# CMM PLN at MentalRiskES: A Traditional Machine Learning Approach for Detection of Eating Disorders and Depression in Chat Messages

Rodrigo Guerra[1,2], Benjamín Pizarro[3], Carlos Muñoz-Castro[3,4,6], Andrés Carvallo[4], Matías Rojas[5], Claudio Aracena[2,6] and Jocelyn Dunstan[1,3,5,6]

[1]*Center for Mathematical Modeling, Universidad de Chile, Chile*
[2]*Faculty of Physical and Mathematical Sciences, Universidad de Chile, Chile*
[3]*Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile*
[4]*National Center for Artificial Intelligence, Chile*
[5]*Institute for Mathematical and Computational Engineering, Pontificia Universidad Católica de Chile, Chile*
[6]*Millenium Institute Foundational Research on Data, Chile*

Abstract

This paper describes our approaches to solving the MentalRiskES task, which belongs to the IberLEF (Iberian Languages Evaluation Forum) shared task. The task aims to identify the eating disorders and depression of a user using a series of Telegram messages. Our proposed system uses the traditional TFiDF method to represent the messages and then utilizes these representations as input for machine learning models. The best results for classification were obtained using the Naive Bayes classifier, while in the regression task, the best models were Gradient Boots Regressor and Linear Regressor. Despite its simplicity, we demonstrated that our traditional approaches can still achieve competitive results in recent NLP tasks, obtaining the best results in the case of detecting depression and eating disorders.

**Keywords**

Natural Language Processing, Text Classification, Text Regression, Chat Messages, Eating Disorders, Depression

## 1. Introduction

The global mental health crisis [1], as emphasized by alarming statistics from the World Health Organization, demands novel and effective solutions for early risk identification. The emergence of social media as a window into individuals' mental states offers an unprecedented opportunity to identify and mitigate risks before they escalate. However, a gap exists in addressing the non-English-speaking population, mainly Spanish speakers.

This paper focuses on a proposed method that utilizes data from the MentalRiskEs task [2], aimed at the early identification of mental disorders in Spanish messages from Telegram users. Our solution seeks to guide through a data stream, identifying risks at the earliest point, an approach that prioritizes accuracy and detection speed.

Our method tackles two primary tasks: 1) the detection of eating disorders and 2) the detection of depression. Depending on the task's specifics, we approach the problem from

*IberLEF 2023, September 2023, Jaén, Spain*

various angles, including binary classification, simple regression, multiclass classification, and multi-output regression. Each approach is designed to discern whether a user has the given mental disorder and to what degree. We combine machine learning algorithms and different text representations, including traditional techniques such as TF-IDF and modern approaches that leverage transformer-based language models trained in the Spanish language, such as BETO [3, 4], DistillBETO [5], or BERTIN [6], to solve these tasks. While these modern language models have significantly advanced in natural language understanding, we give evidence that traditional machine learning models can still yield competitive performance when paired with appropriate text representations. We aim to show that our method, which combines traditional and contemporary approaches, can identify individuals with potential eating and mental disorders more effectively and rapidly. Detailed information about our approach, its implementation, and its evaluation will be provided in the subsequent sections of this article.

In this paper, we first present the tasks and data. In methodology, we present the preprocessing and methods to solve these tasks. Then in experimental methodology, it shows the pipeline for each task and the model's selection criteria. Then the results and analysis, and finally, conclusions and future work.

## 2. Tasks and Data Description

### 2.1. Tasks

In our MentalRiskES [2] approach, we focus on two tasks: detecting eating disorders and depression through the analysis of Telegram messages. Both tasks use various techniques, including binary and multi-class classification and regression models, to predict the risk of either condition or related sub-conditions. The first task involves detecting eating disorders and estimating the likelihood of anorexia or bulimia. The second task identifies depression and its probability and categorizes it into specific types. More details on each task and their subtasks are provided as follows.

#### 2.1.1. Task 1

This task aims to detect eating disorders. The subtasks are the following:

1. **Subtask 1a**: A binary classification. Labels are 0 for "control" and 1 for "suffer".
2. **Subtask 1b**: Regression of the probability of suffering anorexia or bulimia. 0 means 100% negative, and a 1 would be 100% positive.

#### 2.1.2. Task 2

This task aims to detect depression. The subtasks we approached are:

1. **Subtask 2b**: Regression of the probability of suffering depression. 0 means 100% negative, and a 1 would be 100% positive.
2. **Subtask 2d**: Multi-output regression for each of the previous classes. The system should provide a probability of belonging to that class.

## 2.2. Data

To tackle the tasks of eating disorders and depression detection from Telegram messages, we used the dataset provided by the MentalRiskES from the IberlEF 2023 challenge [2]. The eating disorder detection task consists of 50 rounds of messages, while the depression detection task has 100 rounds of messages. We strategically divided the dataset, using 80% for training our models and reserving the remaining 20% for dev-test. Furthermore, our approach ensured stratification based on the label for the classification tasks.

The evaluation of a system is, according to an *evaluation perspective*. For classification, the perspectives are:

- *Absolute classification*: measure the absolute classification from the first round. If the classifier output in a round is 1, the system will consider that output until the last round. The performance per round is measured with Micro and Macro F1.
- *Early detection effectiveness*: measures how fast the system detects the disorder. The metrics are *Early Risk Detection Error (ERDE)* which measures the correctness and the delay of the classification at a specific round.

For regression, the evaluation perspectives are:

- *Error basis*: measures the error between the prediction of the model and the observed values. It uses the *Root Mean Square Error* (RMSE).
- *Ranking basis*: measures a ranking of the output and its concordance with the observed values. The metric is *Precision@k* (p@k), which means the top k values, and for this problem, at a specific round.

The following list shows the evaluation perspectives and the metrics to rank the systems for each type of problem:

| Classification Tasks | Metrics to Rank |
|---|---|
| Absolute Classification | Macro-F1 |
| Early Detection Effectiveness | ERDE30 |

**Table 1**
Classification metrics to rank the systems

| Regression Tasks | Metrics to Rank | Multiregression Tasks Metrics |
|---|---|---|
| Error basis | RMSE | RMSE mean |
| Ranking basis | P@30 | P@30 |

**Table 2**
Regression Metrics to rank the systems

Apart from these metrics, other performance and efficiency metrics are measured in the submission.

