in Google Scholar is straightforward, making it easy to use and store patient answers in a Google Sheet.

For future work, another attempt can be made to implement the necessary changes in the application and check if they can be successfully executed. Additionally, real patient data, once classified by neurologists, can be added to the training dataset. This inclusion of real-world data can address the challenge of the synthetic dataset's inability to mimic real-world scenarios accurately.

3.6 Chapter Summary

This chapter showed the development and implementation of a questionnaire for patients at Serviço de Neurologia. The questionnaire is designed to collect valuable data on headaches and related symptoms. It covered the questionnaire's content, dataset description, creation process, limitations, and future work.

The questionnaire is a collaborative effort with medical experts to ensure its relevance. The dataset description highlights the collected variables and their significance for headache analysis.

Dataset creation involved attempts to modify an existing application, MyHealthDiary, but technical difficulties led to using Google Forms for data collection. Although patient follow-up was limited, the questionnaire was effectively administered, and responses were saved in a Google Sheet spreadsheet.

Figure 3.1: My Health Diary UI

The chapter acknowledges limitations such as data variability, potential biases, and challenges in capturing complex dependencies.

Future work involves reattempting the implementation of desired changes to the MyHealthDiary application and incorporating real patient data verified by neurologists to address limitations.

Overall, this chapter lays the foundation for subsequent analyses and machine learning models, contributing to headache diagnosis and treatment advancements.

Chapter 4

Development and Evaluation

In this chapter, different models will be implemented. After all, models were implemented, many experiences were conducted so then comparisons could be made between them.

With the neural networks, more experiments were done to achieve the best hyperparameters.

4.1 Model Aplication

Every model was implemented similarly. First, a CSV containing the dataset is loaded using the Pandas Library. The features are extracted into a variable X, and the target variables(classes) are stored into another variable Y.

The features are then preprocessed using StandardScaler, a function from the scikit-learn library, to standardize numerical features by transforming them to have zero mean and unit variance. It brings all features onto the same scale, ensuring each contributes equally to the learning process. Standardizing the features ensures that each feature contributes similarly to the learning process and prevents certain features from dominating based on their scale.

The models are evaluated using k-fold cross-validation, dividing the dataset into k-folds. Each model is trained and tested on different combinations of these folds. This helps to assess the model's performance more robustly by using multiple train-test splits of the data.

During cross-validation, the models make predictions on the data, and evaluation metrics such as precision and accuracy are calculated based on these predictions. This allows us to assess how well each model performs on different subsets of the data.

By leveraging cross-validation, we can better understand the model's performance and compare its effectiveness in predicting the class labels. This approach allows the selection of the most suitable model for the given task, as it considers the model's performance across multiple folds, providing a more reliable and robust assessment.

The models used can be seen in Table 4.1 and their respective evaluation metrics. All of these models are from the library scikit-learn. All these values are the average of 5 times running each model. We can see in Annex B all results for every try and the margin of error these values can have.

All evaluation metrics results shown next range from '0', which means the model wrongly classified every sample, to '1', which means the model classified everything correctly.

Table 4.1: Evaluation metrics e using five folds cross-validation

Model	Accuracy	Precision.	Recall	F1 Score
LR	0.9544	0.8194	0.7931	0.8038
DT	0.9833	0.8879	0.8733	0.8792
k-NN	0.8242	0.6832	0.6142	0.6347
RF	0.9858	0.8907	0.8806	0.8841
NB	0.8402	0.7851	0.7212	0.7236
SVM	0.9266	0.6930	0.6938	0.6892

Without the use of the StandardScaler function, some models presented worse results. As seen in Table 4.2, only the methods based on decision trees, like the Decision Tree itself and Random Forest, were unaffected.

Table 4.2: Model evaluation metrics without using Preprocessed Data

Model	Accuracy
LR	0.2180
DT	0.9821
k-NN	0.4480
RF	0.9853
NB	0.4858
SVM	0.2883

4.1.1 Decision Tree Graph

Figure 4.1 shows the first three levels of the decision tree used. To understand it better, an explanation is required to see what the tree represents. This will be useful when analyzing the whole decision tree in Annex C.

Each box (or node) in the decision tree represents a decision or a split based on a particular feature or attribute. The branches that originate from a node represent the possible outcomes or paths the data can take based on the decision made at that node. The leaf nodes, the terminal nodes of the tree, represent the final predicted classes or outcomes.

Each node presents some values, and to better understand what each one of them represents, let's take a look at the first one (top one):

- **P34** $\leq$ **1.5** : This part indicates the splitting condition of the node. It suggests that the decision tree is splitting the data based on the feature P34, which corresponds to the answer to question 34, 'Do you usually have periods of at least three months, during the year, without a headache?'. In this case, the condition is that the value of that feature should be less than or equal to 1.5. As said in Chapter 3, if answered 'No', it will be saved as '1', and if answered 'Yes', it will be saved as '2'. So as seen in Figure 4.1, if the condition is less than or

equal to 1.5 is false, which means the answer is 'Yes', leads to a leaf node that corresponds to class '10' Episodic Cluster Headache. If the condition is less than or equal to 1.5 is true it follows the left path till a leaf node is achieved based on other splitting decisions.

