HAW HAMBURG

BACHELORTHESIS
Olímpio Machachane

Supervised Machine Learning Methods for Classification

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Olímpio Machachane

Supervised Machine Learning Methods for Classification

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science European Computer Science*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Olaf Zukunft
Zweitgutachter: Prof. Dr. Zhen Ru Dai

Eingereicht am: 12. June 2020

Olímpio Machachane

Thema der Arbeit

Supervised Machine Learning Methods for Classification

Stichworte

Künstliche Intelligenz, Maschinelles Lernen, Tiefes Lernen, Data Mining, Überwachtes Lernen, Unbeaufsichtigtes Lernen, Halbüberwachtes Lernen, Verstärkungslernen, Einstufung, Regression, Clustering, Modellierungstechniken, Lineare Regression, Logistische Regression, K-nächste Nachbarn, Entscheidungsbäume, Support-Vektor-Maschinen, Künstliche neurale Netzwerke, Zufällige Wälder

Kurzzusammenfassung

Diese These befasst sich mit der Rolle von Methoden des überwachten maschinellen Lernens für die Klassifikationserstellung. Diese ist ein wichtiges Teilgebiet der Umgebung des maschinellen Lernens und ein gutes Beispiel dafür, dass die Informatik mit den verschiedensten wissenschaftlichen Bereichen konvergiert, wodurch sich feststellen lassen kann, dass sie sich mit fast allen Wissenschaften fusionieren lässt.

Im Wesentlichen werden die Grundlagen des maschinellen Lernens behandelt, wobei ähnliche Themen wie künstliche Intelligenz, mehrschichtiges Lernen und Data-Mining verglichen werden, um den Leser zu situieren und einige wichtige Konzepte zu veranschaulichen. Die Arbeit beinhaltet außerdem eine Auseinandersetzung mit der Entstehung verschiedener Arten von Systemen des maschinellen Lernens, wie z.B. beaufsichtigtes Lernen, unbeaufsichtigtes Lernen, halbbeaufsichtigtes Lernen, Klassifikation und Regression. Abschließend werden die Grundlagen des maschinellen Lernens besprochen, wobei Algorithmen des überwachten Lernens wie k-Nearest Neighbours, lineare Regression, logistische Regression, Support Vector Machines (SVMs), Entscheidungsbäume und Random-Forests, Naive Bayes und neuronale Netzwerke behandelt werden.

Die Fallstudie enthält die Implementierung eines Textklassifizierungssystems (ein System, das ein vordefiniertes Thema, Tag oder eine Kategorie eines Textdatensatzes auf der Grundlage seines Inhalts zuweist) mit einigen verschiedenen überwachten Klassifizierungstechniken unter Verwendung des Scikit-Learn-Rahmens. Der Studienzweck dieser Diplomarbeit umfasst eine Aufbereitung der Schritte des maschinellen Lernprozesses von der Datenerfassung bis zur Klassifikation oder Vorhersage. Das Ziel dieser Untersuchungen besteht darin, verschiedene überwachte Klassifizierungstechniken zu vergleichen, zu analysieren und die Vor- und Nachteile, beziehungsweiße Stärken und Schwächen, dieser Techniken in deren spezifischen Situationen zu identifizieren.

Das Ziel dieser Arbeit besteht darin, einen Beitrag zur Entwicklung des maschinellen Lernens als Wissenschaft zu leisten, die Wichtigkeit der Klassifikationsmodelle hervorzuheben und die Diskussion über maschinelles Lernen fortzusetzen, indem eine mögliche Anwendung von überwachten maschinellen Lernverfahren zur Klassifikation unter Verwendung verschiedener überwachter Algorithmen vorgestellt wird.

Olímpio Machachane

Title of Thesis

Supervised Machine Learning Methods for Classification

Keywords

Artificial Intelligence, Machine Learning, Deep Learning, Data Mining, Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Reinforcement Learning, Classification, Regression, Clustering, Modelling Techniques, Linear Regression, Logistic Regression, K-nearest Neighbours, Decision Trees, Support Vector Machines, Artificial Neural Networks, Random Forests

Abstract

This thesis approaches the role of *Supervised Machine Learning Methods for Classification* which is an important fragment of the machine learning environment and a good example of the computer science convergence with various other fields, since it is fairly acceptable to risk to say that it can be fused with almost all the sciences.

Essentially, it covers the basics of the machine learning world briefly comparing similar topics such as Artificial Intelligence, Deep Learning, and Data Mining to situate the reader and clarify some important concepts. It also includes the discussion of the genesis of different types of machine learning systems such as Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Classification and Regression. It ends the basics of machine learning covering Supervised Learning algorithms such as k-Nearest Neighbours, Linear Regression, Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests, Naive Bayes and Neural Networks.

The case study contains the implementation of a Text Classification system (a system that assigns a predefined topic, tag or category of a text dataset based on its content) with some different Supervised Classification techniques using the Scikit-Learn framework. For learning purposes, it covers the steps of Machine Learning process from Gathering Data to the Classification or Prediction. The main purpose of this study is to compare and analyse different supervised classification techniques and identify the advantages and disadvantages, strengths and weaknesses for each of them in a specific situation.

The main purposes of this thesis are to contribute in the development of the Machine Learning as a science, highlight the importance of the classification models, and continue the machine learning discussion, showing a possible application of supervised machine learning methods for classification using distinct supervised algorithms.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Introduction

1.1 Motivation

Socio-economically speaking, the world has seen different ages such as the hunter-gatherer age, the agrarian age, the industrial age and now the humanity is living in the information age also known as the computer age or digital age[51]. The information age, as the name suggests, is based on the information which is directly related with the technology.

The emergence of the internet, together with the development of the computer science, originated large amounts of data liable to change rapidly and unpredictably. Data (or simply facts), in its essence, is anything that can be used to generate information and it plays a very important role on the process of learning. The human brain learns from gathering data, then collecting and grouping the data about a particular subject to generate information, then connecting the information to achieve knowledge (familiarity or understanding gained through experience or study) and ultimately wisdom which can be described as the ability to make right and accurate decisions based on the existing knowledge[20].

The development of the human-being is largely due to the ability to extract knowledge and wisdom from data. In the information era, the internet became filled with multiple data and it was necessary to teach the machines to learn and process the information as human brains would do to improve the decision making process in terms of performance and more, hence the term machine learning.

The motivation for writing this thesis paper comes from a personal belief that machine learning is the future of the technology. Since it allows machines to learn and act in a similar way to humans, although it is already changing the world in different ways considering the fact that it can be applied in different fields of the academy, based on the plausible results, it has been clear that it has a lot of potential to grow and continue to make the difference in the long term.

1.2 Overview

In the first place, this thesis begins with a brief history of Machine Learning highlighting the most important milestones of the journey. Subsequently, it provides the definition of the Machine Learning concept describing the data science field as well and compares it with similar concepts describing the difference between the three of the main concepts of data science, namely Artificial Intelligence, Machine Learning and Deep Learning.

Next, focusing the attention back to Machine Learning, it provides the definition of different types of Machine Learning such as Supervised Learning, Unsupervised Learning, Semi-supervised Learning and Reinforcement Learning.

After raising awareness of the types of Machine Learning, in the sequence, different types of Supervised Learning approaches are described, particularly Classification and Regression. Also, within the Supervised Learning environment, different Supervised Learning modelling techniques are highlighted, specifically Linear Regression, Logistic Regression, k-Nearest Neighbours, Decision Trees, Support Vector Machines (SVMs), Artificial Networks, Naive Bayes and Random Forests.

The Supervised Learning modelling techniques previously mentioned are, subsequently, compared and technically analysed in the Classification point of view.

After the analysis, some of the techniques are tested in the case study using the Scikit-Learn framework. The case study consists on Text Classification, therefore, some of the techniques are used to assign predefined different topics to corresponding text datasets based on the content. After the experiments, comes the evaluation of the results and consequently the conclusions of the techniques used in the case study process based on the performance, accuracy and complexity.

This thesis end with a general conclusion about the Supervised Learning and Classification techniques and a possible description of the future of the Machine Learning field as a developing science.

1.3 Goals

The main purpose of this research work is to implement and perform a Text classifier using supervised machine learning techniques for classification with the aid of the Scikit-Learn framework and conclusively decide which is the optimal strategy in terms of performance, accuracy and simplicity. In other words, the primary question to be answered in this research work is "how to classify a random text dataset by topic using the content of the text?", and secondly this work also tries to answer the following questions: "what is the best technique for this type of classification problems?", and subsequently "why the other techniques are not appropriate for this type of classification problem?".

For the first question the main purpose is to perform different supervised algorithms using Scikit-Learn and for the second question the major learning objective is to come to a proper conclusion concerning supervised classification algorithms in order to be able to help the reader to decide what is the most appropriate technique to take into account in future similar experiences.

The first question is very important since it is more technical and practical, however, the second and third questions are more analytical and in certain situations might be slightly subjective since it leads to a decision-making type of conclusion which is ultimately the essence of machine learning.

2 Brief History

This chapter, essentially based on the information available at [21] , aims to briefly point out the history of Machine Learning in a timeline-based approach.

Machine Learning plays an important role on today's prediction and decision-making programs. It represents a powerful mechanism that can help big, medium and small-sized enterprises to be more profitable and accurate [17]. It can be fused with almost any science since its main feature is to perform the human brain personified in a machine [22]. Using sample data, also known as "training data", Machine Learning uses algorithms to build learning models that can be standardly used in any other similar situations to decide without being previously explicitly programmed.

Machine Learning can be classified as a subset of Artificial Intelligence [59]. The term Artificial Intelligence coined by the computer scientist John McCarthy in 1956 is frequently used to describe machines or computers that mimic "cognitive" functions that we as humans associate with the human brain such as learning, predicting, problem solving and more [52]. Nowadays it is often consensual to say that "Artificial Intelligence is the science and engineering of making intelligent machines" et. Al John McCarthy[40].

We can find various definitions from different authors but most of them agree that "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed " et. Al. Arthur Samuel [8].

Following specific steps with a defined methodology, a machine can actually learn to perform accurate decisions based on datasets previously analysed by the same machine. Nowadays, the concept is a reality, in other words, there is a synergy of human brain and machine learning. Machine learning applications include syntactic pattern recognition, natural language processing, search engines, computer vision and machine perception.

In 1949, Donald Hebb (influential psychologist in the area of neuropsychology), in a book titled "The Organization of Behaviour" [29] created a model of brain cell interaction,

presenting his own theories of neural excitement and communication between neurons. His model can be described as a way of altering the relationships between artificial neurons and the changes to individual neurons. According to Hebb, the relationship between two nodes is considered stronger if the nodes are activated at the same time and is considered weaker if they are activated separately. When the nodes tend to be both positive or both negative are described as having strong positive "weights", which is the term used to describe these relationships. Likewise, nodes tending to have opposite weights originate negative weights (e.g. 1 * 1=1, -1x - 1=1, -1 * 1=-1) [21] . "When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell." [29] (et.al Donald Hebb).

In the 1950s, Arthur Samuel (computer scientist, pioneer in the field of computer gaming and artificial intelligence), developed one of the world's first successful self-learning computer program for playing checkers, the Samuel Checkers-playing Program [54]. The program represents a very early demonstration of the fundamental concept of Artificial Intelligence. The fact that the program had a very small amount of computer memory available lead Samuel to implement what is now known as the Alpha-beta pruning search algorithm. The program chooses its move based on a minimax strategy which means it makes the move that optimizes the value of its function, which eventually evolved into the minimax algorithm. Among other mechanisms designed by Samuel, the Samuel Checkers-playing Program recorded/remembered all positions previously seen and combined this with the values of reward function. [53] Arthur Samuel coined the term "Machine Learning" in 1952.

In 1957, Frank Rosenblatt (psychologist notable in the field of artificial intelligence), combined Donald Hebb's model of brain cell interaction with Arthur Samuel's Machine Learning efforts and created the Perceptron. Rather than a program, the perceptron algorithm was originally intended to be a machine. Its first implementation was in software for the IBM 704, it was subsequentially installed in a custom-built hardware called the Mark 1 perceptron, designed for image recognition. This made the software and the algorithms transferable and available for other machines [21].

In 1967, Marcello Pelillo invented the "nearest neighbour rule" for the "Nearest Neighbour Algorithm", it was considered the beginning of basic pattern recognition. It was one of the earliest algorithms used to solve the travelling salesman problem of finding the efficient

route. Using the algorithm, a salesperson starts in a chosen city and is able to visit all the cities by cyclically visiting the nearest cities available.

In the 1960s, it was discovered that using more than one layer in the perceptron could increase the processing power considerably. Since then, the variety of neural networks versions continue to grow. Therefore, it can be said that Feedforward neural networks and backpropagation are concepts born out of the use of the multiple layers.

In the 1970s Backpropagation concept was introduced. Nowadays, usually used in Deep Neural Networks situations, it is an algorithm that describes "the backward propagation of errors". After the error being processed at the output, it is distributed backward through the network's layers for learning purposes. Artificial Neural Networks (ANN), have hidden layers used to respond complex tasks, sometimes outperforming the human brain in terms of finding new patterns.

There is a difference between Machine Learning and Artificial Intelligence. In the late 1970s and early 1980s, AI focused on using logical, knowledge-based approaches instead of algorithms. The goal of the Machine Learning industry shifted from training for AI to solving practical problems providing services using probability theories and statistics. ML flourished in the 1990s and the internet growth was favourable to Machine Learning which led to its success due to the availability of large amounts of digital data and the ability to share data through the internet.

The term Boosting, in Machine Learning, refers to a family of machine learning algorithms that seek to improve weak learners and consequently the predction power and quality. In 1988 – 1989 Michael Kearns and Leslie Gabriel Valiant posed the following question: "Can a set of weak learners create a single strong learner?". In 1990, Robert Schapire first presented the concept of Boosting in a paper titled "The strength of Weak Learnability" and stated that "A set of weak learners can create a single strong learner" [56]. The term Weak Learner, usually refers to a set of rules that are unable to, individually, be strong enough to successfully classify or provide an accurate result. In contrast, Strong learner is a classifier that easily classifies and is well-correlated with true prediction.

Through weighting and evaluation, Boosting algorithms are able to improve weak learners to strong ones and consequently improve prediction accuracy. Historically, AdaBoost is relevant as it was the first algorithm able to work with weak learners.

It can be assumed that the Artificial Intelligence, Machine Learning or Deep Learning worlds are experiencing one of the bests (if not the best) eras now in terms of signifi-

cant development, for example: Speech Recognition has been one of the most successful applications of the world of Machine learning, in fact, it is being done by a Deep Learning method called Long Short-Term Memory (LSTM) – a neural network algorithm introduced in 1997 by Jürgen Schmidhuber and Sepp Hochreiter.

Facial Recognition became a tremendous reality. A dream came trough when in 2014 Facebook developed Deepface – an algorithm that is able to recognize persons in pictures with the same accuracy as human beings.

The present of Machine Learning is being very exciting for the scientific and technology world in general. It is a very powerful and promising field to invest in. The models have become adaptative in continuously learning, therefore, more accurate. Nowadays, Machine Learning is shining in some of the most fascinating industries such as: self-driving vehicles, internet of things, robotics, analytic tools, chatbots and more.

In the business world Machine Learning has been successfully applied in the following forms [21]: Analysing Sales Data (Streamlining the data), Real-Time Mobile Personalization (Promoting the experience), Fraud Detection (Detecting pattern changes), Product Recommendations(Customer personalization), Learning Management Systems (Decision-making programs), Dynamic Pricing (Flexible pricing based on a need or demand), Natural Language Processing (Speaking with humans).

Nowadays, it is accurate to affirm that the industry in general has become increasingly dependant on machine learning techniques to achieve its targets with efficiency.

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed."

- Arthur Samuel

3 Data Science - Artificial Intelligence, Machine Learning and Deep Learning

3.1 Data Science

Data Science is an inter-disciplinary field that has as its main objective the extraction of knowledge and insights from structured or unstructured data. Data Science combines different fields such as statistics, mathematics, computer science and information science to interpret information for decision-making purposes in different areas. In short, it can be assumed that Data Science is ultimately a concept that unify different fields to achieve new, meaningful and valuable knowledge from raw data. [13] As stated by the Dr. Hal Varian, "The ability to take data - to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it - that is going to be a hugely important skill in the next decades"[1] .

