Intelligent Funds Assistant

Inês Sáragga Leal Saraiva
ines.saraiva@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

December 2021

Abstract

Currently, Portugal benefits from financial support given by the European Union, however it has been verified that these funds are not being fully used. One of the main setbacks in the EU funding application system is the identification of the best call for proposal. This work aims to develop a funds assistant to match a call from a written text, using Natural Language Processing techniques for text processing and Machine Learning to facilitate the construction of the models. This problem was addressed as a 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. In this work, it was observed that SVM outperforms the rest of the classifiers with a few exceptions. In addition, the results from calibrated SVM classifiers, considering a second prediction when the models were uncertain about their 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

1. Introduction

Currently, the European Union has a supporting economic development plan in place to assist each country in improving certain areas of their economy. Portugal benefits from the financial support given to this plan, a total of 25.9 billion Euros for the period 2014-2020 [1]. However, in Portugal, only 60% of the budget available per year is actually spent, making it essential to comprehend why these funds are not fully used. To approach this problem, it was decided to use the Portuguese operational programme (OP), PDR 2020 (Programa de Desenvolvimento Rural - Rural Development Program) as a study case, which is the main support of agriculture projects in the country. Despite the fact that the data available is related to the funding of 2014-2020, the same structure of the funds will be maintained in the following years, with some possible minor changes.

One of the main setbacks in the current EU funding application system is the identification of the best call (or notice) to answer for each applicant. A call is a funding opportunity that is inserted in a particular topic and area of action, and given that different calls have distinct benefits, its identification can be quite important. The funds assistant is expected to assist beneficiaries by matching a call from a written text, hoping to improve the overall experience of the community funds system. However, on account of the data available, it was only possible to associate a project with an operation, which consists of the last level of the hierarchy in PDR 2020, with specific scopes and objectives. The identification of the environment that a project should be implemented is still pertinent to this problem and can be later be associated with the temporary calls that PDR 2020 opens. To solve problems such as this one, involving text information, it can be very advantageous to explore the applications of Natural Language Processing (NLP), involving machine learning (ML) algorithms to facilitate the construction of this type of model. For this work, the use of text classification stood out, where the input would be a description of the project that the beneficiary wishes to develop and the output would be the best fitting position inside PDR 2020.

For this thesis, it was necessary to implement a hierarchical multi-class text classification, as seen in [2], to organize the data and classifiers, taking advantage of the organization inside PDR 2020. Four models were chosen to compare the classification performance: Naive Bayes as a baseline solution, as seen in [3] and [4], Support Vector Machines implemented in [2] with a hierarchical approach, Random Forest and k-Nearest Neighbors. The numeri-

cal representation selected was the TF-IDF vectorization, also present in [5] that deals with a similar problem. An analysis of calibrated SVM classifiers' outputs and associated probabilities [6] was accomplished for the consideration of second predictions. The performance of the twenty-two models that constitute the whole classification process was analyzed with accuracy, F_1 score and Matthews Correlation Coefficient (MCC).

Following this introduction, the most important parts of the theoretical background researched are explained, such as the steps of text classification and the hierarchical approach followed. Next, the proposed framework is detailed, in terms of the implementation of a hierarchical approach to this problem, the justification of the models and metrics selected to evaluate, and other pertinent information. After that, an analysis of the hierarchical classification and the individual classifiers is provided, alongside the analysis of probabilities with second predictions and with the results from those. Last, a conclusion of this thesis and some ideas for future work are presented.

2. Text Classification

As stated above, the goal of this work is to correctly predict an operation for a description made by a beneficiary. In short, the data used for the text classification model is a text description of the project (text input i) and a corresponding position inside the OP (class j of input i).

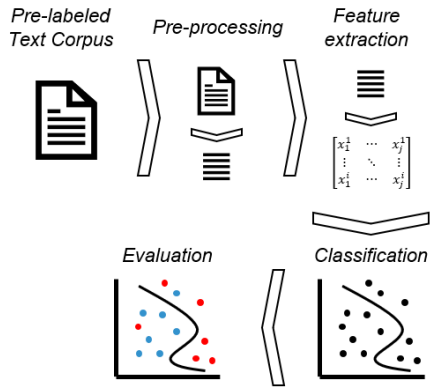

Figure 1: Steps of Text Classification.

