Rodrigo Miguel Maia Ferreira

Mineração de dados sociais para classificação de doenças mentais em fóruns públicos

Social mining for the classification of mental illnesses in public forums

Rodrigo Miguel Maia Ferreira

Mineração de dados sociais para classificação de doenças mentais em fóruns públicos

Social mining for the classification of mental illnesses in public forums

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica da Doutora Alina Trifan, Professora auxiliar convidada do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Luís Oliveira, Professor catedrático do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Carlos Manuel Azevedo Costa
Professor Associado com Agregação da Universidade de Aveiro

vogais / examiners committee

Alina Liliana Trifan
Professora Auxiliar Convidada da Universidade de Aveiro

Cátia Luísa Santana Calisto Pesquita
Professora Auxiliar do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa

agradecimentos

Quero agradecer aos meus familiares e amigos, por todo o apoio moral que me prestaram nesta fase do meu percurso académico. Quero também agradecer aos meus orientadores, por estarem sempre disponíveis para me dar feedback, quando precisei. Por fim, quero agradecer aos organizadores do eRisk pelo trabalho que têm feito estes anos a organizar estes desafios.

acknowledgments

I want to thank my family and friends, for all the support they provided me, during this stage of my academic journey. I also want to thank my advisors for always being available to provide feedback whenever I needed it. Finally, I want to thank the eRisk organizers for all the work they have been doing throughout these years to organize these shared tasks.

Palavras-chave Mineração de Dados, Aprendizagem de Máquina, Processamento de Linguagem Natural, Saúde Mental

Resumo O aumento de problemas de saúde mental é uma das maiores adversidades que enfrentamos atualmente, enquanto sociedade, e os métodos de assistência tradicionais nem sempre conseguem assistir quem precisa.
Neste trabalho implementamos e avaliamos a eficácia de uma ferramenta de triagem que pode complementar alguns dos pontos fracos dos métodos tradicionais, ao sinalizar sujeitos em risco de desenvolver doenças mentais, que podem beneficiar de assistência médica. Esta ferramenta é baseada em aprendizagem de máquina, e deteta os indivíduos em risco, analisando os seus dados disponíveis públicamente na rede social Reddit.
Este trabalho teve como base a participação na edição de 2022 do CLEF eRisk, nos desafios 1 e 2, com o objetivo de detetar sujeitos em risco de serem jogadores compulsivos, e de desenvolverem depressão respetivamente, onde tivemos como foco, o uso e comparação de diferentes métodos de vetorização de texto. Apesar dos resultados iniciais obtidos no evento não terem sido os melhores, com afinamentos e experiências adicionais, conseguimos obter um bom desempenho, com F1-scores finais de 0.886 e 0.653 para os melhores modelos dos desafios 1 e 2 respetivamente.

Keywords Data mining, Machine Learning, Natural Language Processing, Mental Health

Abstract The increasing amount of mental health issues is one of the biggest adversities that we face nowadays as a society, and the traditional assistance methods often fail to help those in need.
In this work, we implement and evaluate the performance of a screening tool that may complement some of the traditional methods' weaknesses, by signalling subjects at risk of developing mental illnesses, that could benefit from receiving medical assistance. This tool is based on machine learning, and it detects individuals at risk using their publicly available data from the social network Reddit. This work was based on our participation in tasks 1 and 2 of the 2022 edition of CLEF eRisk, with the goal of detecting subjects at risk of pathological gambling and depression respectively, where we had a special focus on the use and comparison of different text vectorization methods. Despite the fact that the initial results obtained at the event were far from those desired, with some tweaks and additional experiments, we managed to improve them, achieving final F1-scores of 0.886 and 0.653 for the best models of tasks 1 and 2 respectively.

Table of contents

List of figures

List of tables

List of abbreviations

APA . . . . . American Psychiatric Association
AUC . . . . . Area Under the Curve

BERT . . . . . Bidirectional Encoder Representations from Transformers
BoW . . . . . Bag-of-Words

cBoW . . . . . continuous-Bag-of-Words
CCC . . . . . Consecutive Confidence Criterion
CLEF . . . . . Conference and Labs of the Evaluation Forum
CLME . . . . . Contextualised Language Model Embeddings
CV . . . . . Cross-Validation

DL . . . . . Deep Learning
DSM-5 . . . . . Diagnostic and Statistical Manual of Mental Disorders 5th edition
DSWE . . . . . Distributional Semantics Word Embeddings
DT . . . . . Decision Tree

ELMo . . . . . Embeddings from Language Models
ERDE . . . . . Early Risk Detection Error
ET . . . . . Extra-Trees Classifier

FN . . . . . False Negative
FP . . . . . False Positive
FPR . . . . . False Positive Rate

GPT-3 . . . . . Generative Pre-trained Transformer 3

ICD-11 . . . . . International Classification of Diseases 11th revision

KNN . . . . . K-Nearest Neighbors

LDA . . . . . Latent Dirichlet Allocation
LR . . . . . Logistic Regression
LSTM . . . . . Long Short-Term Memory

MCC . . . . . Multi Confidence Criterion

ML . . . . . Machine Learning

NB . . . . . Naive Bayes
NDCG . . . . . Normalized Discounted Cumulative Gain
NLP . . . . . Natural Language Processing
NN . . . . . Neural Networks

OOV . . . . . Out-Of-Vocabulary

P . . . . . Precision
PHQ-9 . . . . . Patient Health Questionnaire 9

R . . . . . Recall
RBF . . . . . Radial Basis Function
RCC . . . . . Ratio Confidence Criterion
RF . . . . . Random Forest
RNN . . . . . Recurrent Neural Network
ROC . . . . . Receiver Operating Characteristic

SCC . . . . . Single Confidence Criterion
SGD . . . . . Stochastic Gradient Descent
ssToT . . . . . semi-supervised Topic-modeling over Time
SVM . . . . . Support-Vector Machine

Tf-Idf . . . . . Term frequency-Inverse document frequency
TN . . . . . True Negative
TP . . . . . True Positive
TPR . . . . . True Positive Rate

WHO . . . . . World Health Organization

Chapter 1

Introduction

1.1 Motivation

According to the World Health Organization (WHO), "Mental health is a state of well-being in which an individual realizes his or her own abilities, can cope with the normal stresses of life, can work productively, and is able to make a contribution to his or her community" [1]. Generally speaking, it is related to an individual's cognitive, behavioural and emotional state. Mental health disorders, given their not so physical nature compared to something like muscle, skeletal or organ issues (which can be extracted/poked/seen through imaging) can be harder to properly diagnose. Sadly, the lack of proper support surrounding these disorders and the stigma attached to them by society, often prevent individuals from seeking out help, which may lead to the worsening of their conditions and to the potential risk of self-harm or even suicidal behaviours. According to statistics published by the WHO, over 700 000 people die of suicide each year, with many more having failed attempts [2]. Many of these could be prevented with the appropriate care.

This work's goal is to develop a tool capable, in theory, of better handling some of these cases that do not get the help they deserve by signalling users that may benefit from professional help. It is interdisciplinary in the sense that it involves different areas, health and computer engineering related, with the goal of providing a better mental illness screening method. To this end, the main concepts regarding the issue will be introduced in this chapter while the computational tools and analysis of the research in this field will be left for the following one.

1.2 Mental health

As the name suggests, mental health is the field concerned with an individual's emotional, psychological and social well-being. It differs from other fields in medicine from the fact that diagnoses do not rely as heavily on the subject's physical properties, which can make some conditions harder to diagnose. The earlier an individual is diagnosed, the better, since it allows for healthcare professionals to intervene, often leading to better

outcomes.

1.2.1 Mental Illnesses and Diagnosis

There are a lot of catalogued mental illnesses, among the most common types, those most relevant in the scope of this work are anxiety disorders, mood disorders, eating disorders and impulse control or addiction disorders, the following list contains a brief introduction to each of these types:

- Anxiety disorders provide individuals with continuous feelings of anxiousness, worry or fear, even in situations that would not call for such a response in a healthy individual. Common examples are generalized anxiety, social anxiety and phobias.

- Individuals with mood disorders suffer with constant disturbances when it comes to their mood, they might be in a constantly depressive state, constantly manic state, or present heavy fluctuations between both. Common examples are depression, manic syndrome and bipolar disorder.

- Eating disorders like anorexia nervosa, bulimia nervosa and binge eating disorder regard conditions in which the individual develops an unhealthy relationship and habits with food.

- Impulse control or addiction disorders are those in which individuals are unable to resist urges or behaviours that can be detrimental to themselves or others. Some examples are compulsive gambling or kleptomania for impulse control disorders, and alcohol or drug addiction for addiction disorders.

Regarding the diagnostic process, the Diagnostic and Statistical Manual of Mental Disorders 5th edition (DSM-5) by the American Psychiatric Association (APA) and the mental and behavioural disorders chapter of the International Classification of Diseases 11th revision (ICD-11) by the WHO are the standard bodies of knowledge when it comes to mental health disorders. Diagnostic tools, often in the form of questionnaires are frequently based on the items or symptoms present in those bodies of knowledge. The traditional process of diagnosing a mental illness requires the physical presence and honest account of the patient's feelings and mental state to the medical entity listening or employing some screening tool.

This whole setup is far from optimal since it may fail in many different parts. First, the requirement of the patient's physical presence may be hard to fulfil due to lack of access, or due to the naturally secluding nature of many mental illnesses and the social stigma surrounding them. For the same reasons, the subject might not feel comfortable being totally honest describing their situation to the healthcare professional. It may also be the case that the subject does not realise that there is an issue, or they may just not think of it as something serious enough to warrant a visit to a professional. Patient aside, it may fail from the lack of experience of the healthcare professional, which is not always a

qualified specialist but sometimes only a general health practitioner from a health center or emergency room for example, that despite their best efforts, sometimes lack the expertise. Naturally it might also fail from the screening tool used, but those tend to be sturdier since they are backed up by large bodies of research.

It should come as no surprise that with all these possible failing points, the current system is not providing optimal results. This is where the idea behind this work comes into play, since it addresses some of these weak points.

1.2.2 Social Data/Monitoring

Social data corresponds to the data that social media users publicly share in an online scenario, though it sometimes includes data that the users do not explicitly wish to share like search query data and post metadata like geographical location, language spoken or shared links [3]. Social data's public and easily accessible nature makes it a big facilitator for a lot of big data applications by substituting or complementing traditional data, which is harder to acquire. Naturally, it also brings up a fair share of concerns. Despite the fact that it is public, users might not be comfortable having their data being used for purposes other than the typical use of social media.

The process of analysing social data with the end goal of understanding some trends, opinions or behaviours within a population is called social monitoring, it has been used across many fields with goals such as measuring consumer sentiment, measuring political sentiment, forecasting sales, estimating traffic congestion, forecasting elections, among others [3]. It has also been commonly employed in the public health field in use cases like Influenza surveillance, measuring the prevalence of substance abuse, monitoring cases of foodborne illness, and monitoring gun violence, among others. To better prevent or tackle existing issues regarding a population's well-being.

As we saw earlier, the traditional screening solutions in the mental health field can fail at various points, a social monitoring approach may be useful since it may be able to tackle some of the issues found in the traditional methods. Imagining a depression screening tool employed in social media, one can easily see how the first obstacle of being physically present with a health specialist is broken (though this should still happen later on). Since there is no direct interaction with a health specialist, the users might feel more comfortable to accurately portray their thoughts, as they typically do on social media platforms. And with a signalling tool like this, the user can be made aware that they may benefit from seeking professional help. Naturally, this is not a perfect solution as there are still some concerns, namely regarding the effectiveness of such screening tool and the public's acceptance to being subjected to it. In the past, some similar approaches have faced the public's backlash due to ethical concerns on the use of personal data.

1.3 Objectives

The main objective of this work, upon analysing the available research and tools, is to be able to implement Machine Learning (ML) models capable of detecting instances of users suffering or at risk of suffering from mental illnesses using their publicly available data in a social media context. In other words, the focus is on evaluating the effectiveness of ML based screening tools in detecting mental illness, not on how such tools would be implemented in a real life scenario. Additionally, the resulting models will be tailored for the shared tasks of the 2022 edition of Conference and Labs of the Evaluation Forum (CLEF) eRisk [4], using the data supplied by the organizers for training and testing purposes. Both the workshop and the shared tasks are introduced in the following chapter.

1.4 Outline

This document is composed of 5 chapters.

- Chapter 1 provides an introduction to the issue being addressed as well as this paper's proposed solution. It explains what the issue is and where it stems from, why it needs to be fixed, and how the approach we are exploring may be helpful.

- Chapter 2 focuses on the technical side of this work, providing background information on the process of implementing ML solutions, going from a general to a Natural Language Processing (NLP) specific context. There is also a review of some works that were found to be relevant to this work's goal, where the main approaches, results and interesting takeaways are noted. Finally, an introduction is provided of relevant workshops and shared tasks also responsible for bringing interest and advancements in this area of research.

- Chapter 3 describes the methodology followed in the experimental stage, building from the more theory-based content of chapter 2. Tools, procedures and validation results are shown to explain to the reader how the final models were achieved.

- Chapter 4 goes over the results achieved in our experiments, using various different metrics and comparing our performance with the results achieved by other participants in the shared tasks.

- Chapter 5 contains the main conclusions drawn at the final stages of this work, with some observations made from the final results, different approaches that can be explored in the future, and the overall effectiveness of our methods.