# 3. Methodology

This section outlines the methodology employed to tackle the tasks and subtasks of the MentalRiskES Challenge. Our process begins with a detailed overview of our data preprocessing strategy, which centers on preparing text data for input into our chosen machine learning models. Following this, we present and contrast our proposed methods against the challenge's baseline models, which are state-of-the-art transformer-based architectures. Our objective is to illustrate the competitiveness of traditional machine-learning techniques against these baselines. Comprehensive details about our preprocessing steps and each method are thoroughly explained in the subsequent subsections.

## 3.1. Preprocessing

As a first step, we concatenated for each user all the messages. We analyzed this data to see which words were more common in the complete and per class. The most important finding in Taks 1 is that the word "Ana" was common in class "suffer". This information was useful in finding out that the teenagers, mainly females, identify as Ana for anorexia and Mia for bulimia.

After we lemmatized all the messages, with *spacy-spanish-lemmatizer* [7], and removed the word "mia" from the stopwords set. Then, for feature extraction, at first, we used Term Frequency Inverse Document Frequency (TF-IDF) [8] with 2-gram to maintain context. This is the input for the machine learning models.

## 3.2. Methods

This subsection outlines our methods for our experiments, centering on traditional machine learning models for solving these intricate tasks related to mental disorders. Initially, we discuss the challenge-proposed baselines, incorporating state-of-the-art language models like RoBERTa [9] and DeBERTa [10]. Subsequently, we shed light on our solutions for the binary and multi-class classification tasks related to eating disorders and depression. We have leveraged traditional machine learning classifiers, including Support Vector Classifier, Multinomial Naive Bayes, K-Nearest Neighbors, and Extreme Gradient Boosting. For regression tasks, our toolkit comprises SGD Regressor, Ridge Regressor, Linear Regressor, Gradient Boost Regressor, and Random Forest Regressor. We inquire into more specifics for these models in the following subsections.

### 3.2.1. Baselines

The baselines for MentalRiskES tasks are the following:

- RoBERTa-Base [9]: is a model developed by Facebook AI and is a variant of BERT optimized for better performance. It uses dynamic masking and trains on larger batches of data. It discards the Next Sentence Prediction (NSP) task used in BERT, enabling it to produce more nuanced and context-aware text representations. This makes it a powerful tool for various natural language processing tasks, namely regression and classification, after adding a linear layer.

- RoBERTa-Large [9]: similar to the RoBERTa Base, optimizes the BERT approach for superior performance. The 'Large' variant denotes a more extensive architecture with 355 million parameters, compared to RoBERTa Base's 125 million, allowing it to model more complex data patterns at a higher computational cost.
- DeBERTa [10]: is a transformer-based model used for natural language processing tasks. It builds upon the BERT model by introducing two main enhancements: disentangled attention and an enhanced mask decoder. Disentangled attention in DeBERTa separates the content and position streams in the self-attention mechanism. The model processes the content (the what) and position (the where) information separately in the attention scores, allowing for a more nuanced understanding of the context. The enhanced mask decoder applies a scaling factor to the dot-product self-attention mechanism, improving the model's ability to model dependencies in masked language modeling tasks. These improvements lead to a more refined understanding of the context in the text and allow DeBERTa to excel in various NLP tasks, from classification to sentiment analysis and beyond.

#### 3.2.2. Classification

To accomplish classification tasks, we employed four traditional supervised algorithms that have demonstrated exceptional results across diverse tasks in Natural language Processing (NLP) from scikit-learn library [8].

- **Support Vector Classifier (SVC)**: The Support Vector Machine or Support Vector Classifier (SVC) is a powerful method for regression and classification tasks [11, 12]. The main objective of SVC is to find an optimal hyperplane that separates different classes, maximizing the margin between them in a dimensional space. In the same way, the algorithm uses a training subset, and it identifies the support vectors; that correspond to points nearest to the hyperplane. These support vectors play a crucial role in determining the position and orientation of the hyperplane [12], finally directly impacting the decision function. Additionally, SVC has the capability to handle both linearly and non-linearly separable datasets by employing kernel functions [13]. These functions transform the input data into a higher-dimensional space where a linear separation may exist. The most popular Kernels are Linear, Radial Basis Function (RBF), Polynomial, and Sigmoid.
- **Multinomial Naive Bayes (Multinomial NB) Classifier**: The algorithm is supported by the well-known Bayes' theorem [14], which provides a foundation for a simple probabilistic method known as Naive Bayes (NB). This algorithm aims to predict the conditional probability of an event by making certain naive assumptions. It calculates the probability of occurrence based on previous knowledge of circumstances associated with the event, leveraging the principles of Bayesian probability. In addition to the above, an approach to a multinomial model [15] becomes particularly relevant when dealing with discrete features (e.g., word counts present in text). The multinomial classifier effectively captures the patterns and information embedded in the discrete features of the text, enabling accurate classification.
- **K Nearest Neighbors Classifier (KNN)**: The KNN algorithm [16] is a classical and uncomplicated classifier but highly efficient and effective in several tasks [17, 18]. The

procedure classifies the unlabeled data based on the k-nearest neighbor points labeled from a previous sample [19], commonly named as training examples. The essence of this approach depends mainly on measuring the distance between the examples, the number of neighbors represented by k, and the neighborhood combination criterion. Where k is an integer that takes values $\geq 1$ and depicts the observed neighborhood's scope.

- **Extreme Gradient Boosting Classifier (XGB)**: it operates under the principle of gradient boosting, which involves building multiple weak predictive models, typically decision trees, and combining them in an ensemble for stronger and more accurate predictions. When applied to text classification, XGBoost starts with preprocessing the text data. This involves steps such as tokenization, removing stop words, and vectorization. After preprocessing, each document is represented as a vector, forming a high-dimensional dataset. XGBoost then builds decision trees iteratively. For each iteration, it adds a new tree that attempts to correct the errors made by the ensemble of existing trees. It does this by assigning more weight to the instances that were misclassified in previous iterations. This process continues until a specified number of trees are added or no further improvements can be made. XGBoost also employs a technique called regularization, which helps to prevent overfitting by penalizing complex models, leading to better performance on unseen data.