It's crucial to understand the pipeline for text classification to be able to construct the best classification model possible. The main steps for a text classification problem are shown in Figure 1 and described as follows.

1. Pre-processing is the phase where the text is clean of unnecessary information, with a set of labeled texts or documents.

2. Feature extraction consists of transforming the text into a numerical representation that can be used as input for a classification model.

3. Training of a classification model where connections are created between the vector representation of the text and the associated labels of the input.

4. Evaluation of the model is the last phase, where the predictions made are verified. If the predictions are correct, it can be concluded that the model was able to learn distinct features for each label.

2.1. Hierarchical Classification

In [2], a hierarchical structure is explored 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 would learn to distinguish a second-level class from all other second-level classes and usually, the model built would be a binary classifier, which means that a certain input may be predicted into none, one or more than one class. 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.

To better understand the flow of the classification, in a generic form, Figure 2 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 at a certain level.

2.2. Other information

As it was explained previously, text classification requires choosing pre-processing and feature extraction techniques, models to build and metrics to evaluate them with. All of these elements were studied and can be consulted in the original document if necessary.

3. Proposed Framework

For the operational programme that was used as a study case, PDR 2020, there is a hierarchical division in terms of projects' scopes. In total, there are

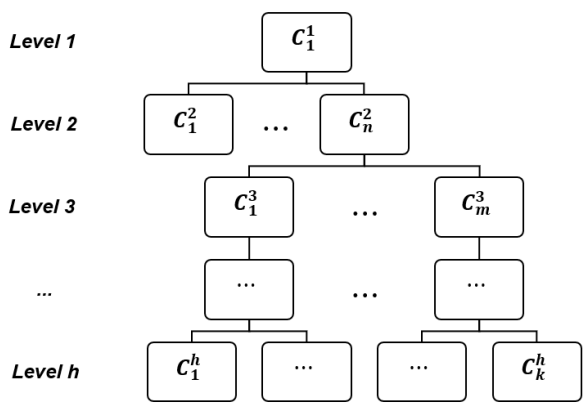

Figure 2: Generic classifier tree.

four levels of hierarchy, starting with 5 classes, in the first level, called **areas**; 10 classes in the second level, referred to as **measures**; 22 classes, in the third level, called **actions**; and 46 classes, in the last level, named **operations**. Relating this hierarchy with the literature found [2], the nomenclature shown in Figure 2 translates to the classifiers in Figures 3 and 4, with the following levels:

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

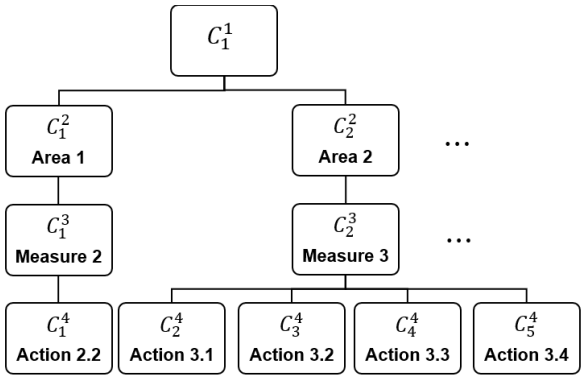

Figure 3: Classifier tree used for PDR2020 - Part 1.

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 4. 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 in terms of operations, for measure 1, no further classification is necessary since it has only one action and one operation. In terms of actions, 11 out of 22 were excluded from classification given that they have only one operation.

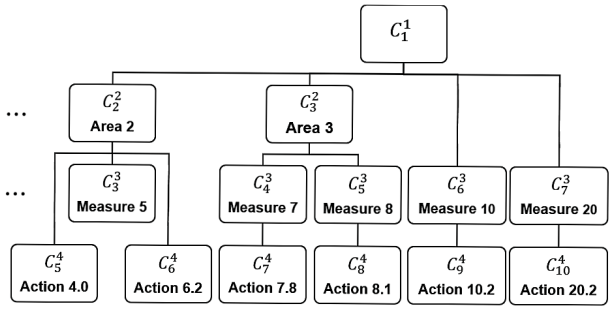

Figure 4: Classifier tree used for PDR2020 - Part 2.

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

3.1. Model Selection

To have the best performance possible for the funds assistant being developed, four classifiers were chosen to be tested and compared against each other. The **Naive Bayes** (NB) 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 [3] and [4].

