uc3m | Universidad Carlos III de Madrid

Dual Bachelor in Data Science and Telecommunications Technologies Engineering 2020-2025

Final Project

Machine Learning Applications

Alejo González García (100454351)

Alonso Madroñal de Mesa (100454449)

Daniel Toribio Bruna (100454242)

Andrés Navarro Pedregal (100451730)

Table of Contents

1. Introduction

This project consists of a multiclass classification of previously preprocessed poems, that will be assigned to a period. In order to carry out this task, we make use of the tools explained in class regarding Natural Language Processing and Machine learning tools, such as feature extraction and selection.

2. Task 1: Text Preprocessing and vectorization

2.1. The dataset

In order to start the project, we first needed to obtain the dataset to work with. We could either download one available online or create it ourselves. We opted for the second option, and we obtained the information from the web Poetry Foundation whose poems are licence-free for educational uses.

For the creation we used a library called *request* to obtain the response of the GET request and then we obtained the JSON response where we have extracted information like author, title, link to the poem, tags and snippet. With the link to the poem and using BeautifulSoup we get the HTML information. Poems were mainly in two formats, text or images. For the first one we just extracted the text that was inside the div with class 'o-poem', and for the second one, we used an OCR, in this case, *pytesseract*, to extract the text from the image that was in the div with class ‘c-assetStack-media’.

This process was slow. Applying the OCR to thousands of images and sending GET requests to gather information of more than 50.000 of documents, at least in the way that we did it, took a lot of time. As we could not afford to keep running the computer for one day, we gathered the data in small segments, and at the end we jointed all the datasets together. We just had to store the id of the last gathered poem and the next day start the scrapping beginning with the next Id.

The reason for this delay, was the way in which the page was created, it took lot of time, not due to our internet connection or computational power, but due to the fact that the server hosting the web page was answering our requests slowly.

In the web Poetry Foundation only 5535 poems from the 47388 total poems have a period assigned, so we will have two different datasets, one composed with all the poems that will be used for the topic modelling task, see Figure 1, and another with only the poems that have a period and they can be used for the classification task, see Figure 2.

The resulting dataset contains the following information:

- An id
- The title of the poem
- The author
- A snippet of the poem
- The link to where the poem is (text or image)
- The categories that the poem has
- The period of the poem
- The text of the poem

	id	title	author	snippet	link	categories	poem
0	162275	The 80’s Miracle Diet	By Melvin Dixon	Yours free without the asking	https://www.poetryfoundation.org/poetrymagazin...	['Living', 'Health & Illness']	Yours free without the asking Quick delivery v...
1	162250	All Saints’	By Corey Van Landingham	CaravaggioÃ¢â‚¬â„¢s face in the sunken pumpkin.	https://www.poetryfoundation.org/poetrymagazin...	[]	CaravaggioÃ¢â‚¬â„¢s face in the sunken pumpkin...
2	162279	And These Are Just a Few ...	By Melvin Dixon	This poem is for the epidemic dead and the liv...	https://www.poetryfoundation.org/poetrymagazin...	['Living', 'Health & Illness', 'Social Comment...	This poem is for the epidemic dead and the liv...
3	162248	ASMR	By Corey Van Landingham	Why not climb up the mountain	https://www.poetryfoundation.org/poetrymagazin...	[]	Why not climb up the mountain of delight? To t...
4	162249	ASMR	By Corey Van Landingham	HelloÃ¢â‚¬â€Tonight	https://www.poetryfoundation.org/poetrymagazin...	[]	HelloÃ¢â‚¬â€Tonight weÃ¢â‚¬â„¢ll trace the st...

Figure 1. Sample of the dataset obtained without periods

	id	title	author	snippet	link	categories	period	poem
0	43926	The Canterbury Tales: General Prologue	By Geoffrey Chaucer	Whan that Aprille with his shoure</str...	https://www.poetryfoundation.org/poems/43926/t...	['The Body', 'The Mind', 'Love', 'Activities',...	Middle English	Here bygynneth the Book of the tales of Caunte...
1	44295	Confessio Amantis, Book III: The Tale of Apoll...	By John Gower	Appolinus his levE tok,	https://www.poetryfoundation.org/poems/44295/c...	[]	Middle English	Appolinus his leve tok, To God and al the lond...
2	159137	Ego Dormio: [All perishes and passes]	By Richard Rolle	[Alle perisches and passes pat we with eghe see]	https://www.poetryfoundation.org/poems/159137/...	['Love', 'Classic Love', 'Religion', 'Christia...	Middle English	[Alle perisches and passes pat we with eghe se...

