TÉCNICO LISBOA

Intelligent Funds Assistant

Inês Sáragga Leal Saraiva

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. João Miguel da Costa Sousa
Prof. Susana Margarida da Silva Vieira

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira
Supervisor: Prof. Susana Margarida da Silva Vieira
Members of the Committee: Inv. Aux. Cátia Matos Salgado
Prof. Rui Miguel Carrasqueiro Henriques

December 2021

Acknowledgments

I would like to express my deep and sincere gratitude towards everyone who helped develop this paper. First, I would like to thank Professor João Sousa for his suggestions and support in this work. I would also like to thank Daniela Moniz, Ana Valente and Alexandre Almeida for their continuous help and encouragement in the development of this thesis.

Second, I would like to thank Axians and IST for providing me with the opportunity to research and explore such an interesting topic.

Last, I would like to thank my family and friends for the support they gave me during the course of this thesis.

Resumo

Atualmente, a União Europeia tem um plano de desenvolvimento económico em vigor para ajudar cada país a melhorar certas áreas de sua economia. Portugal beneficia deste apoio financeiro, no entanto, verificou-se que este financiamento não é totalmente utilizado. Um dos principais desafios do sistema atual de candidatura ao financiamento da UE é a identificação da melhor chamada a responder para cada candidato, o qual foi cuidadosamente explorado nesta tese.

Este trabalho tem como objetivo desenvolver um assistente para os fundos, utilizando técnicas de *Natural Language Processing* para processamento de texto e *Machine Learning* para facilitar a construção dos modelos. O assistente deverá ajudar os beneficiários ao encontrar uma chamada a partir de uma descrição escrita do projeto, na esperança de melhorar a experiência geral dos fundos comunitários.

Este problema foi tratado como classificação multi-classe hierárquica de texto, aproveitando a estrutura hierárquica dentro dos fundos europeus. Quatro modelos foram escolhidos para comparar o desempenho da classificação: *Naive Bayes*, *Support Vector Machines*, *Random Forest* e *k-Nearest Neighbors*. A técnica de extração de *features* utilizada para a representação numérica do texto foi a vectorização TF-IDF, e o desempenho dos modelos que constituem o processo de classificação foi analisado em termos de *accuracy*, F_1 *score* e *Matthews Correlation Coefficient*.

Neste trabalho, foi observado que o SVM supera o resto dos classificadores com algumas exceções. Além disso, os resultados dos classificadores SVM calibrados, ao considerar uma segunda previsão quando o modelo está incerto sobre sua primeira, alcançaram resultados ainda melhores em todos os níveis da hierarquia.

Palavras-chave: *Machine Learning*, *Natural Language Processing*, Classificação hierárchica de texto, *Support Vector Machine*, Vectorização TF-IDF

Abstract

Currently, the European Union has a supporting economic development plan in place to assist each country in certain areas of their economy. Portugal benefits from this financial support, however, it has been verified that this financing is not being fully used. One of the main setbacks in the EU funding application system is the identification of the best call for proposal, which was carefully explored in this paper.

This work aims to develop a funds assistant, using Natural Language Processing techniques for text processing and Machine Learning to facilitate the construction of the models. The funds assistant is expected to match a call from a written description of the project, hoping to improve the overall experience of the community funds system.

This problem was addressed as hierarchical multi-class text classification, taking advantage of the hierarchical structure inside the European funds. Four models were chosen to compare the classification performance: Naive Bayes, Support Vector Machines, Random Forest and k-Nearest Neighbors. The feature extraction technique used for the numerical representation of text was the TF-IDF vectorization, and the performance of the twenty-two models that constitute the whole classification process was analysed in terms of accuracy, F_1 score and Matthews Correlation Coefficient.

In this work, it was observed that SVM outperforms the rest of the classifiers with a few exceptions. In addition, the results from SVM calibrated classifiers, considering a second prediction when the model was uncertain about its first one, achieved even higher performances in all levels of the hierarchy.

Keywords: Machine Learning, Natural Language Processing, Hierarchical Text Classification, Support Vector Machines, TF-IDF vectorization

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

(η_i, γ_i)	Vector space representation of a sample i.
α	Smoothing parameter in Naive Bayes.
β	Balance between recall and precision in F-measure.
ω_ϕ	Weight given to variable m.
ω_ρ	Weight given to variable ρ.
ω_τ	Weight given to variable c.
ϕ	Parcel of the objective function F related to the evaluation metrics.
ρ	Parcel of the objective function F related to percentage of the second predictions made.
τ	Parcel of the objective function F related to percentage of correct second predictions made.
θ_j	Parametrization of class j
θ_{jq}	Likelihood of feature q appearing in a sample of class j.
ξ	Slack.

Roman symbols

(C, r)	Centroid and radius of a node (Ball Tree Algorithm).
$\hat{h}_i$	Output given by h for a sample i.
A, B	Parameters estimated by the maximum likelihood estimation.
b	Bias.
C	Regularization parameter in SVM.
C_c^h	Classifier c in level h.
d	Distance measure.
E	Entropy function.

F	Objective function.
f	Number of features.
f_j	Number of features in class j.
f_{jq}	Number of appearances of feature q in a sample of class j.
G	Gini-index function.
h	Decision function.
l	Number of correct predictions in total.
M	Confusion Matrix.
m	Number of classes.
N	Node.
n	Number of training samples.
o_j	Number of times that class j is the true class.
P	Probability.
p	Parameter to manipulate the Minkowski distance.
r_j	Number of predictions of class j.
R_t	Percentage of second predictions out of all predictions made.
S	Score given to a sample in reference to a certain class.
s	Number of predictions made in total.
T_t	Percentage of correct second predictions out of all second predictions made.
W	Coefficients of the hyperplane.
X	Training set.
x	Input vector.
y	Class/output.

Subscripts

c	Classifier in a certain level.
i	Sample $i \in [1, ..., n]$.
j	Class $j \in [1, ..., m]$.
q	Feature $q \in [1, ..., f]$.

t	Decision Threshold.

Superscripts

h	Level of the hierarchy.

Glossary

AI	Artificial Intelligence
DL	Deep Learning
DT	Decision Tree
ESIF	European Structural and Investment Funds
FN	False Negatives
FP	False Positives
KNN	k-Nearest Neighbors
ML	Machine Learning
NB	Naive Bayes
NLP	Natural Language Processing
NN	Neural Networks
OP	Operational Programme
PDR	Programa de Desenvolvimento Regional
RF	Random Forest
PT	Portugal
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negatives
TP	True Positives
EU	European Union

Chapter 1

Introduction

1.1 Motivation

Currently, the European Union has a supporting economic development plan in place to assist each country in improving certain areas of their economy. The funding given to this plan is called the European Structural and Investment Funds (ESIF) and it is managed by the European Commission and the EU countries together [1]. These funds encourage the development of five major areas: research and innovation; digital technologies; supporting the green ecological economy; sustainable management of natural resources; and small businesses.

Portugal benefits from total funding of 25.9 billion Euros for the period 2014-2020 [2]. The Portuguese government and the European Commission have a partnership agreement, Portugal 2020 (PT 2020), and therefore is able to access a portion of the European Structural and Investment Funds (ESIF). Even though PT 2020 is implemented for the period of 2014-2020, the same structure of the funds will be maintained in the following years, with some possible minor changes. Out of all the existing funds, Portugal has received the most access to the fund ERDF (European Regional Development Fund) with the goal of developing the rural areas of the country.

Besides the agreement PT 2020, Portugal also has access to funding from the European Recovery Plan [3], to help repair the economic and social damage caused by the coronavirus pandemic of 2020. So the EU budget consists of the EU's long-term budget joined with NextGenerationEU, a temporary tool to boost the recovery of the European countries, which will sum up to a total of 1.8 trillion Euros to help restore a post-corona Europe.

It is also important to know the implementation progress of the various funds through the years, taking into consideration that the financial resources assigned to each country may go unused and need to be returned if that is the case. From 2015 to 2020, Portugal had a linear improvement in implementing projects however, only 60% of the budget available per year is actually spent, as shown in the official data from the EU [2]. In Figure 1.1, it is possible to verify this boost, given that not only the spent budget (yellow column) has increased but also the decided budget from the approved projects (dark blue column). Even though in the last years, the budget used improved from 4% of the total budget in 2015 to 62% in 2020, it's still necessary to comprehend why the community funds are not being completely used.

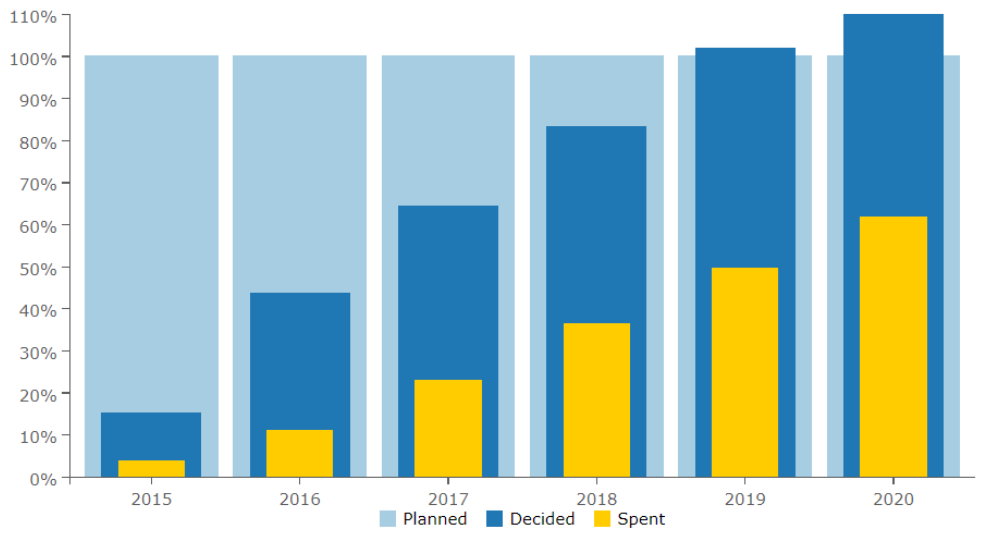


Figure 1.1: Implementation Progress (total cost) for Portugal (ESIF 2014-2020) [2].

To better understand the problem at hand, the Portuguese program PDR 2020 (Programa de Desenvolvimento Rural - Rural Development Program) is used as a study case for this thesis, which is the main support of agriculture projects in the scope of Portugal 2020. The objectives of PDR 2020 are consistent with the main goals of the strategic plan developed by the European Union and are based on the sustainable growth of the agroforestry sector, the promotion of effective management of resources and the creation of circumstances that enable the economic revitalisation of rural areas [4].

1.2 Funds & Grants Context

As it was introduced in the section 1.1, the funding given to Portugal originates from the European Structural and Investment Funds. These funds are, in fact, divided into five different funds [1] with separate sub-objectives and a certain amount of the total budget available for Portugal, as follows:

1. European regional development fund (ERDF), which promotes balanced development in the EU (amounts to 45.8% of available financing);
2. European social fund (ESF), which supports employment-related projects (amounts to 26.2% of available financing);
3. European agricultural fund for rural development (EAFRD), that focuses on projects in the EU's rural areas (amounts to 15.2% of available financing);
4. Cohesion fund (CF), which invests in transport and environment projects (receives 9.9% of available financing);

5. European maritime and fisheries fund (EMFF), that helps improve quality of life along the European coasts (amounts to 1.5% of available financing).

For Portugal, through the agreement Portugal 2020, the funds are, currently, divided into 28 programmes [5] that can be further separated through their scope - National, Regional, Crossborder, Transnational or Interregional - and is managed through several Operational Programmes (OP). For each OP, there is a clear hierarchy that can be used to navigate and find a particular theme of interest. It was found that each OP had its own nomenclature and given a large number of OPs, understanding all of them would be a very complex problem to tackle. For purpose of this thesis, **it was chosen to focus solely on the operational programme PDR**.

With respect to the Portuguese operational programme, PDR 2020, it's presently divided into four main areas of development [6], also known as priority areas: 1) Innovation and Knowledge; 2) Competitiveness and Production Organization; 3) Environment, Resource Efficiency and Climate; and 4) Local development. Besides those four central areas, there are also calls available for Technical Assistance, used inside PDR 2020 itself, and an exceptional measure for farmers and SMEs affected by the COVID-19 pandemic, as shown in Figure 1.2.

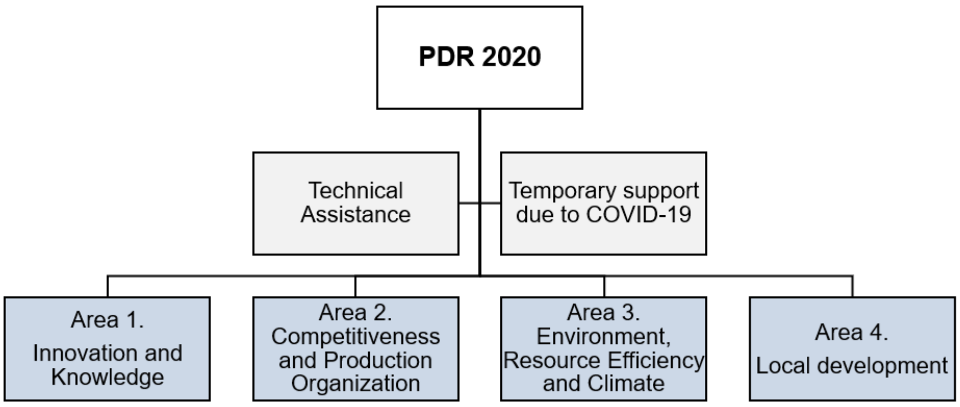

PDR 2020
Technical Assistance
Temporary support due to COVID-19
Area 1. Innovation and Knowledge
Area 2. Competitiveness and Production Organization
Area 3. Environment, Resource Efficiency and Climate
Area 4. Local development

Figure 1.2: Architecture of PDR 2020.

For all the areas mentioned, present in [6], one can find a clear structure (detailed in Chapter 3), which ends in a level with specific scopes and objectives, known as operations (in PDR 2020). Each operation has an ordinance, a law or rule made by the government, where the objectives, beneficiaries of the operation, the criteria of eligibility of beneficiaries and proposed projects, details about the financial support that can be given, the application process, the analysis that will occur after the application submission and other important information, are defined. It's from this ordinance that each call, notice or announcement is made. The calls consist usually of a document released by a certain Operational Programme with restrictions in regards to activity sector, budget available, location, among others possible specifications. Besides those restrictions, calls are temporary and change after a period of time. Each project, that wants to receive EU funding, **needs to answer to a specific call**. Unfortunately, the data available doesn't have a correspondence between calls and projects, but it is possible to associate a project with a certain **operation** (the last level of the hierarchical structure).

1.3 Challenges and Approaches

In the context of access to EU funds, it is crucial to understand the problems involved in this process. As it was mentioned previously, the community funds are not being completely used, leaving, at least, 40% of the funding available unused [2]. Hence, to take full advantage of the financial support given by the EU, it is essential to simplify the process of application in order to attract more beneficiaries and facilitate the attribution of financial access to the funds in question, without having failed application procedures.

One of the main setbacks in the EU funding current application system is the identification of the best call to answer for each applicant. Given that different calls have distinct benefits that can only be applied if some conditions are met, the identification of the call can be quite important. The conditions and associated benefits are usually found, in a more detailed form, inside the ordinances. As it was explained previously, each operation has an ordinance, which is a law or rule made by a government or authority. Consequently, it is an intricate document and may prove to be too difficult to understand for a typical citizen. So, when a citizen is looking for information and comes across such a document, it may cause some discouragement and even alienate a few people from applying.

There are a lot of approaches that can be followed to solve these types of problems. Given that this is an issue that requires the understanding of a text about a beneficiary's project, it is important to explore the applications in Natural Language Processing (NLP), a branch of Artificial Intelligence (AI). NLP is a field of study that focuses on making computers understand and generate human language, usually, involving machine learning (ML) or deep learning (DL) algorithms to facilitate the construction of this type of model. For this problem, in particular, the use of text classification stands out, where the input is a description of the project that the beneficiary wishes to develop and the output is the best fitting call. The traditional approach to follow would be rule-based modelling; however, that is an extensive process that requires time and effort, not only for developing but also maintaining and was dismissed as a possible solution. Consequently, a machine learning approach is an ideal path that can be easily employed without the necessity of experts, which could potentially compromise the results by creating a model with specific vocabulary, associated with the funds in question, that wouldn't necessarily correspond to the language used by the beneficiary. Furthermore, an ML model can also be built quickly for a relatively low cost.

On account of the data available, instead of identifying the call, it was only possible to associate a project with an operation, that specifies a particular topic and area of action. The identification of the environment that a project should be implemented is still pertinent and can be later be associated with the temporary calls that each OP opens. Considering the number of operations existent, it was necessary to explore multi-class classification, where there are more than two classes to consider, versus the typical binary classification, with just two classes to distinguish. Besides exploring multi-class classification, a hierarchical approach was also researched and implemented to organize the data and classifiers, taking advantage of the organization in the OPs.

With these primary challenges and approaches found, the optimization of the application process is a significant step to make the beneficiaries more autonomous, by creating a meaningful application and maximizing the use of funding available. The beneficiary of this progress would be, not only the EU citizens (more particularly, for now, the Portuguese population) but also the SMEs (Small and Medium-sized Enterprises) that are forced to give up on applying and cannot receive proper funding for their projects.

1.4 Contributions

The main objective of this thesis was to develop a Machine Learning model to assist the beneficiary to find the best investment possible for his project, through a description of that project. In regards to the work done, the following contributions of this work can be listed:

- Development and implementation of the proposed framework for an intelligent funds assistant, using Machine Learning algorithms and a hierarchical approach to organize the data available and the classification as a whole. The funds assistant is excepted to assist beneficiaries in a positive manner, hoping to improve the overall experience of the community funds system.
- Comparison between Machine Learning models, analyzing accuracy, F_1 Score and Matthews Correlation Coefficient to find the best performance of the twenty-two models that constitute the sequential classification process. All models were tuned according to accuracy measures.
- Implementation of calibrated SVM classifiers to examine the predictions and associated probabilities, with the objective of improving the hierarchical classification with the consideration of second predictions.
- Evaluation of different text pre-processing techniques for the Portuguese language, such as stemming algorithms and the effect of the removal of accentuation, using accuracy as the measurement.

1.5 Thesis Outline

Past this introductory chapter, this thesis is organized as follows:

- Chapter 2 details the theoretical background into the techniques that were used in the following chapters.
  - Section 2.1 provides a broad introduction of text classification to guide the reader into the steps that occur in this type of problem.
  - Section 2.2 offers detailed information about the pre-processing that occurs when working with text features and the vectorization technique of said input.

  - Section 2.3 details the functioning behind the classification models chosen is explained with attention to the problem at hand (multi-class text classification) and the parameters that were optimized further ahead.
  - Section 2.4 is dedicated to explaining hierarchical classification and the nomenclature used in these sort of problems.
  - Section 2.5 explains the evaluation metrics that were used, such as accuracy, precision, recall, among others, and also strategies that allow the application of these binary metrics to a multi-class problem.

- Chapter 3 focuses on the proposed framework developed to create a classification model for the funds assistant.
  - Section 3.1 explains the hierarchical structure of the PDR 2020 and how that structure is used to create a process of hierarchical classification.
  - Section 3.2 and 3.3 address the selection of the models and metrics chosen to evaluate.
  - Section 3.4 explains the procedure followed to implement second predictions in SVM models, and the heuristics problem to understand when a second prediction should considered, in terms of a decision threshold.
  - Section 3.5 explains the pre-processing of the text, with some experimental tests in order to decide which algorithm performs the best and which approaches should be followed.
  - Section 3.6 provides a summarization of the public data available and details the information that it contains, in regards to the classification's input and output.
- Chapter 4 provides all the analyses and results from the models built for the funds assistant.
  - Section 4.1 focuses on parameter tuning of each model tested.
  - Section 4.2 brings the analysis of the hierarchical classification, using the predictions from past levels to chose a classifier and retrieve the following results.
  - Section 4.3 shows an individual analysis of each classifier built, considering that the predictions from the previous levels were correct.
  - Section 4.4 focuses on the enhancement of the SVM models, solving the heuristics problem defined in 3.4 and implementing decision thresholds on a set of particular classifiers.
- Chapter 5 describes, in a general matter, the problem in this thesis, the approaches followed and the results.
  - Section 5.1 focuses on the summarization of the results in this work.
  - Section 5.2 addresses future works and further improvements in regards to the work developed here.

Chapter 2

Text Classification

In this chapter, there is an overview of the text classification pipeline that is usually followed and also a brief resume of the most used methods and algorithms for this type of problem. In section 2.2, text processing techniques, such as removal of stop-words, stemming, lemmatization and vectorization, are explained. Next, the theoretical background of the implemented models is detailed (2.3), followed by an explanation of the hierarchical classification used (2.4). The final section of this chapter is dedicated to the evaluation metrics considered, with detailed formulas and explanations (2.5).

2.1 Introduction to Text Classification

Text classification algorithms use unstructured data to extract features and assign the text to a certain category according to its content. There are several approaches to automatize this problem, such as rule-based systems, that involve creating a set of rules by hand, where each rule consists of an antecedent or pattern to recognize and a predicted category if that pattern is found. This particular method is time-consuming to implement, especially if the problem at hand is a complex one, and in general, requires a lot of analysis and testing. Besides those disadvantages, rule-based systems are difficult to preserve since the introduction of new rules may affect the old ones. Given the downsides of rule-based systems, other options are more attractive such as Machine Learning based systems, that learn to make classifications based on past observations. ML algorithms use pre-labeled examples (and sometimes unlabeled) as training data to learn the different connections between the texts given. Contrary to the rule-based systems, ML models are more accurate, especially on more complex problems, and are easily maintained seeing that it always possible to fed more examples into the system to learn new tasks.

To be able to construct the best classifier possible for this problem, it's crucial to understand the pipeline for text classification, explained in detail in Kowsari et al. [7]. The main steps for a text classification problem are shown in Figure 2.1: with a set of labeled texts or documents, the first phase is to process the text and clean it of unnecessary information, and this is known as pre-processing. The next step is feature extraction, which consists of transforming the text into a numerical representation that

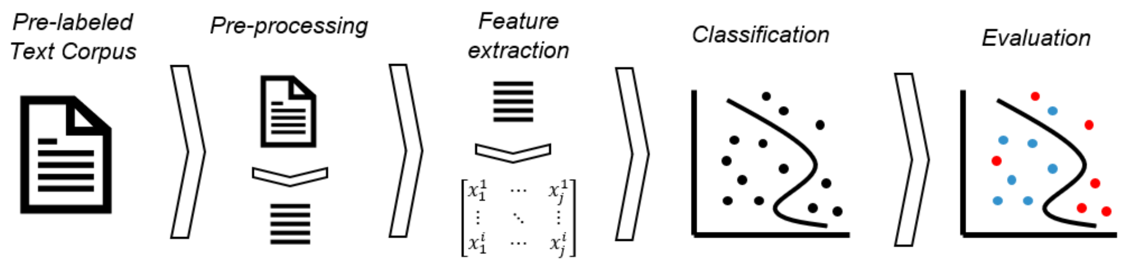


Figure 2.1: Steps of Text Classification.

can be used as input for a classification model. With this vector representation and the associated labels of the input, a classification model is trained, creating connections between the text and the labels. The final step is evaluation of the model and verification of the predictions made. If the predictions are correct, it can be concluded that the model was able to learn distinct features for each label. For each step described here, there are several models and techniques available to develop depending on the purpose of the classifier. In the following sections, the models that were considered most relevant are detailed.

2.2 Feature Extraction

The first step for training a Machine Learning NLP classifier is extracting the important features. As it was mentioned previously, the input is an unstructured text or document, so it's necessary to convert them into a structured feature space, where each text is transformed into a numerical representation in the form of vector.

2.2.1 Text Pre-processing

Before applying the feature extraction algorithms, it's essential to tokenize the text and clean it of unimportant information, such as stopwords (e.g. ”the”, ”she”, ”which”, etc). There are also others methods of pre-processing text such as stemming and lemmatization [7]. Besides those procedures, another important step is to deal with the possible accentuation present in the text, if the language uses diacritics (such as ´, ‘, ˆ , ˜ and ç). In Chapter 3, it is analysed in detail the effects of maintaining or removing accentuation.