Chapter 2

Background

In this chapter, a contextualisation of the more technical topics will be provided with the goal of exposing the reader to the fundamental concepts at work when it comes to using ML algorithms for the classification of mental illnesses from social data. It is structured in a way where the concepts are introduced generally at first and in a more text-focused, NLP targeted perspective later on. Related research works are also analysed to understand what kind of approaches are being used and their respective pros and cons.

2.1 Machine Learning

Recent technological developments have allowed theories and algorithms from the past century to come to life due to the increases both in available data and computational power, that allow computer systems to learn and improve on certain tasks with experience much like humans do. The field dedicated to this is called Machine Learning (ML) and its application extends over many different areas. One such case is the health field, where some models are being relied on by health professionals to detect physical abnormalities such as tumors from medical scans. In this work, we bring focus to the use of ML in the health field but not applied on such a direct, physical level as in the previous example, but focusing specifically on language for the assessment of mental health.

In this section, we first take a look at some important concepts in ML, moving on to the stages common among most of its applications, introducing the general topics within data acquisition and preprocessing, model training and testing, and finally describing some specific learning algorithms which will be helpful to achieve a better understanding of the research that has been getting conducted in this field.

2.1.1 Learning Categories

Before moving on to the actual training and testing of ML systems, we should first introduce the distinct categories in which these algorithms are divided, which differ in the way that feedback is provided to the model's learning system. The correct outputs can be provided along with the rest of the data (supervised learning), they can be partly omitted

(semi-supervised learning), or omitted altogether (unsupervised learning), feedback may also be provided as the model interacts with an environment (reinforcement learning). In this work, we focus mainly on supervised learning since it tends to produce the best results and it is the most applicable for our use case.

The approach of supervised learning revolves around building a mathematical model that successfully maps inputs to their correct outputs, to put it simply, we provide a dataset containing the elements and their correct labels, and train the model based on those, to be accurate and generalised enough to correctly make predictions on new, unseen data. Within supervised learning, we can define two types of models, depending on their outputs. If it is the case that our response variable (the value being predicted) belongs to a continuous range of values, we call it a regression model, a possible example could be predicting the probability that it will rain tomorrow (the output will belong to the continuous range of values between 0 and 1). On the other hand, if the response variable is of categorical nature, we call it a classification model, an example could be predicting to which species a fish belongs given its body measurements.

2.1.2 Data Acquisition and Preprocessing

In order to learn, we need data to learn from, it can be extracted from existing databases, already built datasets, or crawled from the internet with automated scripts to generate new datasets. Datasets are composed of samples (examples or instances of data) that normally make up the rows, and features, which are the values for each field or characteristic, represented in the columns.

It is often the case, that this data does not come in a format that is directly applicable to training ML models, and thus, some cleaning must be done. In fact, this stage can oftentimes be the most time consuming for the developers of ML applications. Data cleaning operations often include the removal of duplicate entries and possible outliers, the handling of entries with missing features (by either generating a synthetic value, removing those entries or removing that feature altogether), and mapping non-numeric values to numbers. Data normalization, or the transformation of some feature's values to some fixed scale, can also be performed since it is beneficial for some algorithms to work with smaller numbers.

As smaller datasets tend to lead to worse predictions, an option that is very useful in extracting as much information as possible from a small dataset is called data augmentation. It basically introduces new data elements to the existing dataset, generated by performing a transformation on the original elements. The transformations vary depending on the data at hand, if our dataset contains pictures we can generate new ones by applying transformations like flipping, zooming, rotating, blurring, among many others, to the original pictures. In the case of text, it gets trickier since language is not as simply defined as a set of pixel values, but we can still perform transformations such as back translation (translating the existing text to another language and then translating

that result back into the original language), synonym replacement, randomly swapping or deleting words, etc.

At this stage, dimensionality reduction techniques of two types might be applied if the data contains too many features. Feature selection based techniques try to find a subset of the most relevant features, reducing dimensionality by simply discarding some features, while feature extraction approaches apply transformations to the existing features to achieve new ones which fit in a space with fewer dimensions.

2.1.3 Model Training and Testing

When the data is completely ready to be used, we can begin the training, this is when the model's parameters are tuned in a way that the final model can be tested and trusted to produce reliable predictions. From a high level we can make the analogy that in general, training in ML is similar to the way that humans train and learn. Throwing a basketball at a hoop for example, we throw it from many different locations, evaluate the results by some metric ("did we score?") and then we try to correct our perceived mistakes by tuning our parameters for the following attempts (apply more/less force, aim more to the left/right, and so on) until we get to a point where we can somewhat reliably score from any location, even those we didn't train from at all.

We will take advantage of this analogy to introduce some important concepts regarding models, those of overfitting and underfitting. Sometimes when training a model, we may get to a point where it can perform good predictions with samples from the data which it was trained on while failing to do so with new unseen data, this phenomenon is called overfitting, it means that the model is too focused on the training examples and is not generalised enough, in a more abstract way, it is memorising the data instead of learning from it. In our analogy this would be equivalent to our player learning to shoot very well from the few locations he trained shooting from but failing to do so from new locations. We may also arrive at a model which completely fails to generalise for unseen data but due to the fact that it learned too little from training, not capturing the nuances of the training data, this phenomenon is called underfitting. Again, in our basketball analogy, this would correspond to the player just not learning enough about how to shoot a basketball and thus producing poor results. Ideally, we want a model that generalises well, where neither overfitting nor underfitting occur.

Training

To assess the reliability of our model we need to train it and to perform tests on it using some evaluation metrics.

But before that, we must answer the question of "what data will we train and test our model on?". A simple solution would be to train and test on all the data present in our dataset, without any division. But we simply would not be able to tell if the model is overfitting in that scenario. To handle this issue, the solution is to divide the full dataset

into subsets to perform training and testing separately, this is called Cross-Validation (CV). With this approach we are able to test our model with data that it has never seen before to truly assess its effectiveness. In its simplest form, CV consists of dividing the dataset into a train and a test set, though in this case, it is a common practise to also divide the train set further into train and validation set, so that we use the resulting training set to train the models and the validation set to evaluate and tune the hyperparameters, once these are optimized, we are free to train a model with the optimal hyperparameters on the full train + validation sets, using the test set for a final evaluation that emulates a real world scenario.

To go one step further, we may use what is called K-Fold CV, in this approach, after splitting the full dataset into train and test sets, instead of dividing the resulting train set into train and validation once, we can split it into k subsets (or folds) and train/evaluate k times, using in each iteration k-1 of those subsets for training the models, reserving 1 which is held-out for validating. Once we have executed all k iterations, we aggregate the results of each (by averaging the resulting metrics for example) to evaluate the choice of models or hyperparameters used.

Evaluation

In order to evaluate models, objective metrics are required to somehow measure their performance. A comprehensible solution is to build what is called a confusion matrix, it can be extended to more classes, but for the sake of simplicity let us think of its simplest case, a binary classification context with classes Positive and Negative. Let's say we trained our model and we ran our test samples and got the outputs, now we can group them into four categories, visualized in the confusion matrix of table 2.1:

Table 2.1: Illustration of a confusion matrix and its components: True Positives (TP), False Positives(FP), True Negatives (TN), and False Negatives (FN).

		Actual class	
		Positive	Negative
Predicted class	Positive	TP	FP
	Negative	FN	TN

In this case, True Positive (TP) constitutes the number of testing examples where both the correct label and the predicted label are equal to Positive, likewise True Negative (TN) represents the number of samples where the correct and the predicted are both Negative. False Positive (FP) are those examples whose correct label is Negative but the model predicted to be Positive, and on the other hand, False Negative (FN) are those whose correct label is Positive but the model predicted to be Negative.

To summarize, TP and TN regard the number of correctly classified examples while FP and FN regard those that were misclassified. These concepts are often used in many

metrics. One example is accuracy, which ranges from 0 to 1 and has the following formula:

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

Another two very common metrics that arise from these concepts are Precision (P) and Recall (R). Precision shows the percentage of examples assigned positive that were correctly classified, and is given by the following formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall shows the proportion of actual positives that were correctly classified, given by:

$$Recall = \frac{TP}{TP + FN}$$

To really evaluate a model, both metrics must be taken into account, though in some instances one may be more valuable than the other, for example, if for some reason we really want to avoid FP, precision is a more valuable metric, on the other hand, if FN are more undesirable in our context, recall is favoured. Since good results on one of these two metrics often come at the expense of poor results on the other, a solution is to calculate the very common F1 or F-score measure, which is a weighted average of both measures that ranges from 0 (worst) to 1 (best), given by the formula:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

When training a binary classification model outputting probabilities, we often assign a threshold on these probabilities to aid in classification, for example with a threshold of 0.5 if a test on some data outputs 0.8 we assign it as Positive since $0.8 > 0.5$, likewise a 0.3 would warrant a Negative classification since $0.3 < 0.5$, addressing the cases equal to 0.5 to one of the classes. This value, 0.5 in this example , is the decision threshold, and its adjustment is fundamental for the model to achieve the desirable results (depending on what we penalize more FP or FN).

The Receiver Operating Characteristic (ROC) curve is a good way to visualize how different decision thresholds produce different results. For this metric, we must calculate the True Positive Rate (TPR) which is just another term for Recall and the False Positive Rate (FPR), given by:

$$FPR = \frac{FP}{FP + TN}$$

For each decision threshold, we plot the FPR on the X-axis and the TPR or recall on the Y-axis. When all points have been plotted for all the tested decision thresholds, we connect the points, achieving the desired ROC curve, examples of ROC curves are illustrated in figure 2.1. A good method of comparing different models on the same task is by comparing their Area Under the Curve (AUC), a value between 0 and 1, where a

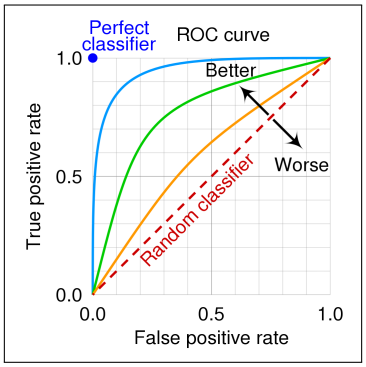

Figure 2.1: Illustration of various ROC curves [5].

higher AUC implies a better ability to make correct classifications over a model with a lower AUC. With an AUC of 1 implying the model is perfectly classifying all examples, in contrast, an AUC of 0 demonstrates that the model misclassifies every example, and an AUC of 0.5 implies that the model is the least informative of all (since even in $AUC < 0.5$ cases, predictions can just be flipped), a model which is not very useful since its predictions are seemingly random, like a coin flip.

2.1.4 Algorithms

With this general introduction out of the way, we can now move on to a brief description of some algorithms. To have a general idea of the ML algorithms most commonly applied in the area, in a review of the research works on the application of data mining algorithms in mental health in 2018, Alonso et al. [6] presented the most common algorithms employed in works involving some of the most prevalent mental illnesses like Dementia, Alzheimer, Schizophrenia and Depression. Though the results vary on the target mental illness, in the case of depression (among these the most relevant one to the scope of this work), the studies analysed employed a mix of the following algorithms: Linear Regression, Logistic Regression, Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), Support-Vector Machine (SVM), K-Nearest Neighbors (KNN) and Neural Networks (NN). A different literature review done in 2019 [7] found that the most frequent models used with social media health data (not limited to the mental health context but it is well represented) were Logistic Regression, SVM, NB, ensemble methods and Deep Learning (DL).

Given that ML is a very vast area with many algorithms, and it would be impossible to go into all of them in detail, a high level description of the intuition behind them is provided for some of those which were found to occur frequently in the literature in this context, which will consequently be relevant in section 2.3 and the rest of this document.

Logistic Regression

Logistic Regression (LR) is one of the most fundamental ML algorithms, it is a supervised probabilistic classifier. It is considered a linear model since the output depends on a linear combination of the input features, and a discriminative model, meaning that it only focuses on distinguishing between classes, not necessarily learning about them (as in generating a descriptive model of each class). It is normally used in binary classification problems, though its multinomial form can be applied for multi-class classification. It works by finding a decision boundary, composed of a linear combination of the input features and a weight vector w plus a bias b, passed in a sigmoid function which will return a probability between 0 and 1 of it belonging to one of 2 classes. The sigmoid function σ is given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

For an input feature vector x, the resulting output $h(x)$ will be given by:

$$h(x) = \sigma(x^T w + b)$$

In the general binary classification case with a class Positive and a Negative, a probability between 0 and 1 is predicted and a decision threshold t is chosen. Results between t and 1 are assigned Positive and results between 0 and t are assigned Negative. The model learns by minimising a cost function C, of cross-entropy, also called log-loss, which compares the predicted output $h(x)$ with the correct one y, given by:

$$C(h(x), y) = -\log h(x) \; if \; y = 1$$

$$C(h(x), y) = -\log (1 - h(x)) \; if \; y = 0$$

This cost function is minimised with the use of gradient descent in order to find the optimal weights and bias. Gradient descent is useful in finding a minimum value of a function (C in this case) by determining in which direction the function's slope rises most abruptly, and moving the opposite direction since the goal is to minimise C, not to maximise it. Since in this case, C is convex, it has just one minimum point, meaning the algorithm will converge to the optimal solution, though in the case of other models, gradient descent can get stuck in local minimums.