- **gini = 0.84:** This part represents the Gini impurity of the node. It measures node impurity or the probability of misclassifying a randomly selected sample from the node. A lower Gini impurity indicates a more pure node.
- **samples = 1904:** This part indicates the number of samples present in the node. In this case, the node contains 1904 samples.
- **value = [145, 137, 329, 305, 27, 28, 61, 10, 27, 422, 394, 19]:** This part provides the distribution of samples in the node across different classes. Each value in the list represents the count of examples belonging to a specific category.
- **class = 10:** It refers to the predicted class or category for the samples that reach that particular node. The decision tree algorithm assigns a class label or category to each leaf node based on the majority class of the samples that reach that node. In this case, the samples at this node are predicted to belong to class 12 based on the decision tree's training and splitting process. Each category uses a different box color for a more straightforward perception.

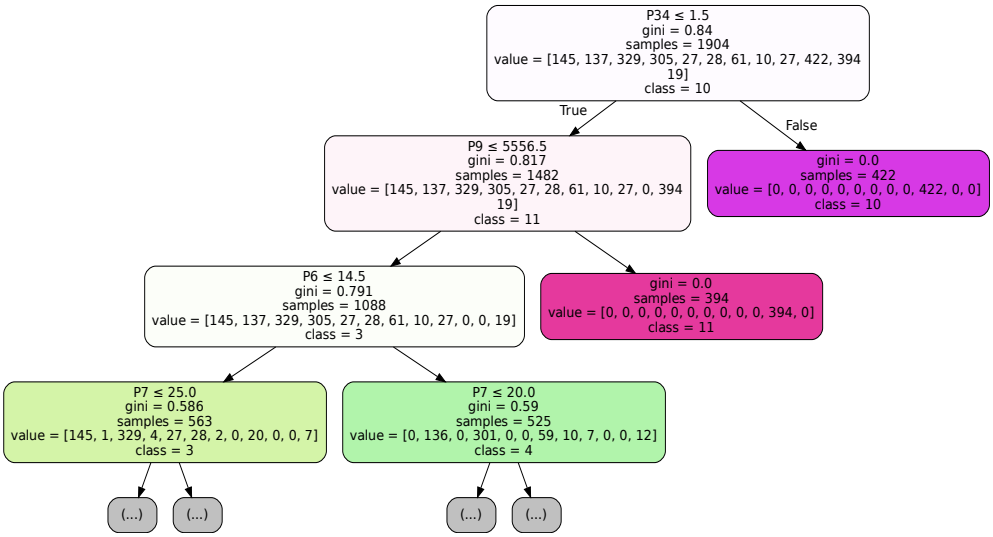

Figure 4.1: Example of created Decision Tree First Levels

4.2 Neural Networks

Every one of the following neural networks was implemented similarly. Initially, a CSV file containing the dataset is loaded using the Panda Library. The features are extracted from a variable, and the target variables(classes) are stored in another variable.

Once the dataset is prepared, the next step is to define the neural network's architecture. This involves deciding on the structure and characteristics of the network. The number of layers in the network, the number of neurons in each layer, the activation functions to be used, and any other architectural choices specific to the framework used (such as scikit-learn, TensorFlow, or PyTorch). The architecture defines how the network is organized and how information flows through it, ultimately influencing its ability to learn and make predictions. The choices made during this step significantly impact the network's performance and ability to solve the given task effectively.

K-fold cross-validation was used to estimate the performance of a model. It involves splitting the dataset into k subsets or "folds". The model is trained and evaluated k times, with each fold taking turns as the validation set while the rest are used for training. This allows the model to be tested on different parts of the data.

During each iteration, the model is trained on the training set and evaluated on the validation set. Performance metrics like accuracy, precision, recall, and F1 score are calculated based on the model's predictions on the validation set. This process is repeated k times with a different fold as the validation set.

After all iterations, the performance metrics from each fold are averaged to obtain a more reliable assessment of the model's overall performance. This helps to ensure that a specific train-test split of the data does not overly influence the model's performance.

K-fold cross-validation provides a robust way to evaluate a model's performance and assess its generalization ability to unseen data. It is a widely used machine learning technique to understand better how well a model is likely to perform in practice. It was always used in the following experiments.

- **Scikit-learn**: The scikit-learn neural network is a flexible implementation based on the Multi-Layer Perceptron (MLP) algorithm. It consists of an input layer, one or more hidden layers, and an output layer. The architecture can be customized by specifying the number of neurons in each layer.

  The hyperparameters can be tuned to optimize the model's performance by experimenting with different values. By adjusting the number of neurons, activation functions, optimization algorithms, regularization, learning rate, and random seed, the neural network can be customized for specific tasks.

  The scikit-learn neural network provides an easy-to-use and adaptable solution for building and training neural network models. Its flexibility in architecture and hyperparameter customization allows for effective model optimization based on the specific requirements of the

problem.

- **TensorFlow**: With TensorFlow, the neural network architecture is defined by creating a sequential model. The model is built layer by layer. After defining the architecture, the model is compiled to configure the learning process. The compilation includes specifying the optimizer, loss function, and evaluation metrics.

  Once the model is defined and compiled, it is trained using the training data. The training involves feeding the model with input data (features) and corresponding target labels (classes). The model's parameters are iteratively optimized to minimize the defined loss function. The number of training epochs determines how many times the model will see the entire training dataset, and the batch size determines the number of samples used in each optimization step.

- **PyTorch**: The PyTorch neural network is a powerful tool for building and training neural network models. It provides a straightforward architecture definition by specifying the layers and their configurations. A customized network structure can be created quickly.