Stemming is the process of chopping off a word and keeping the stem of that word. Stemming offers good results in some cases, for example, with words like ”working”, ”worked” and ”work” that all convert to ”work”, therefore creating a quick and simple connection between three previously different words. However, the same cannot be said for words with the same spelling and different meaning, such as ”lie” that can mean to lay down or to tell something untruthful, nor with words that have the same stem but different interpretation, such as ”meanness” and ”meaning”. (that both steam to ”mean”). The state-of-the-art stemming used for English language is the Porter stemming algorithm, also known as

Porter Stemmer. However, it is not appropriate to use it for Portuguese. In 2001, Orengo and Huyck [8] published an article with a novel stemming algorithm, created specifically for the Portuguese language, named ”Removedor de Sufixos da Língua Portuguesa” (RSLP). This stemmer is composed of 8 steps, each with its own set of rules, where each rule dictates the suffix to be removed and the replacement suffix if necessary. Another option to consider is the Snowball stemming algorithm introduced in [9], created with the purpose of introducing non-English stemming algorithms. Currently, this algorithms supports Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Romanian, Russian, Spanish and Swedish languages.

As for **lemmatization**, it takes a different approach to removing stems. Instead of removing inflections (”ing”, ”ed”, among other), it relies on a lexical knowledge base to obtain the correct stem of the words. In this case, the words ”meanness” and ”meaning” would be lemmatized to themselves instead of ”mean” (that would occur in stemming). To conclude, lemmatization offers better precision than stemming, but at the cost of recall.

2.2.2 TF-IDF Weighting Scheme

The most basic form of vectorization is Term-Frequency (TF) method, where each text or document is represented by a vector composed of the frequency of a certain term (e.g. word) in said text, which also plays a part in the numerical representation of TF-IDF. There are other models that provided more complex weighting techniques like Bag-of-Words (such as word2vec or doc2vec), described in more detail in Kowsari et al. [7], and also the TF-IDF weighting scheme. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorization creates a representation of words where each term is weighted based on its inverse document frequency. This representation is more relevant, compared with the TF method, due to the fact that terms with high frequency are usually very similar and need to be de-emphasized by the IDF weight, as explained in Han and Karypis [10].

The scoring of TF-IDF has two components as specified by the name: TF and IDF. First, the TF representation will assign a weight to each term in a document, that is equal to the number of occurrences of the term in that document [11]. With this approach, the TF weight of a document represents a quantitative compilation of that document and doesn’t take into account the order of terms, which is why this representation can be called a bag-of-words model. Now, this first model considers that all words have the same importance which is not true. For example, some words that could appear a lot in the descriptions for a project, such as ”project”, ”plan”, ”proposal”, have no significance for differentiating projects. These words, even though they do not have an important meaning, aren’t considered stop-words (such as ”the”, ”he”, etc) and therefore, aren’t be removed during the text pre-processing. To fix this problem, the second component in this vectorization comes into play. IDF stands for Inverse Document Frequency and this method will attenuate the effect of terms that occur too often to be meaningful in classification. The main idea here is to reduce the TF weight with a value that grows with its collection frequency, which is the number of occurrences of a term in all the documents available [11]. However, instead of collection frequency, it's more common to use document frequency DF, which is defined as the number of documents in the collection available to contain a certain term. So, to scale the weight of a term as required, the inverse document frequency is defined as the logarithm of the total number of documents n divided by the document frequency DF, as shown in (2.1).

$$\text{IDF} = \log \frac{n}{\text{DF}} \tag{2.1}$$

After defining both term frequency and inverse document frequency, the TF-IDF weighting scheme is given by (2.2).

$$\text{TF-IDF} = \text{TF} \times \text{IDF} \tag{2.2}$$

In conclusion, the IDF of a rare term, that only occurs in a small number of documents, will be high, which means that its TF-IDF will also be high. But if a term is very frequent then its IDF will be lower, resulting in a small TF-IDF weight and if a term occurs only in a few times in a document, the TF will also be low, which will result in an overall TF-IDF small weight.

2.3 Classification Models

To classify text and documents, several algorithms have been developed and explored. The models chosen for this thesis were the following: Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF) and k-Nearest Neighbors (KNN). These models were chosen based on the literature and articles of similar works to the one developed here. For more details, Section 3.2 can be consulted.

2.3.1 Naive Bayes Classifier (NB)

The Naive Bayes algorithms are based on the calculation of the conditional probability (also known as Bayes conditional probability) of each class (output, y) for a given text (input, vector x = [$x_1, ..., x_f$]), shown in (2.3), where the output is the class with the highest probability [7].

$$P(y \mid x_1, ..., x_f) = \frac{P(x_1, ..., x_f \mid c) \times P(y)}{P(x_1, ..., x_f)} \tag{2.3}$$

In (2.3), there are 4 different parcels:

- $P(y \mid x_1, ..., x_f)$ is the posterior probability of a class given a certain set of features.
- $P(x_1, ..., x_f \mid y)$ represents the probability of a set of features $x_1, ..., x_f$ given a certain class and is called likelihood.
- $P(y)$ is the prior probability of the class y.
- $P(x_1, ...x_f)$ is the prior probability of the feature vector x.

The Naive Bayes classifier, besides using the Bayes probability, uses an assumption of class conditional independence, meaning that the values of x on a given class are independent of the other values of x. In other words, the likelihoods $P(x_i \mid y_j)$ are independent given the class y (2.4). With this "naive" assumption, the formula in (2.3) can be simplified to (2.5).

$$P(x_i \mid y) = P(x_i \mid y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_m) \tag{2.4}$$

$$P(y \mid x_1, ..., x_f) = \frac{\prod_{i=1}^{n} P(x_1, ..., x_f \mid y) \times P(y)}{P(x_1, ..., x_f)} \tag{2.5}$$

Given that the prior probability of the features, $P(x_1, ..., x_f)$ is constant, it's possible to create a proportionally relation between the posterior probability and the other terms, as can be seen in (2.6). This is the probability that will be calculated for each input given and every class existent and the output will be the class with the highest probability.

$$P(y \mid x_1, ..., x_f) \propto \prod_{i=1}^{n} P(x_1, ..., x_f \mid y) \times P(y) \tag{2.6}$$

Now, because this is a text classification problem, the model that is used is not the classic Naive Bayes but one of its most used variants: Multinomial Naive Bayes. Other existent variants were dismissed as possible solutions, such as Bernoulli Naive Bayes, where the only the presence of the feature is found, and Gaussian Naive Bayes, that works with continuous data distributions. The multinomial classifier handles inputs with discrete features, such as word counts and TF-IDF representations, which means that the data distribution is parametrized by vectors $\theta_j = (\theta_{j1}, ..., \theta_{jf})$. Each class j will have a certain vector, θ_j, composed of f features and each entry, θ_{jq}, represents the likelihood of feature q appearing in a text of class j, $P(x_q \mid j)$. The parameters θ_j are estimated as shown in (2.7), with the following parcels:

- $f_{jq} = \sum_{x \in X} x_q$ represents the number of times that feature q appears in a sample of class j in the defined training set, X.
- $f_j = \sum_{q=1}^{n} f_{jq}$ is the total count of all features in class j.
- α represents the smoothing parameter.

$$P(x_q \mid j) = \hat{\theta}_q = \frac{f_{jq} + \alpha}{f_j + \alpha \times f} \tag{2.7}$$

As it can be seen in (2.7), the multinomial model adds a new variable: α. This parameter is related to the Laplace smoothing that occurs. Laplace smoothing is a technique that addresses the problem of zero probabilities in Naive Bayes. If the probability calculated for a certain input follows the formula in (2.6), and some word in that input has never been seen before, the likelihood, $P(x_1, ..., x_f \mid j)$ will be zero. However, with the new likelihood formula, in (2.7), even if f_{ji} is zero, the α can prevent the probability of being zero if a word in not in the training dataset.

2.3.2 Support Vector Machines (SVM)

Support Vector Machine (SVM) is one of the state-of-the-art classifiers used for their accuracy without needing much training data. Unlike NB classifiers, SVM is more computational expensive and has faster and more accurate results. SVM classifiers are usually used for binary classifiers, but can be adapted to multi-class problems by implementing approaches, such as, One-to-One or One-to-Rest, which are detailed ahead.

SVM classifiers use linear conditions to separate two or more classes. For a binary problem, the SVM algorithm defines that the class y_i, for the train set X_i, is drawn from $\{-1, +1\}$. One of the most important concepts of Support Vector Machines is the maximum margin hyperplane. This hyperplane is accompanied by two possible separating hyperplanes (positive and negative hyperplane), which are parallel to the maximum margin hyperplane and have their corresponding support vectors (closest data points to touch the hyperplanes). To better understand these definitions in a geometric sense, Figure 2.2 contains an example classification between two classes, with captions for the hyperplanes, support vectors and maximum margin.

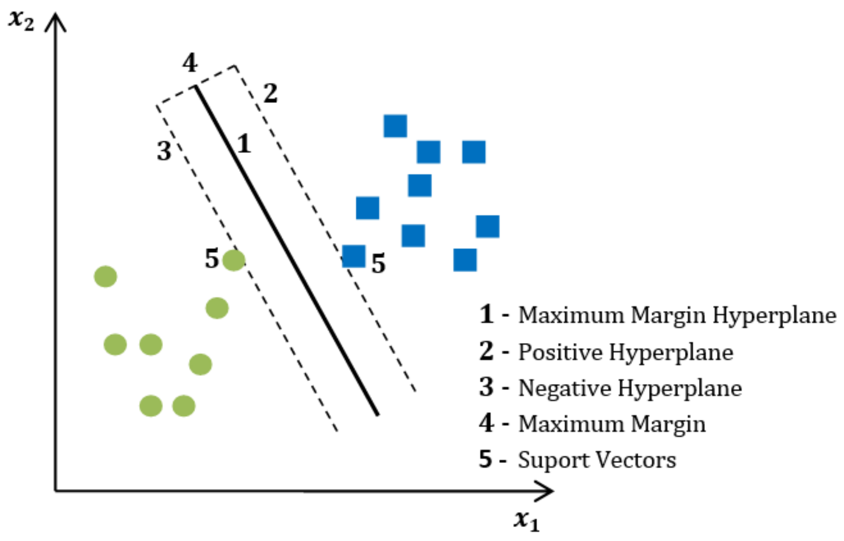


Figure 2.2: Geometric Representation of SVM.

The objective of support vector machines is to achieve maximum margin separation of the hyperplanes. In geometric terms, the separating hyperplane can be represented by $W \cdot X + b = 0$, where X represents the training set, W is a vector representing the coefficients of the hyperplane of separation and b is the bias. The two symmetric support vectors have the following form: $W \cdot X + b = +1$ and $W \cdot X + b = -1$. The distance between the two separating hyperplanes is given by $2/\|W\|$ and maximizing this formula is identical to minimizing $\|W\|^2/2$ [12]. With these definitions, it's possible to describe the problem as a convex quadratic optimization problem with two constraints (positive and negative hyperplane), as shown in (2.8).

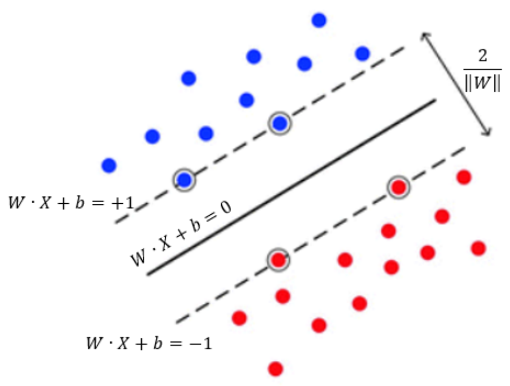
$$\begin{aligned} \text{Minimize} \quad & \frac{\|W\|^2}{2} \\ \text{subject to} \quad & W \cdot X_i + b \geq +1, \forall i : y_i = +1 \\ & W \cdot X_i + b \geq -1, \forall i : y_i = -1 \end{aligned} \tag{2.8}$$

However, the problem defined in (2.8) doesn't account for the fact that the data may not be linearly separable, as it usually happens in real life. Therefore, soft-margins methods were developed to assist in this problem [12]. In the soft-margin algorithm, a slack $\xi_i \geq 0$ is introduced in training for the algorithm to be able to violate the support vector constraint, for a penalty. The optimization problem is modified, in terms of the objective function and constraints, to account for the new slack variables introduced, as shown in (2.9). The new objective function introduces a new constant C, that regulates the importance of the margin and slack requirements. Considering that the objective function is being minimized, small values of C are associated with a soft-margin SVM and large values of C are more appropriate for a hard-margin SVM.

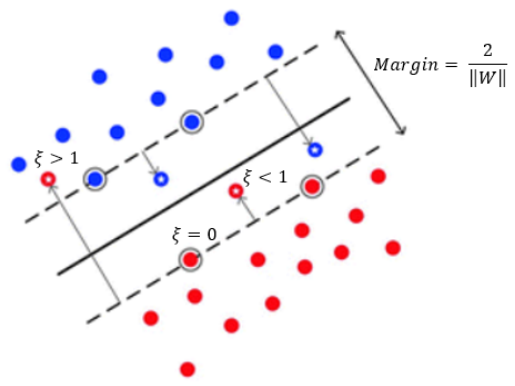
$$\begin{aligned} \text{Minimize} \quad & \frac{\|W\|^2}{2} + C \cdot \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & W \cdot X_i + b \geq +1 - \xi_i, \forall i : y_i = +1 \\ & W \cdot X_i + b \geq -1 + \xi_i, \forall i : y_i = -1 \\ & \xi_i \geq 0 \end{aligned} \tag{2.9}$$

To understand a little bit better the use of the slack variables in this optimization problem, Figure 2.3(b) provides an example of the values given to some data points: the points that touch the negative hyperplane (support vector) have slack variables $\xi = 0$; a point that crosses the negative hyperplane but not the maximum margin hyperplane has a value minor to 1, $\xi < 1$; and a point that crosses the maximum margin hyperplane (to the ”opposite side of his class”) has a slack value bigger than 1, $\xi > 1$, meaning that has a bigger penalty. It's also possible to observe what a hard-margin SVM would look like in comparison to soft-margin in Figure 2.3(a).

Another topic that is necessary to address with SVM models is the kernel used. The most basic kernel is the linear one, which divides the classes with a linear hyperplane. However, a linear model may not be ideal for every classification. The solution to this problem is to use the ”kernel trick”, that allows the model to operate in a high dimensional feature space, without having to compute the exact coordinates of the data into that space. In Figure 2.4, it's possible to observe an illustrative example of this type of transformation: the separation in the input space was impossible to be done by a linear hyperplane but, in the feature space created by the kernel transformation, it was possible to linearly separate the classes. The most common types of kernel are polynomial, Radial Basis Function (RBF)



(a) Hard-margin



(b) Soft-margin

Figure 2.3: Difference between soft and hard-margin SVM.

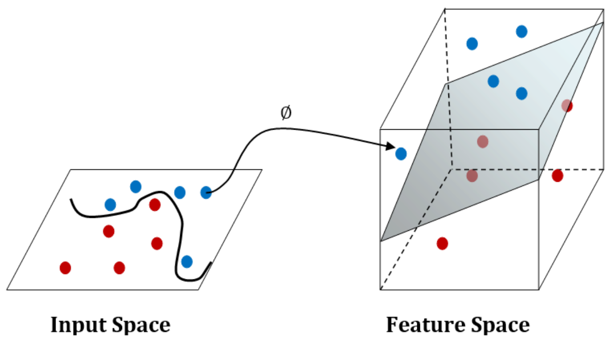
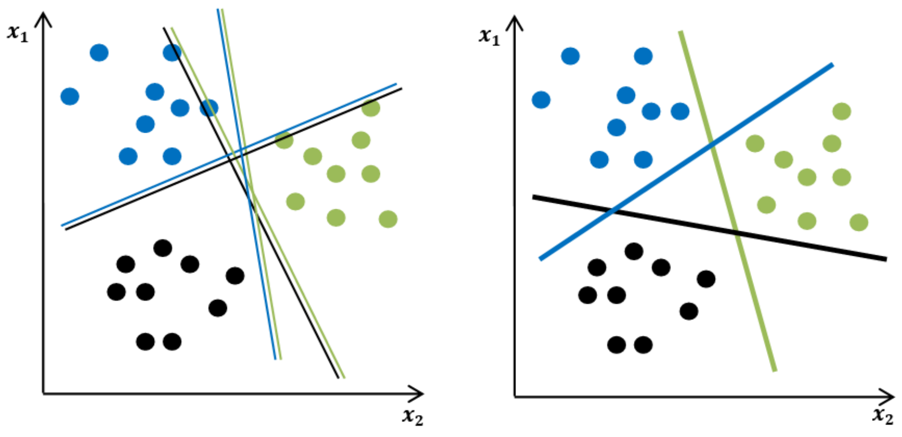


Figure 2.4: Illustrative example of the application of a kernel.

and sigmoid, which implement different types of transformation functions to the input given.

Given that the problem at hand is not a binary one and that the SVM algorithm isn't originally designed for multi-class classification, it's important to understand the strategies that can be followed to overcome this limitation [12]. The first approach is called ”One-to-One” and it divides the problem into multiple binary classification problems, where one binary classifier is built for each pair of classes. With this method, the classifier uses $\frac{m \cdot (m-1)}{2}$ SVM models to classify m classes. In Figure 2.5(a), it can be seen an example of this type of classification (with 3 classes), where each hyperplane takes into account only 2 classes. For example, the green-blue line is maximizing the separation between the blue and green classes and doesn't consider the red class. To choose the multi-class output for this type of approach, a voting procedure is used, where each classifier votes for the predicted class. The output is the class with more votes and if all predictions are correct, then the class chosen should contain $m - 1$ votes, while all others should have less than that.

The second approach that can be used is the ”One-to-Rest”, where each binary sub-problem classifies one class against all others. This approach uses m SVM models, where each SVM predicts membership in one of m classes. In Figure 2.5(b), the hyperplane that separates the classes considers only one class versus all others where, for example, the green line is separating the green data points



(a) One-to-One Approach (b) One-to-Rest Approach

Figure 2.5: SVM multi-class approaches.

from the red and blue data points all at once. In terms of the output shown by the model, if all the predictions calculated by all the binary classifiers are correct, then there is only one class that can be the multi-class prediction. If the binary predictions are wrong, it's necessary to compare the decision values $h(x)$ of the classifiers and choose the largest one, which implies a larger distance from the decision surface. So the multi-class prediction is decided by $\arg\max_{c=1,..,m} h_c(x)$, where h_c represents the decision values of the classifier c.

Probabilistic interpretation of SVM output

As explained previously, the decision function used for SVM works with distances to the hyperplane, more specifically, the signed distance to the hyperplane. Unlike the Naive Bayes model, which works with likehoods and probabilities which can be easily interpreted, the SVM algorithm outputs distances to the hyperplane that can vary given the model, data and other parameters, making it difficult to analyse. To solve the problem of interpretation of the SVM scores, a few approaches have been developed, as stated in [13]. The first approach is Platt's scaling. This algorithm calculates the probabilities considering the SVM scores, $h(x)$, and two parameters, A and B, which are estimated by the maximum likelihood estimation 2.10. For multi-class problems, the function $h(x)$ is the decision function obtained from the multi-class approaches mentioned earlier, One-to-Rest or One-to-One. This method is a parametric approach, where a logistic regression model is trained on the classifier outputs. While this method works for models that have balanced training data, it can be ineffective for imbalanced classification problems.

$$P(y = 1 \mid x) = \frac{1}{1 + \exp\left(A \cdot h(x) + B\right)} \tag{2.10}$$

The second approach considered in [13] was the Isotonic Regression, which, unlike Platt's scaling, is non-parametric and does not depend on the shape of the mapping function (which exists for multi-class problems). This method follows the optimization problem described in 2.11, where y_i represents the true class of sample i and $\hat{h}_i$ is the output given the probabilities calculated for a sample i.

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n} (y_i - \hat{h}_i)^2 \\ \text{subject to} \quad & \hat{h}_i \geq \hat{h}_j \text{ if } h_i \geq h_j \end{aligned} \tag{2.11}$$

The Isotonic Regression is more common than Platt's logistic model and can correct any distortion of the model (e.g. the imbalance of classes), but is also more likely to overfit, especially on smaller datasets (less than a thousand training samples).

2.3.3 Random Forest (RF)

Another famous algorithm used for text classification is the Random Forest Classifier. This classifier is actually an ensemble model, meaning that it uses individual decision trees to make a random forest Kowsari et al. [7]. Random Forest can also be considered as a variant of the Bagging approach (bootstrap aggregating), given that it follows the main steps of bagging and then uses the decision tree algorithm to build base classifiers. This classifier, unlike the SVM model, can be used for multi-classification problems, without having to consider solutions such as One-to-Rest or One-to-One strategies.

To understand the random forest classifier, first, it's important to explain the base classifiers used: decision trees. A decision tree has three components: decision nodes, leaf nodes and a root node, which can be seen in Figure 2.6. Each internal node (non-leaf node) is associated with a certain input feature while each lead node is labeled with a class. Each branch of the decision tree has a set of decision rules, associated with a particular class, which is found at the end of that branch (leaf node).

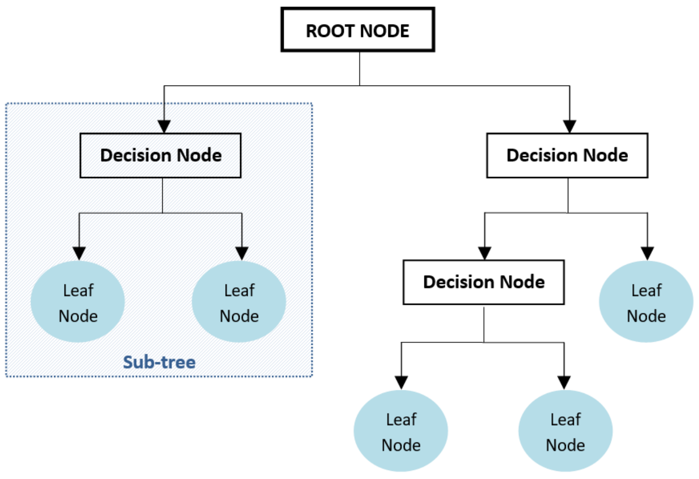


Figure 2.6: Decision tree.

The decision tree algorithm divides the data in a hierarchical form, where the leaf level (lowest level) relates directly to different classes available in the problem. The partition of a node into several branches uses a split criterion that tries to recursively split the training data to maximize the discrimination (distinction) among the different classes (this is also called recursive partitioning). By maximizing discrimination, the level of skew in a given node is maximized, meaning that the distribution of the data in that node is unbalanced and can be divided. To quantify the skew among different classes, a measure like the gini-index or entropy can be used [12]. These measures try to identify the features with the most information regarding a certain class so that, when the data is split, only the most useful information is passed down the tree. The gini-index of a particular node N, with m different classes, where $P(y_j)$ represents the probability of belonging to class j, is defined in (2.12). The value of this measure is in the interval $[0, 1 - 1/m]$ and the smaller this value, the greater the skew is. If the classes are balanced then the value is $1 - 1/m$, which corresponds to the maximum possible value.

$$G(N) = 1 - \sum_{j=1}^{m} P(y_j)^2 \tag{2.12}$$

Another measure that can be used is the entropy, defined in (2.13). This measure is used to measure the randomness (or impurity) of a dataset and the value of it varies between $[0, \log m]$. The maximum entropy possible is $\log m$, which occurs when the samples are perfectly balanced among the different classes. Like with the gini-index, the smaller the entropy value, the greater the skew will be [12].

$$E(N) = -\sum_{j=1}^{m} P(y_j) \cdot \log P(y_j) \tag{2.13}$$

With these measures defined, the split is performed recursively while training the model until a certain termination criterion is satisfied. The stopping criterion can be, for example, that all the data in the node belongs to the same class, which can be difficult to achieve. More realistically, a minimum level of skew (or purity) or a minimum number of samples in the node can be demanded. A recurrent problem with decision trees is overfitting. In order to prevent overfitting, the decision tree has to stop its growth at a specific point. However, it's not possible to predict that point and several approaches can be followed to "prune" a decision tree [12]. One of those is to use a minimum description length (MDL) principle in deciding when to prune a node from the tree. The MDL is an inductive inference method and it states that the shortest description will have the best model and overall, a smaller tree will have better results. Another approach that can be followed is to hold out a small training subset during the decision tree training and then, test to see if replacing a subtree with a single node improves the classification accuracy of the training subset. If that happens, that subtree is pruned and replaced with a node. However, these approaches are quite complex and the models that are going to be used contain different parameters that should be adjusted to prevent overfitting, introduced in section 4.1.