Basically, being a multidisciplinary field[2], to achieve its objective, Data Science goes through different steps, each of them corresponding to combinations with different fields. In a nutshell, the main Data Science steps [32] can be described as:

1. Asking the correct questions and analysing the raw data

2. Modelling the data using various complex and efficient algorithms

3. Visualizing the data to get a better perspective

4. Understanding the data to make better decisions and finding the final result

[1] https://datascience.berkeley.edu/about/what-is-data-science/#fn3b

[2] More information about the multidisciplinary nature of data science can be found at: https://www.oreilly.com/library/view/doing-data-science/9781449363871/ch01.html

With the development of the technology, Data Science tend to have computer science techniques as key elements to achieve its goals. [68] Naturally, it has been a successful connection because Computer Science is a study of processes that interact fundamentally with data and originate automated programs. In that fusion (Data Science and Computer Science), Machine Learning is found, providing training systems for the raw data in order to be able to make the machine (usually computers) learn and execute as a human brain automatically [32].

However, before diving deeply in the Machine Learning concept, for learning purposes, for a better understanding of the environment, it is important clarify the difference between Machine Learning, Artificial Intelligence and Deep Learning.

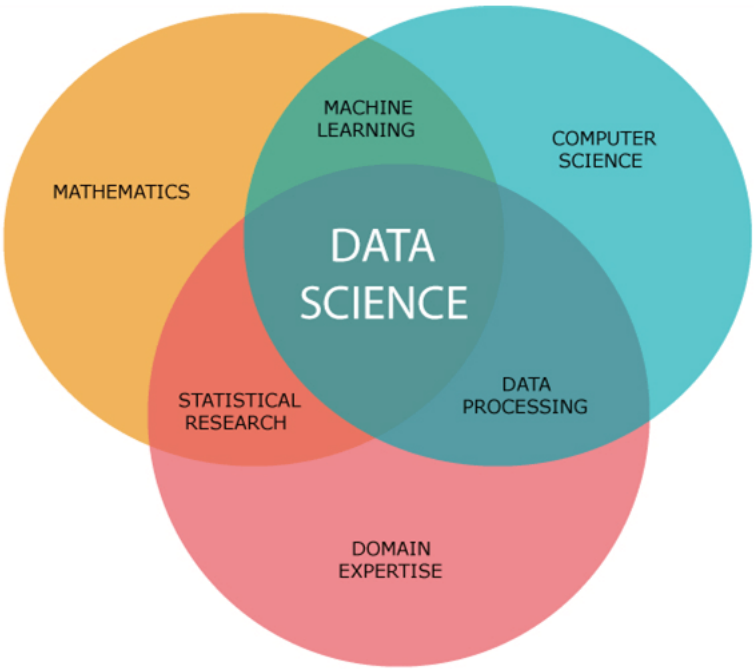

Abbildung 3.1: Definition of Data Science [32]

3.2 Artificial Intelligence

Artificial Intelligence, as the name suggests, is a ramification of Computer Science that has the creation of intelligent machines as its main purpose. Nowadays, it is common to add the term smart to technological machines and devices because they have their own thinking and decision-making abilities equivalent to human-beings. Such devices are able to perceive the environment, make decisions and take actions that maximize their chance of successfully achieving their goals. These capabilities are largely due to the development of Artificial Intelligence as a whole field. According to a PwC report[3], Artificial Intelligence will potentially contribute $15.7 trillion to the global economy by 2030.

Although is has been very hyped recently, Artificial Intelligence as a concept has been around for a long time. It can be assumed that the idea of giving human-like tasks to machines is old, therefore, it can be said that any try to create an automated machine that responds to previously inexistent stimulation (input) is considered a form of and a contribution to the Artificial Intelligence field.

Machine Learning is a subset of Artificial Intelligence and Deep Learning is a subset of Machine Learning, therefore, Artificial Intelligence encompasses both concepts of Machine Learning and Deep Learning. It is also important to note that all Machine Learning (and inherently, Deep Learning) is Artificial Intelligence but not all Artificial Intelligence is Machine Learning.[7]

[3] PwC's Global Artificial Intelligence Study: https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html

From a Computer Science point of view, it can be simply assumed that Artificial Intelligence is an umbrella term (that encompasses Machine Learning and Deep Learning) for any computer program that does something smart.

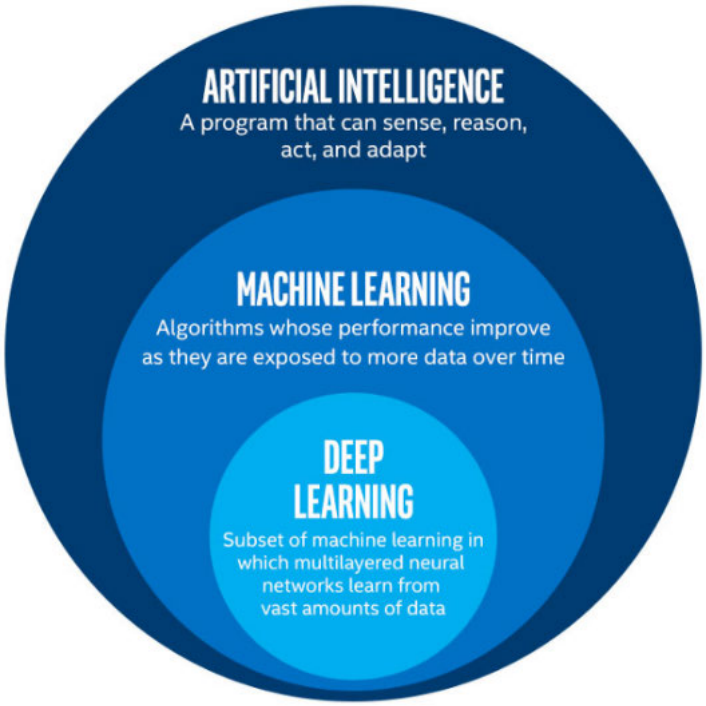

Abbildung 3.2: Correlation between Artificial Intelligence, Machine Learning and Deep Learning [59]

3.3 Machine Learning

Machine Learning is the science (and art) of programming computers so they can learn from data. It is composed of algorithms and statistical models that computer systems use to perform decision-making tasks. Basically, Machine Learning is a subset of Artificial Intelligence that concerns the computational part of the question. It is a simple method to automate the process making it faster and even more accurate than humans in some situations. A Machine Learning system, does not need to be updated frequently as a normal program would need and that is one of the most advantages of working with Machine Learning. [46]

A typical Machine Learning system, only needs to have the raw data available, know the appropriate algorithm to be used for the issue and train the machine to be smart enough to learn for itself. Therefore, given any similar situation, the machine only needs to adjust itself to the specific situation and it learns by itself how to approach it automatically and proceed until it is able to make good decisions.

Nowadays, Machine Learning is frequently used to study large amounts of data and it can contribute to discover new patterns that were not immediately apparent. That phenomena is called Data Mining, in essence, it concerns in extracting relevant information from large amounts of data but it lays the foundation for machine learning [63].

In order to achieve the goal in a Machine Learning context, it is necessary to subdivide the data in two categories: the original dataset and the training dataset for which each example is called training instance (or sample). It is also needed to measure the performance of the training instance and this particular performance measure is called accuracy.

Therefore, a typical solution for a Machine Learning problem is divided in 3 parts: the Task (the goal), the Experience (the training process) and the Performance (the quality of the solution).

Normally, the steps[4] followed in a typical Machine Learning project are [27]:

1. Gather the data
2. Study the data
3. Select a model
4. Train the data on the training data
5. Apply the model to make predictions on new cases (Generalization)

In conclusion, following the steps mentioned and choosing the appropriate model for the problem, a machine learning system can be created. It is also important to highlight the steps 3 and 4 as they are responsible to ensure satisfactory results in terms of performance and accuracy.

[4]The steps on Machine Learning are also precisely explained on: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e

3.4 Deep Learning

Deep Learning is a Machine Learning technique that instruct machines to learn by example as the human brain does. Since it is a subset of Machine Learning, Deep Learning functions in a similar way and that fact can put the two concepts in a blurred situation. Although, they have identical goals, its capabilities are slightly different.

Normal Machine Learning models evolve and improve as they experience similar situation, however, they still need some guidance from time to time. This means that if for any reason, the model does not return an accurate result, the engineer needs to step in to make adjustments.

On the other hand, Deep Learning models use algorithms that allow the system to determine on its own if a result or prediction is accurate or not through its own neural network. As an example, Deep Learning is able to automatically find the right features to use in classification problems (which is not the case in Machine Learning situations where these features need to be provided manually) [25].

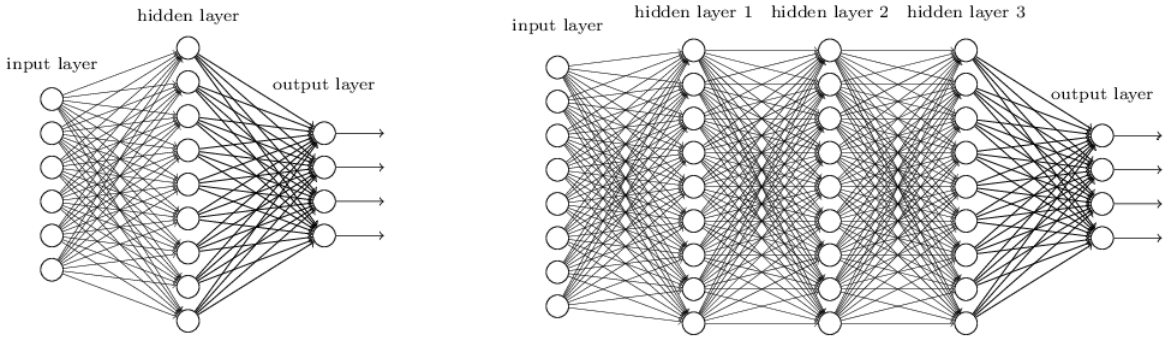

Abbildung 3.3: A non-deep feedforward neural network (left side) and a deep neural network (right side)

A Deep Learning model is composed with different layers of the algorithms, each of them providing different interpretation to the data. As the image 3.3 suggests, the main difference between a non-deep and a deep neural network is the amount of hidden layers applied. That layered structure of algorithms is called *Artificial Neural Network* which will be discussed in more detail later on chapter 5 in section 5.6 (understanding ANN first, is the ideal way to fully understand DNN). For now, it is important to bear in mind that what makes deep neural networks more accurate than the traditional ones is the amount of hidden layers applied.

To make it easy to understand, it can be assumed that the difference between Machine Learning and Deep Learning lies on the fact that Machine Learning models need predefined structured data to learn and Deep Learning models do not necessarily need structured or labelled data to learn as it learns by itself from the beginning using the Artificial Neural Networks as a basis to achieve Deep Neural Networks. In other words, Machine Learning basically learn from similar examples and Deep Learning models learn from the data itself (making its own correlations and assumptions during the learning process).

Since it is not needed to provide similar examples and identifiers to teach the system, its essence, it is plausible to say that Deep Learning is an evolution [30] of Machine Learning.

4 Machine Learning - Types of Systems

Machine Learning systems are first and foremost classified in categories based on three different questions [27]:

1. Is the system trained with human supervision?

2. Can the system learn incrementally on the fly?

3. Does the system simply compare new data points to known data points, or instead detect patterns in the training data and build a predictive model?

In order to answer the mentioned questions, the following sections will demonstrate the sub-categorization particularity of Machine Learning systems describing its important concepts.

4.1 Training methodology

Training methodology of the system pays special attention to the whether the system is trained with human supervision or not.

That question can be answered by subdividing the systems in the following Machine Learning categories:

• Supervised Learning: The training dataset provided to the algorithm includes the desired solution. In these cases, the machine learns from examples as it receives the problem and the desired solutions as primary input. Supervised Learning tasks are mainly based on Classification and Regression problems.

Classification problems consist on assigning or labelling a certain object (dataset) to a previously known and predefined category or tag. The spam filter is a classic example. Basically, system predicts attributes. The machine needs to know when to classify an

email as a spam or not. To achieve that goal, the machine first needs to receive spam emails examples and the elements that makes the email to be considered as a spam. Subsequently, the system is trained to be able to classify similar examples accurately (i.e. an experimental evaluation of a supervised ML classification algorithm on of 1.5 million articles has achieved above 90 percent of successful classification [4]).

In essence, the crucial steps [34] for text classification are:

1. Data collection

2. Text Processing

3. Feature extraction

4. Dimensionality Reduction

5. Classification Techniques

6. Performance evaluation

In a nutshell, it can be said that a classification predictive model implies the categorization of a certain dataset.

Regression problems, on the other hand, consist of predicting target numeric values given a set of features. For example, predicting the price of a computer given a set of information such as brand, processor, memory, screen size, operating system and storage capacity can be considered a typical Regression problem. To train, the system receives many examples of computers including their respective predictors and labels (i.e. prices). Basically, the system predicts continuous numeric values that fits in some specific discrete values (i.e. concepts).

Some of the most important Supervised Learning algorithms to be covered in the next chapter are:

– K-Nearest Neighbours

– Logistic Regression

– Linear Regression

– Decision Trees and Random Forests

– Support Vector Machines (SVMs)

– Neural Networks[1]

It is also fairly important to note that some regression and classification can be combined as some regression algorithms can be used for classification as well and vice versa [27]. For example, outputting the probability of belonging to a specific class (i.e. given a laptop, the system predict chance of it being a windows10 or MacOS).

• Unsupervised Learning: The training dataset is neither labelled nor classified which means that the system tries to learn on its own without guidance. Since there are no categories provided, the system works on grouping unsorted information according to similarities and differences. In these situations, the algorithms of the system perform on the data without prior training, therefore, the result (output) is essentially dependent on the algorithm used.

Even though, Unsupervised Learning techniques can handle more complex processing tasks than supervised learning ones, they can be more unpredictable, therefore, in some cases, erroneous (because of its essence, unsupervised algorithms can create clutter instead of order), or sometimes it can be advantageous as it may find new patterns that have not been previously considered [2].

In short, unsupervised learning can be seen as a system that learns without a teacher[24], a self-taught system. For example, given a basket of different fresh fruits, the task of grouping the fruits based on their similarities without knowing anything about them a priori, is considered a unsupervised learning task.

In most of the times, unsupervised learning algorithms are distributed in three [27] important categories: Clustering, Association Rule Learning and Visualization and Dimensionality Reduction.

• Semi-supervised Learning: As the name suggests, the Semi-supervised Learning, combines Supervised Learning and Unsupervised Learning. In essence, it deals with partially labelled training data. It combines large amounts of unlabelled data (unsupervised) and small amount of labelled data (supervised) during the training process [14].

[1] According to [27] in some scenarios, some Neural Networks architectures can be unsupervised or semisupervised.

A good example of semi-supervised learning system, is a system that given a family photo, it automatically recognizes that the same person A shows up in photos 1, 3,and 5 in the gallery, while the person B shows up in photos 7, 2 and 8 – that is the unsupervised part of the algorithm. Subsequently, the system needs to name each person of the photo and to achieve that goal, it needs to know the names of the family members in advance – that is the supervised part of the algorithm. Consequently, the system becomes able to name every one in every photo of the gallery – a semi-supervised system.

• Reinforcement Learning: The system is fundamentally based in a reward-penalties system. This type of machine learning system lets the system to learn by itself what is the best strategy to use in order to solve a specific problem. This phenomenon is called policy. Basically, the system tries different solutions for a same situation. When the solution is inappropriate, it receives a penalty as feed-back, on the other hand, in a successful situation it receives a reward as feedback. Subsequently, the system updates the policy learning the correct strategy to apply. The steps [27] followed in a reinforcement learning system usually are:

1. Observe

2. Select action using policy

3. Action

4. Get reward or penalty

5. Update policy (learning step)

6. Iterate until an optimal policy is found

All in all, reinforcement learning can be considered a goal-oriented system as it operates in a recompense based system (reward and penalty feedbacks).

4.2 Learning methodology

Learning methodology of the system concentrates on how the system learns in terms of design (whether it learns offline or online).