Naive Bayes

Naive Bayes (NB) classifiers consist of a family of classification algorithms working under the same basic principles, where learning happens by building probabilistic models that can compute a posterior class probability, the chance that the data element x is of a certain class c, given its n features. These probabilities are built around the Bayes theorem

and thus, calculating the posterior class probability boils down to computing:

$$P(Class = c|X = x) = \frac{P(x_0|c)P(x_1|c) \ ... \ P(x_n|c)P(c)}{P(x)}$$

Similarly to LR, NB is a supervised probabilistic model, but instead of being discriminative, it is generative, meaning that the model captures patterns of each class and classifies new data by checking which classes' patterns fit it best. The "naive" part comes from the fact that the model assumes that each feature has an equal and independent influence on the outcome. Basically assuming that no pair of features can be considered dependent on one another, and every feature contributes with the same weight in the prediction calculation.

Working with small datasets, when computing the prediction for a new piece of data that contains a feature x_i with a value z that was not previously seen in training, this feature's part in the formula will be

$$P(x_i = z|c)$$

but since z was never seen for this feature, the probability will be 0, leading the final prediction to 0 since it will zero out the whole numerator. A technique called Laplace smoothing addresses this, assigning by default a small probability to events such as the one previously mentioned by adding a fixed number to each of the $P(x_i|c)$ numerators, making it so that the prediction will not inevitably lead to 0 for unseen feature values.

Decision Tree

A Decision Tree (DT) is one of the more "readable" machine learning algorithms since the predictive model produced is often easy to go through even for a human. The name stems from the fact that the resulting model can be intuitively represented with the structure of a tree, with a root node on top, that splits into other decision nodes that eventually split into leaf nodes, those that do not split any further. In this structure, the root node (also a decision node) represents the entire population or sample, its contents will be split into sub-populations, meaning other decision nodes, according to the results on a splitting test. With every split operation the remaining population becomes smaller, and when a leaf node is reached, the model is ready to assign its elements to a certain class. This is easier to fully understand with a visual example, as shown in figure 2.2.

Its hyperparameters mainly address how the model decides which features should be considered to split on and how the splitting condition is chosen, as well as the conditions for when splitting can or cannot occur.

When training, the algorithm chooses which feature to split on at any point based on the metrics of the chosen criterion, typically gini impurity or information gain. To put it simply, these measure the quality or purity of the populations in the decision nodes that result from a given split, to ensure that at each step, the resulting split divides the initial

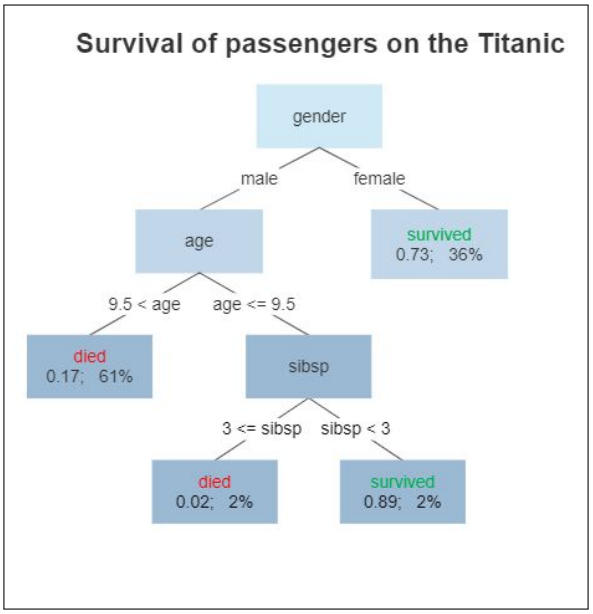

Figure 2.2: Illustration of the resulting Decision Tree model predicting the probability of a given Titanic passenger surviving the accident [8].

population into sub-populations that are homogeneous within themselves but different when compared with other sub-populations. Afterwards, in order to classify some unseen data element, one must simply descend the tree from the root node following the decision nodes whose split condition is satisfied by the features of the data element whose prediction is being made.

This type of model is simple and understandable though it often get outperformed by other algorithms on the same tasks, to combat the unreliability of a single DT, a RF or Extra-Trees Classifier (ET) approach can be taken, where the predictions depend on the results of multiple DTs, producing more accurate results at the cost of some interpretability.

Support Vector Machine

A Support-Vector Machine (SVM) is an algorithm, that when given examples of n-dimensional labeled data will try to find the best hyperplane (n-1 dimensional) called the decision boundary, that correctly separates the space into two, one side for each class, while trying to maximise the distance from the hyperplane to the closest points of either class, called the margin. Its name comes from the fact that it relies on support vectors, the hardest points to classify and those closest to the separating hyperplane which influence it the most.

In some cases, the hyperplane can be arrived at linearly, meaning that the decisions are based on the result of simple linear combinations of the features. In other cases though, this separation cannot be achieved in n-dimensions and a mapping to a higher dimensional

space is required. In that new space we may find a decision surface, thus constituting a non-linear solution and making the SVM also capable of non-linear classification. These transformations happen implicitly using kernel functions by what is known as the kernel trick. Since the SVM requires a lot of vector operations, doing so in higher dimensions would be very computationally expensive, but these kernel functions can simulate the results produced in higher dimensional spaces while still working in the default n-dimensions (in the case of non-linear kernels, linear kernels preserve the original dimensionality since a regular dot product is applied). A visualisation of this mapping to a higher dimension can be seen in figure 2.3.

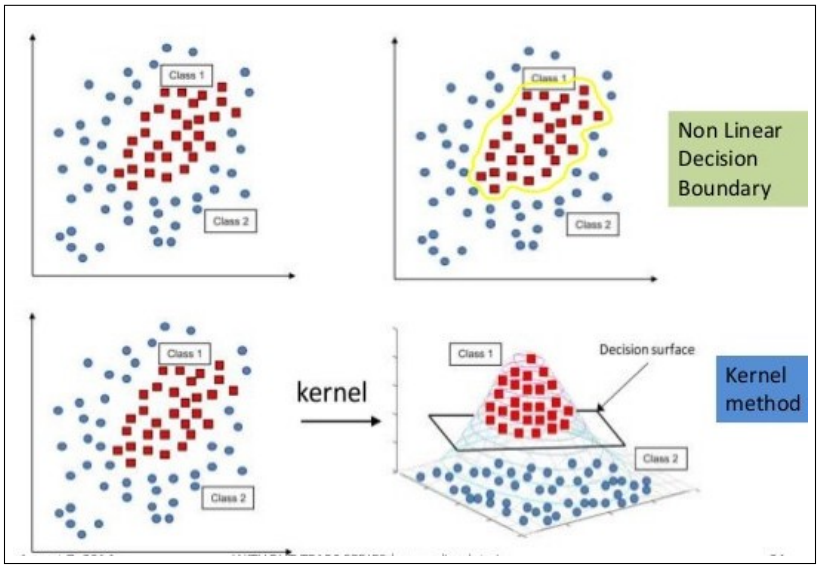

Figure 2.3: Illustration of a nonlinear decision surface being found due to the use of a non-linear kernel [9].

The model's main hyperparameters regard the kernel function to be used and the penalty of misclassifying points, both affecting the resulting decision surface. The kernel function may be linear, polynomial, sigmoid or radial, the latter of which includes the often used Radial Basis Function (RBF). A higher misclassification penalty results in smaller margins and more irregular surfaces, it may also lead to overfitting. In the opposite case, margins can be bigger since the model is more forgiving of errors and thus decision surfaces are smoother, though this case may lead to underfitting.

When it comes to classifying new data, a SVM is very fast since it must only project the data point into space, where the side which it falls on in relation to the hyperplane will dictate the result. In terms of direct use, SVMs can only be applied in problems where the goal is to find the separation between two classes, but ways have been found to extend it for multiclass problems (more than two classes). Ways often used for any binary classifier, the most common approaches are based on dividing the problem into several sub-problems (and consequently sub-models), where each sub-model will still only work

on separating the space into two but in one of two ways, they may either be trained to find the separations between one class and all other classes (done once for all existing classes) called one-vs-all, or they may be trained to find the separation between every possible combination of two classes (class 1 vs class 2, class 1 vs class 3, and so on) called the one-vs-one approach.

K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm follows a very simple logic and is capable of both classification and regression. First, the training data is mapped, often in very high dimensional feature spaces, then, when we want to make predictions for some new data element x, we plot it and measure its distance to all other data elements. The decision is made based on the properties of the k data elements closest to x, in the case of classification, the most represented class among those k elements will be assigned, while in the case of regression, the result is the average of the parameter being inferred across those k data elements.

The main hyperparameters are k, the number of closest neighbors to take into consideration (which is found usually by testing various possible values, since it has a big impact on results as seen in Figure 2.4), and the distance metric used. Something worth noting is that in this algorithm, most of the computations are done when making predictions for unseen data points, earning it the label of "lazy learning". The combination of a large training dataset (forcing more computations) and a high number of features (making each computation more expensive) can make it inadequate for larger problems which require frequent predictions. As a consequence, it greatly benefits from data and dimension reduction (essentially summarizing the dataset, and reducing the number of features per element respectively).

Artificial Neural Networks

Artificial Neural Networks (ANNs), often just called Neural Networks (NNs) are models capable of capturing complex patterns in data, inspired by the connections of a biological brain. These models are composed of nodes often called neurons which are arranged in layers, the first one which receives the input data is naturally named input layer, the last one which outputs the final result is called the output layer and those in between are called hidden layers. A NN with more than one hidden layer is considered a Deep Neural Network (DNN).

Excluding input nodes (those present in the input layer) which receive data directly, all other nodes in the network receive as input the output of neurons from previous layers (changed based on the weight of the connection), these connections are illustrated in figure 2.5. In each node, these numbers are received, summed, added to a local bias and then passed through an activation function which will determine its output. This is all done with matrix operations, let us say X is a vector containing the outputs of the neurons

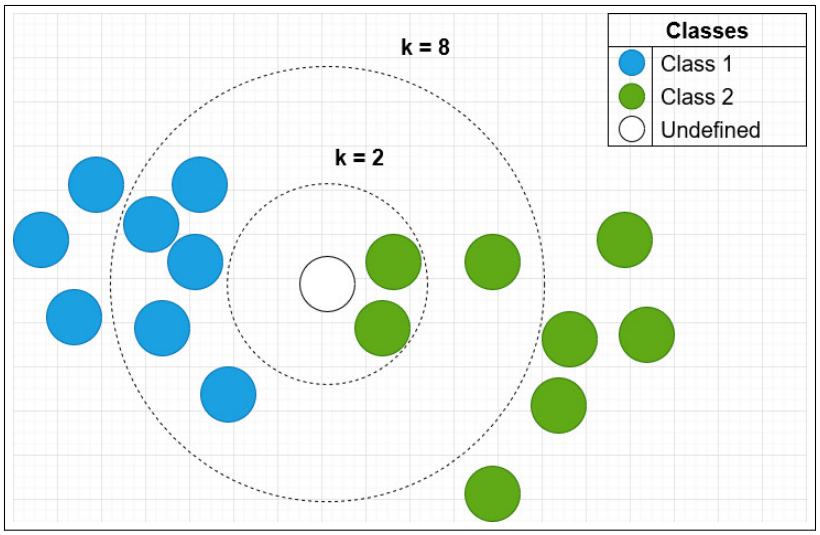

Figure 2.4: Illustration of the KNN algorithm and the impact of different k values. In the case of k=2, the white circle is assigned class 2 while in the case of k=8 it is assigned class 1.

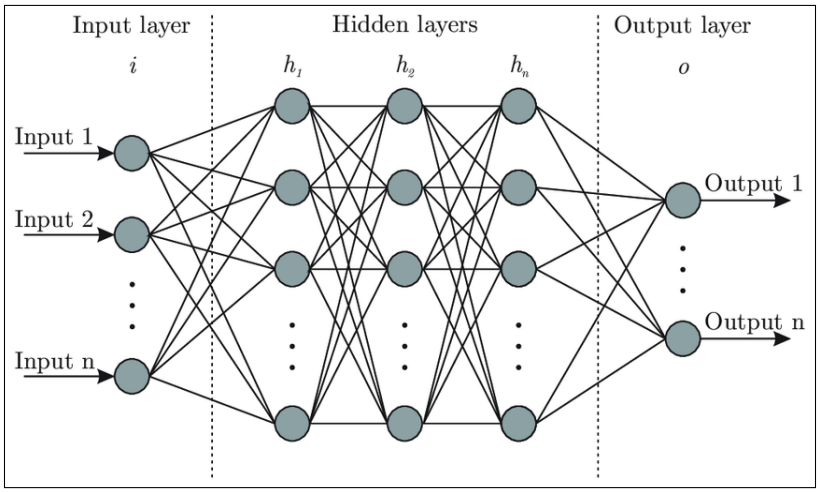

Figure 2.5: Illustration of the architecture of a Neural Network with n input nodes, 3 hidden layers and n output nodes [10].

in the hidden layer n connected with the weights given by the vector W to the node z_i in the hidden layer $n+1$ with bias b and activation function a, this node's (z_i) output $h(X)$ will be given by:

$$h(X) = a(X^T W + b)$$

In order to make a prediction, the input must propagate forward through the network, where each layer will produce results that are used to compute the results of the following layer. The goal is to tune each node's weights and bias which are initialised randomly, to make good predictions. This happens by a process called backpropagation, where by applying gradient descent (or variations of it) on a cost function with respect to each neuron's parameters (starting from those in the last layers) small corrections are made in each iteration leading to smaller cost function values. Though in this case, the gradient descent method can get stuck in local minima. The hyperparameters are the network's architecture, i.e. number of layers and number of nodes per layer, as well as the activation functions, learning rate, loss function, and optimizer. Additional hyperparameters regard the amount of training done and how big the corrections made at each stage are.