#### 3.2.3. Regression

To accomplish regression tasks, we employed four traditional supervised algorithms used in Natural language Processing (NLP) from scikit-learn library [8]:

- **SGD Regressor:** is a linear model that employs a stochastic approximation to the gradient descent optimization. It's particularly effective for large-scale data due to its efficiency and ease of implementation. The model predicts the outcome using a linear combination of feature values and model parameters.
- **Ridge Regressor:** is a technique for analyzing multiple regression data suffering from multicollinearity. Adding a degree of bias to the regression estimates, it reduces the standard errors. This technique employs L2 regularization, which discourages large coefficients by penalizing the square of the coefficients.
- **Linear Regressor:** is a simple form of regression that assumes a linear relationship between the independent and dependent variables. It can be single (one independent variable) or multiple (more than one independent variable). The model calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line.
- **Gradient Boosting Regressor Trees (GBRT)**: The GBRT algorithm corresponds to a variant of Gradient Boosting (GB), highlighting particularly in regression tasks. Where the objective is not to predict classes as XGB; on the contrary, it predicts continuous values. Gradient Boosting Regressor involves the sequential addition of decision trees, assigning greater importance to instances with higher prediction errors. This iterative process continues until a predetermined number of trees are added, or no improvements are found.

- **Random Forest Regressor**: is an ensemble learning method that builds multiple decision trees and outputs the mean prediction of the individual trees. It reduces overfitting common in single decision trees, and improves generalization by introducing randomness in constructing the trees.

## 4. Experimental Methodology

As a preprocessing, we used the Spanish lemmatizer from *spacy* [7]. For feature extraction, we used TFiDF [8] with bigram or trigram. We used the one with better results.

After, we trained the models with all the rounds concatenated for each task, with the users in the training set. To evaluate the models, we predicted the users in the dev-test set also known as the validation set; each round concatenates the previous messages.

Finally, for the selection criteria, we plotted the evolution of metrics for each model, and we chose the model that ended with best with a specific metric and stability (the predictions do not change strongly over a few incoming messages). The experiment modules are in Figure 1.

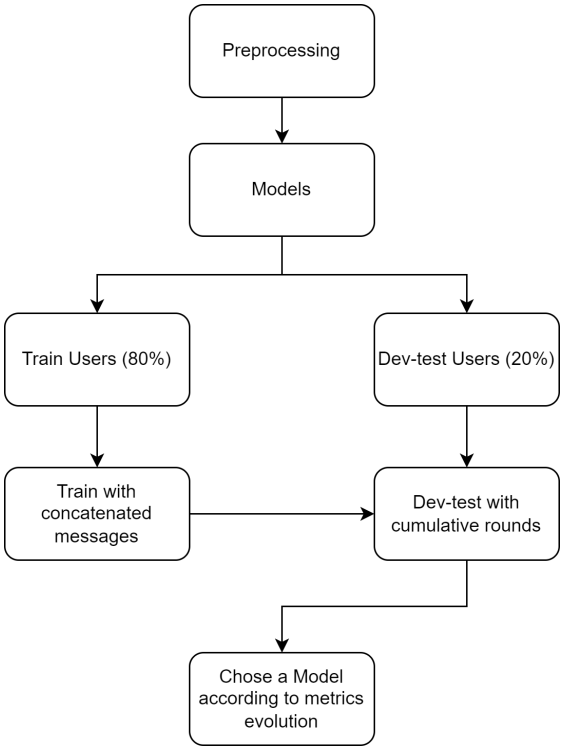



**Figure 1:** Experiments Modules. Preprocessing module: Lemmatizer and TFiDF for feature extraction. Models: Machine Learning Models according to the task. Train: 80% of users with all the messages. Dev-test: 20% of users with a cumulative concatenation of messages. Selection criteria: evolution and stability of metrics according to the task

# 5. Results

## 5.1. Task 1: Eating Disorders Detection

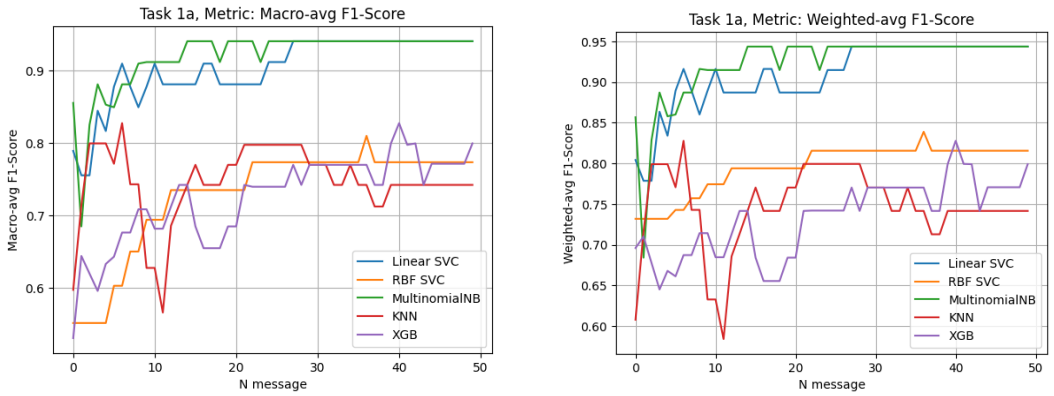
For this task, there are two approaches to detecting an eating disorder. The first is a classification problem of the classes "suffer" and "control". And the second is a simple regression with the probability of the class "suffer".

### 5.1.1. Binary Classification

In this task, our approach in Macro F1 metric is 0.827. And rank in this metric we are 9 out of 25 models, the best model result is 0.966. Our method outperformed baseline models in accuracy, Macro-P, Macro-R, and Macro-F1. In early risk detection, in ERDE30 our model result is 0.074 and in rank, is 5 out of 25. We outperform the baselines. The best model in this metric is 0.018.

The preprocessing of this task was to lemmatize and use bigrams with TFiDF. The classifiers used for this task are Linear SVC, RBF SVC, KNN, XGB, and MultinomialNB, all with *sklearn* default parameters, except for MultinomialNB, with $alpha = 0.01$ for additive smoothing parameter (smooth categorical data).