Essentially, this question have only two existing solutions:

• Batch Learning: The system needs to be trained using all the available data. Since it is typically done offline, this process is usually time-consuming and needs a lot of computing resources. After being trained, the system initiates on real dataset running without learning anymore, it simply implements the algorithms according to what it has learned [58]. This method is also known as offline learning.

When an update is required, a Batch Learning system needs to be trained from scratch with the new dataset type, once the new version is successfully trained, the old version is stopped and is replaced by the new one.

It is important to note that this type of system is limited as if the amount of data to be trained is huge, it might be very hard or even impossible [27] to finish the training process for the same reasons mentioned above.

• Online Learning: The system is trained incrementally, in other words, it receives the data instances in sequences, either individually or by small groups (mini-batches). It is a reactive solution as the new data is learned promptly which makes each learning step fast and cheap.

Online learning is usually used when the system receives data in a continuous flow and need to adapt to change rapidly or automatically (i.e. stock market) [69], in the presence of limited computing resources as it discards the data when not needed anymore (this help to save huge amounts of space) or in the presence of huge datasets that does not fit in the main memory of the machine (out-of-core learning) as the algorithm subdivides the data and trains it sequentially until it has performed on all of the data.

4.3 Purpose of the system

Last but not least, the purpose of the system is also crucial to define its type. Fundamentally, some systems are designed for comparison or pattern definition and prediction purposes.

These options can be explained on the following concepts:

• Instance-based Learning: Instead of just comparing the data to decide the solution for a problem, the system decides through a similarity process. Because of that nature, it is also known as memory-based learning [15]. For example, in the spam filter situation, instance-based learning allows the system to not only flag emails that are identical to known spam emails but also the ones that are very similar to the spam ones already known. It can be achieved programming the spam filter system to flag emails with a certain amount of similarity, which means that in this case, it requires a measure of similarity between two emails.

In short, two important steps can be highlighted in this type of cases: the leaning step and the generalization step.

• Model-based Learning: To generalize from a set of trained datasets examples, the system creates a model of these examples in order to be able to make predictions. The most important step in these types of systems is the model selection step as it reflects the solution type for the problem, which means that all other similar cases are dependent on the selected model (i.e. Linear Regression).

Depending on the selected model it might be necessary to provide a set of predefined parameters and to make sure the selected values takes the best out of the model in terms of performance, it is necessary to apply a performance measure [27] – for this purpose, a utility function (or fitness function) or a cost function can be defined. In short, the utility function measures how good your model performs and the cost function measures how bad your model performs.

Additionally, it is suitable to emphasize that the concepts described above can be freely combined with each other to achieve a specific goal and that is one of the facts that makes Machine Learning more than a science, in fact, some authors even consider it an art.

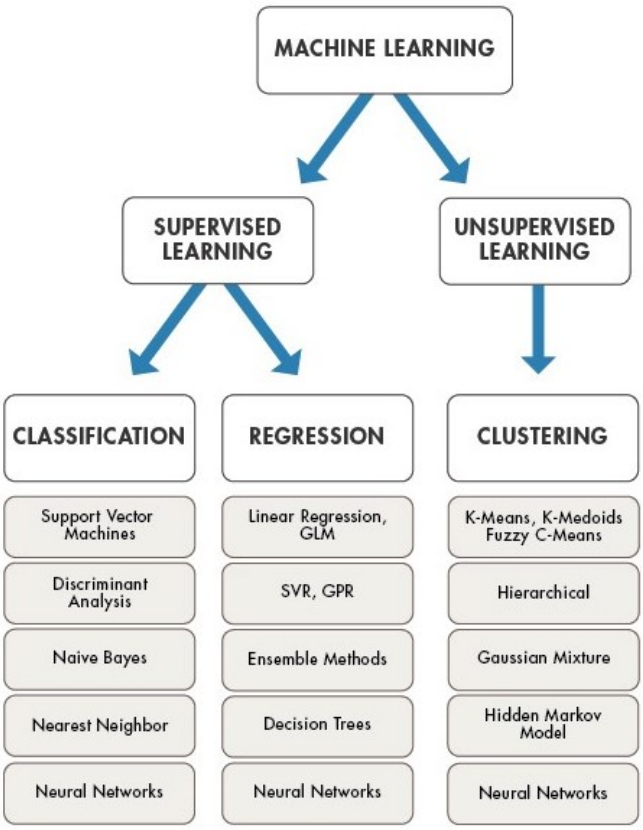

Abbildung 4.1: Some popular Machine Learning algorithms [11]

5 Analysis of Supervised Methods for Classification

This chapter aims to explain what is on the basis of some of the most famous supervised Modelling Techniques (or algorithms) in an easy way. As mentioned in the previous chapter 4 in section 4.1, the algorithms in question are: Linear Regression, Logistic Regression, k-Nearest Neighbours, Support Vector Machines (SVMs), Decision Trees and Random Forests, Naive Bayes and Neural Networks.

5.1 Regression

Since the list above mention two regression algorithms types, it is important to recall the definition of regression as a concept (see section 4.1 in chapter 4) before start explaining the different types of regression.

The difference between regression models is generally based on the type of relationships (between independent and dependent variables) and the number of independent variables. Concerning the regression models, this section summarily explains Linear Regression and Logistic Regression.

5.1.1 Linear Regression

Linear regression is a technique that attempts to model the relationship between two variables (x, y) applying the linear equation to the observed dataset. One of the variables is considered to be the explanatory or independent variable (x) and the other one is the dependent variable (y). The dependent variable (usually represented as y) is the one that tries to explain the forecast while the independent variable (usually represented as x) is the one that explains the other variables of the problem [3]. Linear Regression can essentially be used in two ways: to represent the general relationship between two variables or to represent the statistically significant relationship between two variables. In short, as the name suggests, the model is just a linear function, which means it simply computes a weighted sum of inputs features, plus a constant called the bias term or intercept term.

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

- θ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n.
- θ^T is the transpose of θ (a row vector instead of a column vector).
- $\mathbf{x}$ is the instance's *feature vector*, containing x_0 to x_n, with x_0 always equal to 1.
- $\theta^T \cdot \mathbf{x}$ is the dot product of θ^T and $\mathbf{x}$.
- h_θ is the hypothesis function, using the model parameters θ.

Abbildung 5.1: Linear Regression model prediction (vectorized form) [27]

Based on the brief description provided, it is intuitive to understand that Linear Regression is not applicable on classification problems, however, it is always good looking to mention it as it is a classic ML model. Due to its simplicity, it is most of the times, the first algorithm to be learned by a data scientist, it can be considered the *hello world* of ML[1].

[1] See [27] for more information about Linear Regression and other regression-based algorithms

5.1.2 Logistic Regression

Logistic Regression is a regression algorithm that can be used for classification problems due to its probabilistic nature. Fundamentally, it estimates the probability of an instance belonging to a specific class. Estimating the probability that an email is a spam or not is one of the most famous examples of logistic regression successful experiments (for example, if the probability is greater than 50 percent, the algorithm classifies the email as a spam email).

The logistic is a sigmoid function that outputs a number between 0 and 1. It only needs to estimate the probability of a certain event to be able to make a prediction. The main goal is to set the parameter vector θ that makes the model estimate high probabilities for positives instances (y = 1) and low probabilities for negative instances (y = 0). Therefore, for example, if the sigmoid tend to go to positive infinity, then Y(predicted) will be considered 1, and if it goes to negative, Y will become 0 [62].

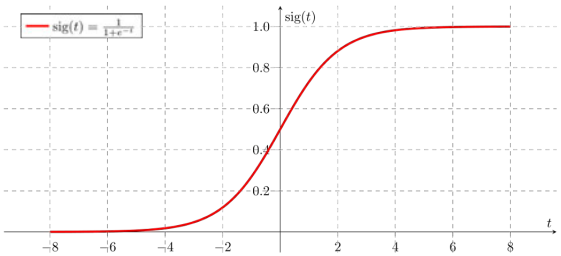

Abbildung 5.2: Logistic Regression Sigmoid function

In the case of Logistic regression, the cost function represents the average cost over all training instances. The cost function is convex which makes it possible for Gradient Descent to find the global minimum (it is guaranteed to be found at some point if the learning rate is not too large) [64].

A Decision Boundary plays an important role in Logistic Regression, it represents a threshold that is used to predict which class a data belongs. The final result of the prediction is based on the relation between the predicted value and the threshold. Decision Boundary can be linear or non-linear depending on the situation, in complex cases, it can even have a increased polynomial nature.[2]

[2]More information about the Logistic Regression can be found on the chapter 4 - Training Models, of the book [27]

5.2 k-Nearest Neighbours

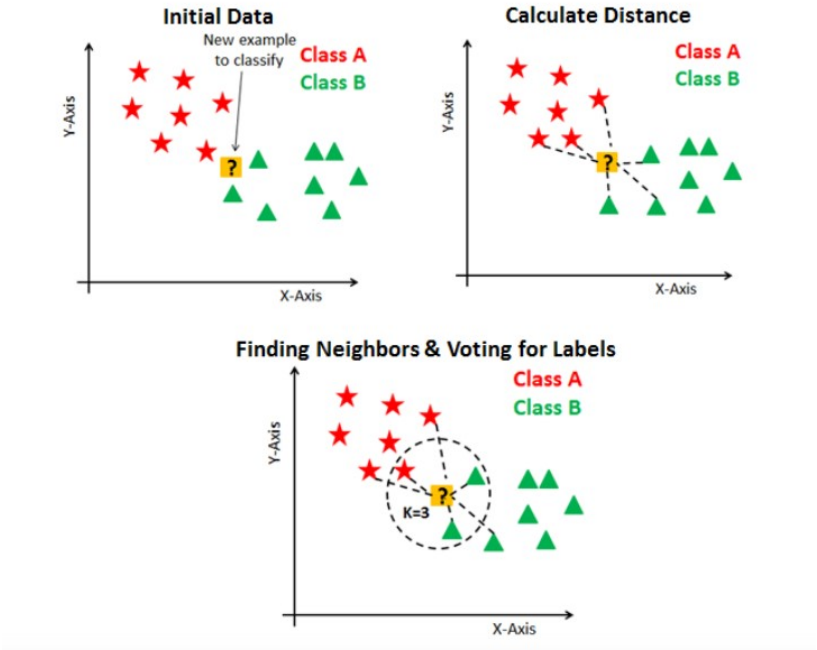

Abbildung 5.3: k-Nearest Neighbours [43]

K-Nearest Neighbours (KNN) is a versatile algorithm that can also be used in classification and regression problems. It assumes that similarity is directly related to proximity – identical data points are generally close to each other. Therefore, it is easy to comprehend that the calculation of the distance between data points on a graph is the fundamental approach of this type of algorithm. Among others, the Euclidean distance is one of the most popular and familiar ways to calculate the distance between two points [44].

$$D(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

Abbildung 5.4: Euclidean Distance formula

The K in k-nearest neighbours algorithm represents the number of the nearest neighbours chosen to take part in the training process. The initialization of the parameter K is pivotal for this algorithm since it determines the environment of the operation. If K is equal to one, the error is zero because the nearest point to any data point is itself (it is comparing with itself because there is only one data point on the training dataset), on the other hand, if K is equal to the total number of the data points, it covers the whole dataset,

that is the reason why it is important to initialize K with a good and strategic number [60].

In a nutshell, having the distance calculation method, KNN only need to know the K parameter to start the grouping and distance comparison process to be able to make a prediction. The algorithm is optimized by finding the optimal value of K which can be done manually or with help of an optimization algorithm such as k-fold cross validation [47] (in python, it can be implemented by importing *cross_val_score* from *sklearn.model_selection*) or the Elbow method [43] .

The beauty of k-Nearest Neighbours algorithm relies on its simplicity[3]. A model is not needed to be built and parameters to tuning is not needed. Despite the fact that it is a very straightforward algorithm, it gets considerably slower as the number of the data or independent variables increase which makes it an unfeasible option when the predictions need to be made rapidly and the only way to overcame that disadvantage is to provide sufficiently powerful computing resources to the system which can be costly.

[3]Further information about k-Nearest Neighbours algorithm available at datacamp page: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

5.3 Support Vector Machines (SVMs)

Support Vector Machines is one of the most popular and well-known models in the Machine Learning world. It is a multifaceted model as it is able to perform linear or nonlinear classification, regression and in some cases outlier detection. Support Vector Machines are especially appropriate for small- or medium-sized classification datasets.

In a nutshell, the main objective of the SVM algorithm is to identify a hyperplane in N-dimensional space (with N as the number of the features) that specifically classifies the data points. There are many possibilities to achieve that goal, however, the main goal is to find a plane that has the maximum margin (for example, the maximum distance between data points of two different classes) [49].

Maximizing the margin distance makes it possible to classify future data points with more certitude in order to avoid the chances of misclassification, this concept is also known as robustness. Therefore, the Support Vector Machine tries to solve a maximization problem (as it tries to maximize the margin the geometric margin between the datapoints and the hyperplane) and a minimization problem (as it tries to minimize the empirical classification error) [61] simultaneously.

Scaling is a crucial concept to take into account as it improves the quality of the decision boundary geometric margin. In figure 5.5, the margin in right side image is clearly better for a LinearSVC approach (in Sci-kit Learn, scaling can be achieved by using *StandardScaler*).

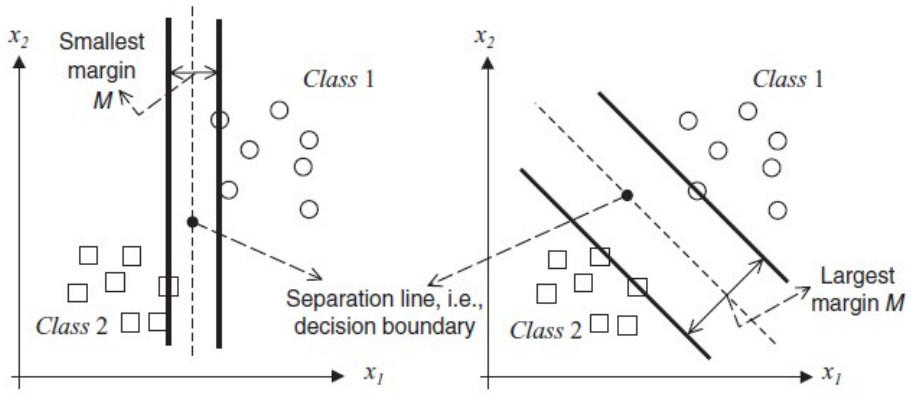

Abbildung 5.5: Scaling in SVM [61]

Hinge Loss is the name of the function that helps the maximization process. Even though maximizing the margin is very important, it is also relevant to note that accuracy has priority [49] over the margin maximization.

It is also important to use a Soft margin classification approach as it allows to tolerate/ignore some outliers (some data located in the wrong side of the boundary. In Sci-kit Learn's LinearSVC class [37], it can be balanced by controlling the values of the hyperparameter C) which makes it a more robust algorithm.

Since the hyperplane is considered a decision boundary that segregates two different classes, it helps to classify the data points. Its dimension depends on the number of the features (i.e. if it has only two features, then the hyperplane would be just a line, on the other hand, if the number of the input features is 3, then the hyperplane would become a two-dimensional plane).

Support vectors are essentially data points near the hyperplane and are capable of influencing the orientation and position of the hyperplane. Therefore, it can be concluded that these Support Vectors are responsible to maximize the margin of the classifier consequently, deleting them, means that the position of the hyperplane might change.

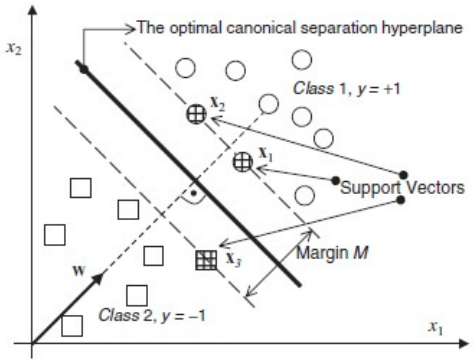

Abbildung 5.6: Support vectors in SVM [61]

In Support Vector algorithms, the LinearSVM is the most commonly used method for classification as it is fast and achieves satisfactory levels of accuracy, however, there is a non-linear solution version of the algorithm for more complex datasets - the Non-LinearSVM (that solution is more time-consuming and, associated with the *kernel trick*, is often used when the dataset is small and very complex in terms of features [27]).