Table 4.3: Hyperparameters available for changing in NN libraries

Hyperparameter	Scikit-learn	TensorFlow	PyTorch
Number of neurons in each layer	X	X	X
Number of layers	X	X	X
Activation functions	X	X	X
Optimizer		X	X
Loss function	X	X	X
Learning rate	X	X	X
Random state	X	X	X
Number of epochs		X	X
Batch size		X	X
Regularization techniques	X	X	X

By observing Table 4.3, We can see some parameters that all three libraries allow to make changes before running the model. With this information, we can compare them using equal hyperparameters when possible.

4.2.1 Neural Networks Experiments

Table 4.4 shows the network evaluation metrics of three different neural networks. The neural networks that were used used the following libraries:

- scikit-learn: The neural network architecture consists of a single hidden layer with 64 neurons. The activation function used is ReLU, and the solver algorithm is Adam. Other hyperparameters include alpha (regularization parameter) and learning rate set to constant.

- PyTorch: Consists of two fully connected (linear) layers. The first layer has an input size equal to the number of features in the dataset and outputs 64 units. The ReLU activation function is applied after the first layer. The second layer takes the output of the first layer and maps it to the number of unique classes, which represents the number of output classes. The neural network is trained using the Adam optimizer with a learning rate 0.001. The training is performed for a specified number of epochs (10 in this case) using mini-batch gradient descent. The CrossEntropyLoss function is the loss function to measure the difference between the predicted and actual class labels.

- Tensorflow: build a neural network with two hidden layers. The first hidden layer has 64 neurons, and the second has 64 neurons. The output layer has several units corresponding to the number of classes in the target variable. The ReLU activation function is applied to the hidden layers. Adam optimizer is used with a learning rate of 0.001. The model uses the CrossEntropyLoss loss function. Finally, the model is trained for ten epochs using mini-batch gradient descent. The batch size is set to 36, meaning the model is updated after processing each batch of 36 samples.

Table 4.4: Model evaluation metrics of the neural networks from different libraries

Model	Accuracy	Precision.	Recall	F1 Score
scikik-learn	0.9512	0.8236	0.8008	0.8106
PyTorch	0.9047	0.6970	0.6699	0.6627
Tensorflow	0.9774	0.9302	0.9190	0.9142

By looking at this, we can see that the neural network that uses the library TensorFlow could obtain a relevant advantage on all metrics used in relation to the other two networks. To see how each hyperparameter affects the final metric results, this network will be used to see if removing some knowledge from the experiments is possible.

Table 4.5: Neurons per Level Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
16 neurons	0.9484	0.8029	0.7973	0.7870
80 neurons	0.9736	0.8943	0.9102	0.8963
196 neurons	0.9769	0.9174	0.9368	0.9198

Table 4.5 shows that, as the number of neurons grows. However, the table doesn't show that the time needed for the neural network to run also grows since more neurons require more calculations and additional time as a consequence.

In Table 4.6, all four metrics also grow as the number of different layers increases. As the layers' diversity increases, so does the evaluation metrics. Like the neurons, as the amount of layers grows, so does the time necessary for the methods to run.

ReLU (Rectified Linear Unit) is an activation function commonly used in neural networks. It offers several advantages, such as being efficient and easy to implement. ReLU also helps address

Table 4.6: Layers per Network Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
1 layer	0.7233	0.4307	0.3458	0.3583
2 different layers	0.9710	0.8926	0.8939	0.8843
2 equal layers	0.4267	0.4689	0.2561	0.2561
other 2 equal layers	0.5447	0.3143	0.4105	0.3186
3 different layers	0.9807	0.9395	0.9376	0.9314

Table 4.7: Activation Functions Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
ReLU	0.9702	0.8958	0.8912	0.8835
Sigmoid	0.9597	0.8574	0.8316	0.8315
Tanh	0.9698	0.8826	0.8865	0.8789

the vanishing gradient problem, which can hinder the training process of deep neural networks. It is particularly suitable for deep networks because it prevents gradient vanishing or exploding. However, one drawback of ReLU is the possibility of "dying" neurons, where the neuron becomes inactive and outputs zero for any input less than zero [43].

Sigmoid is another activation function that provides a smooth non-linear transformation. It is often used in binary classification tasks because it maps the input to a probability between 0 and 1. The smoothness of the sigmoid function allows for gradient-based optimization techniques. However, the sigmoid suffers from the vanishing gradient problem, which can lead to slower convergence during training. Additionally, the sigmoid function can "squeeze" the input space, causing gradients to become small for large input values.

Tanh (Hyperbolic Tangent) is similar to the sigmoid function but maps the input to a range between -1 and 1. It also offers a smooth non-linear transformation and is suitable for classification and regression tasks. Like sigmoid, tanh is affected by the vanishing gradient problem. However, tanh has slower convergence compared to ReLU due to shifting the output towards negative values.

By analyzing the metrics results in Table 4.7, the conclusion is that the most suitable one for this problem is ReLu, but Tanh is not far away. Sigmoid is not ideal for this kind of classification.

Table 4.8: Optimizer Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
adam	0.9714	0.8987	0.8947	0.8856
SGD	0.9391	0.7919	0.7580	0.7601
RMSprop	0.9769	0.9386	0.9363	0.9285
Adagrad	0.9576	0.8473	0.8254	0.8250
Adadelta	0.3626	0.2364	0.2428	0.1920

Adam: Popular optimizer that combines concepts of RMSprop and momentum for adaptive learning rates.

Stochastic Gradient Descent (SGD): Classic and efficient optimizer that updates parameters based on mini-batches of data.

RMSprop: Adaptive learning rate optimizer that adjusts the learning rate based on the magnitude of gradients.