Support Vector Machine (SVM) are one of the most widely used classification algorithms nowadays, known for their high performance, especially in text classification. In [2], a 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 have very good overall results. Given the similarity between the classification in the article and the one that was constructed here, the choice of testing an SVM algorithm is obvious.

Another basic classifier mentioned frequently in the literature is the Decision Tree (DT) model. Compared to the NB classifier, the Decision Tree

algorithm usually shows a worse performance, as it can be seen in [3]. Despite the poor performance of DT algorithms for this type of problem, the **Random Forest** (RF) classifier, an ensemble model that uses decision trees as base classifiers, can overcome the limitations of the DT classifier and produce accurate results. An advantage of this algorithm, over SVM, is that it can be used directly in multi-class classification problems.

Finally, the **k-Nearest Neighbors** (KNN) 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 neighbors search. This optimization problem has other variants, such as the approximate nearest neighbors, which is used in a novel model, Annoy, 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 Annoy as a text similarity search algorithm.

3.2. Metric Selection

The models chosen are going to be compared with 3 metrics: accuracy, F_1 measure 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 and is once again a widely used 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. Due to the unbalanced dataset for this problem, 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** (MCC). 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.

3.3. 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. A decision threshold is a probability below which it is possible to infer that the classifier loses confidence in its prediction and is therefore prompted to consider a second one, as shown in Figure 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 evaluated, it's also verified if the second prediction is correct or not, deciding which prediction to consider based on that.

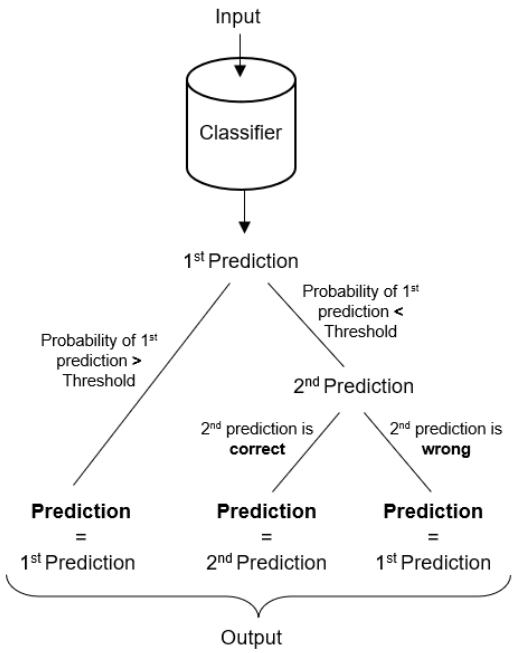

Figure 5: Evaluation pipeline for classifiers with 2nd prediction.

At this point, 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 [6], 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 SVM). 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, such as Platt's scaling or Isotonic Regression.

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

To choose the best threshold possible considering

all these factors, a heuristic technique was used considering an objective function, shown in (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 if the model is considering second predictions or not, $F(t)$ is forced to be zero if no second predictions are made ($R = 0$).

$$F(t) = \omega_\tau \cdot \tau(t) + \omega_\rho \cdot \rho(t) + \omega_\phi \cdot \phi(t) \quad (1)$$

Regarding the variables shown in (1), they 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 the percentages of correct second predictions retrieved, as seen in (2). This variable has a weight of ω_τ to be defined.

$$\tau(t) = \frac{T_t}{\sum_i T_i} \quad (2)$$

- $\rho(t)$ 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). This variable has a weight of ω_ρ to be defined.

$$\rho(t) = \frac{R_t}{\sum_i R_i} \quad (3)$$

- $\phi(t)$ 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 sum of all MCC scores, as seen in (4). This variable has a weight of ω_ϕ to be defined.

$$\phi(t) = \frac{\mathrm{Acc}_t}{\sum_i \mathrm{Acc}_i} + \frac{\mathrm{MCC}_t}{\sum_i \mathrm{MCC}_i} \quad (4)$$

3.4. Text Processing

The first step of any text related algorithm is text cleaning and pre-processing. First, for the text provided, all the numbers contained in the text were removed. Besides that, punctuation was removed and all letters were lowered. Given that the Portuguese language has accents, also known as diacritics, such as ´, ‘, ˆ , ˜ and ç, it's important to understand its influence in the results to decide whether they should be kept or normalized (removed). By removing accentuation from all the text, it's possible that the algorithm could learn better representations discarding spelling errors. It was concluded that, by normalizing the text, the accuracy and the MCC score increase and the running time of the model diminishes.