5.4 Decision Trees and Random Forests

Decision Trees are very intuitive algorithms and as the name suggests, it is based on a tree-like model of decision. Learning decision trees from data (or simply Logical trees) are drawn upside down with the root at the top of the tree and are composed of nodes and the nodes can be distributed in three different types: the root nodes (depth 0, at the top of the tree, with children nodes only), child node (depth x, with a parent node and children nodes) and the leaf nodes (depth x, with one parent node and no children nodes). Each node represents a specific classification with its attributes and if it is not a leaf node, it is linked with its descending nodes through edges. The edges are in its essence, linkages/ways to responses of the questions/conditions represented on the top and the responses are known as features/attributes, in other words, the paths from root to leaf produce the classification rules [57].

Decision trees models can be used to solve classification and regression problems depending on the answer type of the main question, if the variables are numeric (continuous values) – it is considered a regression tree and if they are discrete – it is considered a classification tree.

Basically, the algorithm is based in features and conditions. They are used to show in which direction the tree needs to be splitted. In the splitting process, the algorithm uses a function to calculate the accuracy cost of the splitting and the split that costs least is chosen, therefore, lowering the cost is the strategy used to make the best prediction or classification. In Sci-kit Learn, the *CART (Classification and Regression Trees training algorithm)* basically searches for the optimum split in a greedy way based on features at each level. The essence is good but if applied, it would be time-consuming as the algorithm would never stop seeking the perfection and it would lead to another problem in terms of generalization [28].

Some trees can be very large which makes them more complex. To save time and resources, some conditions such as “the minimum number of training inputs to use on each node” or “maximum depth of the model” are established to stop the splitting process.

A subset is considered *pure* when it contains only samples of one class or category. The purity is an important concept in terms of accuracy and one of the most used algorithms is Entropy.

Entropy is used to measure information or disorder, it calculates the purity of a dataset.

$$Entropy\,(p) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Abbildung 5.7: Entropy - p is the whole dataset, N is the number of classes, nd pi is the frequency of class i in the same dataset

Essentially, if the class is uniform in terms of classes, the entropy is 0 (considered pure), on the other hand, if the formula gets different values, that means that there are different classes in the same dataset (higher entropy, confusion). When splitting, the objective is to achieve satisfactory lower levels of entropy than the original dataset.

However, on the other hand, decision trees are very sensitive to small variations in the training data (variations can lead to a completely different tree, it is known as variance), over-complex trees can lead to overfitting, its greedy characteristic do not always guarantee to return the globally optimal tree and if come classes dominate, biased trees might be created consequently.

To solve the overfiting problem, *Pruning* algorithm can be used to reduce the complexity of the tree and increase its performance removing nodes with low importance for the problem for generalization purposes - basically, a low populated branch of the tree is pruned and the tree is applied to the train again and if the the pruning of the branch does not decrease the accuracy on the training set, then the branch is pruned for good [57]. This method is also known as reduced error pruning. Another common method is the *Early Stopping* method - a minimum number of samples per node established, therefore, a branch stop growing when it achieves the minimum number of data samples.

Random Forest concept is very similar to decision trees. Instead of work with one tree, it works with a large number of decision trees that operate as an ensemble where each tree predicts one specific class and the class with more vote becomes the prediction of the model. This group work characteristic makes this type of algorithm, in some cases, more accurate than algorithms that use individual predictions as the different trees prevent each other from their individual errors. Random forest algorithms are useful for large datasets, can have a high predictive accuracy, have the ability to deal with multiple input features so does not need to delete features and work well with missing data. On the other hand, they are not easy to interpret and overfit with noisy classification or regression.

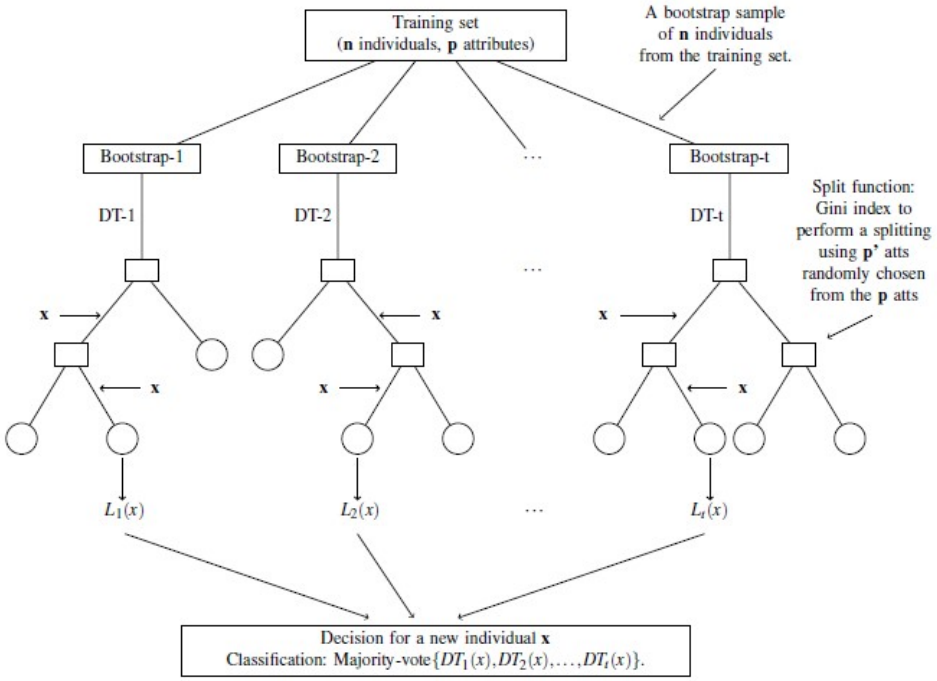

Abbildung 5.8: Random Forest [19]

5.5 Naive Bayes

Naive Bayes is a probabilistic classification algorithm as it work with probabilities and conditional probabilities to make predictions by nature. As the name suggests, it is based on the Bayes Theorem that aims to find the probability of A happening, given that B occured.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

A, B = events
$P(A|B)$ = probability of A given B is true
$P(B|A)$ = probability of B given A is true
$P(A), P(B)$ = the independent probabilities of A and B

Abbildung 5.9: Bayes Theorem formula

Assuming that all the predictors are independent - the presence of a specific feature in a class is not connected to the presence of any other feature [50] which means that each feature contributes independently and equally to the outcome. Each feature is given the same weight/importance.

Bayesian classification aims to predict the probability of occurrence of a label (L) (i.e. category of a text) given some features which can be represented as:

$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

It is possible to decide between two labels (i.e. L1 and L2) by simply computing the ratio of the conditional probabilities for each label, as follows:

$$\frac{P(L_1 \mid \text{features})}{P(L_2 \mid \text{features})} = \frac{P(\text{features} \mid L_1)}{P(\text{features} \mid L_2)} \frac{P(L_1)}{P(L_2)}$$

In order to be able to proceed with the prediction, it is also necessary to know which features compose a specific label, $P(features|Li)$ (having Li as the particular label). Such model is called *generative model* as it specifies the hypothetical random process that

generates the data [48]. Such model is specified for each label making simple assumptions about the configuration of the label.

The *naive* assumption comes with the independence among the features precisely [35]. *Gaussian Naive Bayes* and *Multinomial Naive Bayes* are the commonly used for classification purposes.

The Gaussian model assumes that *the data from each label is drawn from a simple Gaussian distribution* [48] in a naive point of view. On the other hand, the Multinomial model assumes that *the features are generated from a simple multinomial distribution*, essentially, it finds the probability by checking the number of occurrences/frequencies in several categories which makes it suitable for counts or count rate situations.

Despite being a very simple and naive algorithm, Naive Bayes has surprisingly proven to be very efficient achieving satisfactory levels of accuracy and performance in many text classification situations (sometimes outperforming some really sophisticated algorithms).

5.6 Neural Networks

Artificial Neural Networks, briefly mentioned in chapter 2 and 3 in section 3.4, are models used for linear and non-linear relationships between different datasets. To understand the artificial neural network processing method, it is important to highlight its basic concepts such as layers, weights and artificial neurons.

As the name suggests, artificial neural networks are composed by what it is called artificial neurons – nodes that try to act/represent biological neurons. All the artificial neurons from the collection are connected with each other by weights and are organized in layers. There are nodes responsible to receive input data, to process the data and to represent the output data. In its essence, there are two main characteristics to highlight: the learning rule and the transfer function as, in order to have a successful output, the processed data is transferred by the transfer function after had used the learning rule at the data processing stage – these type of connections have the same pattern of the synapses as they have the same signal-transmission as its central procedure.

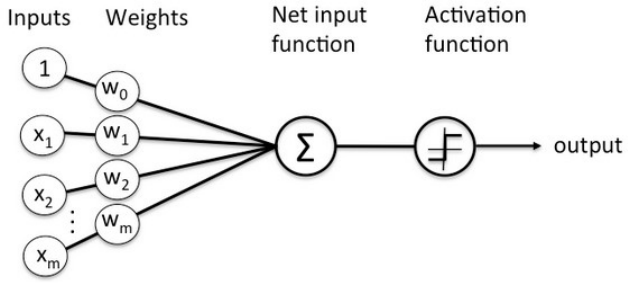

Abbildung 5.10: Neural Network node representation

There are 3 important steps [67] to highlight in the figure 5.12

1. Each input is multiplied by a weight:

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

2. All the weighted inputs are added together with a bias b:

$$(x_1 * w_1) + (x_2 * w_2) + b$$

3. The sum is passed through an activation function:

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

As the name suggests, a Neural Network is simply a network of neurons. Knowing how the neurons work it is quite easy to understand their networks:

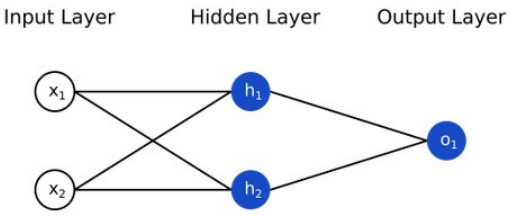

Abbildung 5.11: Neural Network with 1 hidden layer [67]

Basically, the networks are divided in layers (there is no restrictions regarding the number of hidden layers - see figure 3.3 from chapter 3). As illustrated on figure 5.11, the network has 2 inputs in the input layer, 2 neurons in the hidden layer (where the computation occurs) and 1 neuron output in the output layer.

In machine learning, among others types of neural networks, the two most popular ones are: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

Convolutional Neural Networks (CNN) are a Feedforward [67] artificial neural network type which means the data is sent in only one direction (all the way from the input

nodes to the output neurons passing through the hidden nodes if applicable, see image 3.3 in chapter 3), in other words, they do not have memory. Consequently, Convolutional Neural Networks are not able to use any type of cycles which makes them very simple to be interpreted.

In contrast to Convolutional Neural Networks, Recurrent Neural Networks (RNN) are artificial neural networks with memory and assume that the inputs have a dependence relationship between each other, therefore, the time element plays a very important role in this type of networks (since what was done before have an impact in what will be done next). That characteristic makes them very useful for real-life scenarios (such as predicting the next word in a sequence) and to perform cycles during the training process.

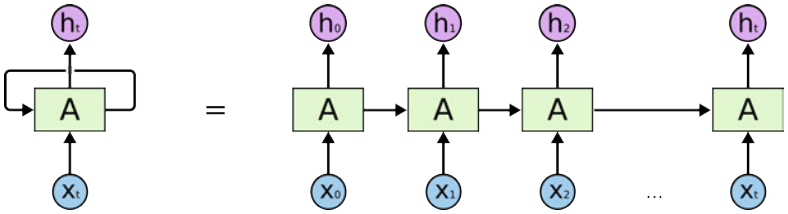

Abbildung 5.12: Recurrent Neural Network node representation [45]

In the figure 5.12, xt represents an input at a specific time t, A is the memory characteristic as it represents the current status base on previous ones, and ht represents the output at time t [45].

The training process is based on performing the same task for each member of a sequence originating different outputs based on the dependence relationship between the previous computations. Backpropagation (mentioned in chapter 2) is the most commonly used algorithm for training purposes as it takes the previous steps into account (i.e. in order to calculate the gradient at a specific time it would need to backpropagate 3 steps and sum up the gradients [12]).

Convolutional Neural Networks and Recurrent Neural Networks can be used for classification problems, however, in terms of accuracy, generally speaking, the RNN have a slight advantage because of their memory attribute in the training process.

6 Experiments and Evaluation

6.1 Supervised Text Classification with different algorithms using Scikit-Learn

The main objective of this chapter is to illustrate the simplicity and the power of Machine Learning tools performing and subsequently comparing different supervised text classification algorithms using the Scikit-Learn framework. Initially, the libraries will be succinctly presented and subsequently the concepts used on the experiments will be explained as the reader proceeds.

First and foremost, before proceeding with the experiments it is important to decide what software to use. On the following experiments, the *Anaconda*[1] was the software used. It is a free and open-source distribution of the Python and R programming languages for data science and machine learning purposes. It is commonly recommended because of its simple and straightforward nature (i.e. it comes with almost all the necessary tools already installed). Using anaconda, the experiments can be performed directly on its environment or on an IDE (i.e. the IDE used on the next experiments was *Spyder*[2] as it is more developer-friendly and comes with anaconda as well).

Scikit-learn (usually referred as sklearn in pyhton codes) is a free open source machine learning library that provides many supervised and unsupervised algorithms. According to its official page [3], it was created as a project that started in 2007 as a Google Summer of Code project by David Cournapeau. Later that year, Matthieu Brucher started work on this project as part of his thesis. Its first public release was in 2010 and since then it has been consistently managed as several other releases has been made cyclically.

[1] https://www.anaconda.com/
[2] https://www.spyder-ide.org/
[3] https://scikit-learn.org/stable/about.html

The first step to perform any machine learning algorithm is the collection of the dataset and in this work, the targeted dataset will be collected form the 20newsgroups collection [4]. 20newsgroups is a dataset collection (originally composed by Ken Lang) of approximately 18000 newsgroups documents/articles subdivided in nearly 20 different newsgroups each of them corresponding a specific topic/categories.

Over the time, the 20newsgroups dataset collection has become popular for experiments with text datasets in machine learning as well as the Scikit-Learn library. The following subsections will cover the experiments and the results taking a particular attention to the levels of accuracy and the time needed to perform the experiments. At the same time, important concepts and procedures will be explained using the performed experiments as the main source to provide not only theoretical but practical illustrations as well. One of the main goals of this chapter is to encourage to realise that nowadays, machine learning is not really difficult to understand and implement as the tools available make it a relatively straightforward science to deal with.