There are various types of NNs, in the context of classification in this work, the most relevant type is the typical feed-forward Neural Network, though as we will see Recurrent Neural Network (RNN) also appear sometimes since our data is of a sequential nature. In the feed-forward type, information flows in one direction while in RNNs there are loops, meaning that output data can be fed back as input (justifying its use with sequential data since future predictions are affected by past predictions).

2.2 Natural Language Processing

Natural Language Processing is the sub-field of computer science focused on getting computers to understand human speech. Substantial progress has been made here, from systems with complex sets of hand-written rules in the second half of the last century, to systems taking advantage of ML algorithms, which as of late are dominating the field, mostly based on NNs.

NLP tasks handles unstructured text with different possible goals, some of which being: Information retrieval: finding in a big collection of documents those more relevant for a given query given their text contents; Text classification: assigning text to different classes; Question answering: generating answers for a given question; Sentiment analysis: extracting emotion or affective states from text; Machine translation: the task of translating text from one language to another.

2.2.1 Data Preprocessing

In the context of natural language, the data must be transformed or preprocessed into representations appropriate for the machines to work with. The steps usually entail some form of basic transformations, tokenizing, filtering, stemming/lemmatization and

vectorization.

The basic transformations usually aim at making the data simpler to work with while trying to avoid the loss of relevant information, examples of this are transforming uppercase characters to lower case, removing non-English and other irrelevant characters/tokens depending on the context of the work, like URLs and some forms of punctuation. Some transformations can also be applied for anonymizing subjects when working with sensitive data.

In the tokenizing step, text is broken down into smaller units, this is typically done on a word-level (each word is a unit), by matching the text with regular expressions. Once we have a list of tokens, derived from the tokenizing step, we might want to filter some out. A common yet optional procedure is stopword removal, which filters out tokens that are so common in a given language that they add little information to the text in most contexts, though they have been found to be useful in others. Common stopwords in the english language are words like "I", "me", "the", "of", etc.

In order to further simplify our data, one of two somewhat similar operations can be done to further preprocess it, those being stemming and lemmatization. Stemming reduces words to their root form, following a rule-based approach, meaning it simply chops off the suffix leaving a word which might be oversimplified and possibly without any meaning. It is the faster method of the two and preferred when word meaning is not the focus, for example in spam detection. Lemmatization on the other hand, also reduces words to their root form but following a dictionary-based approach which produces a chopped word that retains its meaning, taking its context into account (results vary depending on whether a word is used as a verb, noun, adjective, etc), this may be important in tasks like question answering.

A comparison of both methods can be seen in Table 2.2 using NLTK's PorterStemmer and WordNetLemmatizer, where most notably: lemmatization turns "was" to "be" preserving meaning while stemming resulted in "wa", and lemmatization differentiated both contexts of the variations of the word "finding", as a verb and as a noun.

Table 2.2: Stemming and Lemmatizing comparison on tokens from the sentence "She was finding it difficult to report her findings".

Raw token	Stemming	Lemmatizing
she	she	she
was	wa	be
finding	find	find
it	it	it
difficult	difficult	difficult
to	to	to
report	report	report
her	her	her
findings	find	finding

After these procedures, we arrive at the step of vectorization, in other words, the process representing words with numbers, this topic is critical to the performance of a NLP application, and thus, the following subsection is fully dedicated to it.

2.2.2 Vector Representations

When find ourselves at this stage, with our final list of tokens, ready to be vectorized, i.e. turn them into vectors of numeric values, we have to choose among several vectorization approaches.

Bag of Words

A simple Bag-of-Words (BoW) one-hot encoding can be used here, where there is a matrix with as many columns as there are words in the vocabulary and as many rows as there are documents (collections of words), with value 1 if the token of a given column exists in the document of a given row and 0 otherwise. Here, the rows are document representations and the columns are token representations. The BoW name comes from the fact that text is treated exactly as such, simply assessing the presence of the tokens.

It's easy to see how this would fail to scale properly, if managing a large corpus of text, huge vectors would be required and they would be comprised mostly of 0's. Another weakness of this method is that it completely disregards order and thus fails to capture many relations between words, though this can be somewhat mitigated by having a vocabulary of n-grams, where each text unit is composed of n-words or n-characters, for example, the phrase "I like the cold weather" broken into 2-grams of words results in "I like", "like the", "the cold", and "cold weather".

Something commonly done in these simpler approaches is storing the words counts or frequencies in the corpus instead of the binary presence/absence, but this means that

words that do not provide a lot of value but appear very frequently dominate the vectors and may sway results on the NLP task at hand. To counteract this and improve on these simpler approaches, weighting methods can be applied like the Term frequency-Inverse document frequency (Tf-Idf), which scores words based on how common they are in a document compared to the overall corpus, assigning higher weights to words that occur more often within fewer documents. For a given term t, document d found in corpus D:

$$Tf(t,d) = \frac{occurrences\ of\ t\ in\ d}{total\ occurrences\ in\ d}$$

$$Idf(t,D) = \log \frac{|D|}{|documents\ in\ D\ with\ t|}$$

$$TfIdf(t,d,D) = Tf(t,d) \times Idf(t,D)$$

Distributional Semantics Word Embeddings

More sophisticated methods have surfaced in the last decade, most notably Word2Vec, GloVe and FastText emerged to generate word representations called distributional semantics word embeddings, that manage to capture their semantic meaning through the context in which they occur in a large corpus. In the words of the linguist J.R. Firth, "You shall know a word by the company it keeps". Closely related words are kept closer together in vector space with interesting relational properties encoded into them as seen in figure 2.6. It is worth noting that due to the necessity of training with large corpora

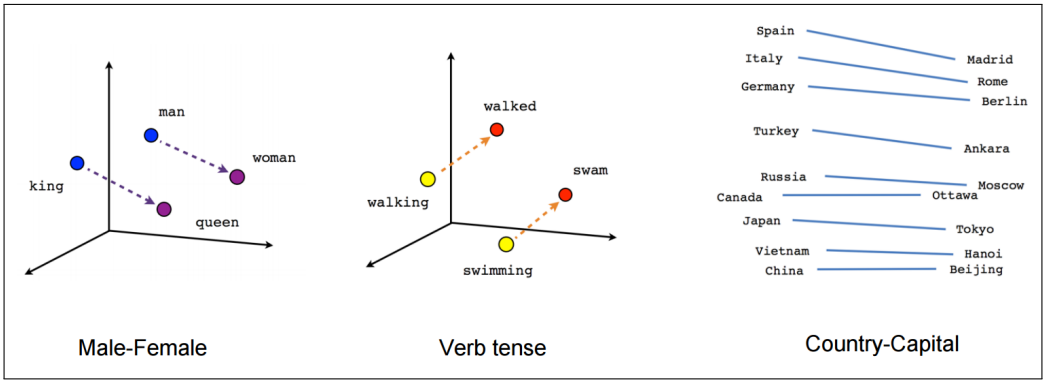

Figure 2.6: Illustration of some properties related to word semantic meaning captured by word embeddings [11].

to find quality word embeddings with any of these methods, all of the following provide pre-trained word embedding models that can be used in plenty of NLP tasks directly as a look-up table or after some fine-tuning to better fit the embeddings to the use case at hand.

Word2Vec [12], developed by Google researchers in 2013 works by training a NN with either of the skip-gram or continuous-Bag-of-Words (cBoW) approaches over a large un-

labeled corpus, by breaking down the text and building tuples containing a context and a target. These will serve as the labeled data that the network will train on. From a high level perspective, the skip-gram model's NN is trained on the task of finding words that are likely to appear in the context of a given input word, while the cBoW way of doing things consists of training the NN to predict a likely target word given the surrounding context (neighboring words in a phrase) as input. A visual explanation can be seen in figure 2.7.

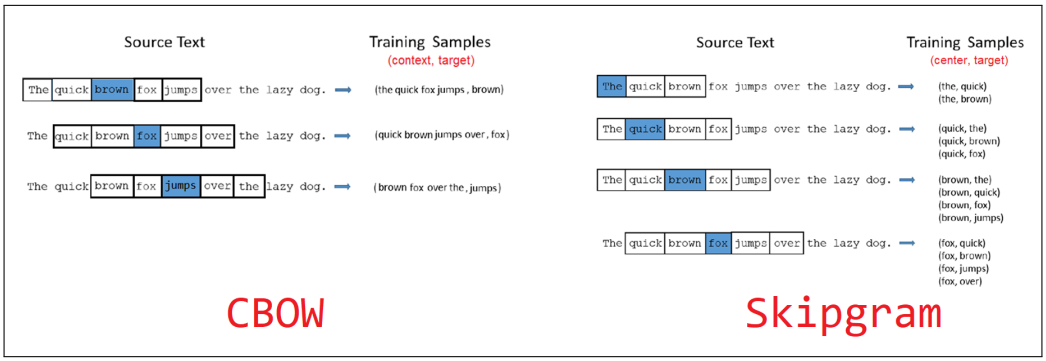

Figure 2.7: Comparison of the different learning tasks of the cBoW and skip-gram models [13].

Later in 2014, GloVe [14] (Global Vectors), developed by a group of Stanford researchers, represented an improvement over Word2Vec due to not being limited to local information as its predecessor was, it captures both local and global statistics based on word co-occurrence. The basic idea is to have a word-context co-occurrence matrix (WC) where each entry (w,c) represents the amount of times that word w occurred with context c. From this point, a form of matrix factorization is applied to break down the matrix to the product of two other matrices, a word-feature matrix (WF) and a feature-context matrix (FC) to achieve something like the illustration in figure 2.8. These WF and FC

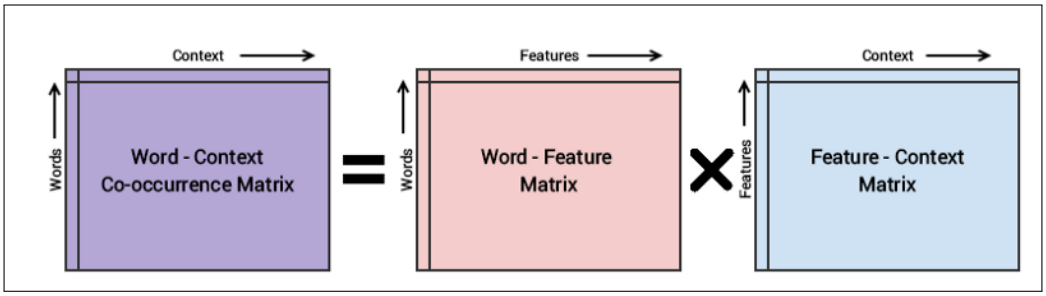

Figure 2.8: Illustration of the intuition behind GloVe's training process [15].

matrices are initialised with random values and trained using Stochastic Gradient Descent (SGD) to minimise the error in order to get to a point where their multiplication is considered close enough of an approximation of the WC matrix. The resulting WF matrix

contains the word embeddings which are later used for NLP tasks.

This brings us to 2015 when FastText [16] was initially released by Facebook AI researchers, which improved on the Word2Vec approach by developing a fast and more adaptable model that can also handle words not seen in training called Out-Of-Vocabulary (OOV) since it treats every word as a bag of sub-words. Every token is transformed in a way that signals its beginning and end, for example "where" becomes "<where>" and from this point n-grams are generated which along with the full word itself will be used to calculate its representation. To further exemplify, with 3-grams and the word "where", the resulting tokens would be "<wh", "whe", "her", "ere", "re>", and "<where>", though it is worth noting that FastText uses n-grams of varying lengths. A word's representation is the result of the sum of the representation of all its n-grams. Much like Word2Vec, it can also be trained on skip-gram or cBoW.

Contextualised Language Models

Research did not stop there, and as of recently there has been an influx of contextualised language models, notably Embeddings from Language Models (ELMo) [17], Generative Pre-trained Transformer 3 (GPT-3) [18] and Bidirectional Encoder Representations from Transformers (BERT) [19], that keep the interesting properties of the previously mentioned Distributional Semantics Word Embeddings (DSWE) of capturing a word's semantic meaning, while also capturing the context in which they are used. For example, the token "fair" in "he went to the fair" and "the game is not fair" would be represented by the same vector with the DSWE methods of the previous section (often called static word embeddings), despite obviously having different meanings, Contextualised Language Model Embeddings (CLME) manage to address these nuances (dynamic word embeddings).

Such language models are derived from an architecture which is very common in NLP, that of encoder-decoder. This architecture is very used in seq2seq models which are designed to handle sequential data. Using the case of the machine translation task as an example, the encoder receives text as input and creates some representation of it, which is then passed to the decoder which will generate text in the other desired language from the given representation. Both BERT and GPT-3 make use of the transformer architecture [20], introduced in 2017 by a team of Google's DL researchers which discarded the use of RNNs or RNN-based models like the Long Short-Term Memory (LSTM) (that ELMo is based on) in encoder and decoder alike, preferring to use just the attention mechanism as the title of the paper suggests. Since this type of model does not process each token sequentially as a RNN would, rather processing all in parallel with their position encoded into them, the whole training process can be done faster than it would with previous approaches.