The second concept that is necessary to define to fully understand random forest models is bagging. Bagging is an approach that combines the results from models that learn independently from each other in parallel using different samples. The training samples in each classifier are chosen by sampling with replacement, and these are referred to as bootstrap samples. The final result from this method is usually found using a deterministic averaging process. Overall, bagging is effective for reducing the variance because of its random nature but not for reducing bias.

In short, Random Forest Ensemble Classifier uses a bagging approach with several decision tree models built with a different set of training data. To combine the results of the different classifiers, the RF model uses a majority voting or an averaging approach of the prediction results of all the trees. An example of a Random Forest classifier can be seen in Figure 2.7.

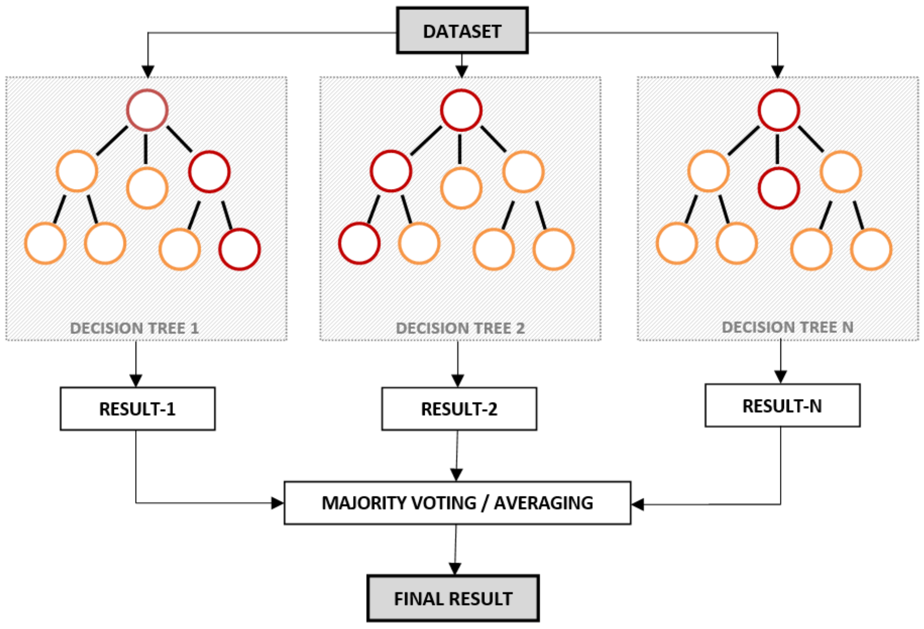


Figure 2.7: Random Forest Ensemble Classifier.

2.3.4 k-Nearest Neighbor (KNN)

Han and Karypis [10] describes the k-Nearest Neighbor (KNN) classifier as an instance-based algorithm that first computes the nearest neighbors of a text and then, puts together the similarities of the text to the k nearest neighbors according to the class of the neighbors and as the final step, the text is assigned to the most similar class. In instance-based learning, the test samples are directly related to the training data to create a locally optimized model, specific to the test instances. Methods that operate in such a manner are referred to as lazy learning algorithms and one advantage of this type of approach is that they can avoid the information loss that usually occurs by not using the full dataset when training a model. However, the drawback of this method has also to do with the fact that it uses all features equally in computing similarities, which can lead to poor measurements [12], and various techniques have been developed to adjust the weight of these features [10]. In terms of the application of this classifier, like the Random Forest, it can be used for both binary and multi-class classification problems without any adaptations.

In the KNN classifier, the top k nearest neighbors in the training data are found for a certain test sample. The search for the k closest data points is called Nearest Neighbor Search (KNNS) and it's a well-known optimization problem. To express the proximity between data samples, the data is transformed to a vector space where the similarity can be measured through distances such as the Minkowski distance, Euclidean distance, Manhattan distance, among others. The Minkowski distance, defined as (2.14), is a metric calculated only in a normalized vector space, where the distances cannot be negative. There is the parameter p in this distance that can be manipulated to give different distances. If $p = 1$, then this metric is called Manhattan distance, where the distance between 2 points is the sum of the absolute differences of their Cartesian coordinates, as shown in (2.15). However, if p is set to 2, this metric becomes the Euclidean distance, a measure of the direct line between 2 points in the Euclidean space (2.16).

$$d(\eta, \gamma) = \left(\sum_{i=1}^{n} (\eta_i - \gamma_i)^p \right)^{1/p} \tag{2.14}$$

$$d(\eta, \gamma) = \sum_{i=1}^{n} \eta_i - \gamma_i \tag{2.15}$$

$$d(\eta, \gamma) = \sqrt{\sum_{i=1}^{n} (\eta_i - \gamma_i)^2} \tag{2.16}$$

There are several algorithms that can be used to solve the nearest neighbors search optimization problem. The most straightforward method to find the desired neighbors of a certain sample is to calculate the distance between all the other data points to that one, known as Brute Force. Despite being a simple solution, this approach is computationally expensive and not practical for large datasets. Another approach that can be used is the K-Dimensional Tree (K-D Tree), where some information about the distances is simply deduced from others, without having to explicitly compute their distance. For example, if point A is very far away from point B and point B is close to point C, then it can be assumed that point A and C are distant from each other. The K-D Tree is a space-partitioning data structure that creates a binary tree structure to organize the data points along the Cartesian axes (meaning that the data is partitioned along the Cartesian axes). In Figure 2.8, an example is shown with 10 points being divided in a tree structure with 4 levels and 9 partitions. While this method is less computationally expensive than the one before, it's still not sufficient for large datasets.

The last method considered here is the Ball Tree data structure. This approach also partitions the data, not along the Cartesian axes, but in a series of nesting hyper-spheres ("balls"). The Ball Tree algorithm recursively divides the data into nodes with a specific centroid C and radius r, and each data point in the node is within a certain hyper-sphere, as shown in the illustrative example of Figure 2.9. By calculating the distance between the test point and the centroid, is possible to determine a lower and upper bound on the distance to all data points in that node, using the triangle inequality: $|\eta + \gamma| \leq |\eta| + |\gamma|$. This algorithm has the best performance for high dimensional datasets although it's actually more complex to build this tree than the K-D Tree.

After finding the top k nearest neighbors, the decision rule for the predicted class is given in (2.17), where $S(x_i, y_j)$ refers to the score given to the sample i in reference to class j out of m classes. After calculating the scores values, which are the similarities between texts (distance) as this is a multi-class problem, the algorithm chooses the class with the highest score [7]. An example of this type of

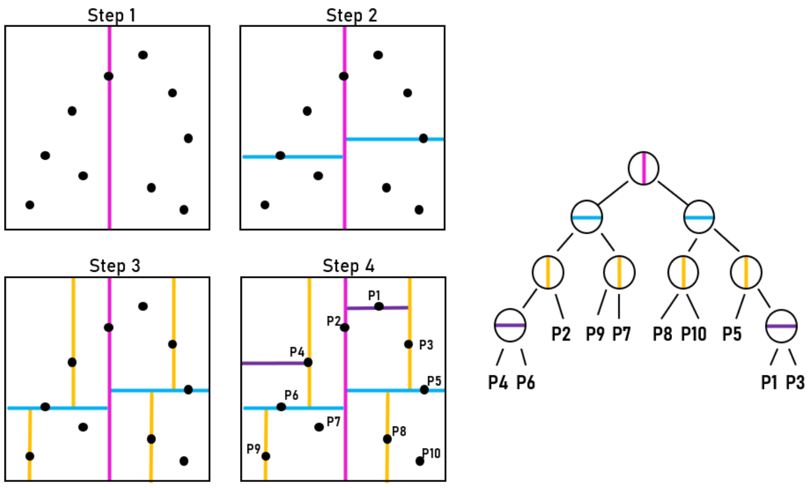


Figure 2.8: Illustrative example of the K-D Tree algorithm.

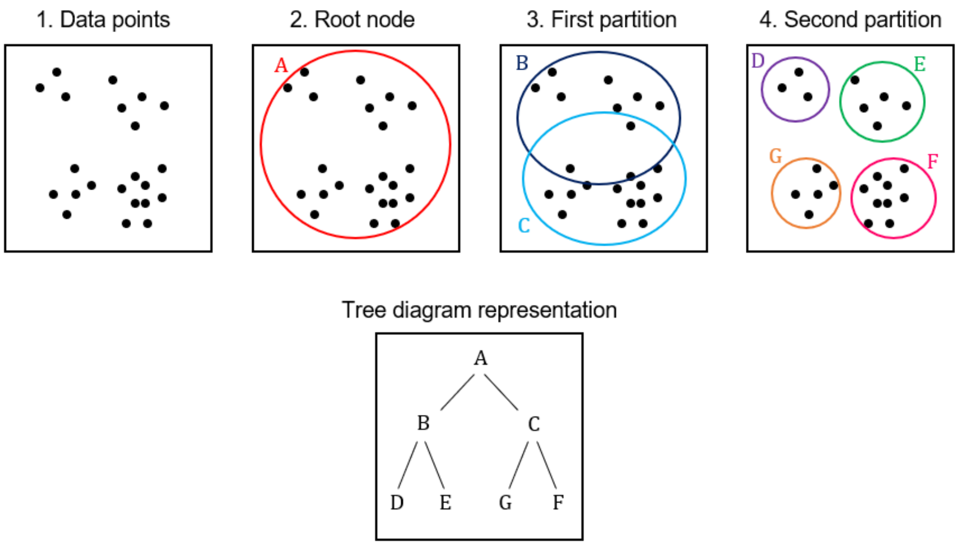


Figure 2.9: Illustrative example of the Ball Tree algorithm.

classification can be seen in Figure 2.10 for 3 classes with 5 nearest neighbors found for a sample x_i.

$$h(x) = \arg\max_{j} S(x_i, y_j) \tag{2.17}$$

2.4 Hierarchical Classification

In the article published by Dumais and Chen [14], it is explored a hierarchical structure for classifying a large collection of web content. The approach followed in this paper considers that a hierarchical structure is built on models that learn to distinguish a second-level class from other classes with the same top level. In the non-hierarchical approach, the model learns to distinguish a second-level class from all other second-level classes and usually, a certain number of binary classifiers are built (on one-to-one approach), which means that a certain input may be predicted into none, one or more than one

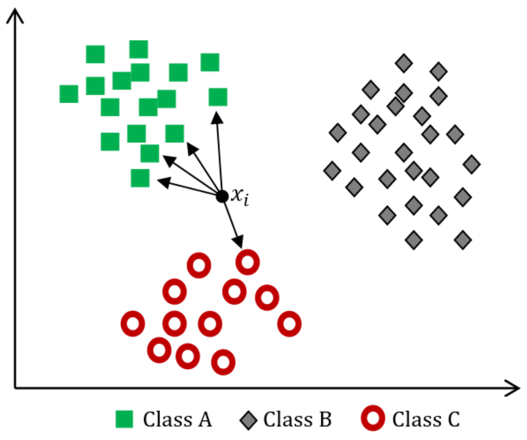


Figure 2.10: Example of k-Nearest Neighbors model for 3 classes.

class. Using Figure 2.11 as an example, this hierarchical approach considers that a model learns to distinguish classes C, D and E, inside the top-level class A, and a second model learns to distinguish class F and class H, which are both inside top-level class B. Other approaches used usually don't fully take advantage of the hierarchical structure and consider a single model to distinguish all classes in the second-level, meaning classes C, D, E, F and H are used in the same classifier and the classification in the top-level is not used.

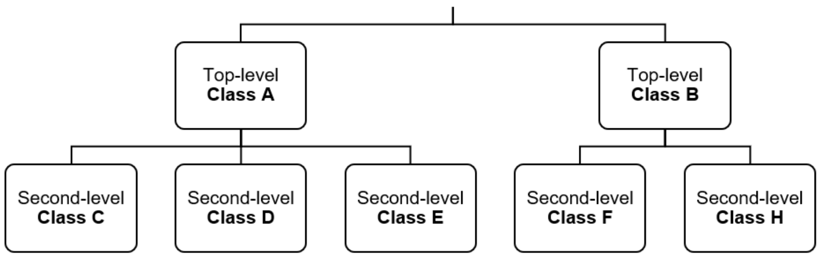


Figure 2.11: Example of the hierarchical structure used in [14].

In this article, it was concluded that the hierarchical models had some advantages over the non-hierarchical ones, in terms of accuracy and F_1 score, and provided large efficiency gains. However, it was also noted that the sequential decision model may create an issue of error cascading down the levels of the hierarchy, which can be solved through improved classifiers and appropriate decision thresholds to guarantee the best results. Another approach would be to use interactive interfaces where the user could help make critical decisions [14].

To better understand the flow of the classification, in a generic form, Figure 2.12 shows the nomenclature that is going to be used for the classifiers and the type of division done. Each classifier is identified as C_c^h, with two indexes: the superior one (h) indicates the level of the hierarchy where the classification is taking place, and the inferior one (c) refers to a specific classifier in a certain level.

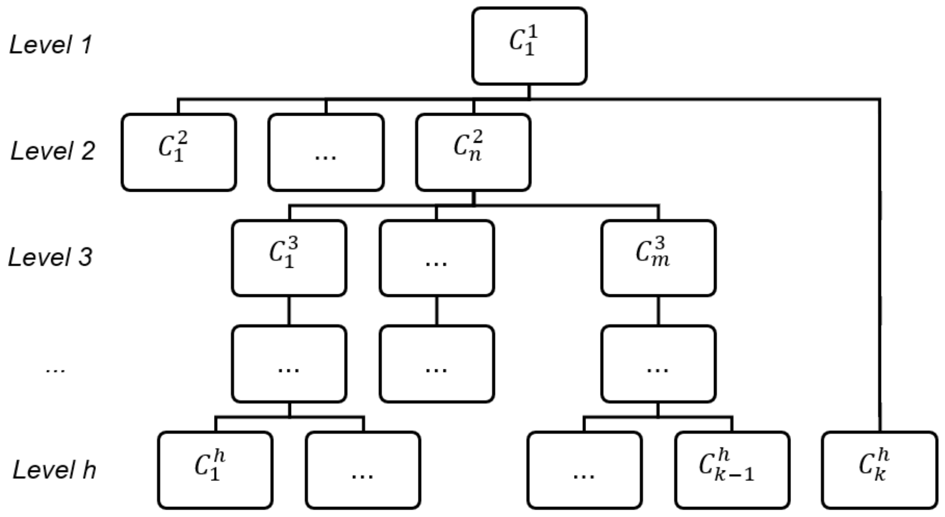
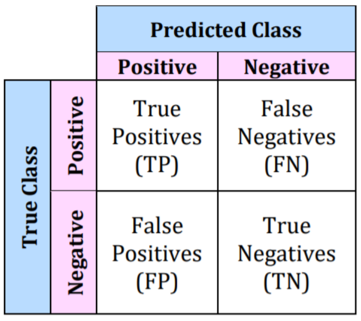


Figure 2.12: Generic classifier tree.

2.5 Evaluation Metrics

To evaluate the performance of classification models, a few different measures can be used. According to Kowsari et al. [7] , the most used metrics include recall, precision, accuracy, F-measure, micro-average and macro-average. All of these metrics are based on the Confusion Matrix, shown in Figure 2.13, which is composed of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN). The 2 types of errors that can happen are false negatives also known as Type I errors and false positives, aka Type II errors. Given that the problem is a multi-class classification, the metrics used are averaged ones. Besides these metrics, it was found useful to also measure the Matthews Correlation Coefficient, given that it's a balanced measure and considers all the entries of the confusion matrix.



		Predicted Class	
		Positive	Negative
True Class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

Figure 2.13: Confusion matrix for binary classification.

2.5.1 Accuracy, Precision and Recall

Accuracy is one of the most used formulas to evaluate both binary and multi-class models. The accuracy measure is simply the fraction of correct predictions in all the predictions made by the classifier, shown in (2.18) [15].

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{2.18}$$

Another important measure is **precision** and it's defined by the fraction of correctly predicted positives in all the positives predicted, shown in (2.19) [7]. Simply put, this metric measures the ability of the classifier not label a negative sample as positive, which means that false positives (samples labeled as positive when they're negative) should be minimized in order to increase this value.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.19}$$

Finally, **recall** is the fraction of known positives that are correctly predicted in all predictions and is also known as sensitivity, defined in (2.20) [7]. While the precision focus on the ability of the classifier to distinguish false positives, recall is the ability of the classifier to find all positive samples, meaning that false negatives (samples that are positive but labeled as negative) are the important value for improving this metric.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.20}$$

2.5.2 F-measure

Usually, recall and precision have a trade-off of performance, i.e. to improve recall, precision lowers and to have better precision, recall values drop. To better evaluate these metrics, F_β Score, shown in (2.21), is used in order to balance out recall and precision [7]. The β is the balance between these two metrics, meaning that for $\beta = 1$, recall and precision have equal weights for the computation. If $\beta = 1$, this metric can be called F_1 Score, defined as (2.22).

$$F_\beta = \frac{(1 + \beta^2)(\text{Precision} \times \text{Recall})}{\beta^2 \times \text{Precision} + \text{Recall}} \tag{2.21}$$

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.22}$$

2.5.3 Macro-Average, Micro-Average and Weighted-Average

For a multi-class classification problem, precision, recall and F_1 score need to be averaged given that there isn't any positive or negative label. In summary, three techniques of averaging can be used: macro-averaging, micro-averaging and weighted-averaging [7]. **Macro-averaging** gives an average over all classes, meaning that it calculates the metric independently for every class and then averages it, so that all classes have the equal contribution. The formulas for macro-averaged precision, recall and F_β score are shown in (2.23) for a multi-class problem with m classes [16].

$$\text{Macro-averaging} = \begin{cases} Precision_M = \frac{\sum_{j=1}^{m} \frac{TP_j}{TP_j+FP_j}}{m} \\ Recall_M = \frac{\sum_{j=1}^{m} \frac{TP_j}{TP_j+FN_j}}{m} \\ F_{\beta M} = \frac{(\beta^2+1)(Precision_M \times Recall_M)}{\beta^2 \times Precision_M + Recall_M} \end{cases} \tag{2.23}$$

A **micro-averaging** approach combines the contributions of all classes to compute the average metric required, meaning that it takes into account the possibility that the data is imbalanced and considers each class's frequency. The formulas for this type of averaging are shown in (2.24) [16].

$$\text{Micro-averaging} = \begin{cases} Precision_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m} TP_j+FP_j} \\ Recall_\mu = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m} TP_j+FN_j} \\ F_{\beta\mu} = \frac{(\beta^2+1)(Precision_\mu \times Recall_\mu)}{\beta^2 \times Precision_\mu + Recall_\mu} \end{cases} \tag{2.24}$$

Finally, the last method that can be used is **weighted-averaging** where the average of the binary metrics is calculated but each class's score is given a weight depending on the presence of the class in the data, meaning that it accounts for class imbalance. The weight for each class is calculated as $w_j = (TP_j + FN_j)$ which is the sum of the true samples of a certain class j. The formulas for this type of averaging are shown in (2.25).

$$\text{Weighted-averaging} = \begin{cases} Precision_W = \frac{1}{\sum_{j=1}^{m} TP_j+FN_j} \times \frac{\sum_{j=1}^{m} TP_j \times (TP_j+FN_j)}{\sum_{j=1}^{m} TP_j+FP_j} \\ Recall_W = \frac{1}{\sum_{j=1}^{m} TP_j+FN_j} \times \frac{\sum_{j=1}^{m} TP_j \times (TP_j+FN_j)}{\sum_{j=1}^{m} TP_j+FN_j} \\ F_{\beta W} = \frac{(\beta^2+1)(Precision_W \times Recall_W)}{\beta^2 \times Precision_W + Recall_W} \end{cases} \tag{2.25}$$

2.5.4 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) is a more reliable metric than the previous ones given that it evaluates all the information in a confusion matrix, in a single calculation. A high MCC score is only attributed if all 4 confusion matrix categories have good results, being therefore a good overall measure. This is also considered a balanced measure and can be used in problems with uneven class sizes [7]. Unlike the previous metrics that vary from 0 to 1, the MCC can have values from -1 to 1. If MCC is -1, then the predictions made by the classifier are completely wrong; if MCC is equal to 0, that means that the classifier assigns classes in a random matter; and if MCC is equal to 1, then the predictions are all correct and the classifier is able to distinguish the different classes. The Matthews Correlation Coefficient is defined by the formula present in (2.26), for binary classification.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP}) \cdot (\text{TP}+\text{FN}) \cdot (\text{TN}+\text{FP}) \cdot (\text{TN}+\text{FN})}} \tag{2.26}$$

For multi-classification problems, the MCC is defined by (2.27), according to the confusion matrix M

for m classes and the variables o_j, r_j, l and s defined as follows:

- $o_j = \sum_i^m M_{ij}$ represents the number of times that class j is the true class;
- $r_j = \sum_i^m M_{ji}$ is the number of predictions of class j;
- $l = \sum_j^m M_{jj}$ represents the total number of test samples predicted correctly;
- $s = \sum_a^m \sum_b^m M_{ab}$ is the total number of samples being tested.

$$\mathrm{MCC} = \frac{l \times s - \sum_j^m r_j \times o_j}{\sqrt{(s^2 - \sum_j^m r_j^2) \cdot (s^2 - \sum_j^m o_j^2)}} \tag{2.27}$$

Another change that occurs in multi-class MCC is that the minimum value is no longer -1. While in binary classification, the MCC score varies in the interval $[-1, +1]$, for multi-classification, the minimum value is between -1 and 0 depending on the number of labels. In both types of classification, the maximum value is +1.

Chapter 3

Proposed Framework

In this chapter, the hierarchical classification proposed is justified with a detailed explanation of the hierarchy inside the Operational Programmes, in particular, PDR 2020. The selection of the models and metrics, introduced in Chapter 2, is also justified based on the literature. Besides that, the data available is described and a brief introduction to the text processing techniques used is made.

3.1 Hierarchical Approach

As it was introduced in Section 1.2, there are a total of five areas in PDR 2020, and inside each of them, one can find a clear structure: each area is divided into a certain number of measures, which is further separated in actions (they can range from none to one or two, or even twelve). For each action, there is, usually, one to three operations, that consist of the last level of the hierarchy. In total, there are **four levels of hierarchy** to consider: **areas, measures, actions and operations**. With the goal of clarifying this structure, using Area 1 (Innovation and Knowledge) as an example, Figure 3.1 shows the sort of measures, actions and operations that exist within that area.

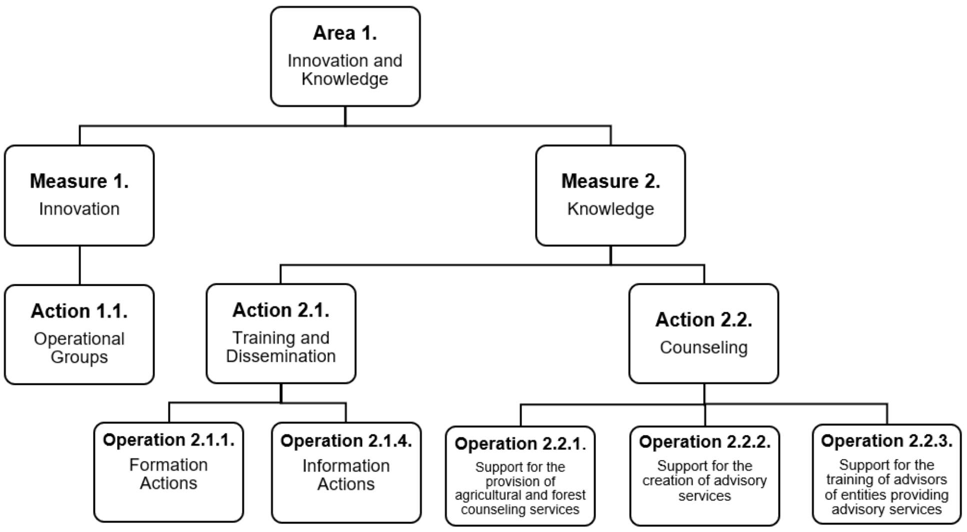


Figure 3.1: PDR 2020 - Area 1 (Innovation and Knowledge).