Adagrad: Adaptive learning rate optimizer that gives larger updates to less frequently updated parameters.

Adadelta: An improved version of Adagrad that dynamically adapts the learning rate using a fixed-size window of past gradients.

The one that obtained the best results according to Table 4.8 was RMSprop.

Table 4.9: Loss function Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
sparse categorical cross-entropy	0.9677	0.8798	0.8951	0.8758
mean-squared error	0.0886	0.0820	0.0709	0.0606
sparse categorical cross-entropy	0.9698	0.8942	0.9029	0.8891
kullback-Leibler divergence	0.0811	0.0757	0.0713	0.0579

Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values. They are used for regression problems.

Categorical Cross-Entropy: Calculates the difference between predicted class probabilities and true class labels and is used for multi-class classification problems.

Sparse Categorical Cross-Entropy: Similar to categorical cross-entropy, but used when true class labels are provided as integers instead of one-hot encoded vectors and used in classification tasks with many classes.

Kullback-Leibler Divergence (KL Divergence): Measures how one probability distribution diverges from a target distribution. They are used in tasks involving probabilistic models.

The one that obtained the best results according to Table 4.9 was sparse categorical cross-entropy.

Table 4.10: Learning Rate Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
0.1	0.9463	0.8584	0.8410	0.8326
0.01	0.9777	0.9379	0.9320	0.9311
0.001	0.9689	0.8792	0.8896	0.8786
0.0001	0.8762	0.6457	0.6187	0.6196

When the learning rate decreases in a neural network, the convergence becomes slower, leading to a longer training process. Lower learning rates allow for finer adjustments and help prevent overshooting the optimal solution. However, minimal learning rates can prolong training and result in suboptimal solutions. Finding the right balance is essential for efficient training.

The one that obtained the best results according to Table 4.10 was '0.01'.

Changing the random state in a neural network has the following effects:

Table 4.11: Random State Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
12	0.9735	0.9073	0.9100	0.8999
36	0.9744	0.9381	0.9282	0.9279
72	0.9777	0.9151	0.9097	0.9106

- Reproducibility: It ensures consistent results each time the model is run, which is helpful for testing and comparison.
- Weight Initialization: It affects the initial values of the model's weights, potentially impacting convergence and performance.
- Data Shuffling: Changing the random state shuffles the training data differently, influencing the order in which samples are seen during training.
- Data Splits: The random state ensures consistent data split into training, validation, and test sets.

The one that obtained the best results, according to Table 4.11, the one that got the best results was 72.

Table 4.12: Number of Epochs Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
2	0.9710	0.9296	0.9255	0.9172
5	0.9761	0.9278	0.9236	0.9197
10	0.9765	0.9117	0.9201	0.9086
20	0.9761	0.9378	0.9387	0.9246

The number of epochs in neural network training determines how often the model will iterate over the entire training dataset. Fewer epochs may lead to underfitting, where the model fails to capture patterns. More epochs allow the model to learn better, but too many may cause overfitting. The optimal number of epochs depends on the problem and dataset, striking a balance between underfitting and overfitting.

The one that obtained the best results according to Table 4.12 was 10.

Table 4.13: Batch Size Comparison

Hyperparameter	Accuracy	Precision.	Recall	F1 Score
8	0.9693	0.9184	0.9181	0.9062
32	0.9790	0.9349	0.9396	0.9306
96	0.9794	0.9416	0.9425	0.9303

Changing the batch size has these effects:

Training Speed: Larger batch sizes can speed up training using parallel computations but require more memory.

Generalization: Smaller batch sizes can lead to faster convergence and potentially better generalization, while larger batch sizes can smooth out updates and improve generalization.

Memory Usage: Larger batch sizes require more memory to store activations and gradients.

Optimization Landscape: Different batch sizes may converge to different minima in the loss landscape.

Learning Dynamics: Smaller batch sizes introduce stochasticity in gradients, while larger batch sizes provide more stable gradients.

According to Table 4.13, the one that obtained the best results was 96.

4.3 Discussion

Combining the hyperparameters that resulted in the highest scores, the following values were obtained: Test Accuracy: 0.9756, Precision: 0.9127, Recall: 0.8977, and F1 Score: 0.8964.

However, as shown in Table 4.1, these scores are still lower than the ones achieved by Decision Trees and Random Forest. It is important to note that these two methods have a high probability of overfitting when applied to the synthetic dataset used in this study.

Therefore, it can be concluded that when dealing with real-world data, artificial neural networks are more likely to provide better diagnoses than using decision trees or random forest.

Table 4.14: Time Comparison between Models

Model	Time(s)
LR	1.0340
DT	0.0460
k-NN	0.3420
RF	1.8260
NB	0.0400
SVM	0.6960
scikik-learn	17.4440
PyTorch	1.7780
Tensorflow	4.6020

Table 4.14 shows that the fastest models are Naive Bayes and Decision Trees. But even though they can be trained and applied in less than a fraction of a second if a closer look at the neural network is made, the difference in speed is not in a dimension that would disturb whoever will be using it. So it's safe to conclude that even though it offers more safe results, the neural networks won't take too long longer than other more straightforward methods.

Since it is being compared to simpler methods, it's also important to compare it to knowledge-based systems. Neural networks offer several advantages over knowledge-based systems in complex problem domains like headache classification. These are:

- Automatic Feature Learning: Neural networks can automatically learn relevant features from raw data, eliminating manual feature engineering required in knowledge-based systems.
- Adaptability to Complex Data: Neural networks can handle diverse data types and representations, making them suitable for domains like headache classification that involve multiple variables and data sources.
- Data-Driven Learning: Neural networks learn from large datasets, capturing complex patterns and making accurate predictions without relying on predefined rules like knowledge-based systems.
- Handling Uncertainty: Neural networks provide probabilistic outputs, allowing them to handle uncertainty and ambiguity in problem domains where different classes may exhibit overlapping characteristics.