We will now briefly describe BERT in specific since it revolutionised the NLP world by achieving state-of-the-art performance on arrival, while still being relevant today, since many other models emerged with it as inspiration, such as ALBERT [21], RoBERTa [22]

and XLNet [23]. Some of which outperform the original BERT in various NLP tasks. BERT [19] is trained on two tasks, the first entails the guessing of masked words (where around 15% of the tokens are replaced by the "[MASK]" token), i.e. predicting words that have been purposely hidden given their left and right context (bidirectionally). The other is next-sequence prediction, given two sequences the model must assess whether they occur next to each other in the corpus. During training there is a 50% chance of that the second sequence is the following sentence and consequently, a 50% chance that it is just some other random sentence. BERT is constructed in a way that makes the fine-tuning of the pre-trained model to specific NLP tasks easier since it can accept one or two sequences as input if necessary (useful for question-answering or machine translation for example) separated by the "[SEP]" token, to add to this, every sequence's first token is a special classification token "[CLS]" whose final hidden state can be used as the input's representation in classification tasks. It is also worth noting that BERT takes sub-words as input, finding a medium ground between approaches using words and approaches using characters as input, to get the best of both worlds, performance from the side using words, and the capability of handling out-of-vocabulary tokens from the character approach.

2.3 Relevant works

A lot of work has been done with ML in the mental health area, with many works proposing and implementing models to detect or predict instances of mental illnesses such as depression, schizophrenia, anxiety and PTSD, as well as related behaviours, like suicide and self-harm. Though most of the works found tend to focus on the detection of depression and suicide-risk.

Although studies have been made with different data sourcing approaches, the use of social media user generated data has been gaining popularity among researchers. The most popular platforms used, by far, are Reddit and Twitter, likely due to their conversational nature and public APIs. Within studies using social media based data, the type of features used often varies as well, with some researchers training their models on user posted images (visual features), user posted text (textual features), user characteristics (descriptive features), or combinations of them. There has also been a type of feature gaining popularity recently as we will soon analyse, that of sentiment analysis, which captures emotions or affective states from text.

For instance, Coppersmith et al. [24] implemented a model trained on social media data (Twitter) capable of distinguishing users who attempted suicide from their demographically matched control counterparts with good results (AUC in the 0.89 to 0.94 range depending on the amount of data used). The authors chose to use the original pre-trained GloVe embeddings as the vectorization tool, later fine-tuned in the training process (to better capture the language used in social media), along with a NN-based approach (bidirectional LSTM) to output the likelihood of the author of the analysed text being at risk of suicidal behaviour. One interesting aspect of this study, is that excluding

more recent data (in the study the data from 90 days prior was removed) and thus considering only older data (from the past 180 days excluding data from those 90) leads to roughly comparable results, leading one to conclude that it is capturing mainly suicidal traits instead of suicidal states, allowing for a longer term screening which is safer and easier to intervene on rather than relying on a system that would only detect individuals in an immediate state of crisis. The decision threshold may be tweaked to address false positives at the expense of false negatives and vice-versa. Despite the fact that it is hard to get a fair comparison, the model's results seem to outperform those of traditional screening methods, where some studies indicate that 4% to 6% of those who go through with suicidal behaviours were found to be at risk, compared to the 40% to 60% flagged by this model (using a 10% or 1% false alarm rate respectively).

Robert Thorstad and Philip Wolff [25] trained and tested a LR model on a Reddit-sourced dataset to assess how useful a user's everyday language, in both a clinical and non-clinical context, is for detecting present or predicting future mental illnesses. Achieving results better than chance in all experiments, with varying degrees of success. In order to do this, they gathered posts from several clinical and non-clinical subreddits, and considering the clinical subreddit each user frequents as a proxy for a diagnosis (e.g. user who visits r/depression is considered depressed), they trained models capable of assigning a user to one of the four clinical diagnosis (ADHD, anxiety, bipolar disorder and depression) in three different scenarios :

- Case 1: taking posts in a clinical context (the corresponding clinical subreddit).
- Case 2: taking posts from the same users but in a non-clinical context (non-clinical subreddits).
- Case 3: taking posts from the same users in non-clinical contexts, but only those made prior to the date of the user's first interaction with a clinical subreddit (considered a proxy for a diagnosis).

The authors followed a BoW approach with the Tf-Idf weighting methodology to perform word vectorization. It is also worth mentioning that in the preprocessing stage, explicit mentions of the mental illnesses were removed from the dataset. Case 1 achieved the best results, with F1-score=0.77. This was expected since users are more likely to use relevant language in clinical contexts. The second and third cases performed significantly worse (but still better than chance), with case 2 achieving F1-score=0.38 and F1-score=0.36 for case 3. Additionally, a split-half analysis was performed where models were trained with case 3's data divided in two groups for each user, one containing recent and the other older posts. In this experiment the more recent posts performed slightly better than the older ones (F1-score=0.344 vs F1-score=0.338). An additional analysis was performed on the models most predictive words for each class, revealing several clusters of words, some of which coincided with major symptoms present in the DSM-5 of the corresponding illness.

In order to develop a more generalized depression detecting model based on textual

features extracted from social media, Chiong et al.[26], inspired by successful models developed by other authors in their previously analysed literature, trained different models on 2 Twitter datasets (training each model separately for each of the 2 datasets) and tested them on 3 non-Twitter datasets containing only records of the class "depressed", meaning no non-depressed instances. The first consisting of the text present in the electronic diary of a depressed 17 year old girl that committed suicide, the second based on Reddit posts, and the final one on Facebook posts. The experiments were conducted on the 4 best performing classifiers analysed in previous research, those being LR, Linear Kernel SVM, Multilayer Perceptron (MLP), and DT. Features were extracted with a BoW approach, counting the instances of n-gram words with n ranging from 1 to 3 (from unigrams to trigrams).

Although the results were looking very promising using 10-fold CV on both training datasets, their performance on the 3 other datasets was subpar, achieving accuracy values below 50% across the board. Upon further investigation, the researchers found that this was due to the way that both training datasets were built, the first dataset gathered data from users that had posts with variations of "I'm/I was/I am/I've been diagnosed with depression" to build the depressed class, while the second gathered data from users with "depression" in their posts as an indication that they belonged to the depressed class. This resulted in all the records containing some variation of "depression" or "diagnosis", which resulted in the models distinguishing both classes mainly on the presence or absence of such words, which were nowhere nearly as present in the 3 testing datasets leading them to miss less explicit cues, and consequently, bad accuracy results (0%-13.39% on the diary dataset, 4.73% - 49.86% on Reddit data and 13.35% - 31.35% on Facebook data).

Upon removal of the words "diagnose" and "depression" in training, the results were far better in the generalized tests across all models (27.42% turned into 69.19% for the diary data, 46.17% into 90.40% for Reddit data and 64.30% into 70.44% for Facebook data), though as expected, there was a decline in performance in the CV performed on the training datasets. This observation further enforces the importance of carefully tailoring the training data to minimise the differences with the intended application scenarios (testing datasets in this case). The authors also tested the effectiveness of under/oversampling in the twitter training datasets, since one was heavily imbalanced while the other was just slightly imbalanced. They found that it improved the performance of the models trained on the heavily imbalanced dataset detecting the less populated class (at the expense of performance in the most populated class), while having minimal impact in most tests of the models trained on the slightly imbalanced dataset.

As mentioned previously, sentiment analysis has found its place in mental illness classification on social media, an example of a paper relying solely on this type of feature to do so is the one by Chen et al. [27] where the authors collected 1 year worth of tweets from users that had publicly expressed their depression diagnosis and non-depressed users alike. This data was used to monitor 8 basic emotions and their changes over time to find individuals suffering from, or at risk of developing depression. The authors used a tool

called EMOTIVE [28] to extract a strength score for anger, disgust, fear, happiness, sadness, surprise, shame and confusion from tweets, along with a measurement called emotion overall score which is just the sum of all the emotions' strengths. From these measures, 2 feature sets were built. A non-temporal one, referred to as EMO, which contains the measures mentioned found in all tweets of each user (aggregated). The other, taking into account changes over time (using a day as a unit of time) by creating 9 time series (one for each measure) per user. Various metrics (mean, standard deviation, entropy, mean momentum and mean difference) taken from these time series were used as features, creating the feature set referred to as EMO_TS.

LR, SVM, NB, DT, and RF models were trained using a mix of EMO and features drawn with LIWC [29]. The best results were achieved by SVM and RF models using EMO and LIWC features with accuracies around 88%. Given that these were the best performing models, only these were tested with EMO_TS, where the authors found that despite a mix of EMO, EMO_TS and LIWC features achieving the best results (accuracies of 89.71% and 93.06% for SVM and RF respectively), those with just EMO and EMO_-TS were nearly as good. Suggesting that discrete and temporal emotional data contain relevant information to capture depression.

Aladag et al. [30] used various different features to detect suicidal forum posts, by extracting data from the subreddits r/SuicideWatch, r/Depression, r/Anxiety and r/ShowerThoughts. Data from these subreddits was randomly selected and manually annotated, labeled with 1 if its author expressed suicidal thoughts, and 0 otherwise. From here, a mix of LIWC, sentiment analysis and BoW with Tf-Idf features were used to train LR, RF and SVM models, which were compared with a baseline model that just classified all posts as suicidal.

Four experiments were made in total. In the first one, the ability to distinguish suicidal talk from regular everyday talk was tested by training the models on data only from r/SuicideWatch and r/ShowerThoughts. The second experiment introduced posts from r/anxiety and r/depression to the dataset used in experiment 1, some of which were considered suicidal but in a more subtle manner than those of r/SuicideWatch, to measure if the models could capture suicidal patterns in text that were not as extreme. Experiment 3 tried to assess whether assuming every post from r/SuicideWatch was suicidal (labeled 1) and every post from r/ShowerThoughts was not suicidal (labeled 0) would still lead to quality predictions on posts from those subreddits, this allowed the models to be trained with more data since the training data was not fully annotated. Experiment 4 was basically testing the models trained in experiment 3 on all annotated posts collected from all subreddits mentioned in the paper. As expected, the models performed the best in experiment 1 with F1-scores in the 0.89 to 0.92 range. Experiment 2, due to more difficult cases, resulted in worse though still decent performance with F1-scores in the 0.73 to 0.81 range. Experiment 3 proved that it was indeed safe to make those assumptions regarding those 2 subreddits since the models still achieved F1-scores in the 0.83 to 0.92 range. Finally, with the introduction of more difficult cases in experiment 4, there was a

drop in F1-scores across the board but the results were still very respectable, in the 0.75 to 0.79 range. Showing that the features used could in fact capture some nuances in text to assess the user's risk of suicide.

Yazdavar et al. [31] adopted a very creative approach of using semi-supervised learning to monitor depressive symptoms on social media by emulating the Patient Health Questionnaire 9 (PHQ-9), which incorporates the DSM-5 to evaluate the scale of depression over 9 items or symptoms: lack of interest, feeling down, sleep disorder, lack of energy, eating disorder, low self-esteem, concentration problems, hyper/lower activity and suicidal thoughts.

The study compared the performance of two models, one being a top-down approach, using Latent Dirichlet Allocation (LDA), an unsupervised method that finds topics in documents (with topics being based on the distribution of co-occurring words), and the other, a more bottom-up approach called semi-supervised Topic-modeling over Time (ssToT) which is based on the previous one but enriched by using words related to those 9 items as seed terms for the topical clusters and by limiting them to only 1 topic each, to try to force the topics to correspond to each of those 9 depression items (where they could be mixed in the first LDA model). Tweets were manually annotated with the 9 depression symptoms, and for each user, tweets were merged into buckets by time period (of 14 days). Resulting in 10400 annotated tweets distributed into 192 buckets. The ssToT model was used to classify a set of symptoms for each bucket. Results varied for each symptom, the model performed best for "lack of interest" with F1-score=0.9 and worst for "concentration problems" where F1-score=0.3, but in general the results achieved were good, with an overall F1-score of 0.68.

Cacheda et al. [32] compared the performance of using a single RF vs a dual RF model, both trained on descriptive and textual features, to promote the early detection while penalizing the late detection of depressed users on a dataset sourced from Reddit for eRisk 2017. The authors found that using two different models with one threshold function each, where one is tasked with detecting depressed users while the other detects those of the non-depressed control group, allowed for faster decisions when compared to a single model with two thresholds, this was due to the fact that three possible outputs were always competing, those of: Case 1 where there is a final decision and the user is considered depressed, Case 2 where there is a final decision and the user is considered non-depressed, and Case 3 where there is no final decision and more data is needed to successfully classify the user.

With the division of the model in two, the one tasked with finding depressed users will have as outputs only case 1 and case 3 while the model tasked with finding non-depressed users will have case 2 and case 3, the overall final decision depends on which of the two models reaches a final decision first. This finding is important, since in this use case, the model must be accurate but desirably so in the fastest way possible, requiring the minimum amount of input (writings) for a reliable answer. It is also worth mentioning that while analysing the descriptive features of depressed and non-depressed users, the authors came

across some interesting patterns both regarding how the content is posted and when it is posted. Regarding how each class produces content, the data suggested that: depressed users tend to reply to existing posts more than they publish their own; non-depressed users tend to write significantly more in the title area compared to depressed users; depressed users tend to elaborate more in the text section when compared to non-depressed users.

Regarding when each class produces content, the data suggested that: non-depressed users tend to post more often than depressed users with 1 day less of average time interval between 2 posts (4 vs 5), though depressed users show much higher variability; depressed users tended to publish more than the non-depressed during the weekend + monday, with the trend being reversed for the rest of the week; the publication rate is also more homogeneous throughout the whole week for depressed users, where the non-depressed tend to fall off on the weekend, posting more on weekdays; regarding the hour of posting, depressed users tended to post more than their counterpart from midnight to midday, while the control group posted more content than the depressed users in the hours of the afternoon.