After cleaning the text, tokens are created by splitting the text into different words. Stopwords are words that bring no important information to a text such as propositions and determinants and those were promptly removed. 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 [7] and RSLP stemmer [8]. The Snowball stemmer was chosen for this work given the higher accuracy and quicker processing time that it showed.

The following step of text processing is vectorization to create features for the models. The chosen method for this work was the TF-IDF weighting scheme. An example of the use of TF-IDF for a similar problem can be seen in [5], that tackles document clustering with a hierarchical approach (hierarchical agglomerative clustering). Given the fact that the results achieved in [5] show the efficiency of this vectorization method, it was decided to test this representation in this problem, hoping to also obtain high performance. As for the definition of TF-IDF, it stands for Term Frequency - Inverse Document Frequency, and it's a technique for the representation of words where each word is weighed in regard to its importance to a document (title and description of a project) contained in a corpus (the texts available from all the projects). The result of this vectorization is 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.

3.5. Data Analysis

In regards to the data used, all public information was retrieved to have the most complete database possible. Overall, it was possible to gather information regarding 159 363 projects from 15 different Operational Programmes, having 29 005 projects from PDR 2020. In a brief analysis of the title and description of the data available, 91 projects that didn't provide text information in the title or description were found. It was also noted that some projects had text in English and some had information with no significance (for example, "aaa" and "ccccçççç"). For PDR 2020, 10 projects that had text with no significance were identified and removed. Concluding, the dataset used had a total of 28 904 samples.

Regarding the hierarchical information needed for the classification, it was observed that the information available shows only the operation of a project, such as "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") identify an action; and all the numbers to-

gether ("3.2.1") symbolize a specific operation. The first level of the hierarchy is not identifiable with this code, but that information is available on PDR 2020's website [9], making it possible to manually associate the measures with the five existent areas.

4. Classification Results

For the results retrieved, the data used was separated 80%-15%-5%. 80% of the dataset was designed for training and 15% was for the validation of each separate model (shown in section 4.3). The last 5% was used for the sequential testing phase (shown in section 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

For the parameter tuning of the models, each one was optimized with regard to the accuracy and used part of the training data for the first level of the hierarchy with a total of 2000 samples. In Table 1, the best parameters found are shown. For more details about this section, please consult the original document.

Model	**Parameters**	
NB	α	0.2
SVM	C	1.5
	Penalty	$l2$
	Loss	squared hinge
RF	Criterion	gini-index
	Max Depth	150
	Max Leaf Nodes	500
	Min Impurity Decrease	0
	Min Samples Leaf	1
	Nº estimators	75
KNN	Algorithm	Ball-Tree
	Leaf Size	20
	Metric	Euclidean
	Nº neighbors	2

Table 1: Parameters for NB, SVM, RF and KNN classifiers.

4.2. Analysis of Hierarchical Classification

A sequential process of classification was tested, using previous predictions to choose which model to use and retrieve the next prediction with. The data used to test in this phase was a 5% testing set, 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 4.2. 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. 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. These projects were identified as a different class for the first level and then restored their original labels for the rest of the classification.

		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

Table 2: Comparison of classification models using sequential results.

In Table 4.2, 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. While, in the first two levels, the k-Nearest Neighbors showed better accuracies and MCC scores 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 a 0.649 MCC score.

4.3. Analysis of Individual Classifiers

For the 22 classifications tested and analysed, the best results are shown in Tables 3 and 4. These 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;
- Naive Bayes classifier had the best performance in **1** classification;

	Area	Areas			Measures						
	Classifier	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 3: Results of Individual Classifiers - Part 1.

	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

Table 4: Results of Individual Classifiers - Part 2.

- Random Forest model had the best performance in **1** classification;

The classifiers, that are not SVM, are identifiable with a footnote, which states the model used for the results shown, in Tables 3 and 4.

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.

Three of those classifiers (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. Given that two of those models have a class with very few samples to train and test and the other has the problem related to the investment size accepted (normal vs small investments), some misclassifications can be expected in the evaluation. However, given both those issues, these classifiers obtained better results than expected.