The hierarchical structure of PDR 2020 starts with **5 different areas**, which are then divided into **10 measures** (also referred to as priority areas in other OPs). From these measures, a total of **22 different actions** exist for classification and **46 operations**. The tables present in Appendix A, namely A.1, A.2, A.3 and A.4, have this structure detailed along with the number of projects inserted into each area, measure, action and operation. Besides those tables, the Figures A.1, A.2 and A.3 are provided with the detailed form of the hierarchy for this problem, with the classes that require classification and those that do not, alongside the classes of each classification.

Relating this hierarchy with the literature presented in section 2.4, the nomenclature shown in Figure 2.12 translates to the classifiers in Figures 3.2 and 3.3. In these figures, all the areas, measures and actions where a classification takes place are shown with the nomenclature presented before, C_i^j, and also the nomenclature from the PDR 2020. For example, classifier C_1^2 focus on the classification of measures inside area 1 and classifier C_2^3 classifies projects that belong to measure 3. To use the terminology of this specific problem, the classifier will usually be referred as "classification in a certain area, measure or action". In conclusion, the hierarchy defined has the following levels:

- The classifier in **level 1** classifies areas;
- The classifiers in **level 2** are inserted in a certain area and classify measures;
- The classifiers in **level 3** are inserted in a measure and classify actions;
- The classifiers in **level 4** are inserted in an action and classify operations.

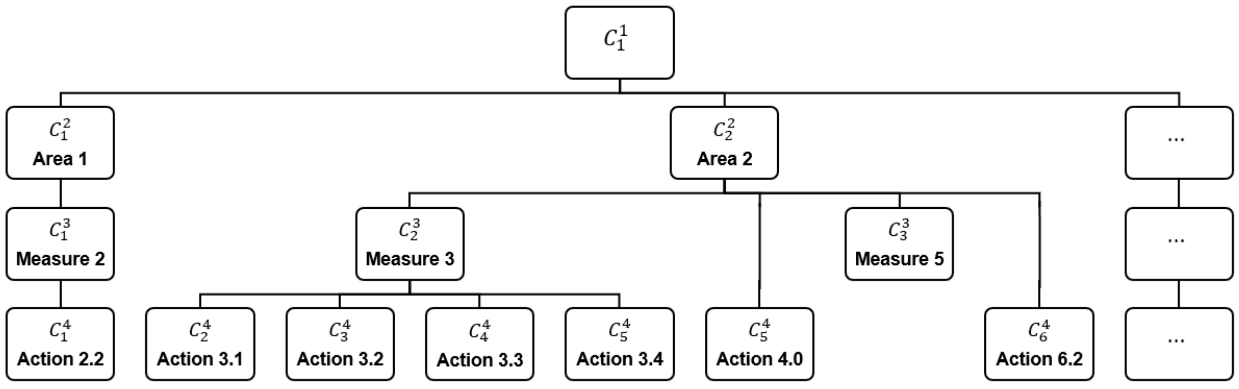


Figure 3.2: Classifier Tree used for PDR 2020 - Part 1.

Although there is a large number of classes in this dataset, by dividing it into these four levels, several classes don't require classification. For example, Area 4 and Area AT have only one measure so there's no need to classify into those areas. However, inside measures 10 (Area 4) and 20 (Area AT), there are different actions to be classified so a model is required for those two measures although the level above did not require it, as shown in Figure 3.3. Out of 10 existing measures, Measure 1, 4 and 6 have only one action and therefore don't need to be classified in terms of actions. While the actions inside measures 4 and 6 both need to be classified into 2 different operations, for measure 1, no further classification is necessary since it has only 1 action and 1 operation. In terms of actions, 11 out of 22

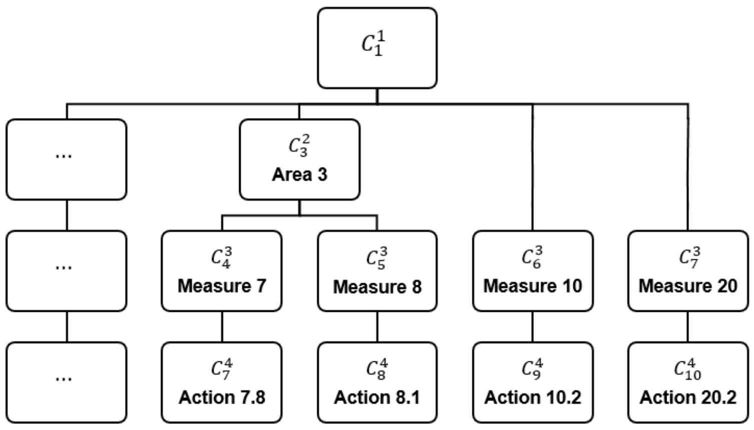


Figure 3.3: Classifier Tree used for PDR 2020 - Part 2.

were excluded from classification given that they have only 1 operation.

Concluding, **1** model is required for the first classification into different areas, then **3** models for areas 1, 2 and 3, after which **7** models are needed for the classification into actions. In the last level, for the classification into operations, only **11** models are necessary out of 22 actions. In total, **22 models** need to be trained and tested for a full classification.

3.2 Model Selection

To have the best performance possible for the funds assistant being developed, four classifiers, Naive Bayes, Support Vector Machine, Random Forest and k-Nearest Neighbors, were chosen to be tested and compared against each other, measuring accuracy, F_1 score and MCC. The Naive Bayes classifier is one of the classical algorithms used for classification, which has proven, over the years, to give reliable results without requiring much training data and computational power. The results from the Naive Bayes classifier are usually used as a baseline for many works, such as in [10] and [17].

Support Vector Machines are one of the most widely used classification algorithms nowadays, known for their high performance, especially in text classification. In [14], an hierarchical approach was developed to deal with the problem of classifying web content, similar to the approach followed in this thesis, and the classifiers used were SVM models, which had very good overall results. Giving the similarity between the classification in the article and the one that was constructed here, the choice of implementing SVM models is obvious. One of the disadvantages of the SVM model is related to output: the final prediction made by this classifier considers distances, not probabilities, like the NB classifier, making it difficult to evaluate other possible solutions besides the predicted one. Some approaches have being studied to overcome this issue, as seen in [13]. These approaches were also tested and implemented in this work. Another downfall of SVM is that the classifier is originally meant only for binary classification, not multi-class problems. Strategies have been developed to solve this type of situations, however, using such methods could result in a decrease in performance, making it important to consider models that support multi-class classification problems, such as Random Forest and k-Nearest Neighbors.

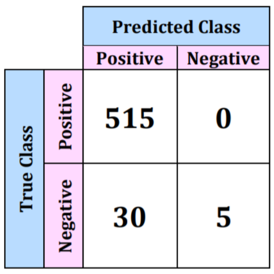
Just like the Naive Bayes model, another basic classifier mentioned frequently in the literature is the Decision Tree (DT) model. However, the Decision Tree algorithm usually shows a worse performance, as it can be seen in [10], which is the reason why it wasn't tested here. Despite the poor performance of DT algorithms for this type of problem, the Random Forest classifier, an ensemble model that uses decision trees as base classifiers, can overcome the limitations of the DT classifier and produce accurate results. As stated previously, this algorithm has the advantage of being suited for multi-class problems.

Last, the k-Nearest Neighbor classifier is a non-parametric classification method, first introduced in 1951, and still commonly used, given the improvements that continue to be made in recent studies. The KNN classifier is composed of a known optimization problem, the nearest neighbor search. This optimization problem has other variants, such as the approximate nearest neighbor, which is used in a novel model, ANNOY [18], developed by Spotify, for music recommendations. One of the reasons for testing the KNN algorithm is the possibility of using text similarity as a solution for this problem and implementing the ANNOY algorithm to find similar texts. If this algorithm has a good performance, it could mean that, as a future approach, a similarity algorithm, such as ANNOY, would bring good results.

3.3 Metric Selection

As stated before, the models chosen are going to be compared with 3 measures: accuracy, F_1 score and Mathews Correlation Coefficient. Accuracy is one of the most used measures of performance and simply expresses the number of correct predictions in all predictions made. F_1 measure is a combination of recall and precision (see section 2.5.1 for a detailed explanation) and is once again a widely used metric to measure the success of a classification problem. However, precision, recall and F_1 score are binary metrics and require an averaging for multi-class problems. A macro-averaging approach was dismissed as it considers that all classes have an equal contribution, which is not true for the problem in this thesis. So, the choice became between implementing a micro-averaging or weighted-averaging approach, where class imbalances are taken into account. It was decided to choose a weighted approach that calculates a weight for each class depending on its presence in the dataset.

The final metric that was chosen to analyze is Mathews Correlation Coefficient. Although not as popular as accuracy and F_1 measure, the MCC has the advantage of considering all the entries of a confusion matrix and also of dealing with uneven class sizes. An example of the benefit of the MCC score can be seen in Figure 3.4, where the confusion of binary classification is shown. In this classification, 515 samples were correctly classified as positive and 5 correctly as negative. However, 30 samples, that should be classified as negative, were misclassified as positive.



		Predicted Class	
		Positive	Negative
True Class	Positive	515	0
	Negative	30	5

Metrics	
Accuracy	0.945
F1 Score	0.926
MCC	0.367

Figure 3.4: Example Confusion Matrix and respective metrics.

The accuracy of 0.945 and F_1 score of 0.926 are both quite high, leading to the conclusion of a successful classification. Despite that, the MCC score shows a much lower value of 0.367, which urges further analysis. By observing the confusion matrix, it can be verified that this classification has an uneven dataset, and it's possible to infer that the classifier is likely unable to correctly classify a class as negative, meaning that the overall classification of these two classes is not reliable. If one was to simply look at the accuracy and F_1 measure, this conclusion could not be made nor understood. By analysing the MCC score, details like this can be caught and properly evaluated.

3.4 Implementation of 2nd Predictions

To improve the performance of the classification, it was decided to explore how to add decision thresholds to the **SVM models only**. A decision threshold is a probability below which it's possible to infer that the classifier loses confidence in its prediction and is, therefore, prompted to consider a second one, as shown in Figure 3.5. For the classifiers where a threshold wasn't implemented, the output is the class with the highest probability. For the classifiers with a decision threshold defined, the second prediction is considered only if the probability of the first prediction is lower than the threshold. When the model is being evaluated, it's possible to analyse if the second prediction is correct or not (Figure 3.5). If the second prediction is correct, then the output of the classifier will be the second prediction; if not, the output will be the first prediction.

Now, to add decision thresholds to SVM models, it is necessary to analyse the confidence that the classifiers have in their outputs. Given that SVM models have a decision function that works with the signed distances to the hyperplane, it becomes quite complicated to interpret those results. Following the strategies implemented in [13], it was chosen to build calibrated SVM models. Calibrated models are used to obtain a probability of a prediction, when such is not possible given the model used (as is the case of Support Vector Machines) or when a model gives poor estimates of the class probability. This solution consists of fitting a regressor that maps the output of the classification to a calibrated probability in a range of 0 to 1. As explained in Section 2.3.2, two regressors were considered for building these calibrated models.

After setting up calibrated SVM models, it's now possible to analyse the output and the probabilities

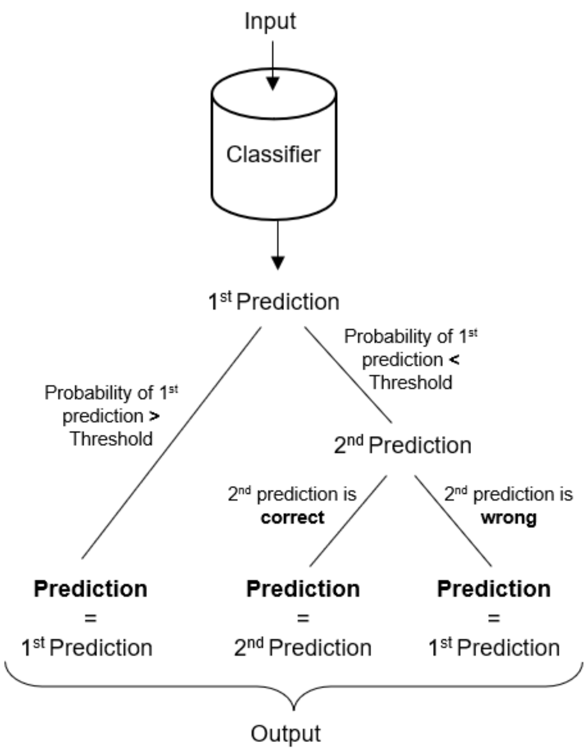

Input
Classifier
1st Prediction
Probability of 1st prediction > Threshold
Probability of 1st prediction < Threshold
2nd Prediction
2nd prediction is correct
2nd prediction is wrong
Prediction = 1st Prediction
Prediction = 2nd Prediction
Prediction = 1st Prediction
Output

Figure 3.5: Evaluation pipeline for classifiers with 2nd prediction.

associated with it. All models will certainly improve by considering second predictions, however, in practicality, it's not acceptable for a person to analyse every option in the predictions. The goal here is to improve the performance enough without overloading the user with different options. To decide the best threshold of probabilities for each model, four scores were retrieved: accuracy, MCC, the percentage of second predictions considered out of all predictions made, R, and the percentage of correct second predictions out of all the second predictions made, T.

By analysing the percentage of all second predictions considered, it's possible to understand the confidence that a model has in its predictions and evaluate how many times the user would be asked to choose between two options, in a real-life situation. On the other hand, the percentage of correct second predictions is useful to find out which threshold should be implemented. It's also important to note that this percentage will likely start decreasing at a certain point after increasing because the model starts to analyse more wrong second predictions, encountering situations where the first prediction is correct. The best threshold to choose is one that maximizes this percentage and minimizes the percentage of second predictions considered in total. To choose the best threshold possible considering all these factors, a heuristics technique was used considering an objective function, shown in (3.1). The best threshold t for a certain model is chosen by trying to find the highest value possible of $F(t)$. Besides that, to make it easier to analyse when the model is considering or not second predictions, $F(t)$ is forced to be zero if no second predictions are made ($R = 0$).

$$F(t) = \omega_\tau \cdot \tau(t) - \omega_\rho \cdot \rho(t) + \omega_\phi \cdot \phi(t) \tag{3.1}$$

Regarding the variables shown in (3.1), these are defined as follows:

- $\tau(t)$ corresponds to the percentage of correct second predictions for a threshold t, T_t, divided by the sum of all percentages of correct second predictions, as seen in (3.2). This variable has a weight of ω_τ to be defined.

$$\tau(t) = \frac{T_t}{\sum_i T_i} \tag{3.2}$$

- $\rho(t)$ corresponds to the percentage of second predictions made for a threshold t, R_t, divided by the sum of all percentages of second predictions made, as seen in (3.3). This variable has a weight of ω_ρ to be defined.

$$\rho(t) = \frac{R_t}{\sum_i R_i} \tag{3.3}$$

- $\phi(t)$ corresponds to the sum of two parcels: accuracy for a certain threshold t, divided by the sum of all accuracies, and MCC for a threshold t, divided by the sum of all MCC scores, as seen in (3.4). This variable has a weight of ω_ϕ to be defined.

$$(t) = \frac{\text{Accuracy}_t}{\sum_i \text{Accuracy}_i} + \frac{\text{MCC}_t}{\sum_i \text{MCC}_i} \tag{3.4}$$

3.5 Text Processing

For text-related tasks, one of the most important steps is the processing of text, that is, creating numeric representations of the text with the essential features.

3.5.1 Text Pre-processing

As it was mentioned in Chapter 2, the first step of any text related algorithm is feature extraction (Section 2.2), more specifically, text cleaning and pre-processing. To better understand the process, a project will be used as an example with the following information available (title and description):

> ”Cunicultura e Fruticultura Duplicação da capacidade da exploração cunicula. Inicio da atividade no setor dos frutos de casca rija.”

Removal of punctuation and numbers and enforce lowercase

First, all the numbers contained in the text were removed, with special attention in order not to eliminate any words attached to said numbers (for example, if the text contains ”15kg”, only ”kg” will remain). Besides that, all punctuation is removed and all letters are lowered using the Python resource `re` (Regular expression operations) and `string` (Common string operations), resulting in the following text:

1. ”cunicultura e fruticultura duplicação da capacidade da exploração cunicula inicio da atividade no setor dos frutos de casca rija”

Tokenization

Then tokens are created by splitting the text into different words, using the `re` package, that finds any word and not word character and removes the spaces between the words, resolving in a list containing all existent tokens:

2. ’cunicultura’, ’e’, ’fruticultura’, ’duplicação’, ’da’, ’capacidade’, ’da’, ’exploração’, ’cunicula’, ’inicio’, ’da’, ’atividade’, ’no’, ’setor’, ’dos’, ’frutos’, ’de’, ’casca’, ’rija’

Removal of Stopwords

Stopwords are words that bring no important information to a text such as propositions and determinants. The library `nltk` was used given that it provides a set of stopwords for several languages, Portuguese being one among them. In the example, the words ’e’, ’da’, ’no’, ’dos’ and ’de’ were all removed and the following tokens were maintained:

3. ’cunicultura’, ’fruticultura’, ’duplicação’, ’capacidade’, ’exploração’, ’cunicula’, ’inicio’, ’atividade’, ’setor’, ’frutos’, ’casca’, ’rija’

Stemming Algorithms

After finding all the tokens in a text, each word is evaluated with a stemming algorithm and replaced with its stem form. For the stemming algorithm, two options were considered: Snowball stemming algorithm and RSLPS stemmer, introduced in 2.2.1. Using the RSLPS Stemmer, the following tokens were returned:

4.1. ’cunicult’, ’fruticult’, ’duplic’, ’capac’, ’explor’, ’cunicul’, ’inici’, ’ativ’, ’set’, ’frut’, ’casc’, ’rij’

And using the Snowball Portuguese Stemmer, the following stems were found:

4.2. ’cunicultur’, ’fruticultur’, ’duplic’, ’capac’, ’explor’, ’cunicul’, ’inici’, ’ativ’, ’setor’, ’frut’, ’casc’, ’rij’

Comparing the two stemmers, it’s possible to verify that they are quite similar and almost all the words will have a similar stem (for example, ’atividade’ passes to ’ativ’ in both cases) but some have different stems such as ’fruticultura’ that transforms to ’fruticult’ with the RSLPS Stemmer and to ’fruticultur’ with the Snowball Stemmer (same happens with ’cunicultura’, that stems to ’cunicult’ and ’cunicultur’, respectively).

To understand which of the two has better results with this type of data, a small test was made with a Support Vector Machine algorithm and the average results of the accuracy, F_1 Score and Matthews Correlation Coefficient, in 5 iterations. These results are shown in Table 3.1. By comparing the average results for each metric, it was found that the **Snowball Stemmer** had better overall results and took less time to process. In regards to the results shown, the classifier analysed is the area classifier, C_1^1.

Table 3.1: Comparison between Snowball Stemmer and RSLP Stemmer.

	Snowball Stemmer	RSLP Stemmer
Accuracy	0.877	0.877
F_1 Score	**0.8704**	0.8695
MCC	**0.769**	0.7655
Time/it (s)	**48**	66

Removal of Accentuation

Given that the Portuguese language has accents, also known as diacritics, such as ´, ‘, ˆ , ˜ and ç, it important to understand its influence in the results to decide whether they should be keep or normalized (removed). A solution to remove accents is to use the `unicodedata` library, that provides access to the Unicode Character Database (UCD) and makes it possible to normalize a string based on canonical and compatibility equivalence. Here, it was used a normal form that first applies a canonical decomposition and then composes the characters again. With the previous example, the text considered is the following:

> ”Cunicultura e Fruticultura **Duplicacao** da capacidade da **exploracao** cunicula. Inicio da atividade no setor dos frutos de casca rija.”

In this example, the words ”duplicação” and ”exploração” are altered. It’s also interesting to note that the words ”inicio” and ”cunicula” actually both have accents (”início” and ”cunícula”) but were written without it. By removing accentuation from all the text, it is possible that the algorithm could learn better representations, discarding spelling errors, such as ”inicio” or ”cunicula”.

A small experiment of 5 iterations was done with an SVM algorithm (using the Snowball Stemmer) and evaluated with accuracy, F_1 metric and MCC score. In Table 3.2, it can be seen that, by **normalizing the text**, the accuracy and the MCC score increase a little bit. Normalizing also proved to be more computationally inexpensive than considering texts with accentuation.

Table 3.2: Comparison of text processing with and without accentuation.

	With Accentuation	Without Accentuation
Accuracy	0.877	**0.8772**
F_1 Score	**0.8704**	0.8692
MCC	0.769	**0.772**
Time/it (s)	48	**44**

3.5.2 TF-IDF Vectorization

As for the feature extraction technique, the TF-IDF weighting scheme was chosen based on the paper [19]. This article brings the problem of document clustering, similar to the topic of this thesis. While in [19], the algorithms used are clustering ones, to deal with the unlabeled data, for this thesis, the algorithms explored were of text classification, in order to take advantage of the labelled data available. However, both consider hierarchical approaches: in [19], it was used hierarchical agglomerative clustering (a bottom-up approach), and here, a hierarchical structure was built with a certain number of levels and classifiers inserted into each level. These strategies exploit the fact that the texts can be grouped in larger aggregations of related content and then separated further to specify the subject or vice-versa, as it happens in agglomerative clustering. Given that the vectorization used in [19] was TF-IDF and considering the similarities between the two problems, along with the fact that the results achieved show the efficiency of this method, it was decided to test this representation in this problem, hoping to also obtain high performance.

As it was introduced in Section 2.2.2, TF-IDF (Term Frequency - Inverse Document Frequency) is a technique for the representation of words where each word is weighed in regard to its importance to a document (in this case, the text with the title and description) contained in a corpus (the texts available from all the projects). For the implementation of this algorithm, the tools from `sklearn` library were used, in particular `TfidfVectorizer`, that utilizes the previous pre-processing (defined in Section 3.5.1) as an analyzer to clean and stem the text as required. After the stemming and vectorization of words, the model creates a matrix with f columns, f being the number of words considered in all the corpus available, and n rows that correspond to each project considered. Using all the data available from PDR 2020, with a total of 28914 projects, the features extracted consist of a matrix of 28914 rows by 32454 columns, meaning that for the 28914 projects, 32454 words were identified and processed. Each entry in the matrix corresponds to the weight of the word j (column) in the text i (row).

3.6 Data Analysis

In regards to the data used, all public information was retrieved to have the most complete database possible. From the website of Portugal 2020 [20], it was possible to access a file with several accepted projects from all OPs. However, it was noted that the projects of PDR 2020 were not available there and had to be retrieved from PDR 2020 website [21]. Overall, it was possible to gather information regarding 159 363 projects from 15 different Operational Programmes. In Table 3.3, it's specified how many projects are from each OP and also the number of projects that do not have an abstract or description available.

The public data available has initially 8 elements of information. Besides this information, it was also found to be relevant to add more information about the hierarchy of the OP of the project (area, measure, action). In Table 3.4, it's possible to observe an example of the described information.

In summary, the 10 elements of the data are as follows:

Table 3.3: Public data available in [20] and [21].

	Accepted Projects	
OPs	Total	Without description
MAR 2020	7 196	43
PDR 2020	**29 005**	**91**
POAPMC	143	0
POAT	28	0
POCI	70 463	66 068
POISE	4 748	0
POSEUR	1 958	0
POCH	6 025	18
Regional OPs	39 797	6 373
Total	**159 363**	**72 593**

Table 3.4: An example of the public information available.