2.4 Workshops and Shared Tasks

It is worth noting, especially since it pertains to the contributions of this work, that some attention has been brought, and progress has been made in the context of events such as workshops. It was one of the objectives of this work, to participate in possibly two events, CLEF eRisk (we did end up participating) and CLPsych (did not participate). These events' influence can be seen in the literature, with many authors making use of the datasets and metrics provided by their organizers.

CLPsych is a computational linguistics and NLP focused workshop that has been hosted every year since 2014 with the goal of directing the progress in the computational area of ML towards the subject of psychology and individual behaviour. Every year there is a shared task that challenges the participant groups to implement models capable of detecting or predicting mental illness related issues in a social media based dataset. In 2021's edition, the teams were asked to develop models capable of assessing suicide risk in two scenarios (subtasks), one using the users tweets from 30 days prior to their reported suicide attempt, and the other using data from 6 months prior to the attempt.

CLEF's eRisk presents shared tasks with focus on early risk prediction on the internet regarding various topics. Every year there is a set of mental health related tasks to challenge the participants to develop models that can fulfill the task's objective. The 2021 edition of eRisk presented three tasks as per usual.

The first one, regarding the early detection of signs of pathological gambling (ludomania), where each subject's posts were received sequentially, in the same way this could be applied in real life social media scenarios, and the models had to provide classifications as early and correctly as possible. This task was "only test" meaning that no training dataset was provided. In the testing phase, after every post was analysed, models that provided

correct final decisions earlier were rewarded by the Early Risk Detection Error (ERDE) metric, taken from the work of D. Losada and F. Crestani [33]. Naturally, the more classic evaluation metrics were also used to evaluate the results. The second task revolved around early self-harm detection, its structure was similar to that of task 1, but in this case a training dataset was provided. The third and final task was about estimating the degree to which an individual suffers from depression, this was done by simulating the answers of the users present in the dataset to a questionnaire (Beck's Depression Inventory) to assess the presence of negative feelings based on their post history.

In the current edition (2022), the three tasks correspond to the early detection of signs of pathological gambling, the same as last year's task 1, the early detection of depression, and the measuring of the severity of signs of eating disorders. The first two regard binary classification (the user either is or is not at risk of pathological gambling in the first case or being depressed in the second) while the third task is more complicated, requiring the estimation of each subject's answers to the Eating Disorder Examination Questionnaire (EDE-Q) to discover the severity of the subject's eating disorder. To further add to this difficulty, no training dataset is provided for this last task, meaning that the participants would likely have to build their own.

2.5 Summary

We began this chapter by introducing the basics regarding ML processes and algorithms, moving on the NLP field more specifically, which has also evolved a lot in recent years. Afterwards, we went through some relevant works, which we summarize in Table 2.3. We observed that many different decisions must be made, at every stage of the implementation pipeline, that affect the end results. From big decisions like what model to use or what type of feature extraction method to apply, to seemingly smaller decisions like what words should be filtered out in preprocessing. All of these decisions matter and they can make or break a ML application's success. We observed that various feature types work well, even when mixed together. Regarding the models used, the results showed that more complex models do not always provide better results, and as such, there are cases where their use is not justified, as classic models such as LR and SVM tend to perform well across the board. Regarding the extraction of textual features, we saw that despite the big advances in DSWE and CLME, plenty of the works analysed relied on simpler BoW approaches, leaving us wondering how well the same models would perform with more complex feature extraction tools such as BERT. It is therefore an objective of this work, to compare the effectiveness of various vectorization methods in the context of this task.

Overall, using ML systems as screening tools for detecting mental illnesses in forum posts seems to have potential as far as the classification goes, open to new and creative approaches. It also brings up some ethical concerns, but that was not the focus of our analysis.

Table 2.3: Summary of the research analysed.

Paper	Detecting	Features	Model	Results	Takeaways
Coppersmith et al. [24]	Suicide risk	Textual (GloVe)	NN (LSTM)	AUC ranging from 0.89 to 0.94 depending on amount of data used	Captures suicidal traits over suicidal states.
R. Thorstadd, P. Wolff [25]	ADHD, Anxiety Bipolar disorder Depression	Textual (Tf-Idf)	LR	Case 1: F1-score=0.77 Case 2: F1-score=0.38 Case 3: F1-score=0.36	Similar results with all data (case 2) and only data prior to proxy diagnosis (case 3).
Chiong et al. [26]	Depression	Textual (BoW)	LR Linear SVM MLP DT	Best for each testing set: Diary: LR Acc=0.69 Reddit: LR Acc=0.9 Facebook: SVM Acc=0.7	Dataset building methodology must be taken into account. Under/Oversampling should be used on heavily imbalanced datasets
Chen et al. [27]	Depression	Sentiment analysis, LIWC	LR RBF SVM NB DT RF	Best results for 2 best algorithms: SVM F1-score=0.87 (EMO and EMO_TS) RF F1-score=0.92 (EMO_TS, LIWC)	Promising results completely based on emotion.
Aladag et al. [30]	Suicide risk	Textual (Tf-Idf), LIWC, Sentiment analysis	LR RF SVM	Best results: Experiment 1 (LR, SVM): F1-score=0.92 Experiment 2 (LR,): 1F-score=0.81 Experiment 3 (SVM): F1-score=0.92 Experiment 4 (SVM): F1-score=0.79	Removing sentiment analysis and LIWC features did not have a great impact on performance.
Yazdavar et al. [31]	Depression	Textual (BoW)	LDA ssToT	ssToT F1-score for each PHQ-9 item: 0.90 0.89 0.78 0.68 0.93 0.82 0.30 0.38 0.68	Unsupervised, only used labeled data for testing. Some items are captured better.
Cacheda et al. [32]	Depression	Textual (BoW), Semantic similarity (LSA), Descriptive	RF	Best singleton model: ERDE@5=12.89 ERDE@50=11.26 Best dual model results: ERDE@5=11.88 ERDE@50=8.68	Dividing the problem in 2 using dual models improves early detection. There are meaningful statistical differences on how each class posts content

Chapter 3

Methods

This chapter describes the approaches used in order to address the problem of detecting subjects at risk of suffering from pathological gambling and depression, corresponding to tasks 1 and 2 respectively of the 2022 edition of CLEF eRisk. As mentioned in the following results chapter, several tweaks, corrections and additional experiments have been made since the submission stage of the event, in this chapter our main focus is to document the final state of the project, though we do mention some of the changes made post-submission. In order to get a better overview of what changed, we recommend the reader to check our official eRisk lab participation paper [34] and to compare the methodology with the one described here. It is worth noting that the final state of the code used is also made available in the form of a GitHub repository[1].

3.1 Tools

Considering that many third party libraries and frameworks were used throughout the experimental stage, we saw fit to briefly introduce the most crucial ones, those being NTLK, Scikit-learn, Gensim, Sentence-transformers, PyTorch and Optuna.

3.1.1 NLTK

NLTK[2] stands for Natural Language ToolKit, its goal is to facilitate the development of programs that use human language data. It provides support for many commonly used NLP operations. In this work, we relied on it mostly on the preprocessing stage, in processes like tokenization and lemmatization.

3.1.2 Scikit-learn

Scikit-learn or sklearn[3] is a very popular ML library that includes implementations of many algorithms across different ML application stages such as preprocessing, model

[1] https://github.com/rodrigosemicolon/eShrink
[2] https://www.nltk.org
[3] https://scikit-learn.org/stable/

selection, dimensionality reduction, clustering, regression and classification. We rely on it mostly for its implementations of model selection algorithms to split our training data (namely StratifiedKFold[4]), its feature extraction tools in the form of the TfIdfVectorizer[5] and its implementation of classification models, as we will later see in more detail.

3.1.3 Gensim

Gensim[6] is a widely used NLP-focused library, made with practicality, memory independence and performance concerns in mind. In this project we used it especially for its KeyedVectors[7] which facilitate the integration (loading and application) of DSWE, in the training and testing stages of our experiments.

3.1.4 Sentence-transformers

Sentence-transformers[8] is a framework dedicated to text and image embeddings, based on HuggingFace's Transformers[9] and PyTorch[10]. It offers a wide variety of tools to generate embeddings that can be used for several ML tasks. It also allows the user to load and fine-tune pre-trained language models. We chose this framework due to its ease of use, since we just have to select a pre-trained model, give it text (already cleaned and preprocessed) to encode and it will take care of the specific tokenization requirements specific to the model selected, run the resulting tokens through the chosen model and output the resulting embeddings by applying a mean pooling operation.

3.1.5 PyTorch

PyTorch is one of the most popular ML frameworks and it allows us to create and train NN models both on CPU and on the GPU, it is very pythonic making it intuitive to work with. It provides implementations of Tensors (its basic data containers), several types of layers, activation functions, optimizers, loss functions, and other useful classes like Datasets and DataLoaders[11]. We use it in our experiments to integrate NNs into our project.

3.1.6 Optuna

Optuna[12] [35] is an optimization framework that works with most ML libraries. It was heavily used in our experiments, whenever some hyperparameter value needed to be

[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[6] https://radimrehurek.com/gensim/
[7] https://radimrehurek.com/gensim/models/keyedvectors.html
[8] https://www.sbert.net
[9] https://huggingface.co/docs/transformers/index
[10] https://pytorch.org
[11] https://pytorch.org/docs/stable/data.html
[12] https://optuna.org/

optimized. In order to optimize some parameters, we simply create a study object[13], pass it an objective function which we want to maximise or minimise (we maximised the mean CV F1-score in our use cases), a sampler for the study to decide which parameter value combinations to test (this can be all possible combinations in the form of grid search, random combinations in the form of random search, etc) and the amount of trials we want to perform. We opted to use the TPESampler[14] in most situations. It is a sampler built on the Tree-structured Parzen Estimator, a form of Bayesian optimization that tries to select the most promising parameter value combinations at each iteration instead of performing an exhaustive search, or choosing parameter values at random.

3.2 Datasets

In order to address these challenges, we used the official datasets provided by the eRisk organizers for each task, which we will now describe.

3.2.1 Pathological Gambling

This dataset corresponded to last year's test dataset for the same task [36]. It was made available on a dedicated server and it consists of a golden truth text file, that maps subjects to their corresponding label of 0 or 1 (for control and pathological gamblers respectively), and a folder containing multiple files in the xml format, one for each subject, containing information regarding their writings (Reddit posts/comments). In its original state, there were records of writings for 2348 subjects, 164 of which were labeled 1, and 2184 were labeled 0.

Naturally, some preprocessing was required, especially since plenty of writings had artifacts of the extraction process, mostly in the format of html tags. At this stage, a jupyter notebook was used, to read through the golden truth text file, save the subject id's and corresponding labels and then iterate through each of the subject's xml files. Each of the writings had the following sequence of transformations applied (performed mostly with regular expressions[15] and contractions[16] libraries):

1. Alphabetical characters were converted to lowercase.

2. Html and decimal unicode artifacts were converted to their corresponding symbols.

3. Various types of emojis such as ":)" and ":(" were converted to "smiling emoji" and "negative emoji" respectively.

4. Mentions of website links, reddit users and subreddits were converted to the tokens "url", "user" and "subreddit" respectively.

[13] https://optuna.readthedocs.io/en/stable/reference/generated/optuna.study.Study.html
[14] https://optuna.readthedocs.io/en/stable/reference/generated/optuna.samplers.TPESampler.html
[15] https://docs.python.org/3/library/re.html
[16] https://github.com/kootenpv/contractions

5. Numbers beginning or ending with "€" (Euro) or "$" (Dollar) were substituted by the token "money".

6. Remaining isolated tokens comprised only of numbers were converted to the token "number".

7. Contractions of common words were transformed into their original form (for example "you're", "I'm" turn into "you are" and "I am").

8. A dictionary of common internet slang terms was constructed to convert words to their original form (for example "omg", "ty" turn into "oh my god" and "thank you").

9. Punctuation characters were discarded except for "!.?,", as these tend to be expressive, and can be useful for some of the features extraction methods used.

The resulting writings were put through a FastText language identification model[17] [37, 38], and those classified as non-English were discarded, while English writings were kept. Regarding the language identification, this model was chosen due to its speed and decent accuracy. With some manual reviewing early on, we saw there was a significant amount of misclassifications, especially with shorter texts and those containing a lot of abbreviations and acronyms. In order to improve on this, the script was changed to return the top 2 most likely languages for each writing, and if English was present in the top 2, the writing would be kept, and discarded otherwise. With this change, upon some manual reviewing, the output seemed to be much better, and of 1129560 total writings, 57534 were filtered out. The remaining preprocessed posts were then aggregated in a csv file in their original, stemmed and lemmatized forms.

The resulting dataset was significantly imbalanced, the control group represented 93% (2184) of the total subjects vs 7% (164) of pathological gamblers, and 95% (1016947) of the total writings vs 5% (54915) for pathological gamblers. The method chosen to handle this imbalance was random undersampling, as it is a very common approach, supported by some of the works in the relevant literature [26]. We performed undersampling in terms of users (not writings), to bring the number of control group subjects down to match the number of positive subjects in a way that results in a somewhat balanced number of writings for each class (since subjects do not have a fixed amount of writings).