4.4 Limitations

The models developed have a limitation in that they can only classify one type of headache at a time. However, individuals can experience more than one type of headache simultaneously. This limitation is an inherent weakness of this solution.

When dealing with multi-class classification problems like identifying different types of headaches, it is essential to consider the possibility of multiple concurrent headache types. While these models effectively classify individual headache types, they may not accurately capture the complexity of cases involving various headaches occurring simultaneously.

Further enhancements or adaptations to the models could be explored to address this weakness. This could involve modifying the model architecture, incorporating additional features, or developing more advanced classification techniques to handle multiple concurrent classes.

It is crucial to recognize the limitations of any solution and understand the specific context and constraints in which it operates. By acknowledging these weaknesses, we can better identify opportunities for improvement and develop more comprehensive approaches to headache classification.

4.5 Chapter Summary

This chapter showed the extensive experiments conducted to compare and evaluate the performance of different machine learning methods. It explored the resulting decision trees but focused on artificial neural networks. By varying hyperparameters such as architecture, activation functions, optimizers, loss functions, regularization techniques, learning rate, batch size, and random state, the study aimed to identify the most effective configurations. The findings emphasized the

significance of hyperparameter tuning in improving model accuracy and generalization. The experiments provided valuable insights for researchers and practitioners in selecting and optimizing machine learning approaches for optimal performance.

Chapter 5

Conclusions

5.1 Conclusion

In conclusion, this thesis delved into the intricate world of headaches, examining their diverse types, unique characteristics, common symptoms, and the crucial task of classification. On the technological front, the study ventured into machine learning, specifically focusing on supervised classification methods, prominently featuring the powerful tool of neural networks.

This research yielded promising results, showcasing the exceptional performance of the employed models in headache classification. Despite using a synthetic dataset, the achieved accuracy and precision values were remarkably high. Particularly noteworthy was the near-parity of effectiveness between neural networks and established techniques such as decision trees and random forest algorithms. This substantiates the notion that the carefully designed dataset, with its predefined rules, aligned perfectly with the strengths of these classification approaches.

However, it is essential to acknowledge that these impressive outcomes require further validation in real-world scenarios to comprehend their practical value fully. Translating these models into the clinical domain is a crucial next step, as it will enable us to assess their potential to aid neurologists in diagnosing and treating headaches. By integrating these advanced classification techniques into existing healthcare frameworks, we can enhance the overall quality of patient care and augment the expertise of medical professionals.

In summary, this thesis is an enlightening exploration of headaches from medical and technological perspectives. Harnessing the capabilities of machine learning, specifically neural networks, in the classification of headaches opens up exciting avenues for future advancements in the field. The potential impact of this research lies in its ability to empower healthcare practitioners with faster and more accurate preliminary assessments of patients' headache conditions. By seamlessly integrating these classification models into existing platforms, such as the My Health Diary application or popular survey tools like Google Forms, the diagnosis, and management of headaches can be revolutionized, ultimately improving the well-being and treatment outcomes of individuals suffering from this common ailment.

5.2 Future Work

The diagnosis is already complete and functioning. However, it still requires manual execution of commands to obtain the classifications, which are then saved locally in a CSV file.

Shortly, deploying the headache classification system is planned so that when the user submits the questionnaire, they can quickly obtain an initial impression of the type of headache they may have. If the development of the MyHealthDiary application continues to pose difficulties, integrating the classification into Google Forms with the questionnaire could be an option.

References

[1] Raquel Gil-Gouveia and Raquel Miranda. Indirect costs attributed to headache: A nationwide survey of an active working population. *Cephalalgia*, 42(4-5):317–325, 2022. PMID: 34521261.

[2] Peter J Goadsby and Stefan Evers. Headache classification committee of the international headache society (ihs) the international classification of headache disorders, 3rd edition. *Cephalalgia*, 38(1):1–211, 2018. PMID: 29368949.

[3] Stuart Spencer. Global burden of disease 2010 study: a personal reflection. *Global Cardiology Science and Practice*, 2013(2):15, 2013.

[4] Nobuo Araki. Tension-type headache. *Nihon rinsho. Japanese Journal of Clinical Medicine*, 63(10):1742–1746, 2005.

[5] David W Dodick, Todd D Rozen, Peter J Goadsby, and Stephen D Silberstein. Cluster headache. *Cephalalgia*, 20(9):787–803, 2000.

[6] Hans-Christoph Diener, Zaza Katsarava, and Volker Limmroth. Headache attributed to a substance or its withdrawal. In *Handbook of Clinical Neurology*, volume 97, pages 589–599. Elsevier, 2010.

[7] Joanna M Zakrzewska and Mark E Linskey. Trigeminal neuralgia. *Bmj*, 348, 2014.

[8] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

[9] Eric I Knudsen. Supervised learning in the brain. *The Journal of Neuroscience*, 14(7):3985, 1994.

[10] Bashar Rajoub. Supervised and unsupervised learning. In *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, pages 51–89. Elsevier, 2020.

[11] Lean Yu, Shouyang Wang, and K.K. Lai. An integrated data preparation scheme for neural network data analysis. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):217–230, 2006.