Figure 2 shows the evolution through 50 messages of **Macro average F1-Score** and *Weighted-avg F1-Score* in the dev-test users.




***(a)*** *Macro F1-Score Evolution in Task 1a* ***(b)*** *Weighted F1-Score Evolution in Task 1a*

**Figure 2:** Evolution of metrics in Task 1a in dev-test users

We see the instability of the metrics in the first messages in every model. KNN, RBF CVC, and XGB have an unstable evolution. This means in absolute classification, the behavior of these models would be lower because of the changes in the decisions.

On the other hand, models *MultinomialNB* and *Lineal SVC* have better metrics and similar stabilities. We choose MultinomialNB because, after 8 rounds, it has the highest metrics and more stable evolution.

Table 3 shows the results of the validation and test set. The metrics of the absolute classification show that Multinomial Naive Bayes performs better than transformers and the mean of the rest of the systems.

| **Model** | Accuracy | Macro-P | Macro-R | **Macro-F1** |
|---|---|---|---|---|
| **Validation** | | | | |
| Multinomial Naive Bayes | 0.886 | 0.883 | 0.883 | 0.883 |
| **Test** | | | | |
| Multinomial Naive Bayes | **0.827** | **0.856** | **0.849** | **0.827** |
| BaseLine - Roberta Large | 0.813 | 0.823 | 0.827 | 0.813 |
| BaseLine - Deberta | 0.813 | 0.842 | 0.835 | 0.813 |
| Other systems (median) | 0.760 | 0.817 | 0.785 | 0.760 |
| Other systems (mean) | 0.759 | 0.816 | 0.777 | 0.741 |
| BaseLine - Roberta Base | 0.700 | 0.783 | 0.736 | 0.694 |

**Table 3**
Task 1a: Overall absolute results

Table 4 shows the results for early prediction. At ERDE5, the performance is not well because, in the submission, we could not send the first 8 rounds. When simulating this in validation, we see similar values. For ERDE30, we can see that our model is superior to the others but with a higher latency, which means that latency-weighted F1 is lower than all the transformers-based models.

| **Model** | ERDE5 | **ERDE30** | latencyTP | speed | latency-weightedF1 |
|---|---|---|---|---|---|
| **Validation** | | | | | |
| Multinomial Naive Bayes | 0.437 | 0.082 | 8 | 0.893 | 0.774 |
| **Test** | | | | | |
| Multinomial Naive Bayes | 0.498 | **0.074** | 10 | **0.817** | 0.679 |
| BaseLine - Deberta | 0.310 | 0.083 | 5 | 0.918 | 0.751 |
| BaseLine - Roberta Large | **0.163** | 0.099 | **2** | 0.979 | **0.792** |
| BaseLine - Roberta Base | 0.186 | 0.132 | **2** | 0.979 | 0.722 |
| Other systems (median) | 0.306 | 0.116 | 4 | 0.938 | 0.679 |
| Other systems (mean) | 0.328 | 0.132 | 6 | 0.908 | 0.695 |

**Table 4**
Overall results in early detection

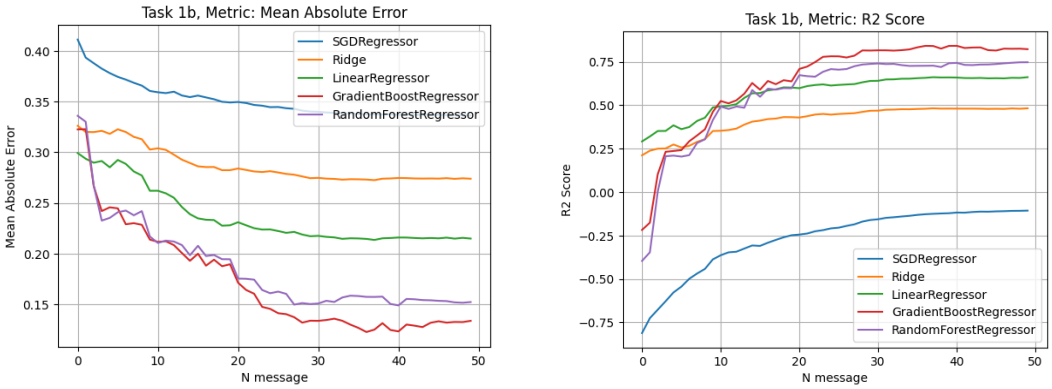
The failed prediction in the test partition is mainly caused by an early prediction of the class “suffer”. This could have been avoided with a higher Multinomial Naive Bayes classifier threshold when predicting probabilities of the class “suffer”.

### 5.1.2. Simple Regression

The result in RMSE of our model is 0.244, and in the ranking is 8 out of 20. The baselines outperformed our model, and the best model is Baseline Roberta Base with an RMSE of 0.178. On the other hand, in ranking-based evaluation in p@30, our model has 0.733, and the best model is Baseline Roberta Large with 0.900.

The preprocessing of this task was to lemmatize and for feature extraction bigram TFiDF. The models we used were all with the default parameters. The regressors were SGD, Ridge, Linear, Gradient Boost, and Random Forest.

Figure 3 shows the evolution of the metrics Mean Absolute Error and R2 Score. In the first round, the Linear Regressor has better metrics than the rest of the models, but after 2 rounds, Random Forest and Gradient Boost are always better. We choose Gradient Boost Regressor because, after 10 rounds, it has the lowest Mean Absolute Error and the highest R2 Score.