	Project Nº1
OP	PDR2020
Operation	3.2.1 - Investimento na exploração agrícola
Project Code	PDR2020-3.2.1-FEADER-000273
Title	Aquisição de equipamento agrícola
Description	Pretende-se adquirir dois tractores agrícolas de 90 cv e uma desladroadora quimica de 2 tuneis.
NUTII	Alentejo
County	Cuba
Support Tax	0.85

- **Operational Programme** - stipulates the OP that each project is inserted into;
- **Operation** - specifies the last branch of hierarchy that each project was accepted into;
- **Project Code** - is a code that specifies the OP, the Operation and European Fund that the project is engulfed in and also a project code with meaning for the OP;
- **Title** - gives the title of each project;
- **Description** - has an abstract made by the beneficiary of the specifications of the project;
- **NUTII** - is a hierarchical system of dividing the territory into regions. NUTS is an acronomy of ”Nomenclatura das Unidades Territoriais para Fins Estatísticos” and have 3 different levels. In this column, it's specified the second level, NUTS II, which divides the country into 7 different regions (North, Center, Metropolitan Area of Lisbon, Alentejo, Algarve, Autonomous Region of Açores and Autonomous Region of Madeira).
- **County** - specifies the county where each project will be implemented;
- **Support Tax** - indicates the level of support given for the project (between 0 and 1, where 0 means no support and 1 means all approved expenses are covered).

3.6.1 Hierarchical Information

It was explained previously the hierarchical structure used for the funds assistant (3.1). In short, using the hierarchy inside PDR 2020, four levels of classes were considered: areas, measures, actions and operations. As seen in Table 3.4, the information available shows only the operation of a project: ”3.2.1 - Investimento na exploração agrícola”. Nonetheless, the operation is identified with a code, ”3.2.1”, which showcases three levels of the hierarchy:

- The first number (3) is related to the measure;
- The first and second number together (3.2) identifies an action;
- All the numbers together (3.2.1) symbolize a specific operation.

The first level of the hierarchy is not identifiable with this code, but that information is available in PDR 2020’s website [6], making it possible to manually associate the measures with the five existent areas. Similar tables, to the ones shown in Appendix A, were created with labels for every area, measure, action and operation, and also information about the number of classes of the following level (important for modelling the hierarchical classification) and the associated label of the previous level. Some of this information is only relevant on some levels. For example, the previous label doesn’t matter for the first level and the nº of classes isn’t important in the last level. Taking area 1 as an example, Table 3.5 shows the organization defined for modelling.

Table 3.5: Example about the hierarchical information defined (Area 1).

	Label	**Name**	**Nº Classes**	**Previous Label**
Level 1	0	1 - Innovation and Knowledge	2	×
Level 2	0	1 - Innovation	1	0
	1	2 - Knowledge	2	0
Level 3	0	1.0 - Operational Groups	1	0
	1	2.1 - Training and Dissemination	1	1
	2	2.2 - Counseling	3	1
Level 4	0	1.0.1 - Operational Groups	×	0
	1	2.1.4 - Information Actions	×	1
	2	2.2.1 - Support for the Provision of Agricultural ...	×	2
	...	...	...	...

3.6.2 Text Information

In a brief analysis of the text information available (title and description), it was noted that some projects had descriptions in English, making it important to verify that the text that is going to be processed is all in the same language (Portuguese). Besides finding projects in English, it was detected some information that had no significance such as titles and descriptions like: ”aaa aaa”, ”svd v¡ds” and ”cccccc çççççççççççççç”. It was tested and discovered that such texts would usually not be detected as Portuguese, nor any similar language (Spanish, Italian, etc), so it could be solved easily by restringing the language of the project.

To verify if the information in the text was valid, the library `langdetect` was used. It was verified that Portuguese could be wrongly classified as Spanish, Catalan (Valencian), Italian and Romanian. Given that most projects that are not detailed in Portuguese, are done in English, it was decided to add all the previously mentioned languages as possible correct classifications. For all the data available, 243 projects were detected in a different language and removed. As for PDR 2020, 10 projects were identified and removed (these projects belonged to the second case mentioned - text with no significance).

Chapter 4

Classification Results

In this chapter, the results of the four chosen models (Naive Bayes, Support Vector Machine, Random Forest and k-Nearest Neighbor), using TF-IDF weighted vectors and a hierarchical approach, were evaluated with the following metrics: accuracy, F_1 Score and MCC. An analysis of the parameters for each model is also conducted to find the best performance possible, optimizing the accuracy. Two analyses of the results were made: one where each model was analyzed separably, considering that solutions from previous levels were correct, and one considering the predictions of the previous levels to choose a classifier and retrieve the following predictions.

The data used, for all the analysis made in this thesis, was separated 80%-15%-5%. 80% of the dataset is meant for training and 15% is for the validation of each separate model, done in section 4.3. The last 5% is used for the sequential testing phase (4.2) that require a dataset that wasn't used for training or testing in any of the previous models. As for the training and validation dataset, the division done considers that an input used for training a certain model may belong to the validation data of a different model. It was preferred to ensure that the training and validation data were divided equally, at the expense of not guaranteeing that training and validation data was the same throughout all the models. However, the 5% testing set is independent of all models and can be used to test the sequential modelling without having to worry about biased results.

4.1 Parameter Tuning

As it was introduced in Section 2.3, Naive Bayes, Random Forest, Support Vector Machine and k-Nearest Neighbor classifiers were chosen as possible solutions. To have the best results possible from these models, several parameters from each model were optimized with regard to the accuracy of the classifier. Given that the number of features used in the first level of classification is quite computationally expensive, it was found best to use a sample of the training data for the area classification with a total of 2000 projects.

Multinomial Naive Bayes

For the implementation of this model on Python, the library `sklearn` was used, in specific, the model `MultinomialNB` to build a multinomial Naive Bayes classifier. The Naive Bayes model has only one parameter to be optimized, α, that regulates the Laplace smoothing added to the model. Laplace smoothing is a technique that addresses the problem of zero probability in Naive Bayes, as explained in Section 2.3.1. If $\alpha = 0$, then there is no smoothing at all, which means that words that are not in the training set will have a probability equal to zero. The default value of this parameter is 1, to prevent numerical issues. Using `GridSearchCV` to find the best α, it was verified that the model had better accuracy results when α was close to zero, with a maximum 0.858 of accuracy when $\alpha = 0$ and a minimum accuracy of 0.783 with a larger α. To prevent numeric issues and prediction errors, $\boldsymbol{\alpha = 0.2}$ was chosen.

Linear SVM

For the Support Vector Machine classifier, it was used a `LinearSVC` model, which implements a linear kernel and uses the one-to-rest approach, explained in 2.3.2, to tackle the multi-class problem. Besides those specifications, the models have the following parameters to be defined:

- `C` affects if the problem is soft-margin or hard-margin. A small value is associated with soft-margin SVM and a larger value with hard-margin.
- `penalty` defines the type of penalty for violating the support vector constraint. There are two options: L1 and L2. If L1 penalization is used then the solution becomes sparse, given that only a subset of features has weights different than zero and actively contribute to the solution. This type of penalization is the one shown in the decision function in (2.9). As for L2 penalization, the objective function is altered to (4.1) to make the SVM algorithm less susceptible to outliers and improve generalization. In general, L2 penalization is preferred, being the default value of this model.
$$\frac{||W||^2}{2} + \frac{C}{2} \cdot \sum_{i=1}^{n} \xi_i^2 \tag{4.1}$$
- `loss` allows the user to define the loss function between a hinge loss function and a squared-hinge loss function. A hinge loss function is a type of loss function used in the training of a classifier, for tracking the number of mistakes made. The hinge loss of an output t (which can be 1 for a positive class and -1 for a negative class), with a classifier score y, is defined as $l(y) = \max(0, 1 - t \times y)$. The other variant of the loss function, squared hinge loss, is a ”smoothed” version and considers the following loss function: $l(y) = \max(0, 1 - t \times y)^2$. Using the squared hinge loss function means that bigger mistakes are punished more significantly and smaller ones more lightly.

Optimizing these parameters, using the `GridSearchCV` from `sklearn`, led to the results shown in Table 4.1. It was found that a hard-margin classifier had better results, and the parameter C was set to 2. The model showed the best results with L2 penalization and a squared hinge loss function. The worst results from this optimization resulted in a 0.9385 accuracy, which was improved to 0.966 with the optimal parameters.

Table 4.1: SVM Optimal Parameters.

C	Penalty	Loss
2	L2	squared hinge

Random Forest Classifier

The model used for the Random Forest Ensemble Classifier, `RandomForestClassifier`, contains several parameters that will prevent overfitting specific for the base classifiers (decision trees) and also some parameters related to the random forest itself. The optimized parameters are defined as follows:

- `criterion` can be used to specify between the gini-index and entropy split criterion introduced in Section 2.3.3.
- `max_depth` allows for the user to define the maximum depth of the tree. If there is no maximum defined, the tree will grow until all the leaf nodes are pure. This is a simple parameter to prune the decision tree but probably not enough for more complex problems and could be quite difficult to find an optimal value without compromising the accuracy of the results.
- `max_leaf_nodes` can also be defined to control the tree growth given that it will limit the number of nodes at the end of the decision tree. The tree will stop growing as soon as it reaches the defined number of leaf nodes.
- `min_impurity_decrease` sets a threshold on the impurity decrease to split the node. This impurity decrease is related to the values given from the gini-index or entropy, depending on the one chosen. Unlike `max_depth`, it is a more sophisticated parameter given that it takes into account the quality of a certain partition being considered. All partitions being made will need to have a decrease of more than `min_impurity_decrease` on the impurity value.
- `min_samples_leaf` defines the minimum number of samples necessary to split a internal node. If this parameter is very small, the tree will be grown fully and for larger values, some nodes will not be divided.
- `n_estimators` is used to restrict the number of base classifiers (decision trees) being built in total. The pre-defined number is 100.

Using the `GridSearchCV` function for the classification of areas, the optimal parameters found are presented in Table 4.2. With the optimal parameters, the model had the best accuracy of 0.919, a big improvement from a 0.783 accuracy that resulted from different settings.

Table 4.2: RF Optimal Parameters.

Criterion	Max Depth	Max LN	Min ID	Min SL	Nº estimators
gini	150	500	0	1	75

K-Nearest Neighbors Classifier

For the K-Nearest Neighbor algorithm, the model `KNeighborsClassifier` was used. This classifier has the following parameters to be optimized:

- `algorithm` specifies the neighbors search algorithm being used. As explained in 2.3.4, there are 3 algorithms that can be used: Brute Force, Ball Tree and K-D Tree. The pre-defined settings for this parameter will find the most convenient algorithm for the data in question.
- `leaf_size` is a setting for the search algorithm. The default value is 30 and increasing this value can lead to closer neighbors but not exactly better accuracy results. This is a parameter similar to the ones from the Random Forest classifier, given that is related to information loss and pruning for the tree.
- `metric` specifies the distance measure used. The distance can be Euclidean, Manhattan or Minkowski (with the parameter p to be defined), explained in section 2.3.4. The default metric is Euclidean.
- `n_neighbors` defines the number of neighbors to use in the nearest neighbors search results.

By optimizing these parameters, it was found that the Ball Tree algorithm had always the best results, alongside 2 nearest neighbors and Euclidean measure as the distance metric. These results can be seen in Table 4.3. With this optimization, it was possible to improve the accuracy from 0.79 to 0.944.

Table 4.3: KNN Optimal Parameters.

Algorithm	Leaf size	Metric	Nº neighbors
Ball Tree	20	Euclidean	2

4.2 Analysis of Hierarchical Classification

A sequential process of classification was tested, using previous predictions to choose which model to use and make the next prediction. The data used to test in this phase was the 5% testing set, not used previously for training nor testing (of each model), meaning that the results from this classification were not biased in any way.

The performance of each level, in terms of the four models considered, is shown in Table 4.4. The first level shows high accuracies, with the maximum of 0.992 and an MCC score of 0.980, obtained with an SVM algorithm. However, these results have an alteration made to the data to realistically classify the projects. This alteration can be seen, in more detail, in section 4.3.1. In short, 2 different classes had projects with the same objective, only with a restriction in location, so it was decided to change the label of these similar projects, to reflect this situation. The classes in question were 3.2.2 - Small Investments in Farming, which belongs to area 2, and 10.2.1.1 - Small Investments in Farming, which belongs to area 4, and also 3.3.2 and 10.2.1.2 - Small Investments in Transformation and Marketing. The projects of 10.2.1.1 and 10.2.1.2 were identified as area 2 for the first level, and then restored their original labels (area 4) for the rest of the classification.

Table 4.4: Comparison of classification models using sequential results.

		SVM	RF	KNN	NB
Level 1	Accuracy	0.992	0.966	0.979	0.974
	F_1 Score	0.992	0.964	0.979	0.973
	MCC	0.980	0.909	0.945	0.930
Level 2	Accuracy	0.988	0.954	0.970	0.959
	F_1 Score	0.988	0.953	0.969	0.958
	MCC	0.982	0.931	0.945	0.938
Level 3	Accuracy	0.803	0.758	0.768	0.798
	F_1 Score	0.805	0.748	0.777	0.794
	MCC	0.756	0.697	0.729	0.746
Level 4	Accuracy	0.717	0.666	0.678	0.689
	F_1 Score	0.716	0.660	0.677	0.678
	MCC	0.682	0.624	0.648	0.649

In Table 4.4, it can be seen that the SVM classifier outperforms the rest of the classifiers from the first level to the last one, starting with a 0.992 accuracy and ending with a 0.717 accuracy. The Random Forest Ensemble classifier showed the worst results in all the classifiers having differences of 0.03 to 0.05 accuracy in comparison to the SVM classifier. While, in the first two levels, the k-Nearest Neighbors showed better accuracy and MCC than the Naive Bayes classifier, in the last two levels, the NB models were able to overpass the KNN, having the second-best overall performance with a 0.689 accuracy and 0.649 MCC score.

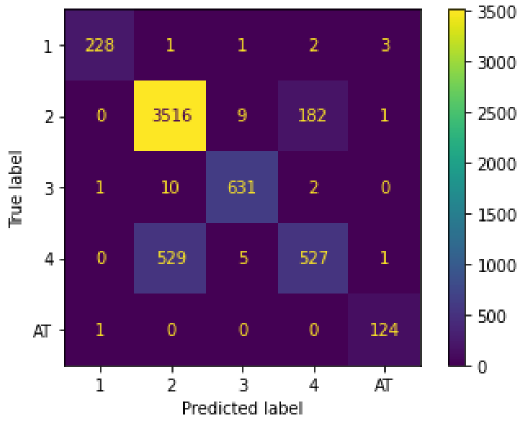
4.3 Analysis of Individual Classifiers

In this section, each classifier is analysed separately, to understand which models are able to learn the features necessary to properly classify projects and which are not. All types of models, NB, RF, KNN and SVM, were built and tested to find the differences between them in each point of the classification. In this analysis, the test dataset used was the 15% reserved for the validation of each individual model.

4.3.1 Level 1: Area Classification

Starting with the first level of the hierarchical classification, there are 5 areas to be classified (Table A.1). The first results are presented in Figure 4.1. The best results were provided with a SVM classifier, which gave an accuracy of 0.870. The KNN model also showed good results with an accuracy of 0.864 and an MCC score of 0.742, a little lower than the SVM score of 0.754. By analysing the confusion matrix of this classification, it's possible to observe that most wrong predictions are related to area 2 and area 4. The reason behind these mistakes relates to the Area 4 projects' scope: Area 4 has projects under

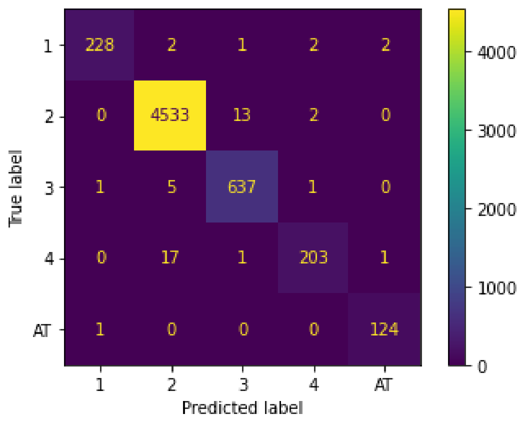
”Local Development”, which are projects from other areas but with a specified location. For example, Operation 3.2.2 - Small Investments in Farming (that belong to area 2) is the same scope as 10.2.1.1 - Small Investments in Farming (that is from area 4) and the operations 3.3.2 - Small Investments in Transformation and Marketing and 10.2.1.2 are also the same. The only difference between these two is the requirement of locations from a selected list, for projects in Measure 10 (Area 4).



	SVM	RF	KNN	NB
Accuracy	**0.870**	0.836	0.864	0.848
Recall	**0.870**	0.836	0.864	0.848
F_1 Score	**0.861**	0.797	0.849	0.825
MCC	**0.754**	0.698	0.742	0.711

Figure 4.1: Confusion matrix and weighted metrics for Area classifier (1st results).

Given the specification of location is not something that is likely to be caught by the algorithm, it could be solved with a rule-based algorithm to ask the beneficiary to specify the location of the project and therefore, understand if it qualifies as area 2 or 4. For now, the operations mentioned before, 10.2.1.1 and 10.2.1.2, had their labels changed in order to be correctly classified as Area 2 instead of Area 4. It was assumed that the descriptions of those operations would be correct if they were classified as Area 2 or Area 4. With these changes, the results of the new model are shown in Figure 4.2, where it’s possible to verify a big improvement in terms of accuracy, from 0.870 to 0.992. Again, the SVM model shows the best results overall, with a larger difference with other algorithms. The confusion matrix reveals that the projects of areas 2 and 4 are no longer misclassified and most errors are related to Area 2, which contains the most projects.



	SVM	RF	KNN	NB
Accuracy	**0.992**	0.971	0.979	0.979
F_1 Score	**0.991**	0.970	0.979	0.978
MCC	**0.977**	0.919	0.942	0.940

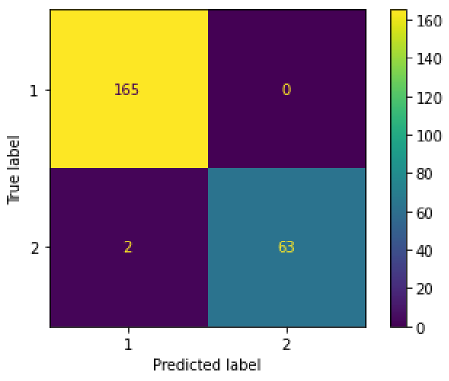
Figure 4.2: Confusion matrix and weighted metrics for Area classifier (after replacing labels for similar operations).

4.3.2 Level 2: Measure Classification

The next level in classification is Measure Classification. In Figure A.1, it's possible to observe that for the 5 existent areas, only 3 of those require classification into measures (areas 1, 2 and 3). In Table A.2, all the measures are listed with the number of projects inserted into each one.

Classifier C_1^2 in Area 1

So for area 1, the classification between measure 1 (Innovation) and measure 2 (Knowledge), with an SVM algorithm, had the best accuracy of 0.991 and MCC score of 0.979, meaning that the classifier was able to distinguish between projects of these two measures. The KNN classifier obtained similar results to the SVM one, followed by RF and NB. In Figure 4.3, the weighted metrics of all the classifiers are shown alongside the confusion matrix from the SVM model, where it's possible to verify that only two projects were misclassified.



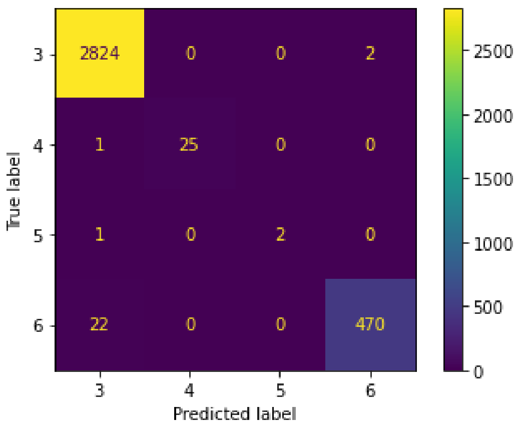
	SVM	RF	KNN	NB
Accuracy	**0.991**	0.974	**0.991**	0.970
F_1 Score	**0.991**	0.974	**0.991**	0.969
MCC	**0.979**	0.936	**0.979**	0.925

Figure 4.3: Confusion matrix and weighted metrics for classification in Area 1.

Classifier C_2^2 in Area 2

For area 2, there are 4 possible measures for the projects to be classified into: measure 3 - Valorization of Agricultural Production, which has the most projects (around 15500), measure 4 - Valorization of Forest Resources, measure 5 - Production Organization, that has the least in all the measures (12 projects), and measure 6 - Risk Management and Restoration of Productive Potential. The classification of these measures had a good accuracy of 0.992 and 0.971 MCC score, with a SVM classifier, which can be seen in Figure 4.4. The second-best model in this classification was the Random Forest Classifier, with an accuracy of 0.985. Also present in Figure 4.4, is the confusion matrix, where it's possible to observe that most misclassified projects are predicted as measure 3 and are actually from measure 6.

Overall, the classifier was able to distinguish measures 3, 4 and 6, and the deviations seen could be due to projects with similar content given that measure 3 focuses on the Valorization of Agricultural Production, measure 4 on Valorization of Forest Resources and measure 6 on Risk Management and Restoration of Productive Potential, meaning that all these would allow investments in agricultural



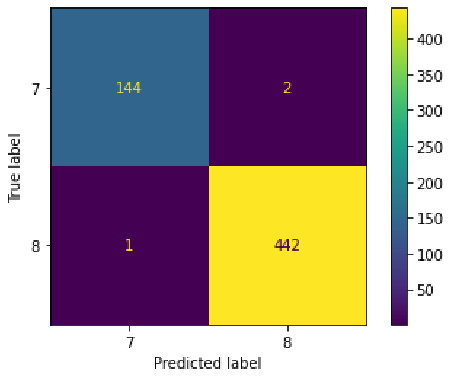
	SVM	RF	KNN	NB
Accuracy	**0.992**	0.985	0.981	0.977
F_1 Score	**0.992**	0.984	0.980	0.976
MCC	**0.971**	0.944	0.927	0.912

Figure 4.4: Confusion matrix and weighted metrics for the classification in Area 2.

resources either for restoration, due to natural catastrophes (measure 6), or for brand new farming exploration (measure 3 and 4). Another possibility to consider for these deviations would be the unbalanced dataset used, where the larger training data of measure 3 could be outweighing the other measures. As for measure 5, given that it has only 12 projects, only 3 projects were part of the testing set so it's most likely that the algorithm **cannot** effectively learn the features of projects of **measure 5**.

Classifier C_3^2 in Area 3

Finally, area 3 has 2 possible measures for classification: measure 7 (Agriculture and Natural Resources) and measure 8 (Protection and Rehabilitation of Forest Stands). This classification, using an KNN classifier, gave an accuracy of 0.995 and only 3 misclassified projects, followed closely by the SVM and NB models with 0.990. On account of these results, it can be concluded that this was a successful classification. In Figure 4.5, it's possible to verify the weighted metrics of all the algorithms tested and the confusion matrix and the confusion matrix for the results of the KNN model.



	SVM	RF	KNN	NB
Accuracy	0.990	0.988	**0.995**	0.990
F_1 Score	0.990	0.988	**0.995**	0.990
MCC	0.973	0.968	**0.986**	0.973

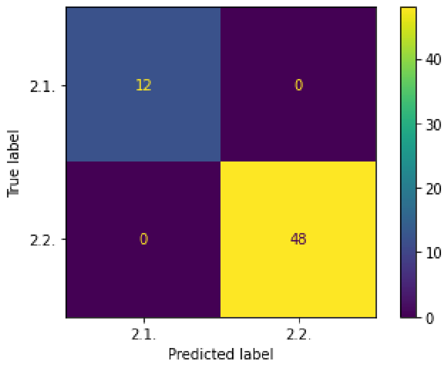
Figure 4.5: Confusion matrix and weighted metrics for the classification in Area 3.

4.3.3 Level 3: Action Classification

After classifying all the measures, the following step is the classification into actions. In Figure A.3, it is shown that out of the 10 existent measures, only 7 require classification into actions (measure 2, 3, 6, 7, 8, 10 and 20). Measure 1, 4 and 6 will be not classified into actions given that they have only 1 action. For further consulting, Table A.3 contains all the actions and the number of projects per action.