6.1.1 The Data

As mentioned before, gathering the data is an essential step to perform machine learning experiments because, as already known, the science itself is data based. Nowadays, the frameworks have some datasets already available to be used in machine learning experiments, these datasets are mainly used by data scientists or students to test algorithms or other aspects of the data science world performing different types of experiments. However, it is important to know that, if needed, one can gather the data from the web (i.e. from a target website) or upload a fresh dataset from a specific archive. As the main objective of the following experiments is to compare models/algorithms, there is no need to scrap the web and gather the data from a specific website, therefore, all the datasets used in this work will be taken directly from the framework. A really important step is to subdivide the dataset into *train dataset* and *test dataset*. The train dataset is used to train the models used, which means that it is actually used to perform the algorithms until it is decently suitable for use on real/important data. Therefore, the test dataset, as the name suggests, is used to test the model, in other words, it is used to perform the algorithms and subsequently check if the results are correct or not (accuracy) and how long it took to achieve such results (performance). Using the scikit-learn library, in the experiments, the 20newsgroups dataset can be promptly assembled for use by simply

[4] 20newsgroups dataset is available for download on the kaagle platform: https://www.kaggle.com/

importing the *fetch 20newsgroups* from *sklearn.datasets*. From there, other interesting actions can be taken to examine the dataset provided:

```
from sklearn.datasets import fetch_20newsgroups

news = fetch_20newsgroups(subset = 'all')
```

Abbildung 6.1: Importing the 20newsgroups dataset from sklearn datasets

```
print("Number of articles: ", len(news.data))
print('Number of different categories: ', len(news.target_names))
print('Show categories:\n', news.target_names)
```

Abbildung 6.2: Printing the number of articles,the number of categories and the names of all categories

The following image shows the output of the previous actions:

```
Number of articles:  18846
Number of different categories:  20
Show categories:
 ['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc',
'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x',
'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball',
'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space',
'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast',
'talk.politics.misc', 'talk.religion.misc']
```

Abbildung 6.3: The output of the previous lines of code from the image 7.2

The following line can be used to see a particular article, in this case, the article 3000 was used as an example:

```
print("\n".join(news.data[3000].split("\n")[:]))
```

Abbildung 6.4: Printing a specific article from the dataset

Subsequently, the respective output would be:

```
From: osinski@chtm.eece.unm.edu (Marek Osinski)
Subject: Re: Turkey-Cyprus-Bosnia-Serbia-Greece (Armenia-Azeris)
Organization: University of New Mexico, Albuquerque
Lines: 12
Distribution: world
NNTP-Posting-Host: chtm.eece.unm.edu

In article <1993Apr15.174657.6176@news.uiowa.edu> mau@herky.cs.uiowa.edu (Mau Napoleon) writes:

>Compromise on what, the invasion of Cyprus, the involment of Turkey in
>Greek politics, the refusal of Turkey to accept 12 miles of territorial
>waters as stated by international law, the properties of the Greeks of
>Konstantinople, the ownership of the islands in the Greek lake,sorry, Aegean.

Well, it did not take long to see how consequent some Greeks are in
requesting that Thessaloniki are not called Solun by Bulgarian netters.
So, Napoleon, why do you write about Konstantinople and not Istanbul?

Marek Osinski
```

Abbildung 6.5: The output of the article number 3000

As mentioned before, it is important to have train dataset and test dataset. The following extract of code shows how to actually divide the original dataset (into train and test) applying the commonly used *train-test-split* approach provided by the sklearn library:

```
from sklearn.model_selection import train_test_split
import time

def train(classifier, X, y):
    start = time.time()
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=11)

    classifier.fit(X_train, y_train)
    end = time.time()

    print("Accuracy: ", classifier.score(X_test, y_test), "Time duration: ", end - start)
    return classifier
```

Abbildung 6.6: Train function

In the train fuction (image 6.6), the function *fit()* is just responsible for training the classifier with the selected training data and the *time()* function is used to control the period of time of the training. It is also important to mention that the method *score()* can also be implemented on our train function to measure the accuracy of the classifier (note that the accuracy result is usually represented in a form of percentage).

For learning purposes, one can extract some more informations about the dataset from the same function train (such as the length of the dataset used for test and prediction, the comparison between the original data and the predicted data and the original data in question) by adding the following portions of code:

```
for i in y_pred:
    #time.sleep(.1)
    print(news.target_names[y_pred[i]], ' - ', news.target_names[y_test[i]])
```

Abbildung 6.7: Train function

The cycle in the figure 6.7 basically compares the predicted result/category with the original one for all the selected articles. Therefore, using an appropriate model for classification, the output would be as the follows:

```
comp.windows.x  -  comp.windows.x
misc.forsale  -  misc.forsale
alt.atheism  -  alt.atheism
sci.med  -  sci.med
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
misc.forsale  -  misc.forsale
sci.med  -  sci.med
comp.graphics  -  comp.windows.x
misc.forsale  -  misc.forsale
sci.space  -  sci.space
misc.forsale  -  misc.forsale
sci.space  -  sci.space
alt.atheism  -  alt.atheism
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
sci.med  -  sci.med
misc.forsale  -  misc.forsale
misc.forsale  -  misc.forsale
rec.motorcycles  -  misc.forsale
sci.med  -  sci.med
rec.sport.baseball  -  rec.sport.baseball
rec.sport.baseball  -  rec.sport.baseball
comp.windows.x  -  comp.windows.x
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
rec.sport.baseball  -  rec.sport.baseball
alt.atheism  -  alt.atheism
sci.med  -  sci.med
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
sci.crypt  -  sci.crypt
sci.med  -  sci.med
rec.motorcycles  -  misc.forsale
sci.space  -  sci.space
sci.med  -  sci.med
sci.space  -  sci.space
comp.os.ms-windows.misc  -  comp.os.ms-windows.misc
sci.space  -  sci.space
alt.atheism  -  alt.atheism
```

Abbildung 6.8: A section of the output for the comparison between the predicted values (left side) and the original values (right side)

The model used as an example in the figure 6.8 is the Multinomial Naive Bayes. As it can be noticed, the majority of the predictions is correct. However, as an example, in order to compare the results, there are two different situations highlighted in the image. The first scenario represents a correct prediction as the predicted value is *sci.space* and it can be verified that the original category is *sci.space* as well. In contrast, in the second scenario, the prediction is incorrect as the predicted classification is *rec.motorcycles* and the original category is *misc.forsale*. From this example, one can notice that the model is not a hundred percent accurate even though it looks very precise in most of the cases.

```
print('\n\n')
print("y_pred length:",len(y_pred))
print("y_test length:",len(y_test))
print("X_test length:",len(X_test))

print(news.target_names[y_pred[1135]])
print(news.target_names[y_pred[1135]], ' - ', news.target_names[y_test[1135]])
print('y_pred result:',y_pred[1135])
print('y_test result:',y_test[1135])

print('\n\n')
print('Show article:\n', X_test[1135])
```

Abbildung 6.9: Train function

The image 6.9 demonstrates how to double check the length of the dataset certifying that the prediction and test variables are using the same dataset. It also shows the comparison of one randomly chosen article as demonstrated in 6.8 and subsequently certifies the predicted and original results in terms of index in the category list. Finally, it prints the output of the original selected article (number 1135 in this case) to confirm and analyse the decision provided by the predictive model used.

The following image displays the respective output:

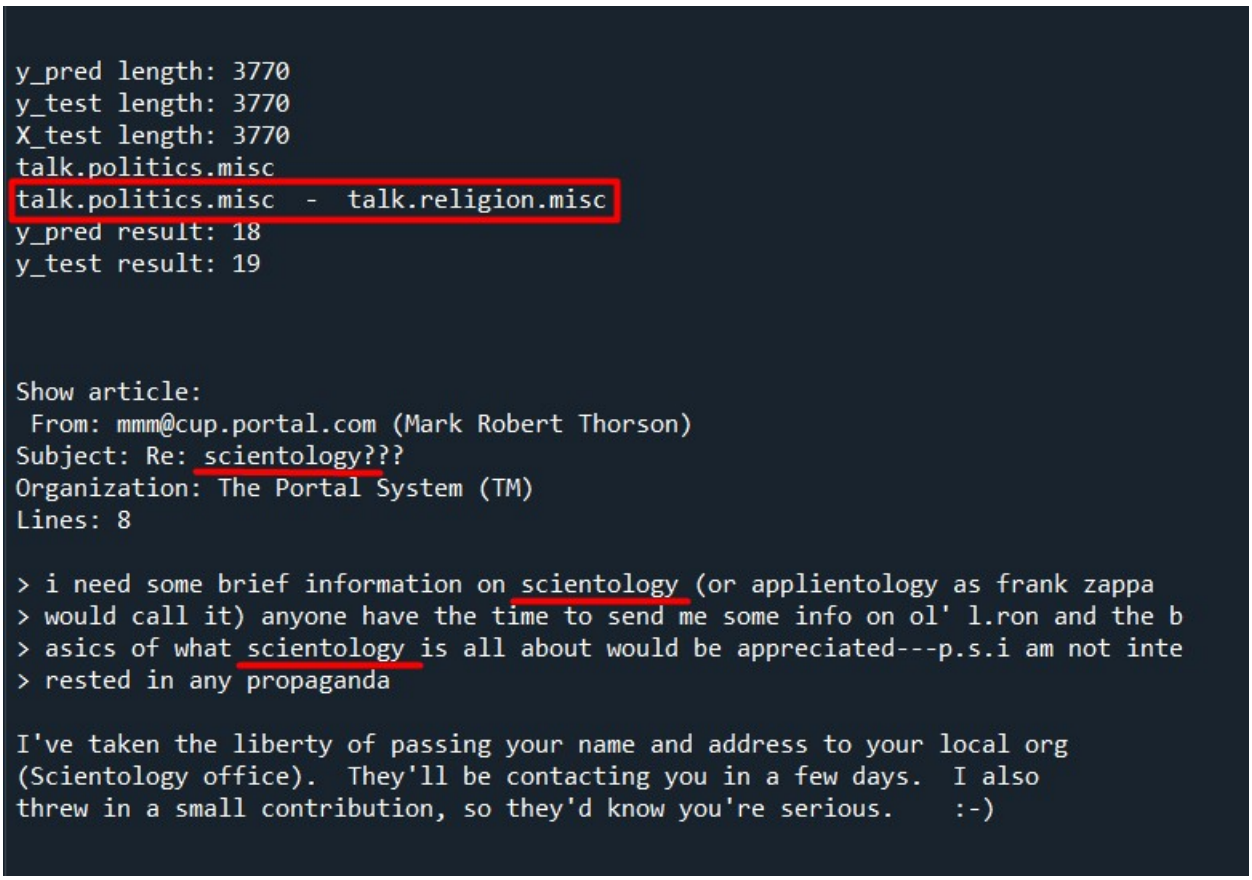

Abbildung 6.10: Train function

The image 6.10 displays the output of the piece of code implemented in the image 6.9. As it can be seen, everything is right in terms of the used data as it can be confirmed that the prediction and test vaiables are using the same segment of dataset previously selected. In terms of prediction, it is interesting to notice that the category was incorrectly predicted as it is highlighted in the image (note that the article 1135 was intentionally chosen to highlight a situation of imprecision. The model predicted *talk.politics.misc* but in fact, the correct category was *talk.religion.misc*). In terms of index, it is also confirmed that the *y-pred result* (18) corresponds to the correct category as well as the *y-test result* (19) and that can be validated by reviewing the image 6.3.

One can certify that situation by displaying the original article. As it is shown in the image 6.10, there are always some words in the original articles that can confirm the obtained results, in this case, the highlighted word *scientology* validates the fact that the category of the text should be *talk.religion.misc* and not *talk.politics.misc* as scientology is a word related to religion beliefs and not with politics.

It is important to note that after being trained, the selected model can also be tested using a personal/private text. This method can be used from individuals or institutions when there is a particular private text or dataset. To illustrate, the same model predicts the category of a randomly written phrase:

```
phrase = ["The term motorcycle (or motorrad in German) was first used in 1894 by Hildebrand & Wolfmüller."]
phrase_pred = classifier.predict(phrase)
print("Phrase:\n", phrase)
print("\nPredicted category:\n", news.target_names[phrase_pred[0]])
```

Abbildung 6.11: Train function

```
Phrase:
 ['The term motorcycle (or motorrad in German) was first used in 1894 by Hildebrand & Wolfmüller.']

Predicted category:
 rec.motorcycles
```

Abbildung 6.12: Train function

In the image 6.11 a random phrase (*The term motorcycle (or motorrad in German) was first used in 1894 by Hildebrand and Wolfmüller.*) was chosen and subsequently, the classification algorithm was applied to check if the model is capable of predicting its category correctly.

In the image 6.12 it can be seen that the model precisely predicted that the category is *rec.motorcycle*. Since original phrase contains words such as *motorcycle* and *motorrad* that emphasize its category, the prediction can be assumed as correct and accurate.

It can be concluded that, when defined, a train function can be characterized by a function that gives some information about the period of time of the training, a function to split the dataset into train dataset and test dataset, function to actually perform the training process and a function to measure the accuracy. Simultaneously, it can be used to perform experiments and extract knowledge about the dataset.

6.1.2 Managing the Dataset

In any text classification context, as the name suggests, the original data is in text format. However, the computer and the algorithms language is always based on numbers, therefore, there is a need to convert or associate the text format to an integer/numbers based format.

A text is a series of ordered letters that make words. A series of ordered words with right punctuations make phrases and a series of ordered phrases can make a text. Based on this premise, it can be stated that a series of words can be seen as an array of integers in the programming environment. Thus, each word of the text can be associated with a specific number and from that point of view it is possible to finally start managing the data and proceed with machine learning activities.

It is relevant to mention that since the texts on the dataset have different length sometimes it is also important to give them the same length in order to move forward on the data analysis process. This can be done by finding a specific length limit and padding smaller data with a specific value (i.e. 0).

The *TfidfVectorizer()* is a really helpful tool from sklearn that can be imported and used to check the frequency of words in a text. In classification circumstances it is commonly used to measure the importance of a particular word on a document based on the number of times that it occurs. The *stopwords* is used to filter words that does not have a significant impact in the classification problems such as 'the', 'is', 'are'. As reported by *pythonspot*, the *nltk* module contains a list of stop words that can be used for classification problems. Similarly, there is also a list of punctuations that can be used and included on the TfidfVectorizer() for classification purposes.

The *Pipeline()* is another important approach to take into consideration. In essence, it makes it possible to implement different operations at once by simply specifying the required action in a tuple (usually, the first initial steps are related to data preprocessing and the last step (last tuple) is the classier - in this case, the classification algorithm). According to Dr. *Saptashwa Bhattacharyya's* article *A Simple Example of Pipeline in Machine Learning with Scikit-learn*[5] posted on *Towards Data Science* on November 12th 2018, the pipeline class allows sticking multiple processes into a single scikit-learn estimator.

[5] https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

Finally, as mentioned before, the last tuple of the *Pipeline* is reserved for the classifier, in other words, the selected classification algorithm.

6.1.3 The Classifiers

According to *Sidath Asiri*, on the article *Machine Learning Classifiers*, published on the 11th of June 2018 for the *Towards Data Science* journal, Classification can be defined as the process of predicting the class of given data points and Classification predictive modelling is the task of approximating a mapping function *f()* from input variables *(x)* to discrete output variables *(y)*.[6]

There are many classifiers available apart from the ones already mentioned on this work. It is a really vast field that is intended to be explored by those who are passionate about the data science area (as an example, most of the classifiers used in scikit-learn can be found on at the scikit-learn official page [7]). Despite the extensive nature of the field, most of the times, it is not fair to say that one algorithm is definitely better than the other one because all of them have its qualities and drawbacks.

Therefore, it can be assumed that some algorithms are better for some specific situations, in other words, it is more assertive to compare the algorithms based on the specific situation or dataset provided than to compare their nature itself.

The classifiers used in the next chapter of this work are: Multinomial Naive Bayes, Logistic Regression, Support Vector Machine and k-Nearest Neighbours.

Further information about the 20newsgroups dataset and its usage with other interesting classifiers can be found on the official scikit-learn page dedicated to the usage of the 20newsgroups dataset.[8]

[6] Available at: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
[7] https://scikit-learn.org
[8] Availabble at: https://$scikit-learn.org/0.19/datasets/twenty_newsgroups.html$

6.2 Results and Analysis

The main goal of this section is to show the results of the selected algorithms and subsequently analyse them comparing the outputs of the models.

The results will essentially consist on the *accuracy* and *performance* of the models which are the most important aspects in any machine learning based experiment. Therefore, for each model, the *predicted* values will be compared to the *actual* values for each variable.

On data analysis situations, graphics are commonly used for analytical purposes and on these experiments, the main graphical representation of the results will be displayed in form of confusion matrix and heatmaps.

A *Confusion Matrix* is a proceedure used for performance measurement in machine learning classification problems. It is a straightforward matrix that uses two or more classes and compares the relationship between them. A *Heatmap* is a graphical representation that uses a color-coding based system to represent the relationship between different variables. It is commonly used to analyse large amounts of data as it can give a more comprehensive overview of the results because it is a more visual representation by nature. The *Confusion Matrix* and the *Heatmp* will be combined in this work in order to give a better understanding of the achieved results.