***(a)*** *Mean Absolute Error in Task 1b*

***(b)*** *R2 Score in Task 1b*

**Figure 3:** Evolution of metrics in Task 1b

Table 5 shows that the transformers perform better than the rest of the systems in both evaluation perspectives. Regression is more difficult, so we expected that complex models outperform machine learning models.

| **Model** | **RMSE** | Pearson coefficient |
|---|---|---|
| **Validation** | | |
| Gradient Boost Regressor | 0.166 | 0.920 |
| **Test** | | |
| BaseLine - Roberta Base | **0.178** | **0.906** |
| BaseLine - Roberta Large | 0.196 | 0.890 |
| BaseLine - Deberta | 0.231 | 0.868 |
| Gradient Boost Regressor | 0.244 | 0.773 |
| Other systems (mean) | 0.316 | 0.665 |
| Other systems (median) | 0.324 | 0.705 |

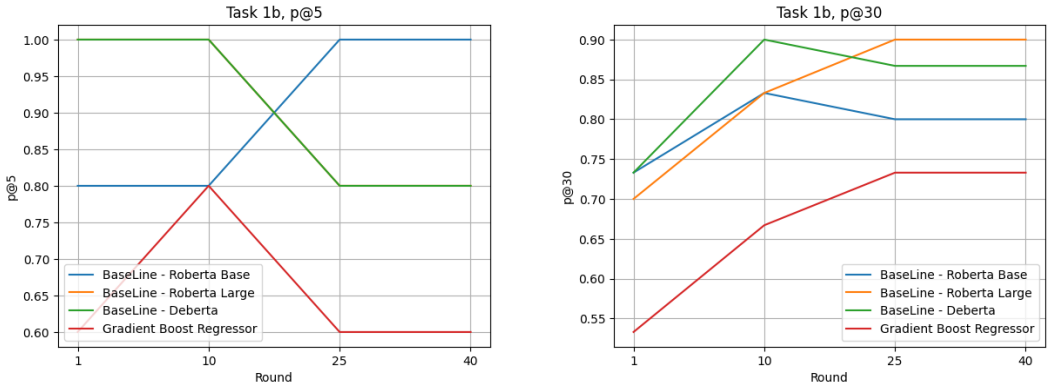
**Table 5**
Overall results in simple regression at round 25

| **Model** | p@5 | p@10 | p@20 | **p@30** |
|---|---|---|---|---|
| **Validation** | | | | |
| Gradient Boost Regressor | 0.800 | 0.700 | 0.700 | 0.600 |
| **Test** | | | | |
| BaseLine - Roberta Large | 0.800 | 0.800 | 0.800 | **0.900** |
| BaseLine - Roberta Base | **1.000** | 0.800 | **0.850** | 0.800 |
| BaseLine - Deberta | 0.800 | **0.900** | **0.850** | 0.867 |
| Gradient Boost Regressor | 0.600 | 0.700 | 0.800 | 0.733 |
| Other systems (mean) | 0.763 | 0.694 | 0.719 | 0.685 |
| Other systems (median) | 0.700 | 0.700 | 0.700 | 0.700 |

**Table 6**
Overall results in ranking simple regression at round 25

Table 6 shows that our model performs better than other systems, but again, transformer-based models outperform our Gradient Boost Regressor at round 25.

We compared the evolution of p@5 and p@30 in Figures 4a and 4b of the baseline methods and ours in the sampled rounds. We can see that Transformers have had the same or better performance since the beginning of the rounds. This is probably due to the large vocabulary and more accurate representation of words in these types of models.




***(a)*** *Evolution of p@5*

***(b)*** *Evolution of p@30*

**Figure 4:** Evolution of metrics in the test partition in Task 1b

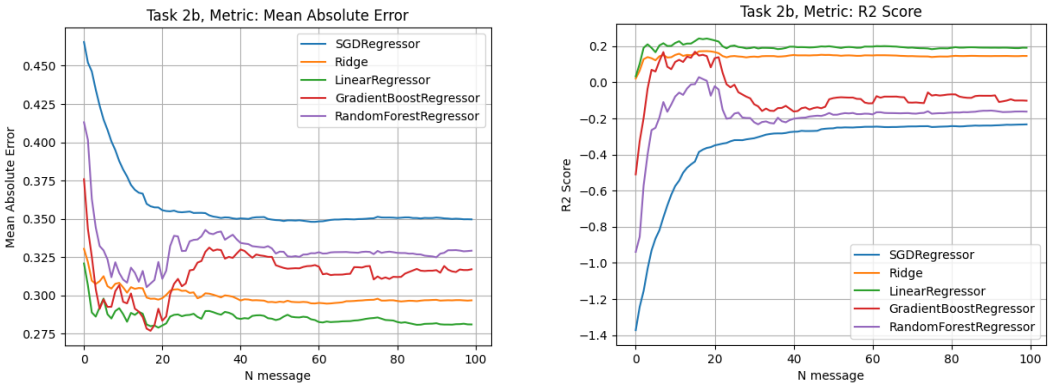
## 5.2. Task 2: Depression Detection

For this task, we only approached the regression problems, corresponding to a simple regression to predict the probability of a user suffering from depression. The second problem is a multi-output regression for the classes "control", "suffering+against", "suffering+in favor", and "suffering+other".

### 5.2.1. Simple Regression

The result of our model in RMSE is 0.309, with a rank of 4 out of 19. The best model is Baseline Roberta Base with an RMSE of 0.277. In ranking-based evaluation, our model has the best results in every metric, with a p@30 of 0.600.

For this task, the feature extraction was bigrams in TFiDF. All the models had the default parameters. The regressors were SGD, Ridge, Linear, Gradient Boost, and Random Forest.
Figure 5 shows the evolution of Mean Absolute Error and R2 Score in the validation set. We can see an erratic Random Forest and Gradient Boost Regressor behavior in both metrics. SGD has a more stable behavior but the worst metrics overall rounds. The best models are Ridge and Linear Regressor. After 20 messages, both are stable, but Linear Regressor has a lower Mean Absolute Error.