Classifier C_1^3 in Measure 2

Starting off with measure 2, there are 2 actions to be classified: 2.1 - Training and Dissemination and 2.2 - Counseling, with 52 and 281 projects, respectively. As it can be seen in Figure 4.6, this classification obtained 1.0 accuracy and an MCC score of 1.0, with no misclassified projects, using a SVM classifier. Therefore, it's possible to conclude that the algorithm was able to learn distinct features between these 2 actions and properly classify the projects.

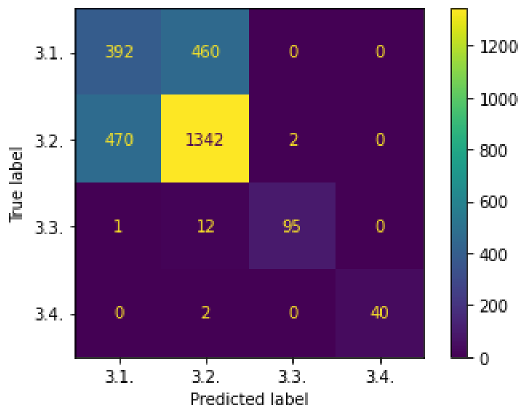


	SVM	RF	KNN	NB
Accuracy	**1.0**	0.983	0.983	0.867
F_1 Score	**1.0**	0.983	0.983	0.838
MCC	**1.0**	0.948	0.948	0.535

Figure 4.6: Confusion matrix and weighted metrics for the classification in Measures 2.

Classifier C_2^3 in Measure 3

Measure 3 incorporates projects for investments in agricultural production. In this measure, there are 4 actions for the projects to be classified into: 3.1 - Young Farmers, 3.2 - Investment in Farming, 3.3 - Investments in the Transformation and Marketing of Agricultural Products and 3.4 - Collective Infrastructures. Action 3.1 and 3.2 are quite similar, given that action 3.1 just adds a restriction on the age of the beneficiary, something that is not likely to be inserted in the description of the project. As for actions 3.3 and 3.4, these are both quite different from 3.1 and 3.2, given that action 3.3 focuses on projects with investments in the marketing of agricultural products, not on agricultural exploration, and action 3.4 accepts investments to improve irrigated soles and drainage. The results of this classification are present in Figure 4.7, which shows a rather lower maximum accuracy of 0.678 (of an NB model), compared with the previous classifications, and also the confusion matrix, where it's possible to observe the misclassified projects between actions 3.1 and 3.2, as expected and explained previously. The best MCC result, 0.347, obtained with a KNN model, also shows that some of the predictions are random and not in line with the real classes, most likely due to the projects of actions 3.1 and 3.2. It was chosen a SVM classifier as the best overall model, given that its measures were the second best, between NB and KNN, balancing the best results of accuracy and F_1 score (NB) and MCC (KNN).

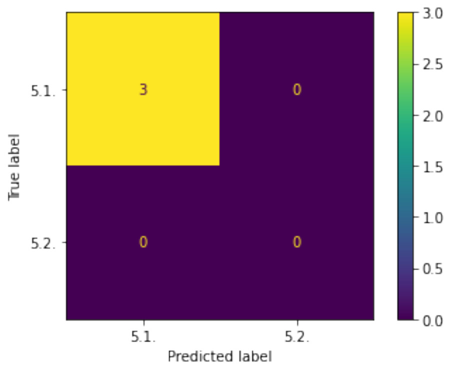


	SVM	RF	KNN	NB
Accuracy	0.664	0.647	0.632	**0.678**
F_1 Score	0.665	0.621	0.645	**0.670**
MCC	0.314	0.198	**0.347**	0.304

Figure 4.7: Confusion matrix and weighted metrics for the classification in Measures 3.

Classifier C_3^3 in Measure 5

As for measure 5, it was explained previously in the classification of measures in Area 2, this measure has only 12 projects, with 2 actions: 5.1 - Creation of Producer Groups and Organizations and 5.2 - Interprofessional Organizations, with 8 and 4 projects, respectively. With 12 projects, it's impossible for the algorithm to correctly learn distinct features and even though the accuracy of this classification is 1.0, the MCC score of 0.0 in all models tested, shows that it's not a significant result (e.g. the predictions are random).



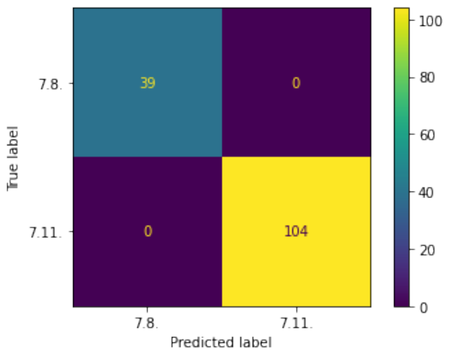
	SVM	RF	KNN	NB
Accuracy	1.0	1.0	1.0	1.0
F_1 Score	1.0	1.0	1.0	1.0
MCC	0.0	0.0	0.0	0.0

Figure 4.8: Confusion matrix and weighted metrics for the classification in Measures 5.

Classifier C_4^3 in Measure 7

In measure 7, the classification of the 2 existing actions was very successful, with accuracy and MCC of 1.0 in all classifiers. The algorithm was able to differentiate between action 7.8 - Genetic Resources and action 7.11 - Non-Productive Investments, given the difference of content of each one: action 7.8 focuses on the conservation of animal, vegetable and forestry genetic resources while action 7.11 accepts investments related to the recuperation of riparian galleries (which are the interface between land

and a river, with important contributions to the soil conversation and habit biodiversity) and also for the recuperation of laid stone walls. In Figure 4.9, the metrics of this classification and the confusion matrix are shown, with zero misclassified projects and 1.0 in all metrics.

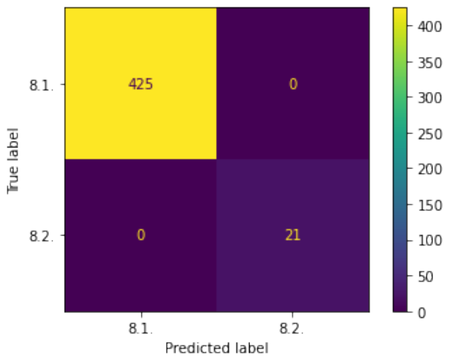


	SVM	RF	KNN	NB
Accuracy	1.0	1.0	1.0	1.0
F_1 Score	1.0	1.0	1.0	1.0
MCC	1.0	1.0	1.0	1.0

Figure 4.9: Confusion matrix and weighted metrics for the classification in Measures 7.

Classifier C_5^3 in Measure 8

For the classification inside measure 8, there are 2 actions to consider: 8.1 - Sustainable Silviculture and 8.2 - Management of Cynegetic and Aquaculture Resources. Like in measure 7, both of these actions are quite different in terms of content, meaning that the algorithm was able to successfully distinguish between these two. In Figure 4.10, it can be seen that the models with the best performances were the SVM classifier and the k-Nearest Neighbor classifier, with 1.0 of accuracy and MCC score. The RF model also had a very good performance, with a 0.993 accuracy.

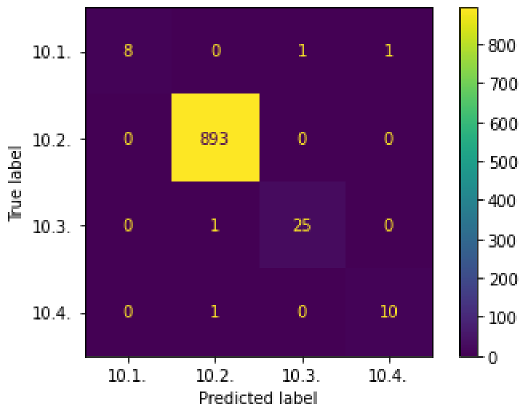


	SVM	RF	KNN	NB
Accuracy	**1.0**	0.993	**1.0**	0.973
F_1 Score	**1.0**	0.993	**1.0**	0.968
MCC	**1.0**	0.923	**1.0**	0.646

Figure 4.10: Confusion matrix and weighted metrics for the classification in Measures 8.

Classifier C_6^3 in Measure 10

Analyzing now the results of measure 10, it's possible to observe that the classification had very good results with an accuracy of 0.996 and an MCC score of 0.955 for the SVM model, in Figure 4.11. In measure 10, there are 4 actions to classify: 10.1 - Preparatory Support, 10.2 - Implementation of Strategies, 10.3 - GAL Cooperation Activities and 10.4 - Operation and Animation. Given the different subjects matter in each of these, the algorithm was able to perform well with only 4 misclassified projects in total. These deviations could be due to the larger number of projects in action 10.2, compared to the other classes, and the lower number of samples in action 10.1.

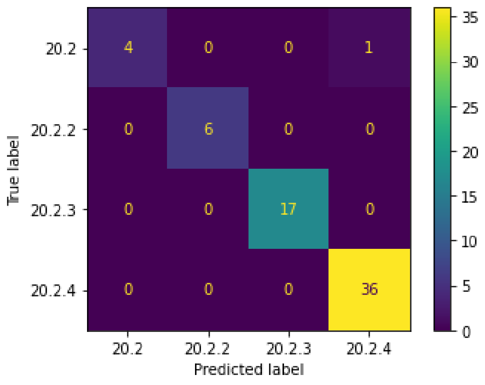


	SVM	RF	KNN	NB
Accuracy	**0.996**	0.982	0.989	0.968
F_1 Score	**0.996**	0.979	0.990	0.956
MCC	**0.955**	0.794	0.896	0.594

Figure 4.11: Confusion matrix and weighted metrics for the classification in Measures 10.

Classifier C_7^3 in Measure 20

Finally, the last measure that is necessary to classify into actions is measure 20, with 3 different classes to consider: action 20.1 - PDR2020 Technical Assistance, 20.2 - National Rural Network and 20.3 - Local Support Structures. In Figure 4.12, it can be verified that the all models obtained very good results with an accuracy and MCC score of 1.0, with only one misclassified project. Overall, it could be said that the models were able to learn distinct features to identify each different class.



	SVM	RF	KNN	NB
Accuracy	1.0	1.0	1.0	1.0
F_1 Score	1.0	1.0	1.0	1.0
MCC	1.0	1.0	1.0	1.0

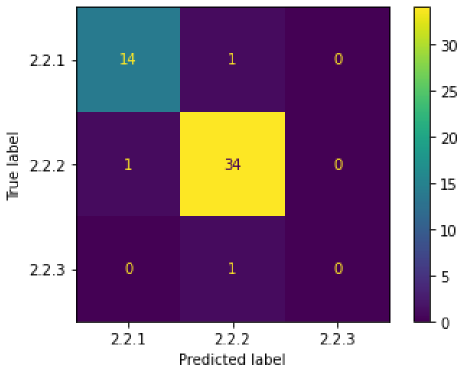
Figure 4.12: Confusion matrix and weighted metrics for the classification in Measure 20.

4.3.4 Level 4: Operation Classification

The final step of the classification is to identify all the operations from the corresponding actions. From all 22 actions, only 12 need to be further classified as can be seen in Figure A.3. Actions 1.0, 2.1, 5.1, 5.2, 7.11, 10.1, 10.3, 10.4, 20.1 and 20.3 only have one operation and therefore don't require classification in this level. All of the operations to be classified in this level are listed in Table A.4 alongside the number of projects inserted into each operation.

Classifier C_1^4 in Action 2.2

Beginning with classification in action 2.2, there are 3 possible operations for the projects to be classified: 2.2.1 - Support for the Provision of Agricultural and Forestry Advisory Services, 2.2.2 - Supporting the Creation of Counseling Services and 2.2.3 - Support for the Training of Counselors of the Counseling Service Provider Entities. This classification wielded good results with only two misclassified projects between operations 2.2.1 and 2.2.2. As for operation 2.2.3, given that it had only 2 projects, the algorithm could not learn enough features to properly classify projects into this operation. The best results were found with the SVM and KNN classifiers, which had a 0.941 accuracy and 0.846 of MCC.

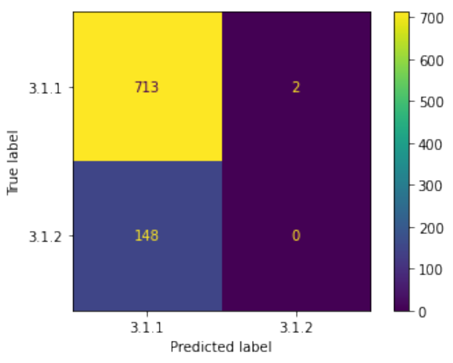


	SVM	RF	KNN	NB
Accuracy	**0.941**	0.902	**0.941**	0.804
F_1 Score	**0.932**	0.891	**0.932**	0.769
MCC	**0.864**	0.770	**0.864**	0.541

Figure 4.13: Confusion matrix and weighted metrics for the classification in Action 2.2.

Classifier C_2^4 in Action 3.1

Moving on to action 3.1 (Young Farmers), this one contains 2 operations to be classified: 3.1.1 - Young Farmers and 3.1.2 - Investment of young farmers in agricultural exploration. The difference between these two operations is the type of investment supported. While operation 3.1.1 focus on brand new agricultural explorations and rewards them with a cash prize, operation 3.1.2 awards a cash prize as well as a state co-payment. That being said, it's excepted that the descriptions of the projects don't contain such information, making it difficult to distinguish these two classes. Overall, the model reached an accuracy of 0.826 and an MCC score of -0.022, meaning that the predictions are random, as can be seen in Figure 4.14. The best results were surprisingly found with a Naive Bayes classifier, which has been able to outperform the rest of the models. Analysing the confusion matrices of the other models, it was possible to understand that the projects from 3.1.1, that were misclassified in other models, diminished significantly with the NB classifier, explaining the high metrics obtained. It's also possible to verify, in Figure 4.14, that out of 148 projects from operation 3.1.2 that were being evaluated, none was correctly classified. As for operation 3.1.1, it has been able to predict appropriately around 99% of the evaluated projects. In conclusion, it's possible to infer that the algorithm could not learn enough features to predict correctly these operations.

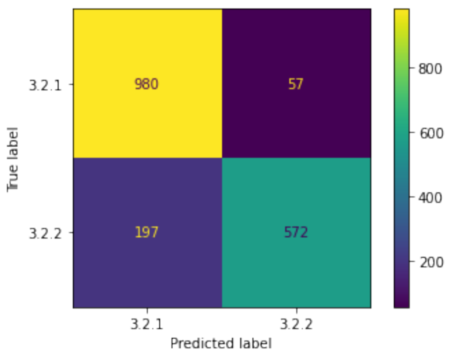


	SVM	RF	KNN	NB
Accuracy	0.730	0.723	0.793	**0.826**
F_1 Score	0.700	0.695	0.734	**0.750**
MCC	-0.142	-0.156	-0.075	**-0.022**

Figure 4.14: Confusion matrix and weighted metrics for the classification in Action 3.1.

Classifier C_3^4 in Action 3.2

Action 3.2 contains the larger dataset in all actions, close to 10 thousand projects. There are 2 operations to be classified within this action: 3.2.1 - Investments in Farming and 3.2.2 - Small Investments in Farming. As the name of these operations suggests, the only difference between them is the value of the investment made by the beneficiary. Like the previous classification in action 3.1, it's unlikely for the description to contain such information, however, the scope of projects with large investments made could be different than the ones with smaller investments. In Figure 4.15, it can be seen that the RF classifier obtained the best accuracy of 0.859 and an MCC score of 0.715. The SVM model had the second-best performance, with the only difference being that, while the SVM predicts 3.2.2 with a higher degree of success, the RF classifier predicts more projects of operation 3.2.1 correctly. Overall, the results were quite positive and the model was able to learn distinct features from each operation, although it misclassifies around 250 projects in a testing dataset of 1800 ($\sim$10%).

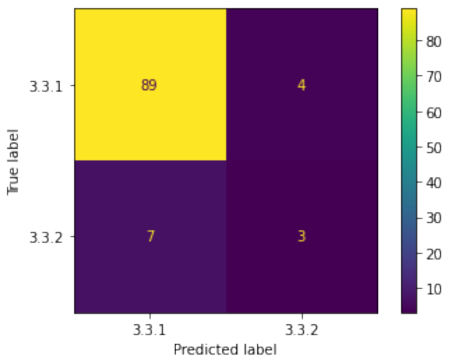


	SVM	RF	KNN	NB
Accuracy	0.854	**0.859**	0.851	0.834
F_1 Score	0.854	**0.857**	0.849	0.830
MCC	0.701	**0.715**	0.696	0.665

Figure 4.15: Confusion matrix and weighted metrics for the classification in Action 3.2.

Classifier C_4^4 in Action 3.3

Like the previous actions of measure 3, action 3.3 contains 2 operations to be classified: 3.3.1 - Investments in the Transformation and Marketing of Agricultural Products and 3.3.2 - Small Investments in the Transformation and Marketing of Agricultural Products. The difference between these operations, like in action 3.2, is the value of the investment made by the beneficiary. Unlike action 3.2, which had a large number of projects to learn from, action 3.3 has only around 500 projects, where more than 400 of those are assigned to operation 3.3.1. In conclusion, the best model, k-Nearest Neighbor, obtained an accuracy of 0.893 and an MCC score of 0.302, a lot lower than the one obtained in action 3.2, which indicates the possibility of the predictions not being trustworthy. Although the RF and NB models had better accuracy, the MCC score shows that the overall performance was not better than that of the KNN classifier. In Figure 4.16, it's possible to verify that only 10 projects of operation 3.3.2 were tested and 3 of those were predicted correctly. However, most projects tested of operation 3.3.1 are classified into the correct class. As was expected, the algorithm was not able to learn enough easily distinguishable features of operation 3.3.2 due to the low number of data for that operation.



	SVM	RF	KNN	NB
Accuracy	0.864	**0.903**	0.893	**0.903**
F_1 Score	0.857	0.857	**0.885**	0.857
MCC	0.150	0.000	**0.302**	0.000

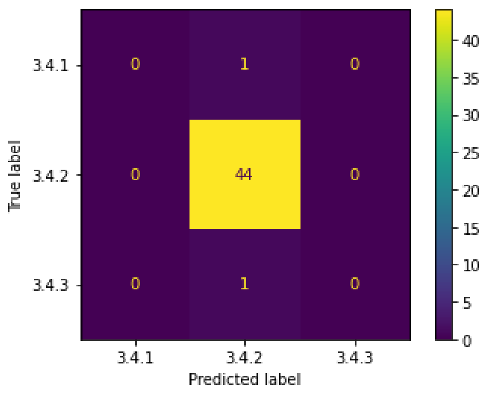
Figure 4.16: Confusion matrix and weighted metrics for the classification in Action 3.3.

Classifier C_5^4 in Action 3.4

The last action of measure 3, action 3.4 consists of projects related to collective infrastructures and contains 3 operations to be classified: 3.4.1 - Efficient Irrigation Development, 3.4.2 - Improved Efficiency of Existing Irrigation and 3.4.3 - Drainage and Land Structuring. The sum of projects in this action is around 250, however, the data is not divided equally to the operations. Operation 3.4.2 contains most of the projects, while operations 3.4.1 and 3.4.3 contain only 8 and 3 projects, respectively. The results for this classification show a 0.957 accuracy and MCC score of 0.0 for all models, meaning that the models correctly predict most projects evaluated but it doesn't have a significant meaning because only 1 project of operation 3.4.1 and of operation 3.4.3 are being tested, which are both misclassified. It's possible to infer that the algorithm cannot learn enough features to properly classify projects into operations 3.4.1 and 3.4.3.

Classifier C_6^4 in Action 4.0

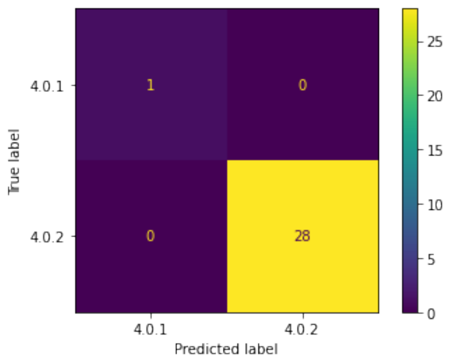
In action 4.0, there are 2 operations to be classified: 4.0.1 - Investments in Forest Products Identified as Agricultural and 4.0.2 - Investments in Forest Products Not Identified as Agricultural. In this classification, there is the same problem of unbalanced data as the previous action. With a total of 160 projects in this action, operation 4.0.1 contains only 14 of those, which means that it's unlikely for the model to



	SVM	RF	KNN	NB
Accuracy	0.957	0.957	0.957	0.957
F_1 Score	0.935	0.935	0.935	0.935
MCC	0.000	0.000	0.000	0.000

Figure 4.17: Confusion matrix and weighted metrics for the classification in Action 3.4.

learn to properly classify this operation. As far as the metrics shown in Figure 4.18, the accuracy and MCC results were of 1.0 for the SVM and KNN model. Given that only 1 project was tested belonging to operation 4.0.1, while all tested projects of operation 4.0.2 were correctly classified, the algorithm is probably able to learn features regarding operation 4.0.2, but not for 4.0.1, resulting in perfect accuracy. Although the performance of the models was quite high, it's very likely that the model is not acceptable for the classification of further projects.

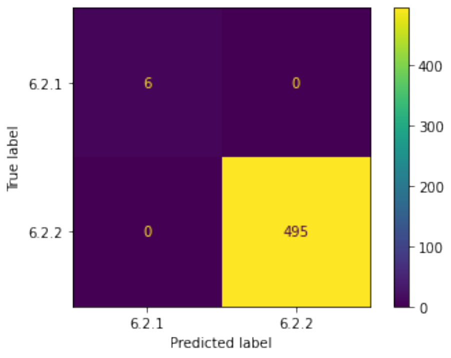


	SVM	RF	KNN	NB
Accuracy	**1.0**	0.966	**1.0**	0.966
F_1 Score	**1.0**	0.949	**1.0**	0.949
MCC	**1.0**	0.000	**1.0**	0.000

Figure 4.18: Confusion matrix and weighted metrics for the classification in Action 4.0.

Classifier C_7^4 in Action 6.2

Action 6.2 belongs to measure 6 and focus on the restoration and prevention of calamities and natural disasters. This action divides into 2 operations: 6.2.1 - Prevention of Natural Disasters and Disasters and 6.2.2 - Restoration of Productive Potential. Given the scope of these operations, it is excepted for the model to be able to distinguish these 2 operations. However, once again, this classification has an unbalanced dataset: operation 6.2.1 contains around 30 projects while operation 6.2.2 has 2737 projects to learn from. The results showed in Figure 4.19 reveal that the algorithm was able to learn distinct features for each operation, reaching an accuracy and MCC score of 1.0 for the SVM classifier. Given the smaller number of projects belonging to operation 6.2.1, only 6 projects were tested and none was misclassified. Alongside the fact that all projects belonging to operation 6.2.2 were all classified correctly, the high metrics found for these classifiers is justified.

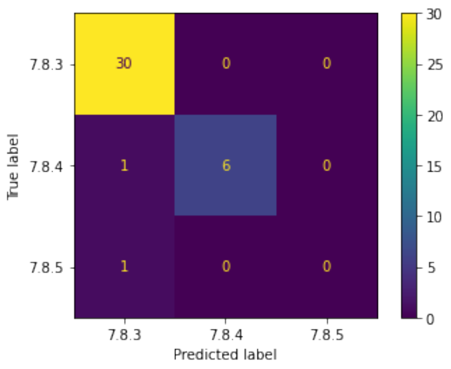


	SVM	RF	KNN	NB
Accuracy	**1.0**	0.994	0.996	0.988
F_1 Score	**1.0**	0.993	0.996	0.982
MCC	**1.0**	0.705	0.864	0.000

Figure 4.19: Confusion matrix and weighted metrics for the classification in Action 6.2.