[12] Bartosz Krawczyk, Dragan Simić, Svetlana Simić, and Michał Woźniak. Automatic diagnosis of primary headaches by machine learning methods. *Central European Journal of Medicine*, 8(2):157–165, 2013.

[13] Junmo Kwon, Hyebin Lee, Soohyun Cho, Chin-Sang Chung, Mi Ji Lee, and Hyunjin Park. Machine learning-based automated classification of headache disorders using patient-reported questionnaires. *Scientific reports*, 10(1):1–8, 2020.

[14] Sonia Domínguez-Almendros, Nicolás Benítez-Parejo, and Amanda Rocío Gonzalez-Ramirez. Logistic regression models. *Allergologia et immunopathologia*, 39(5):295–305, 2011.

[15] Bernard ME Moret. Decision trees and diagrams. *ACM Computing Surveys (CSUR)*, 14(4):593–623, 1982.

[16] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[17] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. *Ensemble machine learning: Methods and applications*, pages 157–175, 2012.

[18] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

[19] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[20] Neha Gupta et al. Artificial neural network. *Network and Complex Systems*, 3(1):24–28, 2013.

[21] Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Inteligence*, 4:26–30, 01 2016.

[22] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[23] Alexios Koutsoukas, Keith J Monaghan, Xiaoli Li, and Jun Huan. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of cheminformatics*, 9(1):1–13, 2017.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[28] Robert Keight, Ahmed J Aljaaf, Dhiya Al-Jumeily, Abir Jaafar Hussain, Aynur Özge, and Conor Mallucci. An intelligent systems approach to primary headache diagnosis. In *International conference on intelligent computing*, pages 61–72. Springer, 2017.

[29] Ahmad Qawasmeh, Noor Alhusan, Feras Hanandeh, and Maram Al-Atiyat. A high performance system for the diagnosis of headache via hybrid machine learning model. *International Journal of Advanced Computer Science and Applications*, 11(5), 2020.

[30] Amanda Trojan Fenerich, Maria Teresinha Arns Steiner, Julio Cesar Nievola, Karina Borges Mendes, Diego Paolo Tsutsumi, and Bruno Samways dos Santos. Diagnosis of headaches types using artificial neural networks and bayesian networks. *IEEE Latin America Transactions*, 18(01):59–66, 2020.

[31] Paola A Sanchez-Sanchez, José Rafael García-González, and Juan Manuel Rúa Ascar. Automatic migraine classification using artificial neural networks. *F1000Research*, 9, 2020.

[32] Ufuk CELIK, Nilufer YURTAY, and Ziynet PAMUK. Migraine diagnosis by using artificial neural networks and decision tree techniques. *AJIT-e: Bilişim Teknolojileri Online Dergisi*, 5(14):79–90, 2014.

[33] Ufuk Celik, Nilufer Yurtay, and Yuksel Yurtay. Headache diagnosis with k-means algorithm. *Global Journal on Technology*, 1, 2012.

[34] Monire Khayamnia, Mohammadreza Yazdchi, Aghile Heidari, and Mohsen Foroughipour. Diagnosis of common headaches using hybrid expert-based systems. *Journal of medical signals and sensors*, 9(3):174, 2019.

[35] Gilles Vandewiele, Femke De Backere, Kiani Lannoye, Maarten Vanden Berghe, Olivier Janssens, Sofie Van Hoecke, Vincent Keereman, Koen Paemeleire, Femke Ongenae, and Filip De Turck. A decision support system to follow up and diagnose primary headache patients using semantically enriched data. *BMC medical informatics and decision making*, 18(1):1–15, 2018.

[36] Karina Borges Mendes, Ronald Moura Fiuza, and Maria Teresinha Arns Steiner. Diagnosis of headache using artificial neural networks. *J. Comput. Sci*, 10(7):172–178, 2010.

[37] Ahmed J Aljaaf, Conor Mallucci, Dhiya Al-Jumeily, Abir Hussain, Mohamed Alloghani, and Jamila Mustafina. A study of data classification and selection techniques to diagnose headache patients. In *Applications of Big Data Analytics*, pages 121–134. Springer, 2018.

[38] Kim Dremstrup Nielsen, Cuno Rasmussen, and MB Russel. The diagnostic headache diary-a headache expert system. *Studies in health technology and informatics*, pages 149–160, 2000.

[39] Shashank Mahajan and Gaurav Shrivastava. Effective diagnosis of diseases through symptoms using artificial intelligence and neural network. *International Journal of Engineering Research and Applications*, pages 2248–962, 2013.

[40] Donald W Lewis. Pediatric migraine. *Neurologic clinics*, 27(2):481–501, 2009.

[41] Jörg Drechsler. *Synthetic datasets for statistical disclosure control: theory and implementation*, volume 201. Springer Science & Business Media, 2011.

[42] Maria Carolina de Almeida Rosa. Adoption of the" my health diary" platform: a health technology assessment study. *https://hdl.handle.net/10216/135647*, 2021.

[43] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing*, pages 203–224, 2021.