Figure 2. Sample of the dataset obtained with periods

2.2. Preprocessing

The first thing we are going to do is to remove missing values which means that poem text was missing. This happens because some of the images of the poems were not displayed because of the web. Then we removed the columns that do not give any useful information: id, snippet, and link.

During the creation of the datasets, we noted that some poems were not in English, but they were translated and published in the web, so we have “duplicated” poems in their original language and in English. To separate the English poems from the others we have used detect from the library *langdetect*. It is important to know that language detection algorithm is non-

deterministic, which means that if you try to run it on a text which is either too short or too ambiguous, you might get different results everytime you run it. To avoid this, we have used a seed. At the end we have obtained that 420 poems are not in English, and as they are translated in the dataset, we can remove them.

After we extracted the poems from the web, we saw that some of the characters were not properly encoded, so we fixed the characters that are important, for example the apostrophe, to be able to expand the contractions with the method fix from library contractions. Then we tokenize the text by words, convert the tokens into lower case and filter non alphabetic tokens. For the homogenization we have chosen lemmatization to keep the semantic meaning and have a better interpretability of the words. Lastly, we have removed the stop words. In addition, we applied N-gram detection so that the words that appear very often together are jointly categorized and we can see the joint words when later doing Topic Modelling.

Also mention that, initially, we were removing just the alphanumeric characters and we ended up removing all non-alphabetic characters. The reason why we took this decision was that at the end, while doing the Topic Modelling, there was a topic full of numbers. We are analysing poems and we detected that most of this numbers were obtained inevitable while doing web scrapping as the indexes, page numbers and dates where mixed and collected. No information gain is obtained if we keep the numbers, so we just removed them.

2.3. Text vectorization

Once all the poems are preprocessed and cleaned, the next step is to transform them into a numerical representation that can be used as input for the learning algorithms. We decided to use the following vectorizations:

- Classical BoW or TF-IDF representation.
- Word2vec representation.
- FastText.
- Extraction of themes and vector representation of the documents using the LDA algorithm

After obtaining all the different vectorizations, the performance of each one of them with several classification methods will be compared. Then the pair of vectorization and classifier with the best results will be later analysed in order to extract even better results from it.

Prior to beginning with the vectorization step, we have to prepare the dictionary. After analysing the most frequent words in the data set, we can see that there are several words that appear much more times than the others. To better visualize it we designed the following graph:

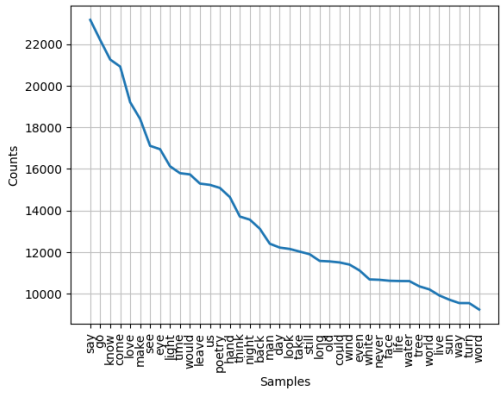

Figure 3. Most frequent tokens in the initial corpus

Also, similar graphs can be obtained in order to check the distribution of the words among the different documents of the dictionary. As it has been seen in class, neither the words that appear in all the documents nor the ones that hardly appear in any one are relevant for the algorithm. Thus, it will be useful to have an estimation of this distribution to manually select the discriminant conditions for the removal of both tails of the following histograms.