Classifier C_8^4 in Action 7.8

Action 7.8 is related to the conservation of different genetic resources and has 3 operations to be classified, divided by the type of genetic resources to handle: 7.8.3 - Conservation and Improvement of Animal Genetic Resources, 7.8.4 - Conservation and Improvement of Vegetable Genetic Resources and 7.8.5 - Conservation and Improvement of Forestry Genetic Resources. Out of these 3 operations, 7.8.3 has the highest number of projects (178 projects), while 7.8.4 has around 20 projects and 7.8.5 has only 4. While the results from the SVM and KNN classifier, shown in Figure 4.20, achieved a 0.947 accuracy and an MCC score of 0.840, the number of tested projects from operation 7.8.4 and 7.8.5 was only 7 and 1, respectively, meaning that the model wasn't able to test fully the knowledge of those classes. Once again, because there are only two misclassified projects, the performance of the models is quite high, which can be misleading the conclusion that the model is fully able to distinguish these classes.



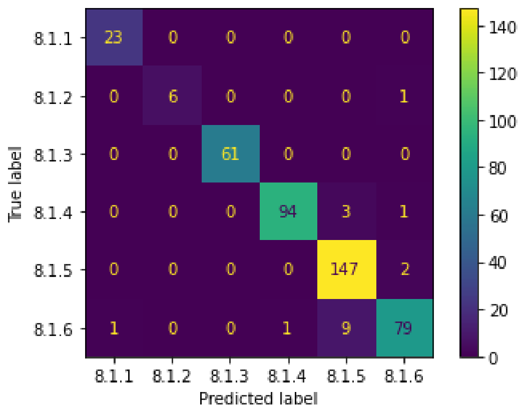
	SVM	RF	KNN	NB
Accuracy	**0.947**	0.842	**0.947**	0.789
F_1 Score	0.934	0.800	**0.944**	0.697
MCC	0.840	0.457	**0.843**	0.000

Figure 4.20: Confusion matrix and weighted metrics for the classification in Action 7.8.

Classifier C_9^4 in Action 8.1

Action 8.1 (Sustainable Silviculture) has a large number of operations to be classified: 8.1.1 - Afforestation of Agricultural and Non-Agricultural Land, 8.1.2 - Installation of Agroforestry Systems, 8.1.3 -

Forest Prevention against Biotic and Abiotic Agents, 8.1.4 - Restoration of Forest Affected by Biotic and Abiotic Agents, 8.1.5 - Improving the Resilience and Environmental Value of Forests and 8.1.6 - Improving the Economic Value of Forests. The class with the largest dataset in this classification is operation 8.1.5, with a total of 851 projects, and the class with the lowest is operation 8.1.2, which has only 33 projects. The rest of the operations have a decent amount of data, ranging from 150 projects (operation 8.1.1) to 530 projects (operation 8.1.4). Although the scopes of these operations are all different, it could prove difficult to distinguish such closely related projects. The best model for this classification was the SVM classifier, which achieved an accuracy of 0.958 and an MCC score of 0.945, both quite good results. However, due to the larger number of projects in the training set for operation 8.1.5, there were some misclassified projects as 8.1.5, especially from 8.1.6 and 8.1.4, alongside some misclassifications to operations 8.1.1, 8.1.4 and 8.1.6. Operation 8.1.2 tested only 6 projects, with only one misclassified. Overall, the model was able to distinguish these classes with some excepted deviations. The results from the other models, alongside the confusion matrix from the SVM model, are shown in Figure 4.21.

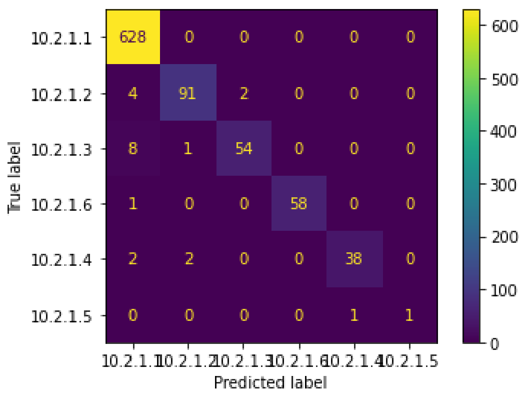


	SVM	RF	KNN	NB
Accuracy	**0.958**	0.942	0.900	0.778
F_1 Score	**0.958**	0.941	0.899	0.765
MCC	**0.945**	0.924	0.868	0.723

Figure 4.21: Confusion matrix and weighted metrics for the classification in Action 8.1.

Classifier C_{10}^4 in Action 10.2

Action 10.2, like the previous action, has a large number of operations to be classified: 10.2.1.1 - Small Investments in Farming, 10.2.1.2 - Small investments in processing and marketing, 10.2.1.3 - Diversification of exploration activities, 10.2.1.4 - Short chains local markets, 10.2.1.5 - Promotion of local quality products and 10.2.1.6 - Renovation of villages. Operation 10.2.1.1 has the largest dataset with 3508 projects and operation 10.2.1.5 has the smallest with only 14 projects, while the rest of the operations have a dataset ranging from 200 to 500 projects. This classification obtained good results with a maximum accuracy of 0.976 and MCC score of 0.950, with a SVM classifier, shown in Figure 4.22. From the confusion matrix, it's possible to verify that the wrong predictions were usually projects misclassified as operation 10.2.1.1 and that most classes had a few misclassified projects, except operation 10.2.1.6. In conclusion, it was expected that the model would be able to distinguish the classes better. However, possibly due to the larger dataset from operation 10.2.1.1, the features from this operation could be outweighing the other classes.

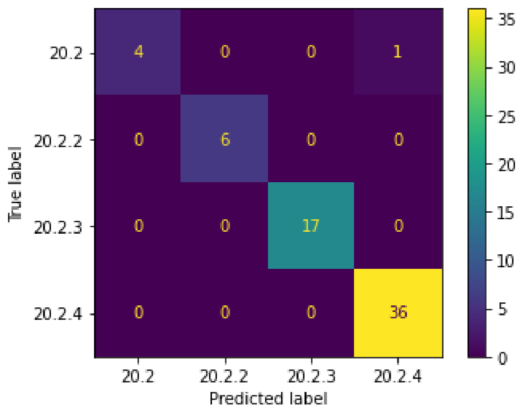


	SVM	RF	KNN	NB
Accuracy	**0.976**	0.915	0.923	0.878
F_1 Score	**0.976**	0.909	0.919	0.860
MCC	**0.950**	0.816	0.833	0.732

Figure 4.22: Confusion matrix and weighted metrics for the classification in Action 10.2.

Classifier C_{11}^{4} in Action 20.2

The last classification that needs to be evaluated is for action 20.2, which has a total of 4 operations to be classified: 20.2 - Technical assistance, 20.2.2 - LEADER, 20.2.3 - Dissemination and Knowledge and 20.2.4 - Innovation. While operations 20.2, 20.2.3 and 20.2.4 have self-explanatory titles, operation 20.2.2 has the name of LEADER and is composed of projects with the objective of contributing to the application and evaluation of the rural development policy measures. The dataset for this operation ranges from 28 projects (operation 20.2) to 187 projects (operation 20.2.4). Given the low number of training data available, the models were surprisingly able to provide very high results. The SVM classifier, as well as the Random Forest model, both had an accuracy of 0.984 and MCC score of 0.975, shown in Figure 4.23, with only one misclassified project. In conclusion, the models were able to learn to distinguish the operations, most likely due to the different scopes of each operation in this classification.



	SVM	RF	KNN	NB
Accuracy	**0.984**	**0.984**	0.969	0.906
F_1 Score	**0.984**	**0.984**	0.969	0.902
MCC	**0.974**	**0.974**	0.949	0.846

Figure 4.23: Confusion matrix and weighted metrics for the classification in Action 20.2.

4.3.5 Discussion

To summarize the results obtained from all the models, Tables 4.5 and 4.6 show the best results of the area, measure, action and operation classification, respectively. For the 22 classifications, the best results were found with the following models:

- Support Vector Machine had the best results in **18** classifications;
- k-Nearest Neighbors had the best performance in **2** classifications (in area 3 and action 3.3);
- Naive Bayes classifier had the best performance in **1** classification (in action 3.1);
- Random Forest model had the best performance in **1** classification (in action 3.2);

The classifiers that are not SVM are identifiable, with a footnote, which states the model used for the results shown, in Tables 4.5 and 4.6.

Table 4.5: Results of Individual Classifiers - Part 1.

	Area Classifier	Areas			Measures						
		1	2	3 KNN	2	3	5	7	8	10	20
Accuracy	0.992	0.991	0.992	0.995	1.0	0.664	1.0	1.0	1.0	0.996	1.0
F_1 Score	0.991	0.991	0.992	0.995	1.0	0.665	1.0	1.0	1.0	0.996	1.0
MCC	0.977	0.979	0.971	0.986	1.0	0.314	0.0	1.0	1.0	0.955	1.0

Table 4.6: Results of Individual Classifiers - Part 2.

	Actions										
	2.2	3.1 NB	3.2 RF	3.3 KNN	3.4	4.0	6.2	7.8	8.1	10.2	20.2
Accuracy	0.941	0.826	0.859	0.893	0.957	1.0	1.0	0.947	0.958	0.976	0.984
F_1 Score	0.932	0.750	0.857	0.885	0.935	1.0	1.0	0.934	0.958	0.976	0.984
MCC	0.864	-0.022	0.715	0.302	0.000	1.0	1.0	0.840	0.945	0.950	0.974

Even though the SVM models obtained sometimes the best results with a significant difference from other classifiers, 5 classifications had equal results between the SVM model and the RF or KNN model. Overall, it was possible to obtain accuracies above 0.95 and MCC scores above 0.70 for a total of **17 models**:

- Area classifier;
- Classifiers in areas 1, 2 and 3;
- Classifiers in measures 2, 7, 8, 10 and 20;
- Classifiers in actions 2.2, 3.2, 4.0, 6.2, 7.8, 8.1, 10.2 and 20.2.

The models for classification in actions 2.2, 3.2 and 7.8 obtained very good results also, while not as high as the rest of the classifications mentioned, with MCC scores ranging from 0.7 to 0.8. Action 2.2 and 7.8 both have a class with very few samples to train and test and action 3.2 has the problem related to the investment size (normal vs small investments), which can bring some misclassifications in the evaluation. However, given both those issues, these classifiers obtained better results than expected.

The last classifications to mention are the ones that had the worst performance (an MCC lower than 0.5), which were the following **5 models**:

- Classifiers in measures 3 and 5;
- Classifiers in actions 3.1, 3.3 and 3.4;

Although the accuracy results in these models were relatively high, from 0.65 to 0.90, the MCC score reveals that the classification is actually not viable, given that an MCC score close to zero shows that the predictions made by the model are completely random. The low scores in these classifications are due to one of the following issues:

- Classes that are difficult to distinguish given the input provided (as it happens in the classifiers in measure 3 and action 3.1);
- Small number of training samples (classifier in measure 5);
- Unbalanced dataset combined with few data available to train and test (actions 3.3 and 3.4)

The classifications in actions 2.2, 3.3, 3.4, 4.0, 6.2 and 7.8 have been mentioned previously, however, there is a common characteristic in these models, despite each of them having a different type of performance: one or more classes of the classification do not have many samples to test, leaving just one class being tested. The class being tested can be correctly predicted, which justifies the results found in actions 2.2 and 6.2, or it can misclassify some of the samples, lowering the evaluation metrics as it happens in actions 3.3 and 3.4.

4.4 Analysis of 2nd Predictions Using Probabilities

As it was explained in section 3.4, to improve the results described previously, it was decided to add decision thresholds to the SVM models. However, SVM models don't deal with probabilities, making it necessary to build new calibrated SVM models, used to obtain probabilities of predictions. Two regressors were tested for building calibrated models, introduced in section 2.3.2. Using the same strategy as in 4.1, it was chosen to use the Isotonic Regression, which is the best method for larger datasets. After training these new calibrated models, it was noted that there were some differences in performance compared to the original results, although not significant. The results of these classifiers are shown in the Tables B.1, B.2 and B.3, presented in Appendix B.

4.4.1 Analysis of Probabilities

In section 3.4, a heuristic strategy was defined for finding the best threshold in each classifier. The objective function $F(t)$, defined in (3.1), had some weights to be defined that rewarded and penalized certain behaviours. Using the partial knowledge acquired from the system developed, these weights were defined as follows: ω_τ was given a positive value of $+1$, considering that the goal is to optimize the correct predictions; ω_ρ was also attributed a weight of $+1$, penalizing large values in this variable; and ω_ϕ was defined as $+5$, to stress the importance of improving the evaluation metrics. With these weights defined, the objective function implemented is shown in (4.2).

$$F(t) = \tau(t) - \rho(t) + 5 \times \phi(t) \tag{4.2}$$

In regards to the thresholds t examined, all the models were tested with seventeen possible thresholds starting in 0.3 and ending in 0.7. It was considered that a predicted class with a probability higher than 0.7 was very likely to be correct, so thresholds above 0.7 were not tested. Out of 22 SVM calibrated models tested, only 8 showed improvements with the consideration of a second prediction:

- Area classifier.
- Classifiers in areas 2 and 3.
- Classifiers in measure 3.
- Classifiers in actions 3.2, 3.3, 8.1 and 10.2.

The classifiers in actions 3.1, 3.4, and measure 5, which had the worst results, were unable to improve. The rest of the models had already very high performance, which implies that the probabilities in those predictions were above 0.7 and therefore, didn't consider a second prediction in the tests made.

Starting by analysing the results from the area classifier, shown in Table 4.7, it was found that there were no projects with probabilities lower than 0.3, and that considering a threshold of 0.38, the accuracy improved from 0.993 to 0.994 and the MCC score from 0.982 to 0.983. With this threshold, the algorithm would analyse 0.02% of projects and all of those had a correct second prediction. By raising the threshold to 0.7, accuracy would improve to 0.996 and MCC to 0.990, nonetheless, the number of projects seen would rise and the number of correct second predictions would be lower.

Table 4.7: Evaluation of thresholds in the Area classifier.

	Threshold, t	0.3	0.38	0.4	0.5	0.6	0.7
	% Correct 2nd Pred	0.000	100.000	100.000	50.000	50.000	36.590
	% 2nd Predictions	0.000	0.020	0.020	0.040	0.310	0.790
	Accuracy	0.993	0.994	0.994	0.994	0.995	0.996
	MCC	0.982	0.983	0.983	0.983	0.987	0.990
Parcels of $F(t)$	$\tau(t)$	0.000	0.113	0.113	0.056	0.056	0.041
	$\rho(t)$	0.000	0.006	0.006	0.011	0.086	0.218
	$\phi(t)$	0.059	0.059	0.059	0.059	0.059	0.059
	$\mathbf{F(t)}$	**0**	**0.401**	**0.401**	**0.339**	**0.265**	**0.118**

Note: Not all threshold tested are shown in this table.

For the area classifier, the different parcels of $F(t)$ for the defined thresholds were calculated, as shown in Table 4.7. With these values, it is possible to create a graphic representation of the optimization problem, shown in Figure 4.24, and verify that **0.38** provided the highest value of $F(t)$.

Using the same logic as explained for the area classifier, the rest of the models were analysed based on the information shown in Appendix B.2. The best thresholds found are shown in Table 4.8.

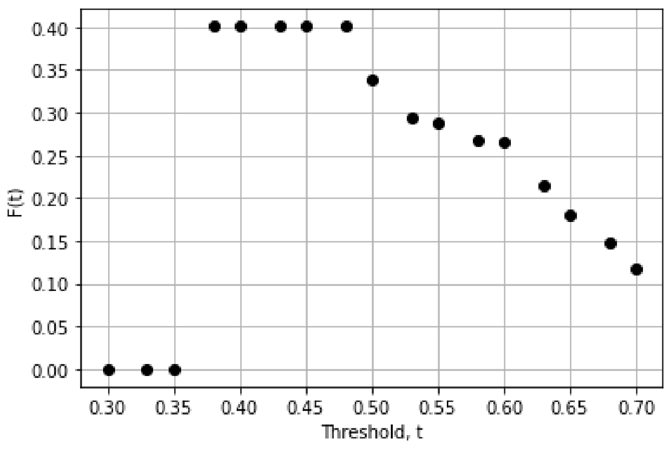


Figure 4.24: Threshold heuristics problem in the area classifier.

Table 4.8: Chosen thresholds for each classifier.

	Area Classifier	Areas		Measure	Actions			
		2	3	3	3.2	3.3	8.1	10.2
Threshold	0.38	0.53	0.63	0.6	0.53	0.60	0.48	0.48

4.4.2 Classification Results with 2nd Predictions

Analysis of Individual Classifiers

After analysing the probabilities and choosing appropriate thresholds to separate predictions that the model is confident about and those that the model is not, each threshold was implemented for the models mentioned previously. Testing all the models again, considering a second prediction, the results showed some improvements, as can be seen in Table 4.9, which compares the results retrieved previously with just one prediction and the results with a second prediction.

Table 4.9: Comparison between results with 1st and 2nd prediction.

	Area classifier		Area 2		Area 3		Measure 3	
Predictions	1st	2nd	1st	2nd	1st	2nd	1st	2nd
Accuracy	0.992	0.994	0.992	0.993	0.995	0.997	0.664	0.958
F_1 Score	0.991	0.994	0.992	0.993	0.995	0.997	0.665	0.958
MCC	0.977	0.983	0.971	0.974	0.986	0.991	0.314	0.916

	Action 3.2		Action 3.3		Action 8.1		Action 10.2	
Predictions	1st	2nd	1st	2nd	1st	2nd	1st	2nd
Accuracy	0.859	0.886	0.893	0.922	0.956	0.956	0.976	0.983
F_1 Score	0.857	0.885	0.885	0.913	0.958	0.955	0.976	0.983
MCC	0.715	0.766	0.302	0.478	0.945	0.942	0.950	0.965

In the case of the classifiers in areas 2, 3, and actions 8.1 and 10.2, they already had a very good performance and with this approach, it was possible to raise some of the metrics evaluated. The most significantly affected models were the classifiers in measure 3, actions 3.2 and 3.3. All of these showed several misclassifications due to the very similar classes: for measure 3, as it can be seen in the confusion matrix of Figure 4.7, most misclassified projects were between action 3.1 and 3.2 and the second prediction being considered for most of these cases was the correct one.

Analysis of Hierarchical Classification

As it was stated before, the new SVM calibrated models showed some differences in performance to the original SVM classifiers built, and to compare the results with the 2nd prediction, it was decided to retrieve the results from the sequential classification with the SVM calibrated models. These results are present in Appendix B, in specific Table B.1. In this section, the hierarchical classification is tested once again, comparing the results considering just one prediction and also considering a second one. The metrics evaluated in all four levels of the hierarchy can be seen in Table 4.9, comparing the previous and new results.

Table 4.10: Comparison of sequential results with 1^{st} and 2^{nd} prediction.

	Level 1		Level 2		Level 3		Level 4	
Prediction	1^{st}	2^{nd}	1^{st}	2^{nd}	1^{st}	2^{nd}	1^{st}	2^{nd}
Accuracy	0.992	0.994	0.989	0.990	0.826	0.952	0.756	0.878
F_1 Score	0.992	0.994	0.989	0.990	0.808	0.951	0.738	0.867
MCC	0.980	0.984	0.983	0.985	0.788	0.941	0.729	0.864

Regarding the first two levels, classification of areas and measures, the results are very similar, showing a slight increase in the evaluated metrics. However, level 3 (classification into actions) shows a significant improvement with an accuracy of 0.952 and a 0.941 MCC score, compared to the previous results of 0.826 and 0.788, respectively. Given that this level had the more severe drop in performance, around 0.16 of accuracy, it's possible to verify that considering a second prediction, the performance decrease becomes much smaller. As for the last level, there is a decrease in performance, going from 0.952 to 0.878 of accuracy (a decline of 0.07, similar to the previous results), which leads to the conclusion that the classification into operations cannot be improved any further. Overall, the classification considering second predictions had a final performance of **0.878** in accuracy and a 0.864 MCC, a considerable enhancement from the first results.

Chapter 5

Conclusions

5.1 Main conclusions

The goal of this thesis consisted in developing an intelligent assistant to find the best choice for financial support (by means of a call or notice), through a description of the project that the beneficiary wants to implement. This problem was approached as hierarchical multi-class text classification and involved the study and development of text processing for the Portuguese language and of Machine Learning models to produce the excepted outcome. Four types of ML algorithms were chosen and implemented as a possible solution to this problem: Naive Bayes, Support Vector Machine, Random Forest and k-Nearest Neighbors. In order to evaluate the performance of these models, three measures stood out: accuracy, F_1 score and Mathews Correlation Coefficient.

Concerning the testing procedure in this work, two types of results were retrieved and analysed, one in regards to the individual classifiers and the other in regards to the hierarchical classification as a whole. By analysing the hierarchical classification, all classifiers were brought together and provided clear results for the application of the funds assistant. Out of the four models considered, a classification built only with SVM classifiers obtained the best performance, starting in the first level with high accuracy of **0.992** and ending with a **0.717** accuracy in the last level. While SVM outperformed the rest of the models in all levels, the KNN algorithm showed the second-best results in the first two levels, although the NB classifiers surpassed the KNN in the last two levels, with a final accuracy of 0.689.

By analysing each individual classifier defined, it was possible to closely compare the models tested and also evaluate the classes of each model, allowing the reflection of possible reasons why it failed or not. In this analysis, it was again verified that the SVM algorithm outperforms all others in a large majority of cases. A close second was the k-Nearest Neighbors algorithm that obtained the same or the second-best results in half the classifiers. In conclusion, out of 22 classifiers, only 5 of them had an MCC score lower than 0.5 but still showed high accuracy. The rest of the classifiers obtained high accuracy, ranging from 0.8 to 1.0, and also high MCC scores above 0.7. However, 6 classifiers showed a clear issue regarding the unbalanced class dataset, which could lead to misclassifications in a real-life situation. For a more detailed review of these results, please consult section 4.3.

Given the preliminary results obtained, it was decided to enhance the SVM models, which showed the most promising performance in both analyses. This was done by considering a second prediction when the model was uncertain about its first one. When analysing the predictions with their associated probabilities, it was possible to infer a threshold where a classifier became unsure of its prediction, without over-considering correct predictions. It is also interesting to note that the best thresholds found had values between 0.38 and 0.60, given that considering a higher probability than those meant analysing too many correct predictions and lower meant missing some wrong predictions that could be correctly predicted. With this strategy, a total of 8 out of 22 classifiers implemented a second prediction and were able to improve their performance. The hierarchical classification also showed a great improvement in the last level of the hierarchy, obtaining a **0.878** accuracy. The rest of the levels also showed slight increases, maintaining a **0.952** accuracy until the last level.

In terms of the Portuguese language text processing, it was found that the Snowball Stemmer, which can be applied to a number of non-English languages, provided the best results, even though the RSLPS Stemmer did not fall very far behind. Another interesting point studied was the normalization of the text, where it was concluded that the removal of accentuation brings better results in terms of accuracy and MCC score, alongside a smaller running time for training a model. An additional observation made is that removing accents results in the elimination of possible spelling errors, producing more accurate performances.

In conclusion, by approaching this problem as a hierarchical classification, the very first results proved to be quite satisfactory, in terms of the evaluated metrics. The following step of improving the results with the consideration of second predictions was truly useful in some classifiers and managed to improve the overall hierarchical classification. A downfall of these models is the vocabulary used. The TF-IDF weighting scheme implemented meant that the models have limited knowledge in terms of words, semantic and syntax. Other approaches could potentially provide a more robust solution for real-life applications.

5.2 Future Work

For future work, a lot of alternatives could be explored. Several neural networks (NN) have been developed for text information, such as BERT, Glove, FastText, among others, and these could be implemented in two different ways. The first would be to use it as a vectorization process, taking advantage of the larger vocabulary corpus that the models have been pre-trained on. Given that there are very few models developed in European Portuguese, one option would be to use a multi-lingual model, which probably cannot provide better results than models trained on the target language (as the solution developed here). Another approach would be to use models trained on Brazilian Portuguese, such as BERT Timbau. A third option would be training a BERT from scratch, using a dataset composed of European Portuguese texts only. Training a model this way would involve a high number of labeled samples, which could be achieved through web scrapping, but would be a highly complicated and time-consuming task. With any of these methods, using the text information available as input to the neural network, the output would come directly from the embeddings in the NN and could be used as input for classification models, such as Support Vector Machines. The second approach for using neural networks would be to work fully with these pre-trained models, from vectorization to classification, given that many of them have specific models for text or sequence classification (for example, the model *BertForSequenceClassification*).