6.2.1 Results

Multinomial Naive Bayes

To perform the *Multinomial Naive Bayes* algorithm, one can simply import the *MultinomialNB* from *sklearn.naive bayes* and proceed with the experiment as it shows on the following portion of code:

```
news = fetch_20newsgroups(subset = 'all')

start = time.time()

classifier = Pipeline([ ('vectorizer', TfidfVectorizer(stop_words=stopwords.words('english') + list(string.punctuation))),
                        ('classifier', MultinomialNB(alpha=0.005))])

X_train, X_test, y_train, y_test = train_test_split(news.data, news.target, test_size=0.2, random_state=11)
classifier.fit(X_train, y_train)
end = time.time()

print("Accuracy: " + str(classifier.score(X_test, y_test)) + ", Time duration: " + str(end - start))
```

Abbildung 6.13: Using Pipeline, TfidfVectorizer for MultinomialNB training process

For the Multinomial Naive Bayes algorithm the results are as follows:

1. The accuracy and the duration:

```
Accuracy: 0.9175066312997348, Time duration: 8.620946884155273
```

Abbildung 6.14: MultinomialNB - Accuracy and Time

2. The metrics - precision, recall, f1-score and support per variable:

	precision	recall	f1-score	support
rec.sport.baseball	0.93	0.95	0.94	172
rec.sport.hockey	0.84	0.85	0.85	184
sci.space	0.84	0.82	0.83	204
comp.sys.ibm.pc.hardware	0.82	0.82	0.82	195
comp.graphics	0.91	0.90	0.90	195
sci.med	0.94	0.89	0.91	204
talk.religion.misc	0.83	0.80	0.82	164
alt.atheism	0.91	0.92	0.92	180
sci.electronics	0.95	0.99	0.97	173
talk.politics.mideast	0.95	0.96	0.95	217
sci.crypt	0.93	0.97	0.95	178
rec.motorcycles	0.95	0.99	0.97	197
soc.religion.christian	0.92	0.92	0.92	199
comp.windows.x	0.96	0.98	0.97	183
rec.autos	0.95	0.97	0.96	207
talk.politics.guns	0.95	0.94	0.94	211
comp.os.ms-windows.misc	0.94	0.96	0.95	208
misc.forsale	0.97	0.97	0.97	200
comp.sys.mac.hardware	0.91	0.90	0.91	175
talk.politics.misc	0.93	0.77	0.85	124
accuracy			0.92	3770
macro avg	0.92	0.91	0.91	3770
weighted avg	0.92	0.92	0.92	3770

Abbildung 6.15: MultinomialNB - Metrics

3. The graphical representation (*Confusion matrix and Heatmap*):

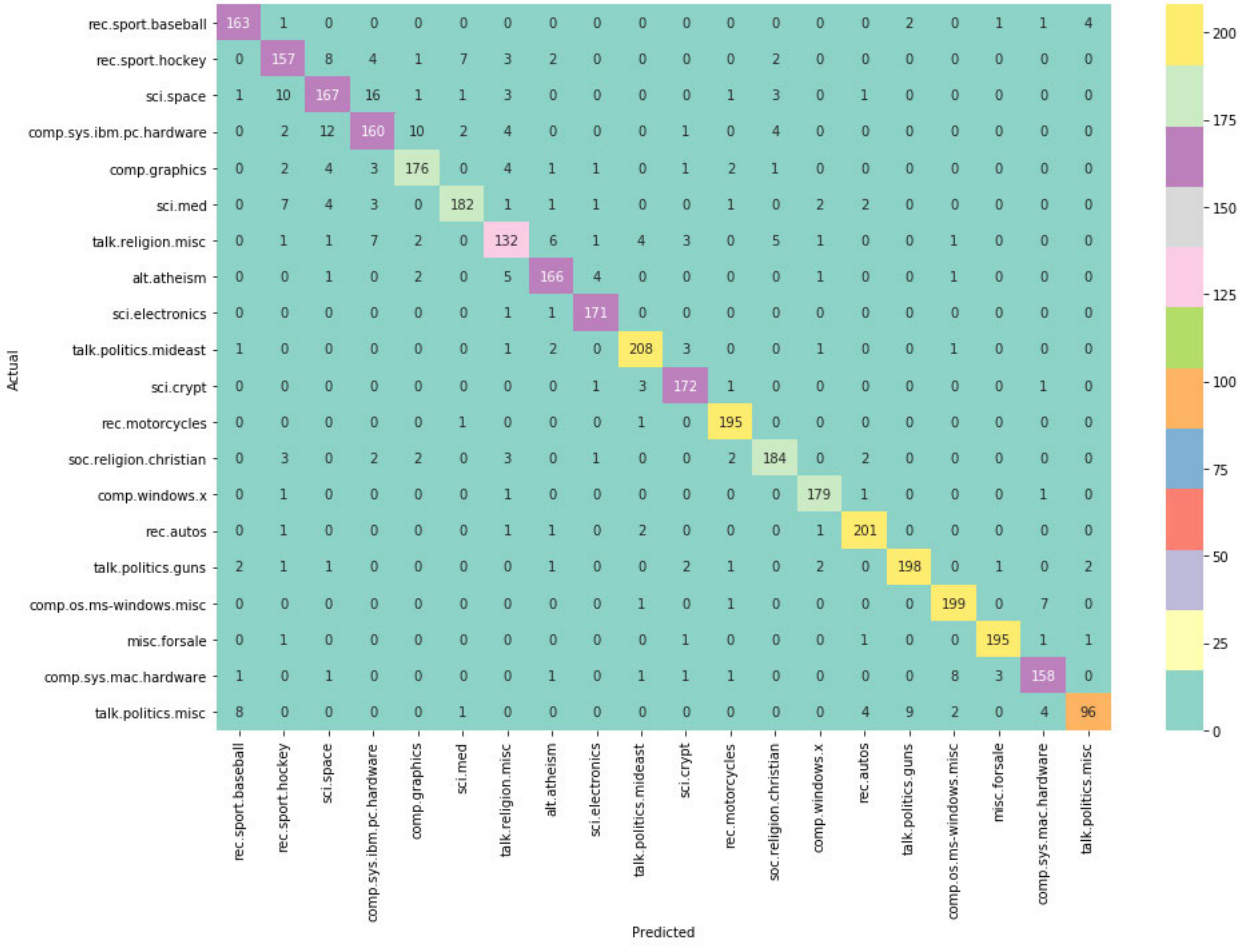

Abbildung 6.16: MultinomialNB - Confusion Matrix and Heatmap

Support Vector Machine

Secondly, to perform the Support Vector Machine for classification purposes, it is needed to import the *LinearSVC* from *sklearn.svm* and proceed as follows:

```
news = fetch_20newsgroups(subset = 'all')

start = time.time()

classifier = Pipeline([('vectorizer', TfidfVectorizer(stop_words=stopwords.words('english') + list(string.punctuation))),
                       ('classifier', LinearSVC(C=10))])

X_train, X_test, y_train, y_test = train_test_split(news.data, news.target, test_size=0.2, random_state=11)
classifier.fit(X_train, y_train)
end = time.time()

print("Accuracy: " + str(classifier.score(X_test, y_test)) + ", Time duration: " + str(end - start))
```

Abbildung 6.17: Using Pipeline, TfidfVectorizer for LinearSVC training process

The results for the Linear Support Vector Classifier in this case are:

1. The accuracy and the duration:

```
Accuracy: 0.9352785145888595, Time duration: 25.838910818099976
```

Abbildung 6.18: LinearSVC - Accuracy and Time

2. The metrics - precision, recall, f1-score and support per variable:

	precision	recall	f1-score	support
rec.sport.baseball	0.96	0.95	0.96	172
rec.sport.hockey	0.86	0.91	0.88	184
sci.space	0.92	0.85	0.89	204
comp.sys.ibm.pc.hardware	0.83	0.87	0.85	195
comp.graphics	0.93	0.92	0.93	195
sci.med	0.93	0.87	0.90	204
talk.religion.misc	0.85	0.88	0.86	164
alt.atheism	0.93	0.94	0.93	180
sci.electronics	0.97	0.97	0.97	173
talk.politics.mideast	0.97	0.97	0.97	217
sci.crypt	0.97	0.99	0.98	178
rec.motorcycles	0.95	0.98	0.96	197
soc.religion.christian	0.93	0.93	0.93	199
comp.windows.x	0.94	0.99	0.97	183
rec.autos	0.96	0.99	0.97	207
talk.politics.guns	0.94	0.96	0.95	211
comp.os.ms-windows.misc	0.97	0.96	0.96	208
misc.forsale	0.99	0.99	0.99	200
comp.sys.mac.hardware	0.96	0.93	0.94	175
talk.politics.misc	0.94	0.83	0.88	124
accuracy			0.94	3770
macro avg	0.94	0.93	0.93	3770
weighted avg	0.94	0.94	0.94	3770

Abbildung 6.19: LinearSVC - Metrics

3. The graphical representation (*Confusion matrix and Heatmap*):

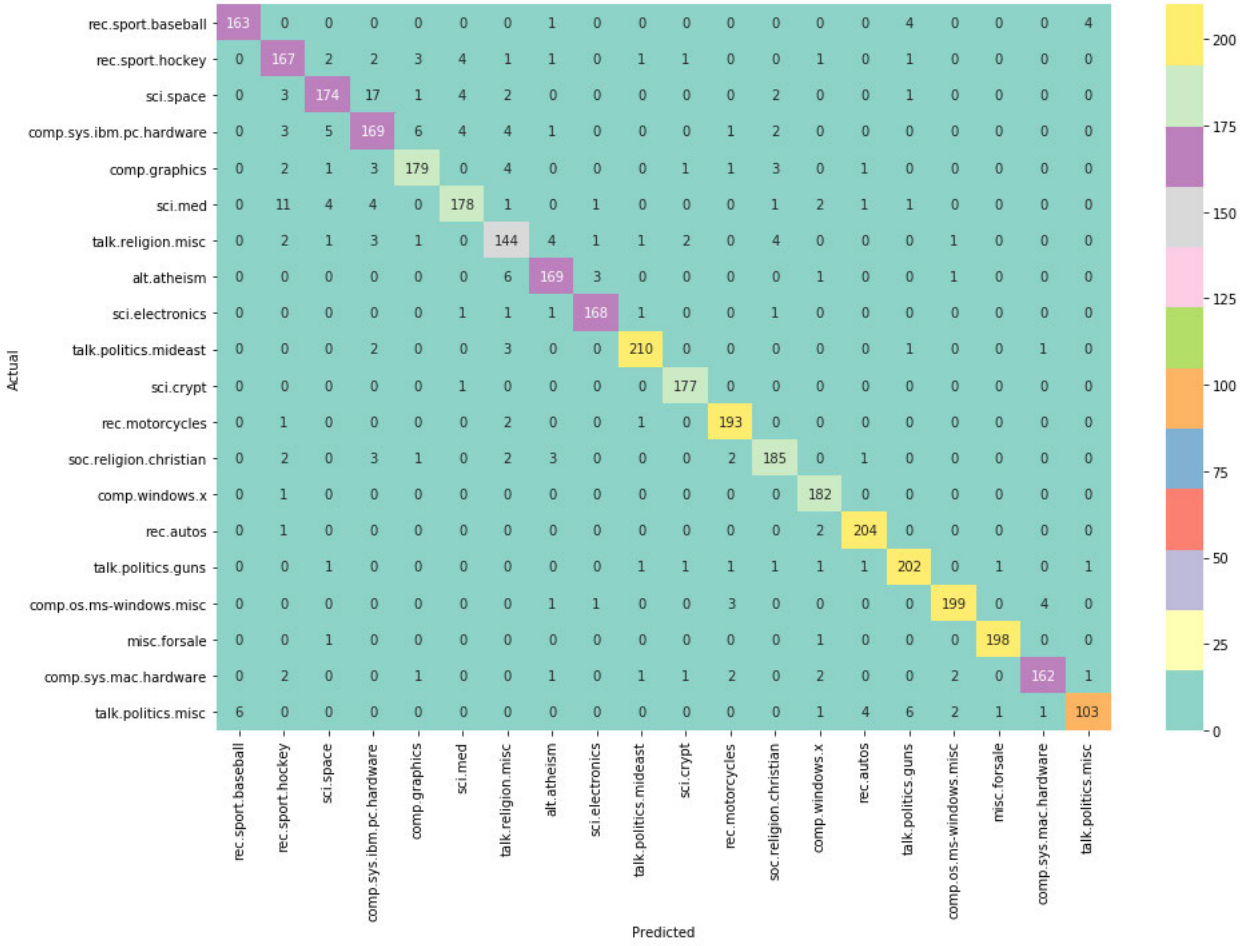

Abbildung 6.20: LinearSVC - Confusion Matrix and Heatmap

Logistic Regression

Finally, in order to perform the Logistic Regression, it is important to import the *LogisticRegression* from *sklearn.linear model* and similarly, implement the model as follows:

```
news = fetch_20newsgroups(subset = 'all')

start = time.time()

classifier = Pipeline([('vectorizer', TfidfVectorizer(stop_words=stopwords.words('english') + list(string.punctuation))),
                       ('classifier', LogisticRegression(max_iter = 1000))])

X_train, X_test, y_train, y_test = train_test_split(news.data, news.target, test_size=0.2, random_state=11)
classifier.fit(X_train, y_train)
end = time.time()

print("Accuracy: " + str(classifier.score(X_test, y_test)) + ", Time duration: " + str(end - start))
```

Abbildung 6.21: Using Pipeline, TfidfVectorizer for LogisticRegression training process

1. The accuracy and the duration:

```
Accuracy: 0.9079575596816977, Time duration: 95.59745359420776
```

Abbildung 6.22: LogisticRegression - Accuracy and Time

2. The metrics - precision, recall, f1-score and support per variable:

	precision	recall	f1-score	support
rec.sport.baseball	0.93	0.92	0.92	172
rec.sport.hockey	0.83	0.90	0.87	184
sci.space	0.87	0.82	0.85	204
comp.sys.ibm.pc.hardware	0.80	0.83	0.81	195
comp.graphics	0.91	0.89	0.90	195
sci.med	0.91	0.90	0.90	204
talk.religion.misc	0.81	0.87	0.84	164
alt.atheism	0.90	0.95	0.92	180
sci.electronics	0.97	0.97	0.97	173
talk.politics.mideast	0.96	0.95	0.96	217
sci.crypt	0.96	0.97	0.97	178
rec.motorcycles	0.97	0.97	0.97	197
soc.religion.christian	0.86	0.90	0.88	199
comp.windows.x	0.92	0.98	0.95	183
rec.autos	0.95	0.98	0.96	207
talk.politics.guns	0.90	0.94	0.92	211
comp.os.ms-windows.misc	0.92	0.92	0.92	208
misc.forsale	0.99	0.95	0.97	200
comp.sys.mac.hardware	0.92	0.85	0.88	175
talk.politics.misc	0.89	0.59	0.71	124
accuracy			0.91	3770
macro avg	0.91	0.90	0.90	3770
weighted avg	0.91	0.91	0.91	3770