***(a)*** *Mean Absolute Error Evolution in Task 2b*

***(b)*** *R2 Score Evolution in Task 2b*

**Figure 5:** Evolution of metrics in Task 2b

Table 7 shows that Roberta Base is the only baseline that outperforms a Linear Regression in metric RMSE. But in the Pearson coefficient, Linear Regressor is outperformed by Roberta and Deberta Base.

| Model | RMSE | Pearson coefficient |
|---|---|---|
| **Validation** | | |
| Linear Regression | 0.330 | 0.466 |
| **Test** | | |
| BaseLine - Roberta Base | **0.277** | **0.770** |
| Linear Regression | 0.309 | 0.642 |
| BaseLine - Deberta | 0.339 | 0.683 |
| Other systems (mean) | 0.377 | 0.390 |
| Other systems (median) | 0.381 | 0.318 |
| BaseLine - Roberta Large | 0.390 | 0.503 |

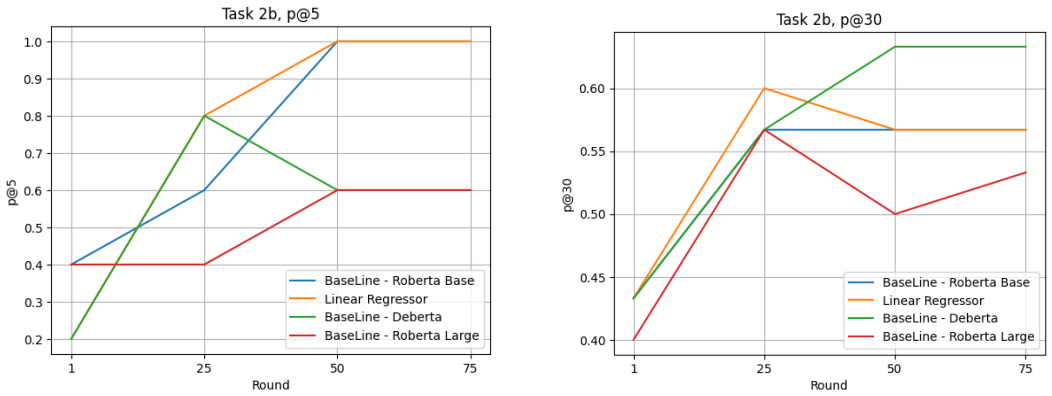
**Table 7**
Overall results in simple regression at round 25

Table 8 shows that at round 25, Linear Regressor has the same or better performance in every ranking-based evaluation.

| Model | p@5 | p@10 | p@20 | **p@30** |
|---|---|---|---|---|
| **Validation** | | | | |
| Linear Regression | 0.800 | 0.700 | 0.700 | 0.600 |
| **Test** | | | | |
| Linear Regression | **0.800** | **0.800** | **0.700** | **0.600** |
| BaseLine - Roberta Large | 0.400 | 0.500 | 0.550 | 0.567 |
| BaseLine - Roberta Base | 0.600 | **0.800** | **0.700** | 0.567 |
| BaseLine - Deberta | **0.800** | 0.600 | 0.550 | 0.567 |
| Other systems (mean) | 0.320 | 0.300 | 0.317 | 0.305 |
| Other systems (median) | 0.400 | 0.300 | 0.350 | 0.300 |

**Table 8**
Overall results in ranking simple regression at round 25

If we look at the evolution of the rankings p@5 and p@30 in Figures 6a and 6b respectively, our model, the Linear Regressor, at p@5, has lower than Roberta Base and Large in round 1, but from round 25, it has the best performance, and the metrics are similar to Roberta Base. On the other hand, in p@30, the Linear Regressor is the best in rounds 1 and 25, but after, the metric gets lower.




***(a)*** *Evolution of p@5*

***(b)*** *Evolution of p@30*

**Figure 6:** Evolution of metrics in the test partition in Task 2b

Unlike Task 1b, a machine-learning model performed better than transformer-based models in ranking metrics. This could be explained because Task 2b has less spaced vocabulary (9046 unique words in 100 messages of the training set) than in 50 messages (8860 unique words in 50 messages of the training set), resulting in a more robust machine learning model for this task, and achieving metrics that can compete with a transformer.

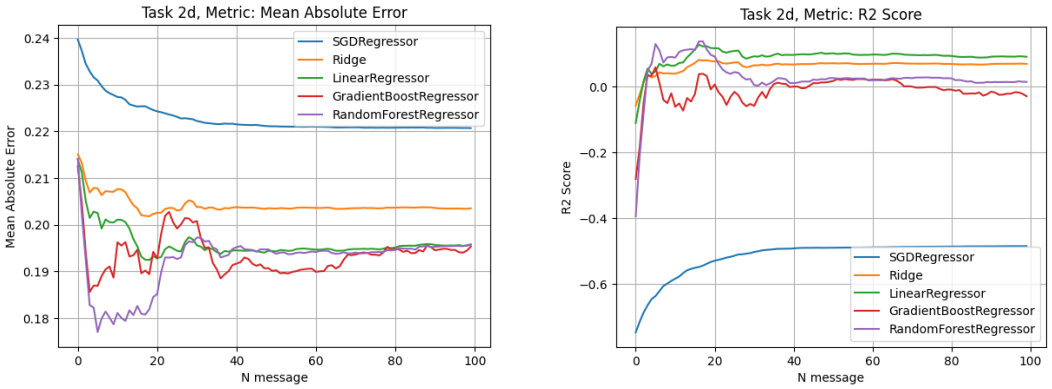
The detail that this model outperformed transformers-based models in Ranking metrics is unexpected, considering the fine-tuning of a language model compared with a Linear Regressor. This leads us to think that with a smaller corpus is not always necessary to apply big complex models, and depending on the behavior of a simpler model, try models that might have a better performance, but always consider that a model that requires higher computing capacity will have higher emissions.

### 5.2.2. Multi-output Regression

The result of our model in RMSE mean is 0.349, ranked 5 out of 7. The best model for this task is Baseline Deberta with 0.246. In ranking-based on p@30 metric our approach is 7 out of 7 models with 0.175, and the best model has 0.350.

For this task, we used trigrams with TFiDF. The regressors for this task were SGD, Ridge, Linear, Gradient Boost, and Random Forest, all with default parameters.

Figure 7, shows an unstable evolution of Gradient Boost and Random Forest Regressor, with a lower Mean Absolute Error at first rounds. However, as the round passes, the error is similar to the Linear Regressor. On the other hand, R2 Score shows that Linear Regressor, after the 20th round, always has a higher score, and it remains stable. So we chose it for the submission.