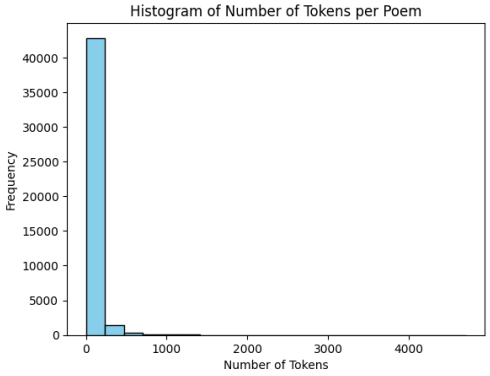

Figure 4. Distribution of the number of tokens per poem

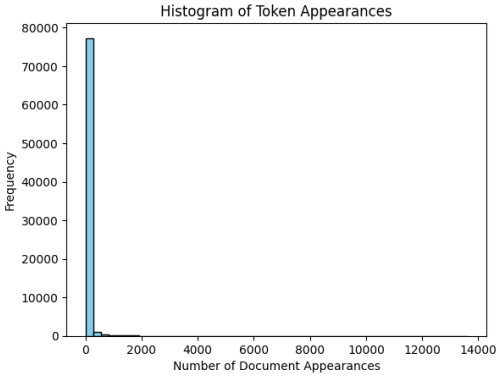

Figure 5. Distribution of the token appearances in the poems

After analysing the distributions, the creation of the final dictionary was done. To filter these types of words, it was decided to remove those that did not appear in more than 10 poems and those that appeared in more than 75% of them. After removing these tokens, we have a dictionary of 22567 terms. We are now ready to vectorize our corpus of poems.

BOW and TF-IDF

BOW was implemented with doc2bow method and TF-IDF with TfidfModel from gensim. After obtaining both vectorizations, we plotted the representations of a given poem to compare them. As seen in Figures 6 and 7, while almost all tokens have the same value in the BoW representation, from the TF-IDF it can be seen that “leave” is a common word within the corpus, because its representation is now lower and “mystery” is relevant in this specific document, because its representation has increased a lot.

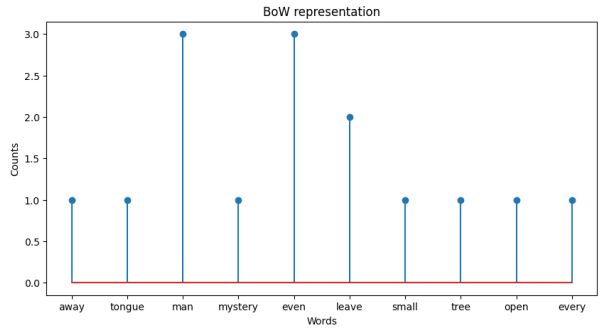

Figure 6. BoW representation of a poem

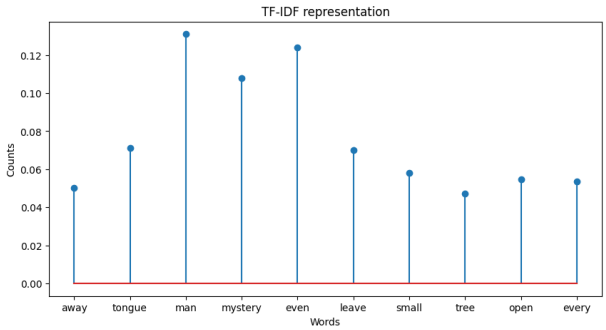

Figure 7. TF-IDF representation of a poem

Word2Vec and FastText

Word2Vec was implemented with Word2Vec from genism and FastText with FastText from genism. For both, we are using vector_size=100, window=5, min_count=5. The vector size is just 100 because in this way the classification models require less time to train. After obtaining both vectorizations, we search for the most similar words to “love”. The results are displayed in the following table:

Word2Vec	FastText
tenderness	unlove
beauty	loveth
hate	lover
lover	love_hate
true	lover_lover

Table 7. Most similar words to “love”

We can see that both gives good results, but **FastText gives more explicit similar words**.

Topic modeling using the LDA algorithm

We have implemented LDA for topic modelling with genism. In order to find the optimal number of topics, we will use the coherence metric “c_v” to evaluate different number of topics, in this case, 5, 10, 15, 20, 25. The results are displayed in the following image:

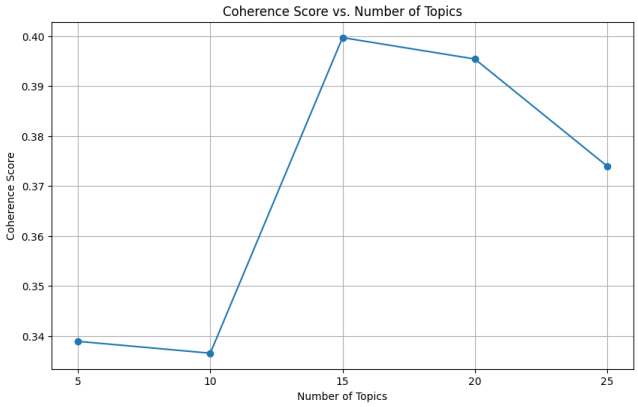

Figure 8. Coherence scores for each number of topics

We can see that the best coherence score is obtained for 15 topics. We have made the WordClouds of these topics to visualize the words that characterize them:

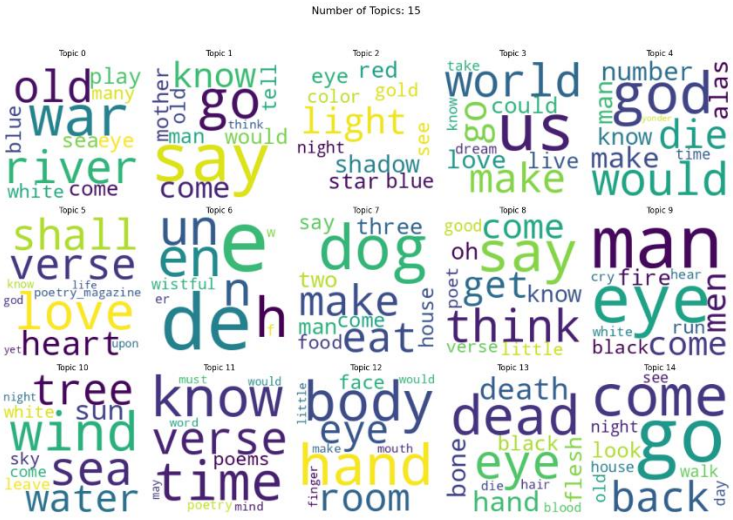

Figure 9. WordClouds of the topics

We can interpret some of the topics, for instance, topic 10 is about nature and topic 13 about death. The topic 6 is containing some words that are not meaningful due to the encoding issues.

3. Task 2: Classification

Our new objective is to classify the poems according to the period when they were written. For this task we have used the embeddings obtained in the text vectorization part as mentioned previously. We have considered random forest, support vector machine and K-nearest neighbours.

Before training the models, we have carried out a small exploratory data analysis of the target variable. The following figure summarises this:

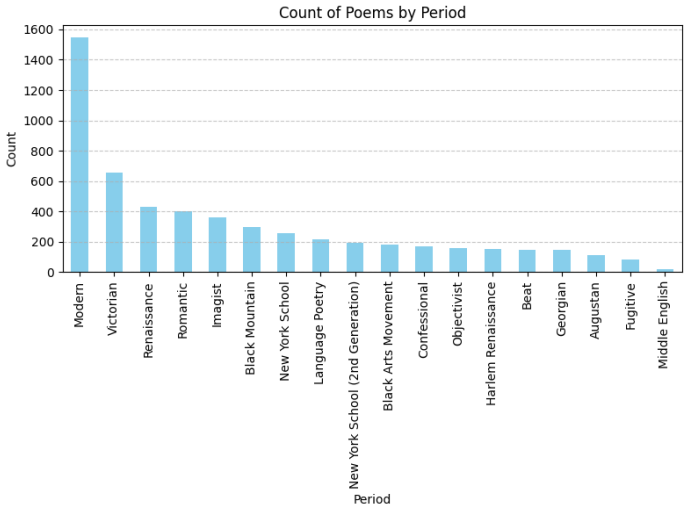

Figure 10. Count of poems in each period

We can see that our dataset is very imbalanced. The poems with period Modern are the 28% of the data, followed by Victorian with 11.8%. As we have a small number of observations, we cannot make under-sampling, so we have considered to keep all poems and use as a benchmark for the accuracy the 28% of the majority class.

In order to select the best pair of text vectorization and classification algorithm, we took advantage of the Pipeline and GridSearchCV functions of sklearn. Each one of the algorithms has its own characteristics and parameters to be tuned, so in the following the chosen implementation of each one of them will be briefly explained. We have also to divide the train and test poems with train_test_split from sklearn.

Random Forest

The first algorithm to be tested will be Random Forest, using the function RandomForestClassifier implemented in sklearn. In order to try to extract the best of it, the RF algorithm (and so will be done with the rest of the algorithms) was tested under different parameters, obtaining the optimum one by means of a grid search. This grid contains two variations for the 'criterion' of the RF classifier (gini and entropy) and tests different values for the 'max_depht' of the forests ranging from 5 to 35 with a step of 10.

Support Vector Machine

The next algorithm used was the support vector machine classifier. This was implemented by means of the SVC function of sklearn. The chosen SVM algorithm was the RBF non-linear Kernel version, as there is hardly expected to be any linearity among the data, and two different values are included in the grid search for the parameter C (the regularization parameter): 1 and 10. For SVM we used the sparse representations of the data to be more computationally efficient.

K-Nearest Neighbours

In the last place, the KNN Classifier will be applied. For the selection of the optimal parameters of the classifier, a grid search is used that will compute this classification for values for the number of neighbours ('n_neighbors') ranging from 5 to 40 and using two different metrics for the weights ('weights'): uniform and distance.