Appendix A

Appendix 1

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2	1 (unilateral)	1 (unilateral)	1 (unilateral)	2 (bilateral)	2 (bilateral)	2 (bilateral)	1,3(uniat. or holocranial)	1 (unilateral)	1 (unilateral)	1 (unilateral)		
3	2, 3(Mod, Sev)	2, 3(Mod, Sev)	2, 3(Mod, Sev)	1,2 (Moderate)	1,2 (Moderate)	1,2 (Moderate)		3 (severe)	3 (severe)	3 (severe)		
4	4-72 hours	4-72 hours	5-60 min per aura (+ 60)	30 min to 7 days	30 min to 7 days	30 min to 7 days	30 min to 7 days	<= 2 minutes	15-180 minutes	15-180 minutes		
4.1									0.5 -> 8	0.5 -> 8	2 (Can happen)	
5											2 (Can happen)	
5.1											2 (Can happen)	
5.2												
6	< 15 days month	>= 15 days mont	< 15 days month	>= 15 days month	<= 1 days month	2-14 days month	>= 15 days month	>= 15 days month				
7			>= 5 minutos	>= 5 minutos								
8	1 (Throbbing)	1 (Throbbing)		4 (Pressure-like)	4 (Pressure-like)	4 (Pressure-like)	4 (Pressure-like)	3,5(Stabbing, Electric)				
9	2 (Can happen)	2 (Can happen)		1 (No)	1 (No)	1 (No)	1 (No)					
10			2 (Can happen)	2 (Can happen)	2 (Can happen)	2 (Can happen)	2 (Can happen)		2 (Can happen)	2 (Can happen)	2 (Can happen)	
11			2 (Can happen)	2 (Can happen)								
12			2 (Can happen)	2 (Can happen)								
13								>=10 month				
14												
15		22	22		11	11	21					
16		22	22		12 or 21	12 or 21	22					
17	2 (10%)	2 (10%)										
18												
19			222	222								
20												
21								>=10 month				
22								>=15 month				
23								>=10 month				
24									20-40 years	20-40 years		
25				2211112211					2212222212	2212222212		
26	2 (yes)	2 (yes)	2 (Can happen)	2 (yes)	2 (yes)	2 (yes)			2 (yes)	2 (yes)		
27												
28			2 (Can happen)	2 (Can happen)								
29											2 (Can happen)	
30											2 (Can happen)	
31			2 (Can happen)	2 (Can happen)				2 (when severe)			2 (Can happen)	
32			2 (Can happen)	2 (Can happen)	2 (Can happen)	2 (Can happen)	2 (Can happen)	2 (Can happen)				
33			2 (yes)	2 (yes)				2 (Can happen)				
34					2 (yes)	2 (yes)			2 (yes)	2 (yes)		
35												
36												
37												
38												
39												
40												

Figure A.1: Possible features per Headache Classs

Appendix B

Appendix 2

Accuracy	LR	DT	k-NN	RF	NB	SVM	scikit-learn	pytorch	tensorflow
1º	0.9533	0.9837	0.8262	0.9879	0.8545	0.927	0.9496	0.9026	0.9752
2º	0.9538	0.9821	0.8151	0.9853	0.8377	0.927	0.948	0.9038	0.9798
3º	0.9543	0.9837	0.8267	0.9863	0.8351	0.9233	0.9517	0.9038	0.9798
4º	0.9564	0.9832	0.8267	0.9832	0.8477	0.9301	0.9559	0.9021	0.9752
5º	0.9543	0.9837	0.8262	0.9863	0.8262	0.9254	0.9506	0.911	0.9769
Average	0.9544	0.9833	0.8242	0.9858	0.8402	0.9266	0.9512	0.9047	0.9774
Margin Error	0.0016	0.0008	0.0058	0.0024	0.0142	0.0034	0.0040	0.0045	0.0023

Precision	LR	DT	k-NN	RF	NB	SVM	scikit-learn	pytorch	tensorflow
1º	0.7786	0.8894	0.7392	0.8902	0.8531	0.7247	0.8533	0.7281	0.9153
2º	0.7897	0.8891	0.6597	0.881	0.7648	0.7205	0.7751	0.6687	0.9402
3º	0.8565	0.8828	0.6664	0.9543	0.7683	0.6437	0.8286	0.7009	0.9422
4º	0.8762	0.896	0.6773	0.8465	0.7738	0.6685	0.8655	0.6871	0.9403
5º	0.796	0.8824	0.6736	0.8815	0.7656	0.7076	0.7957	0.7002	0.913
Average	0.8194	0.8879	0.6832	0.8907	0.7851	0.6930	0.8236	0.6970	0.9302
Margin Error	0.0488	0.0068	0.0398	0.0539	0.0442	0.0405	0.0452	0.0297	0.0146

Recall	LR	DT	k-NN	RF	NB	SVM	scikit-learn	pytorch	tensorflow
1º	0.7477	0.8855	0.6679	0.87	0.8061	0.7227	0.8203	0.6747	0.913
2º	0.7564	0.8607	0.6074	0.8742	0.7041	0.7249	0.7597	0.6607	0.9231
3º	0.829	0.8716	0.6052	0.9467	0.6853	0.6387	0.8067	0.6672	0.9328
4º	0.8477	0.8843	0.6102	0.8424	0.7182	0.6679	0.8378	0.6715	0.9279
5º	0.7845	0.8645	0.5801	0.8696	0.6921	0.7147	0.7793	0.6753	0.8981
Average	0.7931	0.8733	0.6142	0.8806	0.7212	0.6938	0.8008	0.6699	0.9190
Margin Error	0.0500	0.0124	0.0439	0.0522	0.0604	0.0431	0.0391	0.0073	0.0174