***(a)*** *Mean Absolute Error Evolution in Task 2d* ***(b)*** *R2 Score Evolution in Task 2d*

**Figure 7:** Evolution of metrics in Task 2d

Table 9 shows that our Linear Regressor has a better RMST than Roberta Base and Large. But metrics RMSE sa, and Pearson sa, prove that our model is biased to the class “suffer+against”, and we can see the same behavior in Roberta Base and Large. This could be explained because of the imbalanced probabilities for this task. Contrary to Deberta, which is less biased because of the more similar metrics between the classes.

| Model | RMSE mean | RMSE sf | RMSE sa | RMSE so | RMSE c | Pearson mean | Perason sf | Pearson sa | Pearson so | Pearson c |
|---|---|---|---|---|---|---|---|---|---|---|
| **Validation** | | | | | | | | | | |
| Linear Regressor | 0.364 | 0.344 | 0.202 | 0.426 | 0.484 | 0.063 | 0.297 | 0.459 | -0.140 | -0.363 |
| **Test** | | | | | | | | | | |
| BaseLine - Deberta | **0.232** | **0.246** | 0.250 | **0.125** | **0.306** | **0.484** | **0.661** | 0.295 | **0.260** | **0.721** |
| Other systems (mean) | 0.250 | 0.272 | 0.228 | 0.129 | 0.371 | 0.131 | 0.207 | 0.018 | 0.057 | 0.240 |
| Other systems (median) | 0.250 | 0.272 | 0.228 | 0.129 | 0.371 | 0.131 | 0.207 | 0.018 | 0.057 | 0.240 |
| Linear Regressor | 0.349 | 0.328 | **0.210** | 0.391 | 0.469 | -0.052 | 0.051 | 0.394 | -0.153 | -0.498 |
| BaseLine - Roberta Base | 0.410 | 0.547 | 0.272 | 0.235 | 0.585 | -0.145 | -0.496 | 0.355 | 0.185 | -0.624 |
| BaseLine - Roberta Large | 0.437 | 0.682 | 0.312 | 0.158 | 0.598 | -0.209 | -0.678 | **0.890** | 0.059 | -0.306 |

**Table 9**
Overall results in multi-output regression-based evaluation

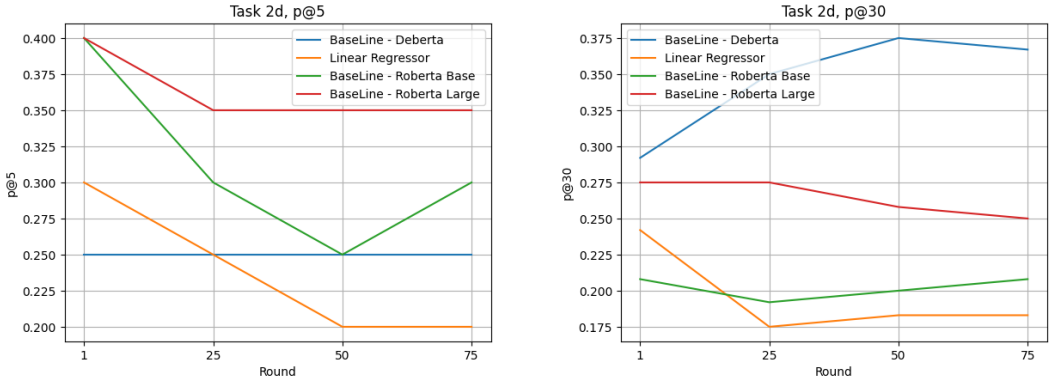
Table 10 shows that Linear Regressor cant rank properly in comparison with the rest of the

models. This task is more complex than the others, and so transformers' initial weights about the language can provide a more precise output.

| **Model** | p@5 | p@10 | p@20 | **p@30** |
|---|---|---|---|---|
| **Validation** | | | | |
| Linear Regressor | 0.350 | 0.225 | 0.175 | 0.133 |
| **Test** | | | | |
| Other systems (mean) | **0.350** | **0.400** | **0.375** | **0.350** |
| Other systems (median) | **0.350** | **0.400** | **0.375** | **0.350** |
| BaseLine - Roberta Large | **0.350** | 0.275 | 0.263 | 0.275 |
| BaseLine - Deberta | 0.250 | 0.300 | 0.338 | **0.350** |
| BaseLine - Roberta Base | 0.300 | 0.300 | 0.225 | 0.192 |
| Linear Regressor | 0.250 | 0.200 | 0.200 | 0.175 |

**Table 10**
Overall results in multi-output ranking-based evaluation

In Figure 8, Linear Regressor after round 25 can not achieve the rankings that transformed-based models have.




***(a)** Evolution of p@5*

***(b)** Evolution of p@30*

**Figure 8:** Evolution of metrics in the test partition in Task 2d

## 6. Error Analysis

This section describes the failed predictions and characterization, possible improvements, and lessons learned from each task.

For eating disorders, the classification error source was mainly associated with sporty men, with a late mention of intentions to lose weight. The tendency of the previous cumulative messages affected the model, which we could solve with the probability prediction of all the text, predict the message by itself, and ponderate by a hyperparameter. Another thing that can affect this behavior is the high dimensional TFiDF. We could limit the features and also prevent the overfitting to certain n_grams.

In the ERDE metric, the main source of error was certain words in the context of gym and sports such as 'subir peso', 'bajar peso', 'alimenticio', or identifying themselves as 'Ana' but their real name is Ana affects a wrong early prediction. We could improve this by adding high-probability thresholds in the first messages and lowering them as the rounds continue.

In eating disorders' regression problems, in RMSE, there could be an overfitting problem because the validation errors are better than all the baselines. Still, in the test set, those models outperform our model, and it happens the same with the Pearson coefficient. The same as before, the feature extractor should have limited n_grams. The errors were mainly in subjects with low probabilities, trying to lose weight healthily.

In the ranking evaluation regression, the failed predictions were similar to the previous. Still, the messages were more associated with food or diets in a gym context, such as fat percentage, "frutos secos", "batidos", "suplemento alimenticio", but also subjects talking about people wanting or starting going to the gym.

In depression detection, the failed regression was mostly associated with people with depression but who are trying to help others or giving advice to someone else. Also, some low probabilities label uses a lot of "cara llorando". We could solve this by adding context to the linear regressor or using other ways to represent the messages but save information from previous messages and weigh them; this way, we could prevent forgetting precious context.

In multi-output regression, the linear regressor was biased to the class "'suffer+againts", which was the highest probability overall after control. There is significant confusion in all classes because the feature extractor and the model cannot get the context of the messages. There are many errors, and there is no pattern in them.