Another solution that could be explored involves reformulating the problem as text similarity. While text classification finds associations between a labeled class and a text input, text similarity focuses on the similarity between words, by projecting words in a vector space and calculating distances between them. A novel algorithm, that focuses on similarity, is the ANNOY model, developed by Spotify for music recommendations. This algorithm uses an Approximate Nearest Neighbor search to find similarities in a vector space, similar to the k-Nearest Neighbor classifier tested in this work. Given the promising results from the KNN classifier, which obtained the same or second-best performance in half the models developed in this work, the ANNOY algorithm could further improve these results in a state-of-the-art solution.

Last, to improve the results from second predictions retrieved, instead of a heuristic strategy to choose the decision threshold, an optimization problem could be introduced to find the threshold best suited for each model, in particular, a state-of-the-art Bayesian optimization.

Bibliography

[1] European structural and investment funds. https://ec.europa.eu/info/funding-tenders/funding-opportunities/funding-programmes/overview-funding-programmes/european-structural-and-investment-funds_en, Aug 2018.

[2] Open data portal for the european structural investment funds - european commission: Data: European structural and investment funds. https://cohesiondata.ec.europa.eu/countries/PT.

[3] Recovery plan for europe. https://ec.europa.eu/info/strategy/recovery-plan-europe_en, Apr 2021.

[4] Pdr 2020. http://www.pdr-2020.pt/O-PDR2020.

[5] Programas operacionais: Portugal 2020. https://www.portugal2020.pt/content/programas-operacionais, .

[6] Pdr 2020 - arquitetura. http://www.pdr-2020.pt/O-PDR2020/Arquitetura.

[7] Kowsari, J. Meimandi, Heidarysafa, Mendu, Barnes, and Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019. doi: 10.3390/info10040150.

[8] V. M. Orengo and C. R. Huyck. A stemming algorithmm for the portuguese language. In *spire*, volume 8, pages 186–193, 2001.

[9] M. F. Porter. Snowball: A language for stemming algorithms, 2001.

[10] E.-H. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. *Principles of Data Mining and Knowledge Discovery*, page 424–431, 2000. doi: 10.1007/3-540-45372-5_46.

[11] C. D. Manning, P. Raghavan, and H. Schütze. *6.2. Term frequency and weighting*, page 117–119. Cambridge University Press, 2018.

[12] C. C. Aggarwal. *Data classification: Algorithms and applications*. Taylor amp; Francis, 2014.

[13] A. Anand, G. Pugalenthi, and P. Suganthan. Predicting protein structural class by svm with class-wise optimized features and decision probabilities. *Journal of theoretical biology*, 253(2):375–380, 2008.

[14] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, 2000.

[15] M. Hossin and M. Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

[16] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[17] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[18] ANNOY library. `https://github.com/spotify/annoy`.

[19] P. Bafna, D. Pramod, and A. Vaidya. Document clustering: Tf-idf approach. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 61–66. IEEE, 2016.

[20] Lista de operações aprovadas: Portugal 2020. `https://www.portugal2020.pt/content/lista-de-operacoes-aprovadas`, .

[21] Projetos pdr2020. `http://www.pdr-2020.pt/Projetos-PDR2020`, .

Appendix A

Data Information

The detailed information regarding the division of the dataset in a hierarchical structure is provided in the section below.

A.1 Hierarchical Data

Table A.1, A.2, A.3 and A.4 show all classes considered inside PDR 2020 hierarchy for each level of the hierarchy.

Table A.1: Areas of PDR 2020.

	Area	Nº Projects
1	Innovation and Knowledge	1275
2	Competitiveness and Production Organization	18520
3	Environment, Resource Efficiency and Climate	3269
4	Local development	5212
AT	Technical Assistance	593

Table A.2: Measures of PDR 2020.

	Measure	Nº Projects
1	Innovation	942
2	Knowledge	333
3	Valorization of Agricultural Production	15578
4	Valorization of Forest Resources	160
5	Production Organization	12
6	Risk Management and Restoration of Productive Potential	2770
7	Agriculture and Natural Resources	784
8	Protection and Rehabilitation of Forest Stands	2485
10	LEADER	5212
20	Technical Assistance	593

Table A.3: Actions of PDR 2020.

	Action	**Nº Projects**
1.0.	Operational Groups	942
2.1.	Training and Dissemination	52
2.2.	Counseling	281
3.1.	Young Farmers	4773
3.2.	Investment in Farming	9995
3.3.	Investments in the Transformation and Marketing ...	564
3.4.	Collective Infrastructures	246
4.0.	Valorization of Forest Resources	160
5.1.	Creation of Producer Groups and Organizations	8
5.2.	Interprofessional Organizations	4
6.2.	Prevention and Restoration of Productive Potential	2770
7.8.	Genetic Resources	204
7.11.	Non-Productive Investments	580
8.1.	Sustainable Silviculture	2385
8.2.	Management of Cynegetic and Aquaculture Resources	100
10.1.	Preparatory Support	54
10.2.	Implementation of Strategies	4941
10.3.	GAL Cooperation Activities	163
10.4.	Operation and Animation	54
20.1.	PDR2020 Technical Assistance	52
20.2.	National Rural Network	348
20.3.	Local Support Structures	193

Table A.4: Operations of PDR 2020.

	Operation	Nº Projects
1.0.1	Operational Groups	942
2.1.4	Information Actions	52
2.2.1	Support for the Provision of Agricultural ...	43
2.2.2	Supporting the Creation of Counseling Services	236
2.2.3	Support for the Training of Counselors ...	2
3.1.1	Young Farmers	3988
3.1.2	Investment of young farmers in agricultural exploration	785
3.2.1	Investment in Farming	5746
3.2.2	Small Investments in Farming	4249
3.3.1	Investments in the Transformation and Marketing...	464
3.3.2	Small Investments in the Transformation and Marketing...	100
3.4.1	Efficient Irrigation Development	8
3.4.2	Improved Efficiency of Existing Irrigation	235
3.4.3	Drainage and Land Structuring	3
4.0.1	Investments in Forest Products Identified as Agricultural..	14
4.0.2	Investments in Forest Products Not Identified as Agricultural..	146
5.1.1	Creation of Producer Groups and Organizations	8
5.2.1	Interprofessionals	4
6.2.1	Prevention of Natural Disasters and Disasters	33
6.2.2	Restoration of Productive Potential	2737
7.8.3	Conservation and Improvement of Animal Genetic Resources	178
7.8.4	Conservation and Improvement of Vegetable Genetic Resources	22
7.8.5	Conservation and Improvement of Forestry Genetic Resources	4
7.11.1	Non-Productive Investments	580
8.1.1	Afforestation of Agricultural and Non-Agricultural Land	150
8.1.2	Installation of Agroforestry Systems	33
8.1.3	Forest Prevention against Biotic and Abiotic Agents	357
8.1.4	Restoration of Forest Affected by Biotic and Abiotic Agents or...	530
8.1.5	Improving the Resilience and Environmental Value of Forests	851
8.1.6	Improving the Economic Value of Forests	464
8.2.1	Management of Cynegetic Resources	100
10.1.1	Preparation and Capacity Building, Training and Networking of GALs	54
10.2.1.1	Small Investments in Farming	3508
10.2.1.2	Small investments in processing and marketing	544
10.2.1.3	Diversification of exploration activities	364
10.2.1.6	Renovation of villages	331
10.2.1.4	Short chains & local markets	180
10.2.1.5	Promotion of local quality products	14
10.3	Interterritorial and Transactional Cooperation of ...	163
10.4.1	Running and Animation Costs	54
20.1	PDR2020 Technical Assistance	52
20.2	Technical assistance	28
20.2.2	LEADER	61
20.2.3	Dissemination and Knowledge	72
20.2.4	Innovation	187
20.3	Local Support Structures	193

The figures A.1, A.2 and A.3 represent the hierarchy for the problem. The blocks with a black line contouring them are the ones that have only one measure/action/operation and therefore don't require a model for classification in that level.

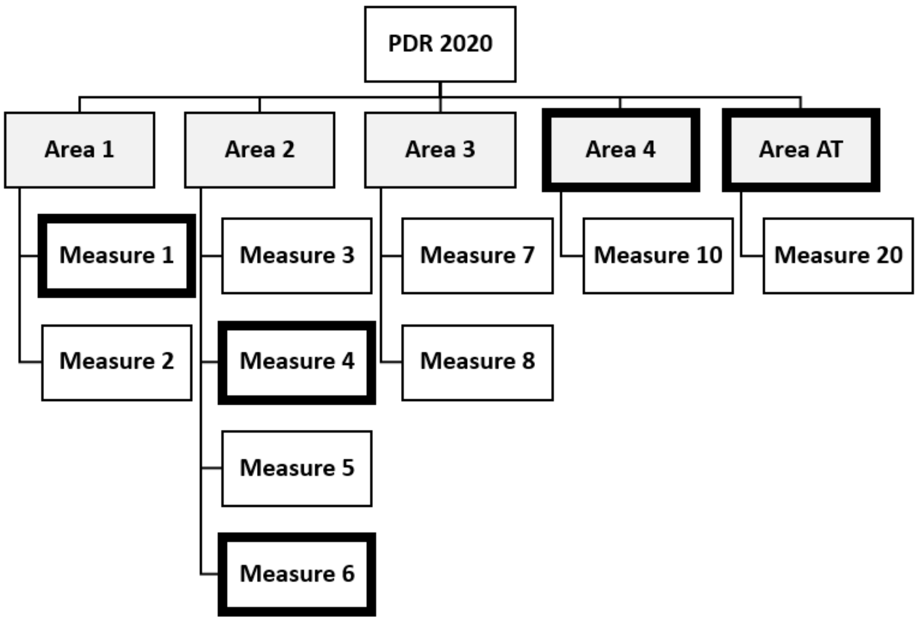


Figure A.1: Division of Areas into Measures.

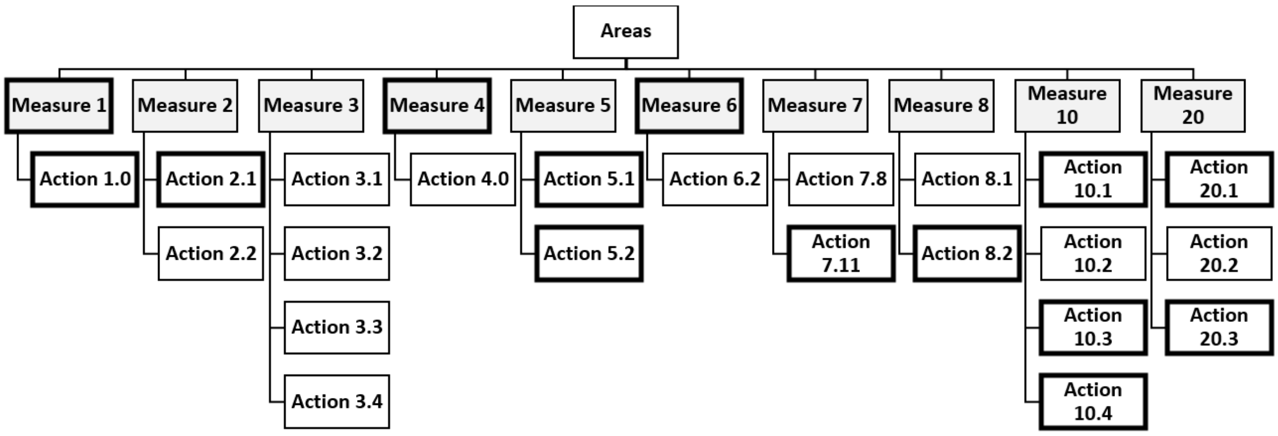


Figure A.2: Division of Measures into Actions.

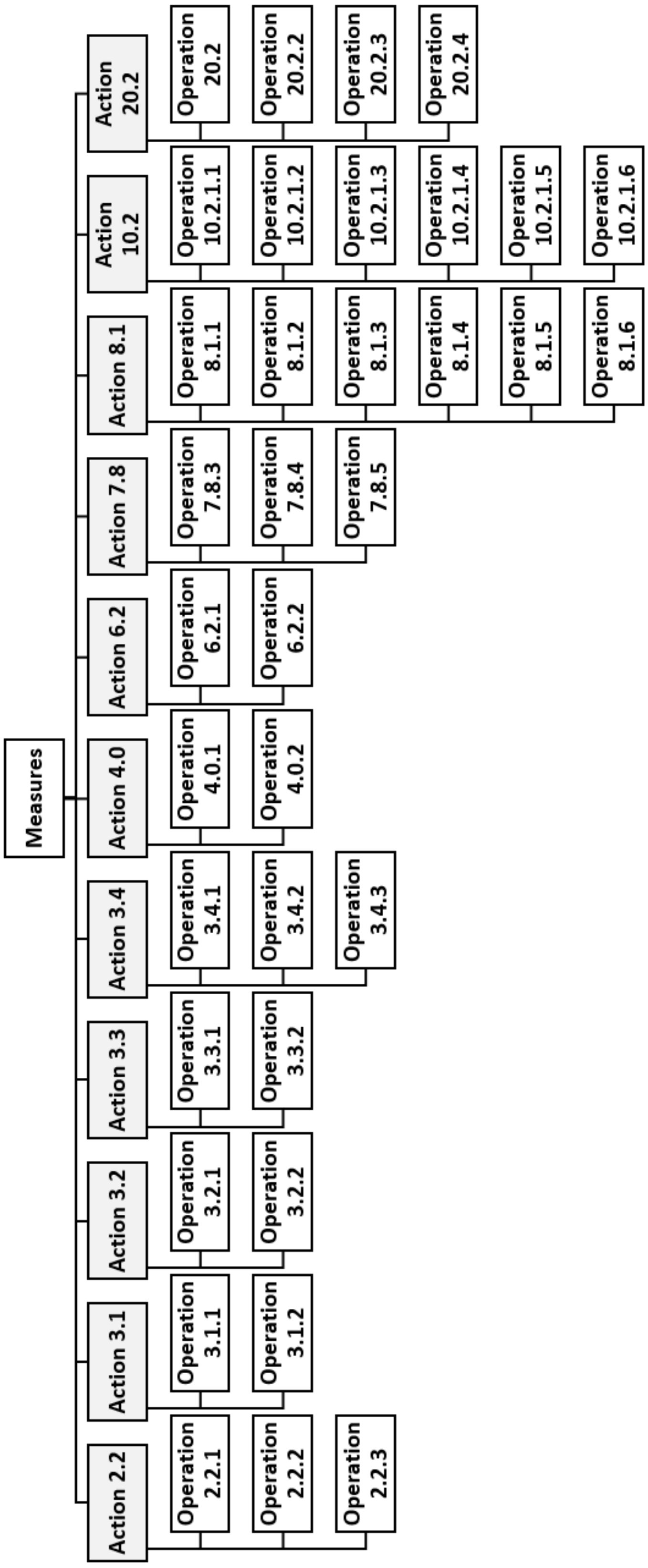


Figure A.3: Division of Actions into Operations.

Appendix B

Calibrated SVM Models

B.1 Results of Calibrated SVM models

The results from calibrated SVM models built for the analysis of the output and associated probabilities are shown in this chapter.

B.1.1 Results of Hierarchical Classification

Table B.1: Results of Calibrated SVM classifiers: Hierarchical Classification:.

	Classification of:			
	Areas	Measures	Actions	Operations
Accuracy	0.992	0.989	0.826	0.756
F1 Score	0.992	0.989	0.808	0.738
MCC	0.980	0.983	0.788	0.729

B.1.2 Results of Individual Classifiers

Table B.2: Results of Calibrated SVM Classifiers - Part 1.

	Area Classifier	Areas			Measures						
		1	2	3	2	3	5	7	8	10	20
Accuracy	0.993	1.0	0.993	0.997	1.0	0.696	0.667	1.0	1.0	0.996	1.0
F1 Score	0.993	1.0	0.992	0.997	1.0	0.657	0.800	1.0	1.0	0.996	1.0
MCC	0.982	1.0	0.972	0.991	1.0	0.328	0.000	1.0	1.0	0.956	1.0

Table B.3: Results of Calibrated SVM Classifiers - Part 2.

	Actions										
	2.2	3.1	3.2	3.3	3.4	4.0	6.2	7.8	8.1	10.2	20.2
Accuracy	0.961	0.829	0.863	0.883	0.935	1.0	1.0	1.0	0.956	0.981	0.984
F1 Score	0.952	0.751	0.862	0.870	0.924	1.0	1.0	1.0	0.955	0.981	0.984
MCC	0.914	0.000	0.719	0.198	-0.024	1.0	1.0	1.0	0.942	0.960	0.974

B.2 Evaluation of thresholds

The information analysed to find the best threshold for the models selected is shown here.

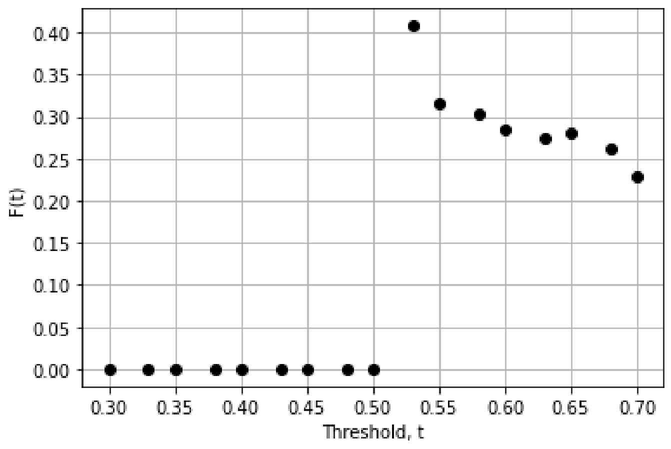


Figure B.1: Threshold heuristics problem (classifier in Area 2).

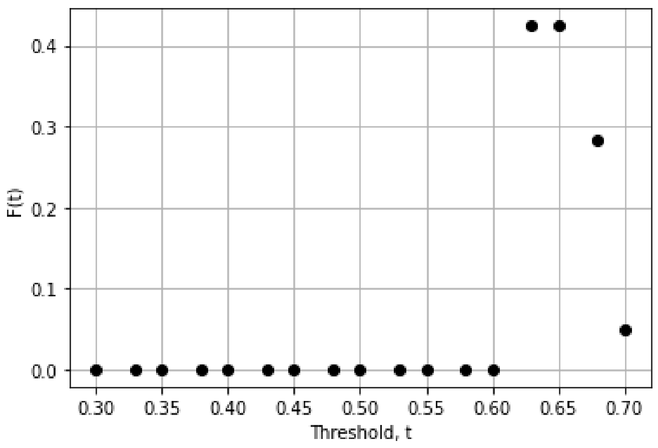


Figure B.2: Threshold heuristics problem (classifier in Area 3).

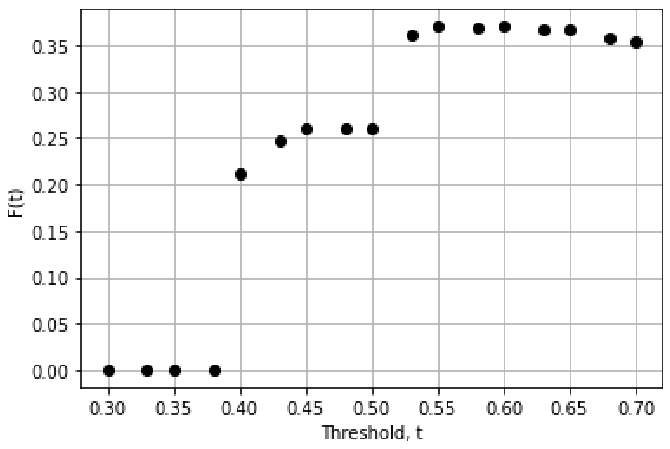


Figure B.3: Threshold heuristics problem (classifier in Measure 3).

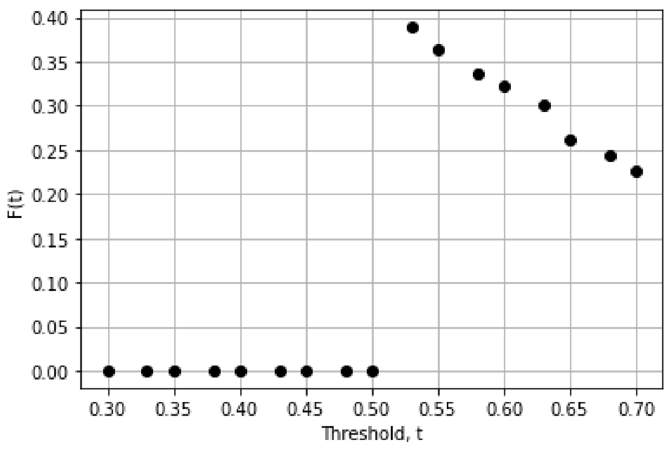


Figure B.4: Threshold heuristics problem (classifier in Action 3.2.

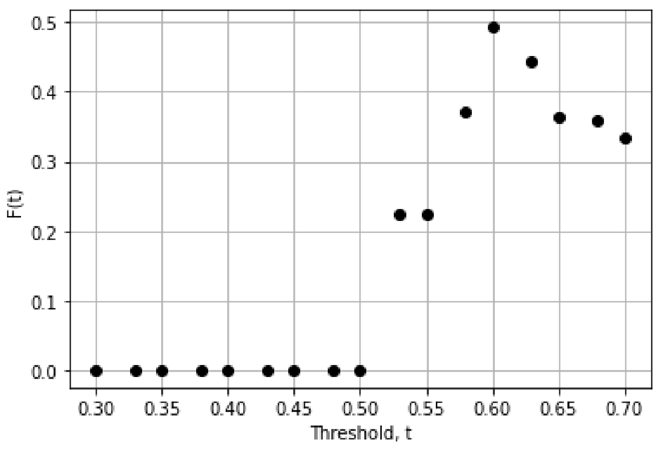


Figure B.5: Threshold heuristics problem (classifier in Action 3.3).

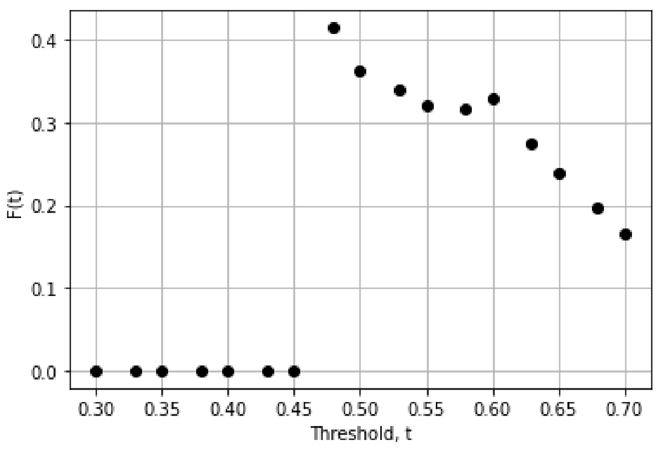


Figure B.6: Threshold heuristics problem (classifier in Action 8.1).

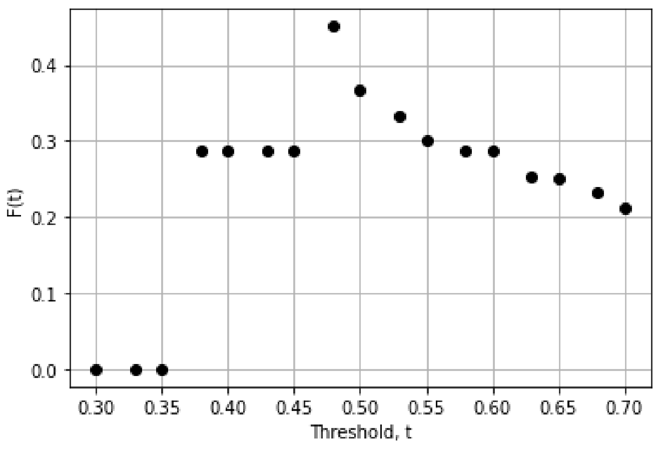


Figure B.7: Threshold heuristics problem (classifier in Action 10.2).