With this approach we end up with a dataset balanced in terms of subjects of each class (164 each), and nearly balanced in terms of posts from subjects of each class (65266 vs 54915). At this point, the data was split 80%/20% into a training and validation sets respectively, since the official testing set was only made available at the time of submission. This was our initial approach, in the final iteration of our experiments these training and validation sets were grouped and cross-validation was performed in a 5-fold manner rather than a single train/validate split for more robust evaluations. Regardless of single training/validation or 5-fold splits, the splitting was performed on subjects, ensuring that

[17] https://fasttext.cc/docs/en/language-identification.html

Table 3.1: Composition of the various sets of data for task 1.

Set	Subjects/Writings		
	Positive	Negative	Total
Original	164 (7%)/55677 (5%)	2184 (93%)/1073883 (95%)	2348/1129560
Preprocessed	164 (7%)/54915 (5%)	2184 (93%)/1016947 (95%)	2348/1071862
Undersampled	164 (50%)/65266 (54%)	164 (50%)/54915 (46%)	328/102181
Training	131 (50%)/44805 (46%)	131 (50%)/53053 (54%)	262/97858
Validation	33 (50%)/10110 (45%)	33 (50%)/12213 (55%)	66/22323
Testing	81 (4%)/14627 (1%)	1998 (96%)/1014122 (99%)	2079/1028749

any given subject's writings may only appear in either training or validation sets/folds. This was made to ensure that there is no information leak and that we can evaluate the trained models on data belonging to completely new subjects with likely different vocabularies.

As we said, the official testing set was only made available at the time of submission, so it was not used in the initial training stage but only afterwards when evaluating, it is comprised of 81 positive and 1998 control subjects, with 14627 and 1014122 writings respectively. More information on all mentioned sets of data can be found in Table 3.1.

Data Analysis

Inspired by some works in the literature that performed some analysis on the different patterns of activity between control and positive subjects, we also decided to see if we could find meaningful differences in our dataset. We began by analysing the average number of writings per class, and found that the control group subjects tend to have more writings available than those of the positive group, with mean values of 465 vs 334, median values of 239 vs 139, and standard deviations of 509 and 391 writings respectively. We also analysed the number of tokens per writing, where the mean amount was of 28 vs 46, with median values of 12 vs 22 and standard deviations of 73 and 76 for classes 0 and 1 respectively. These values seem to suggest that while subjects at risk of pathological gambling might post less, when they do, they tend to elaborate more with longer posts.

Regarding the time of activity, we divided the analysis in terms of days of the week and time of the day as was done by Cacheda et al. [32]. Despite not being focused on the same mental illness, the distributions were surprisingly similar in both cases, perhaps due to the fact that subjects dealing with pathological gambling and those dealing with Major Depressive Disorder (MDD), the focus of the work of Cacheda et al., experience some of the same symptoms. The distribution of writings per day of the week are shown in figure 3.1, where we notice that positive subjects are more active on the weekend while control subjects, in general, tend to post more on the remaining days of the week. When it comes to the time of the day that the writings are published, figure 3.2 shows that positive users dominate the slot between midnight and 5 AM, and from 6 to 11 AM to a

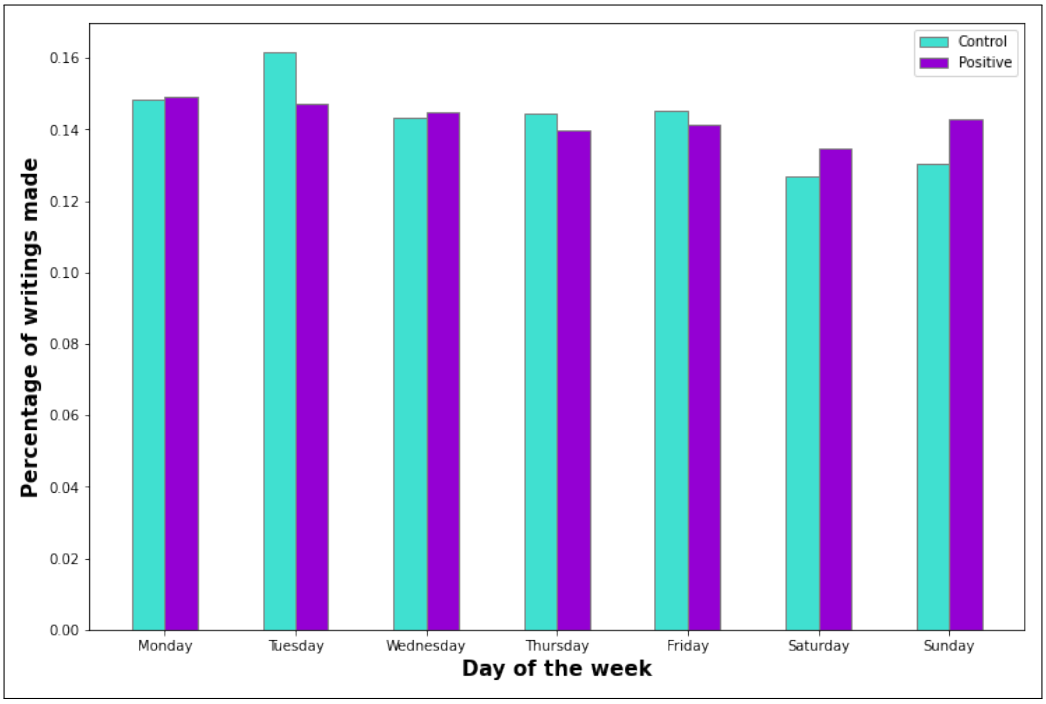

Figure 3.1: Distribution of the subjects' activity in terms of the days of the week in task 1.

lesser degree, while the control group dominates the afternoon slot of midday to 17, and the evening hours of 18 to 23 to a lesser extent. These results may indicate some trouble with sleeping patterns on the side of the positive class.

3.2.2 Depression

This dataset's structure was slightly different than that of the previous task, instead of using a golden truth text file, the xml files containing each user's writings were grouped into positive and negative folders, corresponding to the 1 and 0 labels respectively, also grouped by year, since the organizers provided data from previous editions of eRisk, in the form of 2017 eRisk's training and testing datasets [39] and 2018 eRisk's testing set [40]. Regarding the individual files the format was the same as in the pathological gambling task.

Originally, the data from 2017 contained 752 negative and 135 positive subjects. Likewise, the 2018 folder contained 741 negative and 79 positive subjects. This comes up to a total of 1707 subjects, 214 of which are positive (depressed). The preprocessing was the same as in the previous task, following the same transformations and language identification methods. Out of 1076582 total writings, 16950 were discarded.

Although not to the extent of the pathological gambling case, this dataset was still heavily imbalanced, with 87% of the users being labeled as negative and 13% as positive, and 92% of the writings corresponding to the negative class vs 8% of the positive users.

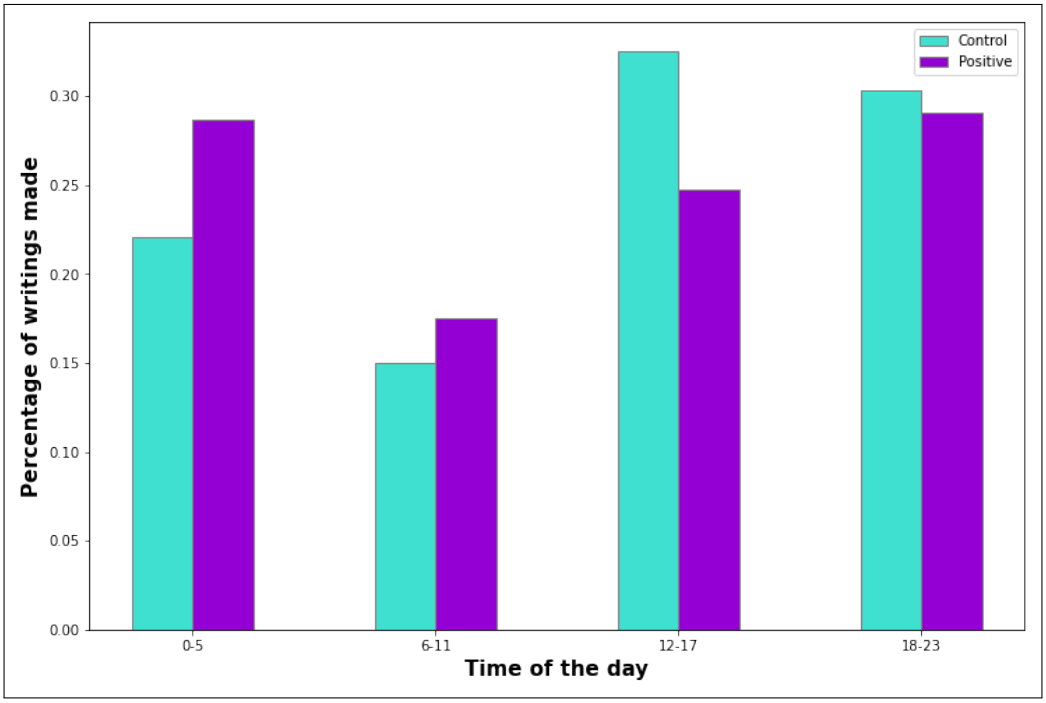

Figure 3.2: Distribution of the subjects' activity in terms of the time of the day when writings are posted in task 1.

Table 3.2: Composition of the various sets of data for task 2.

Set	Subjects/Writings		
	Positive	Negative	Total
Original	214 (13%)/90222 (8%)	1493 (87%)/986360 (92%)	1707/1076582
Preprocessed	214 (13%)/89010 (8%)	1493 (87%)/970622 (92%)	1707/1059632
Undersampled	214 (50%)/89010 (42%)	214 (50%)/121917 (58%)	428/210927
Training	171 (50%)/73670 (42%)	171 (50%)/100503 (58%)	342/174173
Validation	43 (50%)/15340 (42%)	43 (50%)/21414 (58%)	86/36754
Testing	98 (7%)/35332 (5%)	1302 (93%)/687228 (95%)	1400/722560

Naturally, a random undersampling algorithm was employed here as well, to reduce the amount of control group subjects from 1493 to 214 in order to match the amount of depressed subjects. As expected, the number of writings also became more balanced, going from 970622 negative writings to 121917, much closer to the number of positive writings of 89010. Again, initially this resulting dataset was split into a 80%/20% train/validation split (later grouped as in task 1), divided by users, leading to a training dataset containing 171 positive and 171 negative subjects, and 73670 positive vs 100503 negative writings. The validation set contained 43 positive and 43 negative subjects and 15340 positive vs 21414 negative writings. Just like before, we only obtained the testing dataset at the time of submission, it was comprised of 35332 writings belonging to 98 positive subjects and 687228 belonging to 1302 control subjects. A brief summary of the mentioned sets of data can be seen in Table 3.2.

Data Analysis

Likewise, we also performed some statistical analysis in this case, with the assumption that the results would match those of Cacheda et al. even more since we are addressing issues of the same nature and we might even share some of the same subjects, since they relied on eRisk data as well. Regarding the number of writings, the mean amounts were of 649 vs 414, the median amounts were of 386 vs 167, and the standard deviations were of 633 and 462 writings for the control and positive group respectively. In terms of the amount of tokens per writing, the mean values were of 34 vs 43, the medians were 15 vs 19 and the standard deviations were of 107 and 80 for the negative and positive classes respectively. Similar to the pathological gambling case, this suggests that depressive users post less but tend to elaborate more. Regarding the activity distribution in terms of the days of the week in which the content is posted, unsurprisingly, the patterns shown match those observed by Cacheda et al. as seen in figure 3.3. When it comes to the activity in terms of the time of the day in which content is posted, the results also match very closely those of Cacheda et al. as we observe in figure 3.4, depressive users tend to be more active in the 0 to 5 AM and 6 to 11 AM slots while the control group dominates the 12 to 17 slot and 18 to 23 the results are nearly evenly matched. Again, this may indicate some issues with sleep patterns, where perhaps depressive subjects compensate for their lack of sleep at night, during the afternoon.

3.3 Feature Engineering Techniques

We must feed data to our ML models in the form of numbers, not text. When it comes to feature engineering, we saw in the previous chapter that many vectorization approaches can be used. In this work we rely mostly on textual features, though experiments with sentiment analysis features were also integrated.

But before deciding how to extract features from each sample, we must first decide

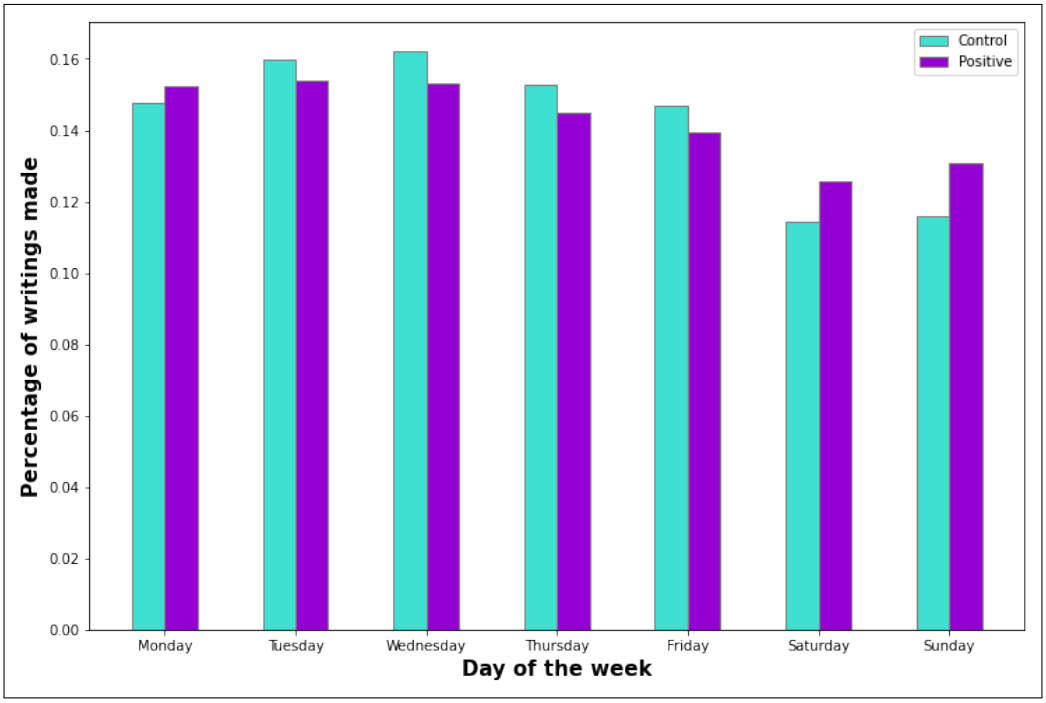

Figure 3.3: Distribution of the subjects' activity in terms of the days of the week in task 2.