To summarize, the main sources of errors are related to overfitting problems with a class because the feature extractor and the model cannot get the context of the messages. To improve the metrics, we could use and weight the probabilities of previous probability predictions and use them to preserve context. The context was essential for the second task.

Also, for feature extraction, we should find better parameters for max features in TFiDF to prevent overfitting with too specific words that do not add information to the messages or introduce noise to the model. Besides, try using other text representations of the messages. They might behave better results with other classifiers or regressors.

Another way to improve the results is cross-validation in specific rounds and choosing a model according to these results to prevent a biased partition in favor of a class or using another strategy, such as histograms, for a more balanced probability.

## 7. Conclusions and future work

Despite the growing interest in transformer-based techniques in NLP, we show that traditional machine-learning models can outperform transformers in more straightforward tasks. Still, we have to consider the size of the corpus and vocabulary. In a smaller corpus is more likely that a transformer performs better than a machine learning model.

Machine learning can compete with a transformer model for classification tasks in absolute classification and early detection tasks. Still, it has to be well-calibrated for the task or at least have stability during the training. We can perform better with a higher probability threshold in the model output to classify the messages. In this type of task, if the training is unstable, this means that the classification changes regularly for the same subject, and machine learning models will not achieve a good performance.

On the other hand, transformers outperform machine learning models in ranking simple regression metrics. It usually has better metrics at the beginning because it is a pre-trained model and can preserve the context of the messages. Still, with simpler models and a bigger corpus for feature extraction, machine learning can achieve similar or better results as the messages increase.

In a more complex task, such as multioutput regression, transformers outperform machine learning because a multioutput will have fewer data to train per class compared to a simple regression and a higher class imbalance than binary tasks. This can result in an overfitted model. So transformers have a high advantage over machine learning models because of the pretaining in a much bigger corpus.

In future work, we propose to test other feature extraction techniques such as TFiDF with different parameters, bag of words, n-grams, or word2vec. Additionally, using other models like neural networks for various problems of rounds. A more theoretical approach for this problem would be to get a boundary or an optimal relation between vocabulary, corpus, and features for specific different models and use this corpus as an experimental set.

As a final thought, we want to emphasize that, although the better performance of the transformers-based model, we can compare the emissions made in training and deployment and question how much improvement of a metric is worth with a higher emissions model. To make that decision, we have to consider how important the decision is and measure the impact on the environment and people's life regarding bias, influence, quality of life, or privacy.

## Acknowledgements

This work was funded by ANID Chile: Basal Funds for Center of Excellence FB210017 (CENIA), FB210005 (CMM); Millennium Science Initiative Program ICN17_002 (IMFD) and ICN2021_004 (iHealth), Fondecyt grant 11201250, and National Doctoral Scholarships 21211659 (Claudio Aracena) and 21221155 (Carlos Muñoz-Castro). This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02) and the Patagón supercomputer of Universidad Austral de Chile (FONDEQUIP EQM180042).

## References

[1] B. N. Rudd, R. S. Beidas, Digital mental health: the answer to the global mental health crisis?, 2020.

[2] A. M. Mármol-Romero, A. Moreno-Muñoz, F. M. Plaza-del-Arco, M. D. Molina-González, M. T. Martín-Valdivia, L. A. Ureña-López, A. Montejo-Ráez, Overview of MentalriskES at IberLEF 2023: Early Detection of Mental Disorders Risk in Spanish, Procesamiento del Lenguaje Natural 71 (2023).

[3] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, J. Pérez, Spanish pre-trained bert model and evaluation data, Pml4dc at iclr 2020 (2020) 1–10.

[4] V. Araujo, A. Carvallo, S. Kundu, J. Cañete, M. Mendoza, R. E. Mercer, F. Bravo-Marquez, M.-F. Moens, A. Soto, Evaluation benchmarks for spanish sentence representations, arXiv preprint arXiv:2204.07571 (2022).

[5] J. Cañete, S. Donoso, F. Bravo-Marquez, A. Carvallo, V. Araujo, Albeto and distilbeto: Lightweight spanish language models, arXiv preprint arXiv:2204.09145 (2022).

[6] J. D. la Rosa y Eduardo G. Ponferrada y Manu Romero y Paulo Villegas y Pablo González de Prado Salas y María Grandury, Bertin: Efficient pre-training of a spanish language model using perplexity sampling, Procesamiento del Lenguaje Natural 68 (2022) 13–23. URL: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6403.

[7] M. Honnibal, I. Montani, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017). URL: https://github.com/pablodms/spacy-spanish-lemmatizer, to appear.

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

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[10] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention, arXiv preprint arXiv:2006.03654 (2020).

[11] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends,

Neurocomputing 408 (2020) 189–215. URL: https://www.sciencedirect.com/science/article/pii/S0925231220307153. doi:https://doi.org/10.1016/j.neucom.2019.10.118.

[12] C. Burges, C. J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 121–167. URL: https://www.microsoft.com/en-us/research/publication/a-tutorial-on-support-vector-machines-for-pattern-recognition/.

[13] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[14] S. Xu, Bayesian naïve bayes classifiers to text classification, Journal of Information Science 44 (2018) 48–59. doi:10.1177/0165551516677946.

[15] S. Xu, Y. Li, Z. Wang, Bayesian multinomial naïve bayes classifier to text classification, in: J. J. J. H. Park, S.-C. Chen, K.-K. Raymond Choo (Eds.), Advanced Multimedia and Ubiquitous Engineering, Springer Singapore, Singapore, 2017, pp. 347–352.

[16] T. M. Cover, P. E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27.

[17] M. R. Huq, A. Ali, A. Rahman, Sentiment analysis on twitter data using knn and svm, International Journal of Advanced Computer Science and Applications 8 (2017). URL: http://dx.doi.org/10.14569/IJACSA.2017.080603. doi:10.14569/IJACSA.2017.080603.

[18] O.-W. Kwon, J.-H. Lee, Text categorization based on k-nearest neighbor approach for web site classification, Inf. Process. Manag. 39 (2003) 25–44.

[19] P. Cunningham, S. J. Delany, K-nearest neighbour classifiers - a tutorial, ACM Comput. Surv. 54 (2021). URL: https://doi.org/10.1145/3459665. doi:10.1145/3459665.