Abbildung 6.23: LogisticRegression - Metrics

3. The graphical representation (*Confusion matrix and Heatmap*):

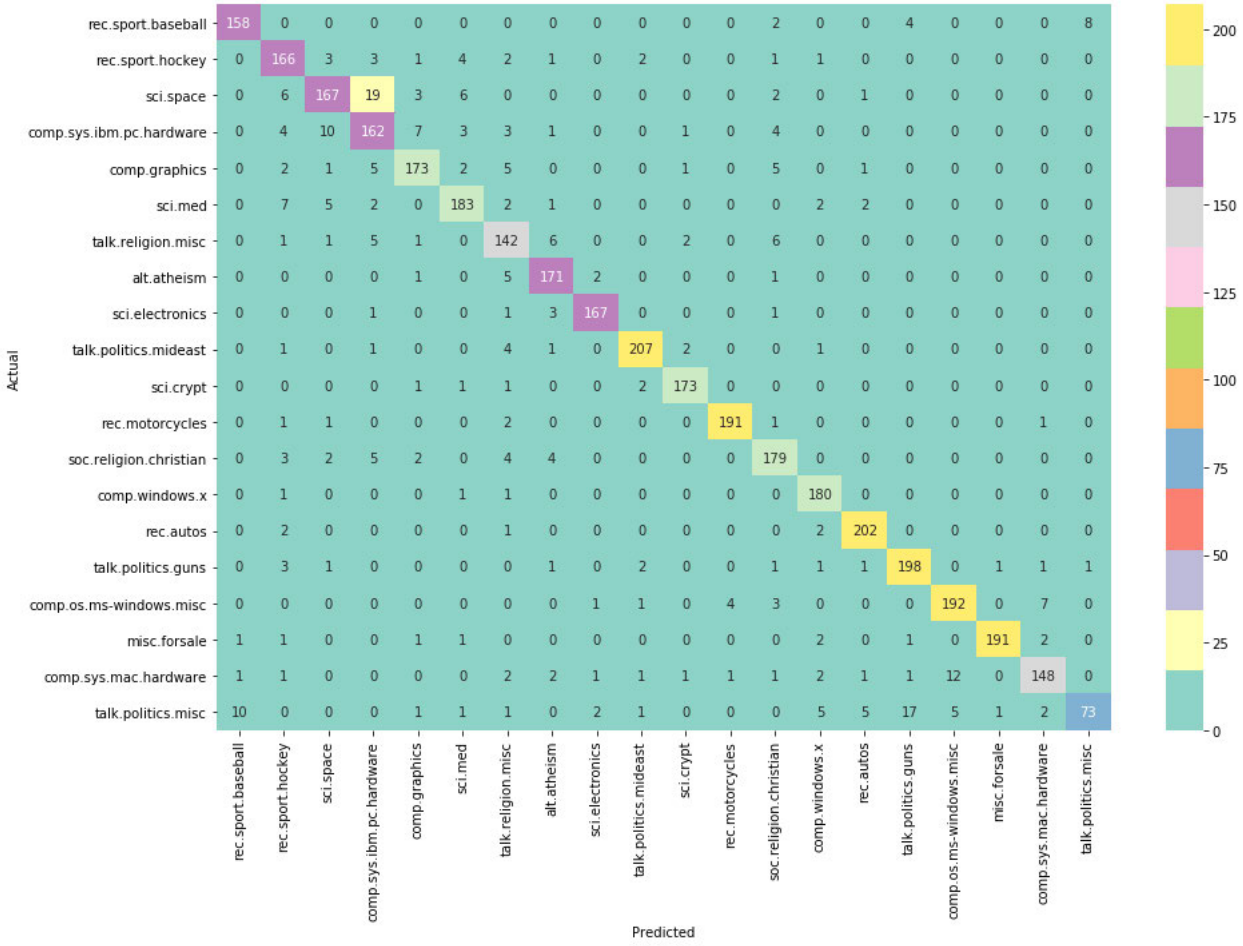

Abbildung 6.24: LogisticRegression - Confusion Matrix and Heatmap

k-Nearest Neighbours

Similarly, to perform k-Nearest Neighbour algorithm, it is needed to import the *KNeighborsClassifier* from *sklearn.neighbors* and proceed as it is shown on the following piece of code:

```
news = fetch_20newsgroups(subset = 'all')

start = time.time()

classifier = Pipeline([('vectorizer', TfidfVectorizer(stop_words=stopwords.words('english') + list(string.punctuation))),
                       ('classifier', KNeighborsClassifier(n_neighbors=5))])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=11)
classifier.fit(X_train, y_train)
end = time.time()

print("Accuracy: " + str(classifier.score(X_test, y_test)) + ", Time duration: " + str(end - start))
```

Abbildung 6.25: Using Pipeline, TfidfVectorizer for k-Nearest Neighbours training process

1. The accuracy and the duration:

```
Accuracy: 0.8013262599469496, Time duration: 6.026879072189331
```

Abbildung 6.26: k-Nearest Neighbours - Accuracy and Time

2. The metrics - precision, recall, f1-score and support per variable:

	precision	recall	f1-score	support
rec.sport.baseball	0.78	0.90	0.83	172
rec.sport.hockey	0.60	0.76	0.67	184
sci.space	0.68	0.71	0.70	204
comp.sys.ibm.pc.hardware	0.65	0.68	0.66	195
comp.graphics	0.74	0.71	0.73	195
sci.med	0.82	0.68	0.74	204
talk.religion.misc	0.62	0.57	0.59	164
alt.atheism	0.79	0.84	0.81	180
sci.electronics	0.86	0.93	0.89	173
talk.politics.mideast	0.88	0.83	0.86	217
sci.crypt	0.85	0.92	0.88	178
rec.motorcycles	0.86	0.94	0.90	197
soc.religion.christian	0.80	0.72	0.76	199
comp.windows.x	0.87	0.82	0.84	183
rec.autos	0.89	0.90	0.89	207
talk.politics.guns	0.90	0.82	0.85	211
comp.os.ms-windows.misc	0.89	0.88	0.89	208
misc.forsale	0.93	0.94	0.94	200
comp.sys.mac.hardware	0.85	0.80	0.83	175
talk.politics.misc	0.78	0.60	0.68	124
accuracy			0.80	3770
macro avg	0.80	0.80	0.80	3770
weighted avg	0.80	0.80	0.80	3770

Abbildung 6.27: k-Nearest Neighbours - Metrics

3. The graphical representation *(Confusion Matrix and Heatmap)*:

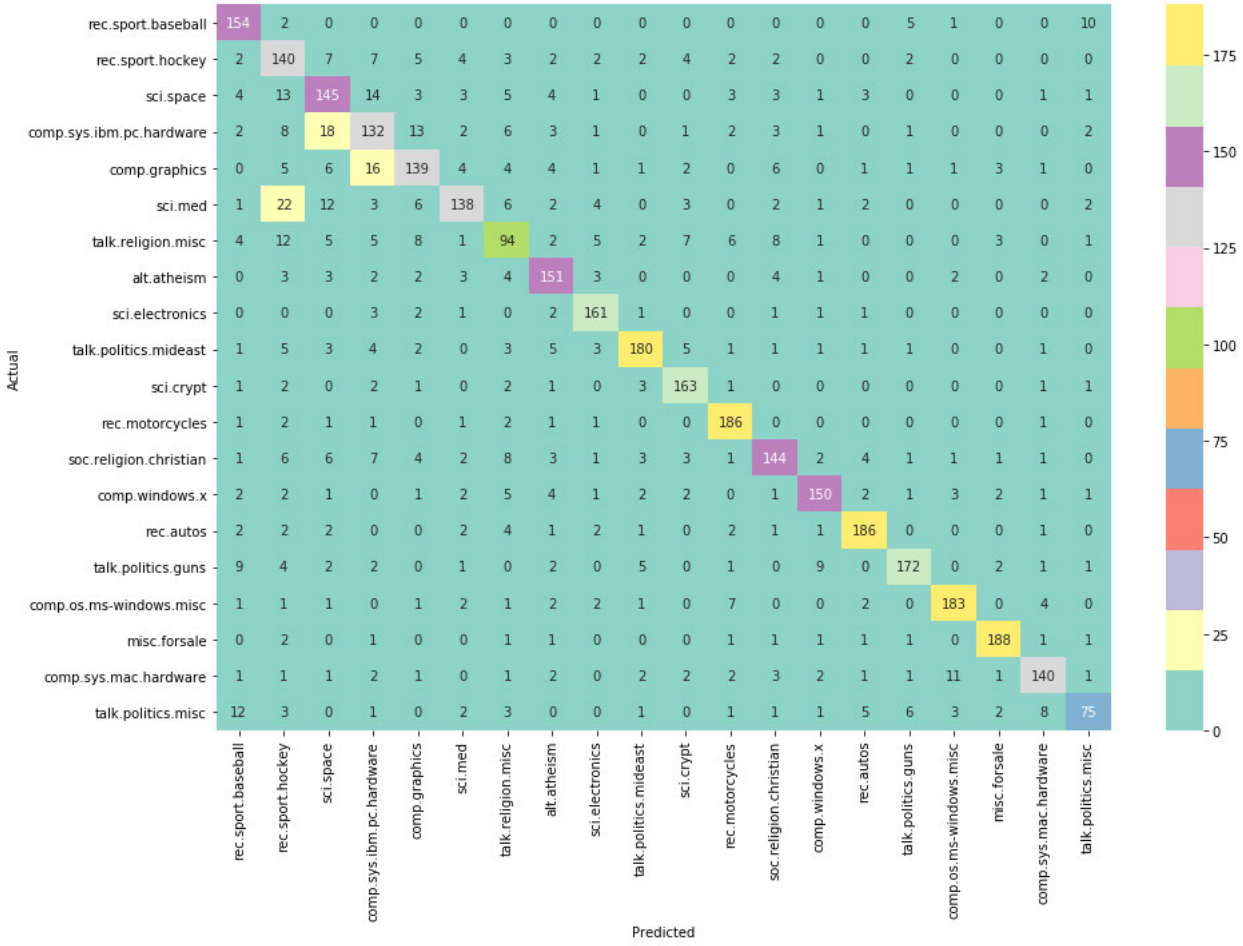

Abbildung 6.28: k-Nearest Neighbours - Confusion Matrix and Heatmap

6.2.2 Analysis

As demonstrated, the implementation of the models in *Python* using *Scikit-Learn* is fairly simple to understand. However, one of the most important (if not the most important) skill of a data scientist is to analyse the achieved results. This step can be a crucial factor for the success of the of the main concern (i.e. to make decisions on business, healthcare or even political environments).

Accuracy

In terms of accuracy, according to the results displayed in the image 6.29, it is important to note that for the *Multinomial Naive Bayes* model (see section 5.5), the precision was not the best compared to the others algorithms but it was not the worst either, in fact, it was the second best result (loosing only for the Linear Support Vector Machine model) which makes it a very acceptable output (see image 6.14). At a more specific level, it can be highlighted that the category *comp.sys.ibm.pc.hardware* had the lowest accuracy and, on the other hand, the category $misc.forsale$ (see image 6.15) had the highest accuracy level. Finally, reading the confusion matrix and heatmap graphical representation (see image 6.16), one can extract some valuable information about each category concretely (i.e. the category *sci.space* was 16 times wrongly predicted/classified as *comp.sys.ibm.pc.hardware* and 167 times correctly predicted).

The second algorithm, *Support Vector Machine* (see section 5.3), have managed to achieve the best accuracy level compared to the other three algorithms which was clearly an outstanding outcome in this classification problem (see image 6.18). It can also be observed that the category *comp.sys.ibm.pc.hardware* had the lowest level of accuracy but the generally speaking the accuracy was good with the category *misc.forsale* having the remarkable level of accuracy of 0.99 (see image 6.19). As an example, analysing the confusion matrix and heatmap, it can be seen that the *sci.med* was 11 times wrongly classified as *rec.sport.hockey* and 178 times successfully predicted (see image 6.20).

For the third method, *Logistic Regression* (see section 5.1.2), it is interesting to remark that with a maximum of 1000 iterations it had the second worst accuracy level. However, it can be considered an acceptable result (see image 6.22), taking into account that the other algorithms had excellent outcomes in general. The category *comp.sys.ibm.pc.hardware* obtained the worst accuracy level compared to the other

categories and the *misc.forsale* managed to have the best accuracy level (see image 6.23). Analysing the confusion matrix and heatmap (see image 6.24), it may be important to highlight the fact that the category *sci.space* had 19 wrongly classified as *comp.sys.ibm.pc.hardware* (despite the fact that it was a wrong prediction, this type of situations can be really valuable and useful for analytical purposes as one can detect what feature is influencing the prediction).

Finally, the *k-Nearest Neighbours* model (see section 5.2) had the worst level in terms of accuracy (see image 6.26). Analysing the categories predictions, one can notice that the category *rec.sport.hockey* had the worst level of accuracy and *misc.forsale* managed to have to best accuracy level again (see image 6.27). Although it was the worst model int terms of accuracy results, there are some interesting informations that can be extracted from the confsion matrix and hetmap representation, for example, three of the categories reached high levels of imprecision (*sci.med, comp.graphics and comp.sys.ibm.pc.hardware*) with *sci.med* being 22 times wrongly classified as *rec.sport.hockey* (see image 6.28). In some cases, this type of situations might be really relevant for decision making purposes as one can analyse the data to find out the reason for the inaccuracy in question.

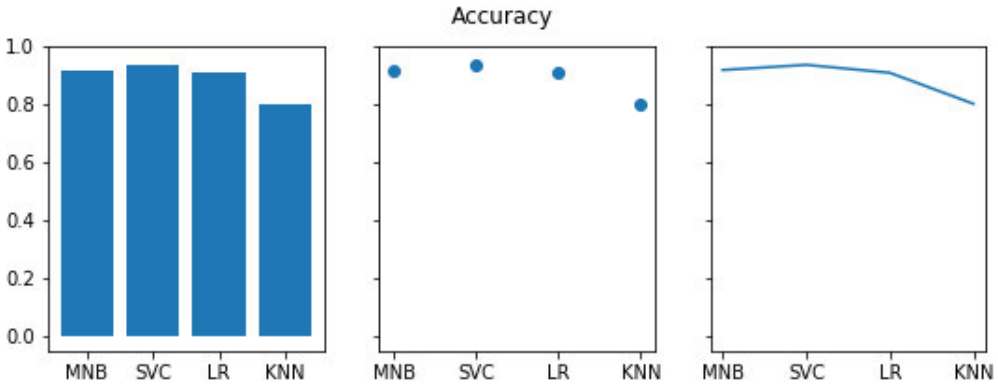

Abbildung 6.29: The Accuracy of the models

The graphic representation (figure 6.29) demonstrates the comparison between the implemented models displaying the considerable disparity between the *k-Nearest Neighbours* model and the other ones. Analysing the accuracy levels, it is important to note that the category *comp.sys.ibm.pc.hardware* happened to have the worst levels of accuracy in 3 models (*Multinomial Naive Bayes, Support Vector Machine and Logistic Regression*). It probably happened because of the amount of similar categories available in the dataset (i.e

comp.sys.mac.hardware, comp.graphics, comp.windows.x or comp.os.ms.windows.misc). Depending on the situation (i.e. business purposes or scientific studies), it would be interesting to investigate what is causing that situation (i.e. studying the dataset profoundly). In contrast, the category *misc.forsale* had the best levels of accuracy in all the models showing that it is composed by singular and easily identifiable articles.

All in all, it can be concluded that despite the remaining differences in terms of accuracy, on average, the models achieved satisfactory results and the *Support Vector Machine* has proved to be the most effective.

Time Duration

The duration of time also plays a significant role when measuring the performance levels of the algorithm. It is an important factor to take into account specially when dealing with large amounts of data (sometimes it might be necessary to revise the model or the hardware components of the machine in order to make the system more productive in terms of time spent).

In terms of Time Duration, the *Multinomial Naive Bayes* model (see section 5.5) had the second best time achieved (see image 6.14) compared to the other models which makes it a really good result for the this specific case. This result might be due to the simplicity and naive nature of the algorithm already discussed (see section 5.5).

The *Linear Support Vector Machine* model finished with the third best duration time. It was not the best in terms of time but it has proven to be a stable model for this situation as the time duration was perfectly acceptable (see image 6.18).

The *Logistic Regression* model obtained the worst levels in terms of duration of time (see image 6.22). It finished as the slowest algorithm in this situation and it might be due to its probabilistic nature as it is a regression algorithm in its essence (even though it can be used for classification problems as well, see section 5.1.2).

Finally, the *k-Nearest Neighbours* model managed to finish the best results in terms of time duration (see image 6.26) and the reason might be the straightforward approach of the algorithm itself (see section 5.2).

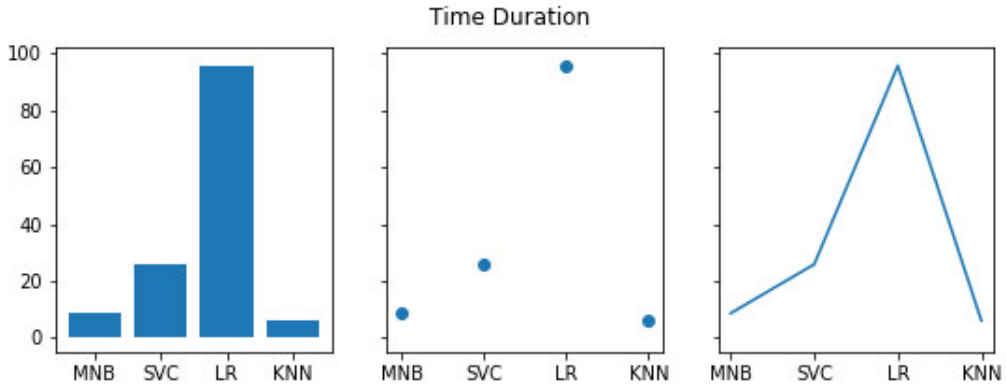

Abbildung 6.30: The Time Duration of the models

In the graphic representation (figure 6.30) it is interesting to emphasize the peak of the outcomes in the *Logistic Regression* which made it the slowest model for this text classification situation. On the other hand, the *k-Nearest Neighbours* has proven to be the fastest model in this case.