With the embeddings of Word2Vec and FastText, we had to compute the mean to have uniform lengths and be able to skip errors in the training of the models. We will also consider a pre-trained embedding, en_core_web_md, which are obtained with a model trained in a huge corpus.

As a summary of the performance of each pair we have the following table:

	BoW	TF-IDF	Word2Vec	FastText	Pre-trained
RF	0.37	0.38	0.39	0.38	0.45
SVM	0.51	0.47	0.35	0.35	0.54
KNN	0.24	0.46	0.37	0.36	0.44

Figure 11. Accuracies of the text vectorizations with classification models

We can see that the best accuracy is obtained with the pre-trained embeddings and the SVM model. This could happen as our embeddings are obtained in a relatively small dataset and the words are better represented in the pre-trained embeddings.

Feature extraction and selection

As the best accuracy was obtained with pre-trained embeddings and we saw that they were of a length of 300, we decided to explore the consequences of reducing the dimensionality of this embeddings to the half and see if the same information could be explained in fewer dimensions and we could reduce the computational cost. To achieve this, we will use the ideas of the feature extraction and selection, to see use the most important variables after the dimension reduction. We have tried PCA, CCA and LDA for feature extraction and mutual information for feature selection. All the techniques are implemented with sklearn functions. The accuracies are displayed in the following table:

	PCA	CCA	LDA
Mutual information	0.51	0.52	0.53

Figure 12. Accuracies after feature extraction and selection

We can see that PCA gives the lowest accuracy since it is an unsupervised algorithm, good for dimensionality reduction, but not as good as CCA or LDA, if we want a supervised approach, like we have seen in class. We have only considered mutual information because is computationally efficient compared to the other ones seen in class, for example, HSIC.

We can also see that the accuracy was not improved, and it has decreased 0.01 so we must try another approach to obtain a higher accuracy. In the following image we can see the confusion matrix of the LDA reduction:

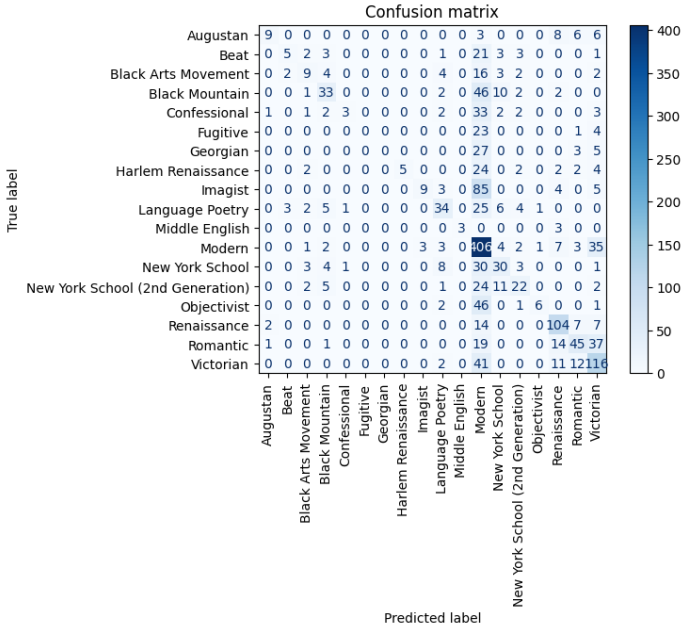

Figure 13. Confusion matrix of the LDA reduction

We can see that most of the predictions are associated with the principal class. As we cannot apply over-sampling, neither under-sampling, we have opted to group classes by their historical period following the mapping of the web. This mapping is:

Pre-1550	1550-1780	1781-1900	1901-1950	1951-Present
Middle English	**Augustan**	**Romantic**	**Fugitive**	**Beat**
	Renaissance	**Victorian**	**Georgian**	**Black Arts Movement**
			Harlem Renaissance	**Black Mountain**
			Imagist	**Confessional**
			Modern	**Language Poetry**
			Objectivist	**New York School**
				New York School (2nd Generation)

Figure 14. Mapping of the historical period with the art periods

Joining classes

By applying the previous mapping our distribution depending on the periods are:

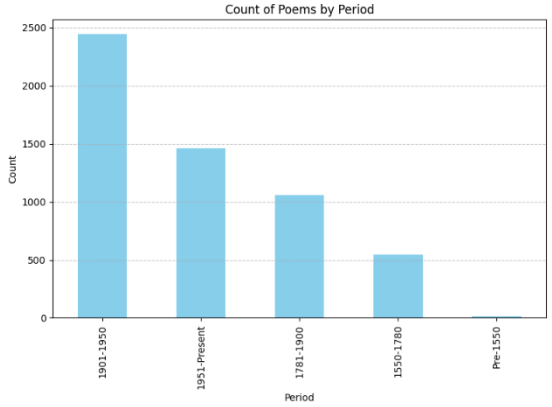

Figure 15.Count of poems in each period

We can see that now the dataset is less imbalanced with the following proportions:
1901-1950: 44.289593%
1951-Present: 26.443439%
1781-1900: 19.131222%
1550-1780: 9.828054%
Pre-1550: 0.307692%

After training the model that gave us the best accuracy (pre-trained embeddings with SVM), we have obtained an accuracy of 0.79 and the following confusion matrix:

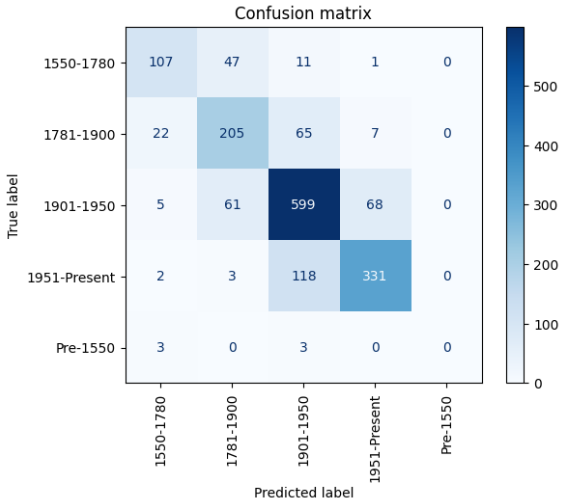

Figure 16. Confusion matrix after joining classes

4. Dashboard

For this part we have managed to create an interactive web page, a Dashboard, that you can check if you run the *"Dashboard.ipynb"* file and open the link: *http://localhost:8050/*

On this board we have implemented a menu where you can select the poem number that you want to inspect, and the Dash will return the content of that poem together with the Predominant LDA Topic associated with that poem. This topic will be presented in the word cloud format. In this way you can read any poem you want and see the topic that mostly characterizes that poem.

Next, we have introduced a Topic Selector, so that you can freely inspect the topic that you want and see the associated word cloud to that topic. It has been really useful for us, as it allows us to check the correctness of the decisions that we took during the preprocessing of the data set.

Then a Topic Distribution is shown, related to the number of topics that are predominant in the documents. We have concluded that only a few of the Topics (4-7) are important in most of the topics and the others are rare and do not really matter in our poems.

5. Conclusions

At the end, for the initial problem we the original periods, we have obtained an accuracy of 0.54, that considering that our benchmark was 0.28, is a good accuracy. The accuracy can be greatly increased if we reduce the number of classes, obtaining an accuracy of 0.79.

On the one hand, this performance could be improved by analysing in more detail the preprocessing part, looking for words that do not give much information and by adding metadata information in the models, for example, the author of the poem or considering the number of syllables and the type of rhyme.

On the other hand, the performance of the models is limited due to the dataset, which contains a small number of poems, and it is high imbalanced, as we have seen with the confusion matrix. Possible ways to solve this problem are by increasing the number of poems or by joining classes, as we have done.

In conclusion, the work gave us a huge comprehension of the different algorithms, ideas and methods seen in class, as well as the different implementations and requirements they have and the results they provide. It has also given us more insights into machine learning and the different challenges that can be solved with it. We can firmly say that our knowledge about machine learning has been significantly improved after coursing this subject.

6. References

During the development of this project, we have used this sources:

- **GitHub-Copilot** that offers a free licence for students.
  (Remark that the implementation of a collaborative system, such as GitHub, is a key part during the development of this kind of projects were many of us work together on different code snippets and problems can arise from time to time.)
- **ChatGPT** Has been employed for code errors fixing mainly and to help us develop some additional plots and issues related to the Dashboard.
- https://dash.plotly.com/tutorial With this tutorial we have learnt how to use Dash.
- Poetry Foundation As we explained, this is the web page from which we have obtained the information whose poems are licence-free for educational uses.
- **C2.350.16503-96* Machine learning applications 23/24-S2 Contents**. From the contents of the course and the notebooks that we have been developing, we have taken most of the code used in this project.