F1 Score	LR	DT	k-NN	RF	NB	SVM	scikit-learn	pytorch	tensorflow
1º	0.7608	0.8871	0.6926	0.877	0.8073	0.7182	0.8352	0.6792	0.9022
2º	0.7686	0.8707	0.6257	0.8742	0.7061	0.7189	0.7662	0.6432	0.9228
3º	0.8406	0.8761	0.6245	0.95	0.6843	0.6373	0.8162	0.6625	0.9345
4º	0.8606	0.8896	0.6307	0.8442	0.7213	0.6643	0.8491	0.6601	0.9217
5º	0.7883	0.8725	0.6001	0.8751	0.699	0.7073	0.7864	0.6685	0.8897
Average	0.8038	0.8792	0.6347	0.8841	0.7236	0.6892	0.8106	0.6627	0.9142
Margin Error	0.0499	0.0094	0.0463	0.0529	0.0615	0.0408	0.0415	0.0180	0.0224

Time	LR	DT	k-NN	RF	NB	SVM	scikit-learn	pytorch	tensorflow
1º	0.95	0.05	0.3	1.78	0.03	0.69	12.14	1.65	4.83
2º	1.01	0.04	0.32	2.02	0.05	0.69	17.62	1.74	4.7
3º	0.96	0.05	0.33	1.79	0.04	0.69	17.64	2.02	4.45
4º	0.94	0.04	0.34	1.76	0.04	0.72	22.23	1.85	4.29
5º	1.31	0.05	0.42	1.78	0.04	0.69	17.59	1.63	4.74
Average	1.0340	0.0460	0.3420	1.8260	0.0400	0.6960	17.4440	1.7780	4.6020
Margin Error	0.1850	0.0050	0.0600	0.1300	0.0100	0.0150	5.0450	0.1950	0.2700

Figure B.1: Experiments Results

Appendix C

Appendix 3

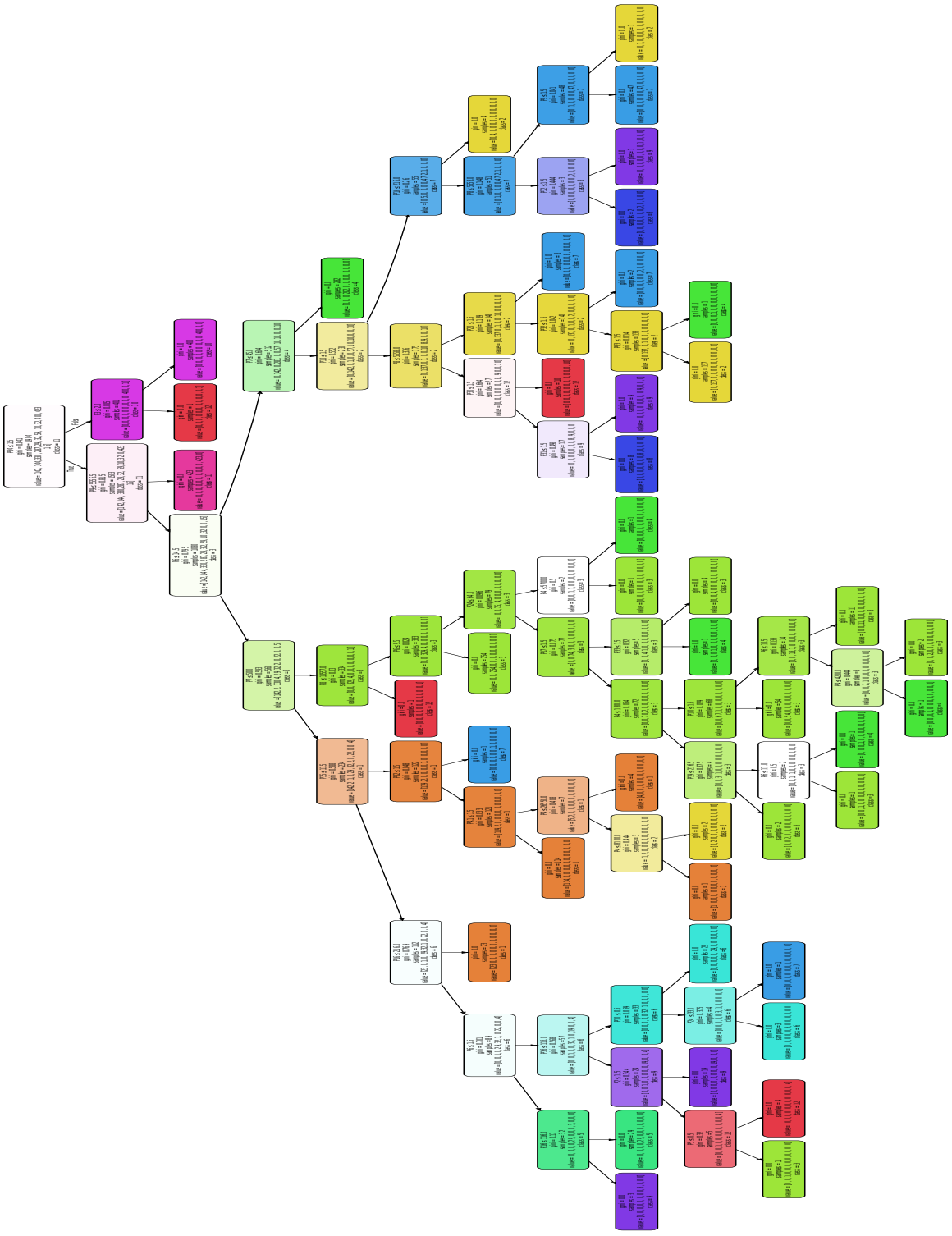

Figure C.1: Decision Tree with Class Distribution