The last classifiers to mention are the ones that had the worst performance (an MCC lower than 0.5), which sum up to a total of **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;
- Small number of training samples;
- Unbalanced dataset combined with few data available to train and test.

Six of the classifiers mentioned (in actions 2.2, 3.3, 3.4, 4.0, 6.2 and 7.8) have a common characteristic: one or more classes of the classification do not have much data to test, leaving just one class being tested. The class being tested can be correctly predicted, which justifies the high metrics found in some models (such as in actions 4.0 and 6.2), or it can misclassify some of the samples, lowering the evaluation metrics (actions 3.3 and 3.4).

4.4. Analysis of 2nd Predictions

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: Platt's Scaling and Isotonic Regression. Testing the two of them for the area classifier and evaluating the accuracy results, it was chosen to use the Isotonic Regression, which is the best method for larger datasets.

Previously, a heuristic strategy was defined for finding the best threshold in each classifier. The objective function $F(t)$, defined in (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 those weights defined, the objective function implemented is shown in (5).

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

Regarding the thresholds examined, all the models were tested with seventeen values 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 classifiers tested, only **8 models** 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.

For the eight classifiers chosen to analyse, the different parcels of the objective function and the value of $F(t)$ for the defined thresholds were calculated. With these values, it was possible to create a graphic representation of this problem, as the one shown in Figure 6 for the area classifier, and verify which threshold brings the highest value of $F(t)$. In conclusion, the chosen thresholds for the eight classifiers analysed are shown in Table 5.

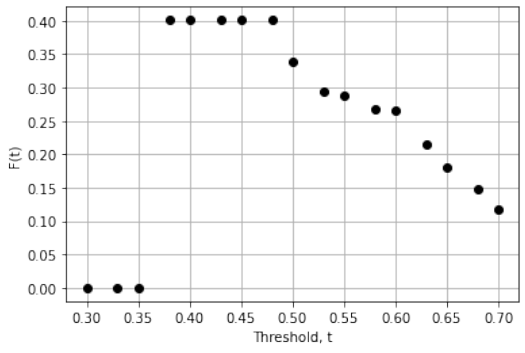

Figure 6: Example of the threshold evaluation (Area Classifier).

After defining the thresholds, the hierarchical classification is tested once again, comparing the results considering just one prediction and considering a second one. The metrics evaluated in all four levels of the hierarchy can be seen in Table 6.

Classifier		**Threshold**
Level 1		0.38
Level 2	Area 2	0.53
	Area 3	0.53
Level 3	Measure 3	0.6
Level 4	Action 3.2	0.53
	Action 3.3	0.60
	Action 8.1	0.48
	Action 10.2	0.48

Table 5: Chosen threshold for each classifier.

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

Table 6: Comparison between sequential results with 1^{st} and 2^{nd} prediction.

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 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 (only 0.04). 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 0.864 in MCC, a considerable enhancement from the first results.

5. Conclusions

The goal of this thesis consisted in developing an intelligent assistant to find the best call for a project with a written description of it, through the identification of the most fitting environment that the project should be inserted into.

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.

In terms of the classification results, out of the four models tested, 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.

By analysing each individual classifier defined, it was again verified that the SVM algorithm outperforms all others in a large majority of cases. Overall, out of 22 classifiers, only 5 of them had an MCC score lower than 0.5, while still showing high accuracies. The rest of the classifiers obtained high accuracies, ranging from 0.8 to 1.0, and also high MCC scores above 0.7.

To enhance the SVM models, which showed the most promising performance, a second prediction was considered when the model was uncertain about its first one. With this strategy, 8 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 **0.878** accuracy. The rest of the levels also showed slight increases, maintaining a **0.952** accuracy until the last level.

6. Future work

A lot of neural networks have been developed for text related tasks, such as BERT, Glove, FastText, among others. These could be explored as a vectorization process, taking advantage of the larger vocabulary corpus that the models have been pretrained on, or used with the existent models for text or sequence classification, such as "Bert For Sequence Classification".

References

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

[2] Susan Dumais and Hao 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.

[3] Eui-Hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results. *Principles of Data Mining and Knowledge Discovery*, page 424–431, 2000.

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

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

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

[7] Martin F Porter. Snowball: A language for stemming algorithms, 2001.

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

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