All in all, on average, one can conclude that, apart from the *Logistic Regression*, the other three algorithms achieved acceptable results.

6.3 Conclusion

To conclude, as already mentioned previously, there is no *best* algorithm but there are some better algorithms for specific situations.

Comparing the three algorithms implemented on this thesis, it is clear that the *k-Nearest Neighbours* and the *Logistic Regression* ones were the less attractive methods for this specific situation for different reasons. On the one hand, the *k-Nearest Neighbours* had the worst accuracy and on the other hand, the *Logistic Regression* had the lowest performance in terms of duration results compared to the others which makes it the slowest model for this situation. However, both of them provided really important information such as the already mentioned situation regarding the relation between the categories *comp.sys.ibm.pc.hardware* and *comp.graphics* or the probabilistic nature of the *Logistic Regression* algorithm.

The *Multinomial Naive Bayes* and the *Support Vector Machine* algorithms were the most attractive algorithms in this experiment. It is a bit hard to state that one is absolutely better that the other because it would need to depend on the situation. As an example, if the accuracy level was the most vital factor, the *Support Vector Machine* algorithm would be considered the best as it had the best accuracy levels. On the other hand, if time was the most imperative factor, then the *Multinomial Naive Bayes* method would be considered the best as it had a better time duration and managed to achieve a satisfactory results in terms of accuracy simultaneously. Nevertheless, it is plausible to state that the *Multinomial Naive Bayes* model was the most balanced algorithm which puts it in a good position to be elected as the best model in this situation.

From this point on, one can play with different algorithms and methods for classification using the available tools in the Machine Learning field. There are many other approaches to this type of classification problems available on different literature sources or websites (note that some links to other interesting methods are available on the bibliography section of this thesis, see for example [31] and [42] which are slightly different versions of the one used as the basis of this work [10]). It is also important to reiterate and emphasize that, as mentioned before, classification is merely one portion of the immense Machine Learning world. Therefore, the purpose of this thesis will be considered achieved if it ignites the desire of the reader to learn more, not only about the *classification* methods but the *Machine Learning* or *Data Science* worlds in general.

7 Summary and Future Work

Based on the already seen development of the machine learning systems, I strongly believe that Data Science (Artificial Intelligence, Machine Learning, Deep Learning and more) as a whole will be really crucial and relevant in the near future.

Specifically speaking about the future of Machine Learning, I am confident that in a few years it will impact even more areas:

1. Cloud-based systems and Machine Learning systems will be merged into one to offer Data analysis services to everyone.

2. In the business world, companies will be more dependant on the Machine Learning Supervised and Unsupervised methods to achieve better results.

3. Hardware providers will need to redesign their products to adjust them with the Machine Learning standards, which means that the hardware will need to be more powerful in general.

4. The fusion of the Machine Learning with Cyber-Security systems will possibly take the cyber-security industry to a higher level.

The digital world is developing rapidly, the Data Science is playing really important role on this process and I am profoundly convinced that it is "just the beginning" of a long journey. I hope this thesis has contributed to the development of the field and has aroused your attention to the Machine Learning world.

8 Acknowledgement

This thesis represents the successful ending of a long journey. Since I was born in Mozambique and raised in Portugal, completing this academic cycle in Germany makes it even more special to me.

I would like to thank Prof. Olaf Zukunft and Prof. Zhen Ru Dai for being always available to help me when I needed.

I am also grateful to all my friends, family, colleagues and professors from the European Computer Science Bachelor, from the Instituto Superior de Engenharia de Coimbra / *Coimbra Engineering Academy* and from the HAW - Hochschule für Angewandte Wissenschaften / *Hamburg University Of Applied Sciences* for being a part of my story. A special thanks to my sister Eunice Machachane for believing in me and for constantly being available and willing to support me. I sincerely wish you all the best and success in your life.

Finally, I would like to dedicate this thesis to my parents Filda Mafumo and Olímpio Machachane who have always encouraged me to follow my dreams. Thank you for always being my inspiration, for always being present in my life and for always support my academic path. I am profoundly grateful and I hope I manage to keep making you proud consistently. This is for you! I love you.

Thank you!

Literaturverzeichnis

[1] : *Machine learning.* 2019. – URL https://www.techopedia.com/definition/8181/machine-learning

[2] Abraham, Annamma ; Sathya, R.: *Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification.* 2013. – URL https://www.researchgate.net/publication/273246843_Comparison_of_Supervised_and_Unsupervised_Learning_Algorithms_for_Pattern_Classification

[3] Agarwal, Animesh: *Linear Regression using Python.* 2018. – URL https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2

[4] Akritidis, Leonidas ; Bozanis, Panayiotis: *A Supervised Machine Learning Classification Algorithm for Research Articles.* – URL https://dl.acm.org/doi/abs/10.1145/2480362.2480388

[5] Al-Janabi, Mohammed ; Quincey, Ed de ; Andras, Peter: Using supervised machine learning algorithms to detect suspicious URLs in online social networks. In: *ASONAM '17: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017* (2017). – URL https://dl.acm.org/doi/abs/10.1145/3110025.3116201

[6] Arevian, Garen: *Recurrent Neural Networks for Robust Real-World Text Classification.* 2007. – URL https://dl.acm.org/doi/abs/10.1109/WI.2007.91

[7] Attaran, Mohsen: *Machine Learning: The New 'Big Thing' for Competitive Advantage.* 2018. – URL https://www.researchgate.net/publication/327215281_Machine_Learning_The_New_%27Big_Thing%27_for_Competitive_Advantage

[8] Awad, Mariette ; Khanna, Rahul: *Efficient Learning Machines*. URL https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_1, 2015

[9] B., Justin: *Introduction to Perceptron: Neural Network*. 2017. – URL https://blog.knoldus.com/introduction-to-perceptron-neural-network/

[10] Belinić, Tena: *Text classification - simple way to organize your data*. 2018. – URL https://krakensystems.co/blog/2018/text-classification

[11] bhattacharjee joydeep: *Popular Machine Learning Algorithms*. 2017. – URL https://medium.com/technology-nineleaps/popular-machine-learning-algorithms-a574e3835ebb

[12] Britz, Denny: *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. 2015. – URL http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[13] Brodie, Michael L.: *What is Data Science?* 2019. – URL https://www.researchgate.net/publication/333752364_What_Is_Data_Science

[14] Castle, Nikki: *What is Semi-Supervised Learning?* 2018. – URL https://blogs.oracle.com/datascience/what-is-semi-supervised-learning

[15] codecnetworks: *Instance Based Machine Learning*. – URL https://www.codecnetworks.com/blog/instance-based-machine-learning/

[16] Dabbura, Imad: *Gradient Descent Algorithm and Its Variants*. 2017. – URL https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

[17] Dhar, Vasant: *Data Science and Prediction*. 2013. – URL https://dl.acm.org/doi/abs/10.1145/2500499

[18] Dietterich, Thomas G.: Machine learning. In: *Encyclopedia of Computer Science* (2003), S. 1056–1059. – URL https://dl.acm.org/doi/abs/10.5555/1074100.1074563

[19] Do, Thanh-Nghi: *Using Local Rules in Random Forests of Decision Trees*. 2015. – URL https://www.researchgate.net/publication/300242355_Using_Local_Rules_in_Random_Forests_of_Decision_Trees

[20] Education Council, Oregon T. in: *Data, Information, Knowledge, and Wisdom.* – URL https://otec.uoregon.edu/data-wisdom.htm

[21] Foote, Keith D.: A Brief History of Machine Learning. In: *dataversity* (2016). – URL https://www.dataversity.net/a-brief-history-of-machine-learning/

[22] forcepoint: *What is Machine Learning? - Machine Learning Defined, Explained, and Explored.* – URL https://www.forcepoint.com/cyber-edu/machine-learning

[23] Fukumoto, Fumiyo ; Yamamoto, Takeshi ; Matsuyoshi, Suguru ; Suzuki, Yoshimi: *Text classification with relatively small positive documents and unlabeled data.* – URL https://dl.acm.org/doi/abs/10.1145/2396761.2398629

[24] Fumo, David: *Types of Machine Learning Algorithms You Should Know.* 2017. – URL https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[25] Garbade, Dr. Michael J.: *Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences.* 2018. – URL https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb

[26] Georgakopoulos, Spiros V. ; Tasoulis, Sotiris K. ; Vrahatis, Aristidis G. ; Plagianakos, Vassilis P.: *Convolutional Neural Networks for Toxic Comment Classification.* 2018. – URL https://dl.acm.org/doi/abs/10.1145/3200947.3208069

[27] Geron, Aurelien: *Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems 1st Edition.* O'Reilly, 2017. – URL https://www.lpsm.paris/pageperso/has/source/Hand-on-ML.pdf. – ISBN 1491962291, 978-1491962299

[28] Gupta, Prashant: *Decision Trees in Machine Learning.* 2017. – URL https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[29] Hebb, Donald: *The Organization of Behaviour.* URL http://s-f-walker.org.uk/pubsebooks/pdfs/The_Organization_of_Behavior-Donald_O._Hebb.pdf, 1949

[30] IESC, Admin M.: *Artificial Intelligence, Machine Learning, and Deep Learning: Same context, Different concepts.* 2018. – URL https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/

[31] Jain, Rohit: *Support Vector Machine - GitHub Tutorial.* – URL https://github.com/Rohit9314/Classifying-20-news-group-dataset-using-support-vector-machine/blob/master/Support%2BVector%2BMachine.ipynb

[32] javatpoint: *Data Science Tutorial for Beginners.* – URL https://www.javatpoint.com/data-science

[33] Kadhim, Ammar: *Survey on supervised machine learning techniques for automatic text classification.* 2019. – URL https://www.researchgate.net/publication/330501364_Survey_on_supervised_machine_learning_techniques_for_automatic_text_classification

[34] Kadhim, Ammar I.: *Survey on supervised machine learning techniques for automatic text classification.* 2019. – URL https://link.springer.com/article/10.1007/s10462-018-09677-1

[35] Khurana, Sambhav: *Naive Bayes Classifiers.* – URL https://www.geeksforgeeks.org/naive-bayes-classifiers/

[36] Krzyk, Kamil: *Coding Deep Learning for Beginners — Linear Regression (Part 2): Cost Function.* 2018. – URL https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f

[37] Learn, Sci kit: *sklearn.svm.LinearSVC.* 2019. – URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html?highlight=svc#sklearn.svm.LinearSVC

[38] Liu, Yuxi (.: *Python Machine Learning by example.* 2017. – URL https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783553112/5/ch05lvl1sec45/decision-tree-classifier

[39] Lorberfeld, Audrey: *Machine Learning Algorithms In Layman's Terms, Part 1*. 2019. – URL https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-1-d0368d769a7b

[40] McCarthy, John: *What is Artificial Intelligence*. 2007. – URL http://www-formal.stanford.edu/jmc/whatisai.pdf

[41] Naik, Krish C.: *Pipelines In SkLearn - GitHub Tutorial*. – URL https://github.com/krishnaik06/Pipelines-Using-Sklearn/blob/master/SklearnPipeline.ipynb

[42] Nandwani, Vijay: *News Classification - GitHub Tutorial*. – URL https://github.com/vijaynandwani/News-Classification/blob/master/News%20Classification.ipynb

[43] Navlani, Avinash: *KNN Classification using Scikit-learn*. 2018. – URL https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

[44] Novakovic, Jasmina ; Veljovic, Alempije ; Papic, Milos: *EXPERIMENTAL STUDY OF USING THE K-NEAREST NEIGHBOUR CLASSIFIER WITH FILTER METHODS*. 2016. – URL https://www.researchgate.net/publication/324918782_EXPERIMENTAL_STUDY_OF_USING_THE_K-NEAREST_NEIGHBOUR_CLASSIFIER_WITH_FILTER_METHODS

[45] Olah, Christopher: *Understanding LSTM Networks*. 2015. – URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[46] Pooja ; Sharma, Aakanksha ; Sharma, Ankush: *Machine Learning: A Review of Techniques of Machine Learning*. 2018. – URL https://www.researchgate.net/publication/329609597_Machine_Learning_A_Review_of_Techniques_of_Machine_Learning

[47] Poojari, Devesh: *K-Nearest Neighbors and its Optimization*. 2019. – URL https://towardsdatascience.com/k-nearest-neighbors-and-its-optimization-2e3f6797af04

[48] Python Data Science Handbook, An excerpt from the: *In Depth: Naive Bayes Classification*. – URL https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html

[49] Ray, ASunil: *Understanding Support Vector Machine(SVM) algorithm from examples (along with code)*. 2017. – URL https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[50] Ray, Sunil: *6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R*. 2017. – URL https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[51] Rouse, Margaret: *Information Age*. 2014. – URL https://searchcio.techtarget.com/definition/Information-Age

[52] Rouse, Margaret ; Burns, Ed ; Laskowski, Nicole: *Definition: Artificial Intelligence*. 2019. – URL https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

[53] Samuel, Arthur: *Computer checkers (draughts) development*. URL https://en.wikipedia.org/wiki/Arthur_Samuel

[54] Samuel, Arthur: *Samuel Checkers-playing Program*. URL http://www.incompleteideas.net/book/ebook/node109.html, 1950

[55] Sastre, Juan C.: *20newsgroups Classification Tensorflow - GitHub Tutorial*. – URL https://github.com/jcsastre/20newsgroups_tensorflow/blob/master/20newsgroups_Classification_Pure_Tensorflow.ipynb

[56] Schapire, Robert E.: *The strength of weak learnability*. 1990. – URL https://link.springer.com/article/10.1007/BF00116037

[57] Silipo, Rosaria: *From a Single Decision Tree to a Random Forest*. 2019. – URL https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

[58] Simon, Stephen: *Batch And Online Machine Learning*. 2019. – URL https://www.c-sharpcorner.com/article/batch-and-online-machine-learning/

[59] Singh, Seema: *Cousins of Artificial Intelligence*. 2018. – URL https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

[60] Srivastava, Tavish: *Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python and R)*. 2018. – URL https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[61] SrivastavaL, Durgesh ; Bhambhu, L.: *Data classification using support vector machine*. 2010. – URL https://www.researchgate.net/publication/285663733_Data_classification_using_support_vector_machine

[62] Swaminathan, Saishruthi: *Logistic Regression — Detailed Overview*. 2018. – URL https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[63] Tran, Hieu: *Survey of Machine Learning and Data Mining Techniques used in Multimedia System*. 2019. – URL https://www.researchgate.net/publication/333457161_Survey_of_Machine_Learning_and_Data_Mining_Techniques_used_in_Multimedia_System

[64] Triangles: *The cost function in logistic regression*. 2019. – URL https://www.internalpointers.com/post/cost-function-logistic-regression

[65] wikipedia: *Data science*. – URL https://en.wikipedia.org/wiki/Data_science

[66] wikipedia: *Machine learning*. – URL https://en.wikipedia.org/wiki/Machine_learning

[67] Zhou, Victor: *Machine Learning for Beginners: An Introduction to Neural Networks*. 2019. – URL https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9

[68] Zhu, Yangyong ; Xiong, Yun: *Towards Data Science*. 2015. – URL https://www.researchgate.net/publication/277944071_Towards_Data_Science

[69] Ziganto, David: *An Introduction To Online Machine Learning*. 2017. – URL https://dziganto.github.io/data%20science/online%20learning/python/scikit-learn/An-Introduction-To-Online-Machine-Learning/

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Supervised Machine Learning Methods for Classification

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort Datum Unterschrift im Original