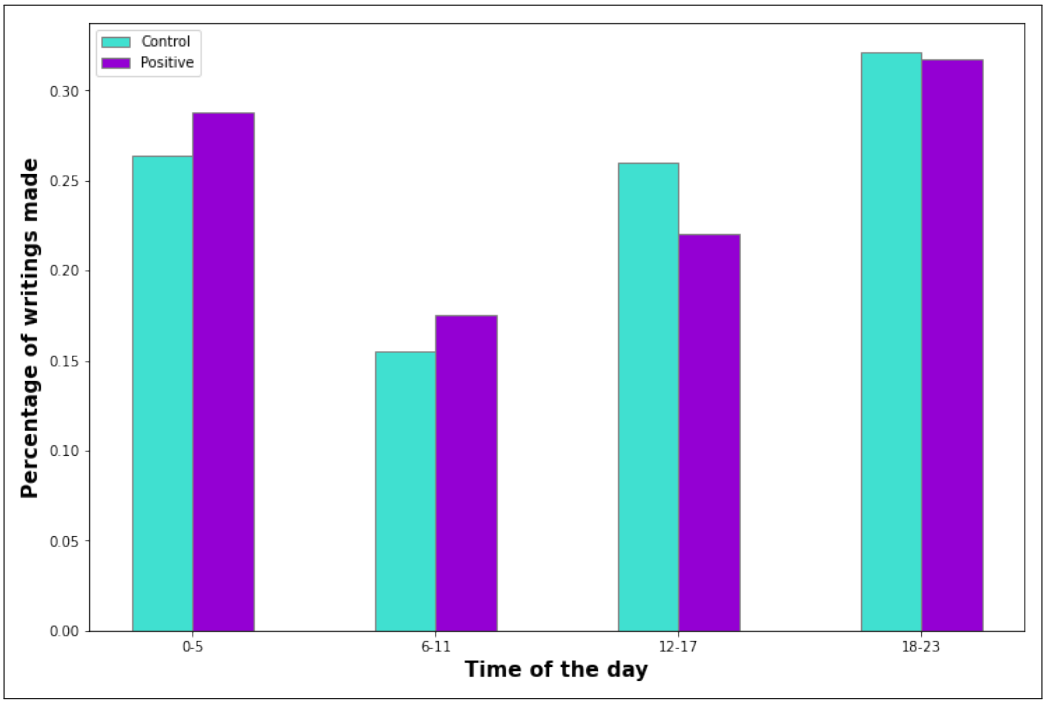

Figure 3.4: Distribution of the subjects' activity in terms of the time of the day when writings are posted in task 2.

what constitutes a sample. Is it a single writing from a subject, multiple writings grouped by some criteria (length, time of posting, etc), or perhaps all of the writings of a given subject? In the very first experiments, user writings were used individually as samples (i.e. a sample is comprised of a single writing), but this quickly led to subpar results, likely due to the fact that some writings are extremely short and it is difficult to assess which class a writing's author belongs to given only a few words. To overcome this, inspired by NLP-UNED's run [41] in 2021's edition of CLEF eRisk, a k-sliding-window method was implemented, where each sample consists of the last k writings seen at the time (since writings are gathered in chronological order) regardless of length. The choice of k is very important here, in early experiments with single writing samples, essentially equivalent to employing a sliding window of k=1, results were rather poor, so we experimented with larger k values as well (3, 5 and 10), resulting in immediate improvements across the board. Further increasing the k would likely be conducive to better results, at least for the first feature engineering approach, which we will discuss next, but given the token limitation of the language models used in approach 3, higher k values would result in large amounts of text being discarded (by truncation) at each prediction. So in order to keep approaches comparable, we decided that the final models chosen for each task all ran on writing windows of the same k, meaning that we have a maximum k of 10, to avoid going over the token limit of the selected language models, since the writing windows are constructed prior to the feature extraction process.

The textual features were essentially split into 3 main categories of increasing complexity: Bag-of-Words (BoW) with Tf-Idf weights, pre-trained Distributional Semantics Word Embeddings (DSWE), and pre-trained Contextualised Language Model Embeddings (CLME). For the first 2 approaches we opted to lemmatize the tokens of each window, while for approach 3 we opted to keep tokens in their raw form which is more similar to the data that the language models were trained on. The previously mentioned Sentiment Analysis (SA) features are those of the sentiment analysis tools in Vader[18] and TextBlob[19]. TextBlob's tool provides us with 2 scores for each text analysed, regarding its subjectivity (opinionated or fact based) and polarity (how positive or negative the text is). Vader's features consist of 4 values corresponding to scores regarding a text's negativity, neutrality, positivity, and a compound value which acts as a single overall measure of the previous values. Experiments were ran with textual features alone and with the inclusion of sentiment analysis features (normalized to fit between 0 and 1), to evaluate their effectiveness in this scenario.

Bag of Words

As we discussed in the previous chapter, BoW with a Tf-Idf weighting methodology is a simple yet effective approach for many text classification problems, though there are some parameters that need to be tweaked to optimise performance, such as:

[18] https://github.com/nltk/nltk/blob/develop/nltk/sentiment/vader.py
[19] https://github.com/sloria/textblob

- max features: maximum amount of tokens in the learned vocabulary.
- n_gram range: how many tokens to group as a unit in the vocabulary (1-gram being groups of 1 token, 2-grams 2 tokens, etc), the range takes a min x and max y values to consider groups of at least x-grams and at most y-grams.
- stopwords list: tokens that are not valid candidates for the learned vocabulary and are discarded.
- max/min document frequency: a threshold to filter out tokens that occur in more/less than x documents, for max and min respectively, with x being an absolute value or a fraction.
- sublinear tf: whether to apply a sublinear scaling to the term frequency values.

Thanks to Scikit-learn, this is all implemented in the form of the class TfIdfVectorizer which allows us to create a vectorizer that given the desired parameters and a training corpus, can later transform samples of text into number vectors.

To find the optimal Tf-Idf parameter values (of those previously mentioned) to use in each task, we made use of Optuna's TPESampler, using 5-fold CV to experiment with different TfIdfVectorizer parameter combinations, feeding the transformed data into a NB classifier to evaluate their effectiveness based on the average of the 5 resulting F1-scores. We used a NB classifier specifically, due to its speed and good results out-of-the-box with minimal tuning, to facilitate these experiments.

The resulting optimal parameters for the TfIdfVectorizer of task 1 and task 2 and shown in Tables 3.3 and 3.4 respectively.

Table 3.3: Optimal TfIdfVectorizer parameters for each window size of writings from task 1.

Window size	max_features	max_df	ngram_range	stop_words	sublinear_tf	F1
1	15000	0.75	1,1	NLTK	True	0.677
3	15000	0.8	1,1	NLTK	True	0.753
5	13000	0.9	1,1	NLTK	False	0.783
10	15000	0.6	1,2	NLTK	True	0.821

Table 3.4: Optimal TfIdfVectorizer parameters for each window size of writings from task 2.

Window size	max_features	max_df	ngram_range	stop_words	sublinear_tf	F1
1	15000	1.0	1,3	NLTK	False	0.544
3	11000	0.35	1,3	NLTK	False	0.616
5	13000	0.5	1,3	NLTK	True	0.642
10	13000	0.85	1,2	NLTK	True	0.669

It is interesting to see that when checking with Optuna's visualization tools, the parameter max_features is consistently considered the most impactful across tasks, as seen in Figures 3.5 and 3.6, highlighting the importance of acquiring a wide vocabulary in BoW methods.

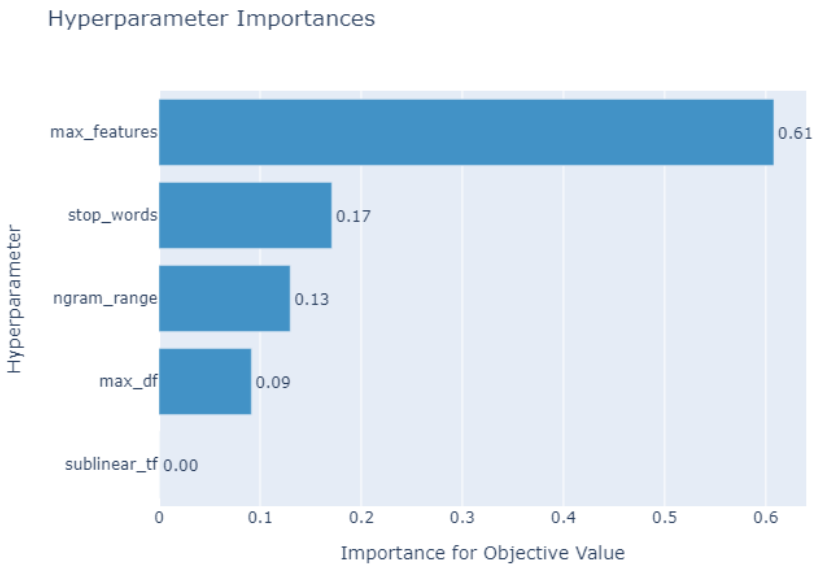

Figure 3.5: Parameter importance for the TfIdfVectorizer on data from task 1 for window size 10.

Distributional Semantics Word Embeddings

In this approach, as previously stated, we use pre-trained DSWE models to represent the windows of writings.

There were lots of possibilities for pre-trained models but 3 were selected. The first 2 were GloVe [14] models while the last one was a FastText [42] model. The first, a

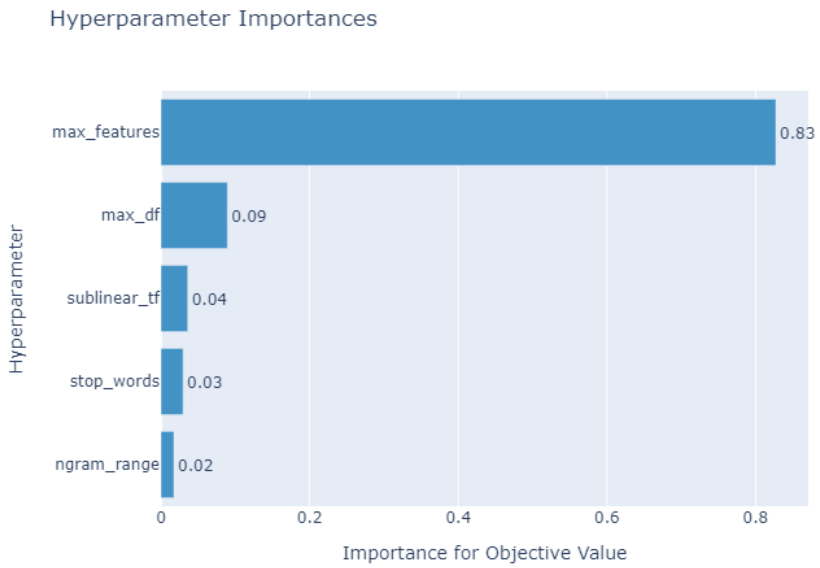

Figure 3.6: Parameter importance for the TfIdfVectorizer on data from task 2 for window size 10.

200 dimensional model[20] trained on Twitter data (which we will refer to as GloVe_TT), chosen due to the reliability of GloVe models (found many times in the literature) and the similarity between the Twitter and Reddit social networks due to their conversational nature. The second one was another GloVe model[20] but 300-dimensional, and trained on Common Crawl (referred to as GloVe_CC), chosen again due to the reliability of GloVe models and due to the fact that Reddit comments were part of this model's training data. The final one was a 300 dimensional FastText [21] model also trained on data from Common Crawl[22] (referred to as FastText_CC), selected due to its versatility handling OOV tokens (since it handles these tokens by breaking them apart).

When testing each pre-trained embedding model's coverage of the vocabulary for task 1, we got:

- GloVe_TT: covered 76.07% of the vocabulary, and 99.24% of all text.
- GloVe_CC: covered 88.04% of the vocabulary, and 99.61% of all text.
- FastText_TT: covered 100% of the vocabulary, and 100% of all text .

Likewise, for data from task 2:

- GloVe_TT: covered 69.17% of the vocabulary, and 99.40% of all text.

[20] https://nlp.stanford.edu/projects/glove/
[21] https://fasttext.cc/docs/en/english-vectors.html
[22] https://commoncrawl.org/2017/06/

- GloVe_CC: covered 86.24% of the vocabulary, and 99.77% of all text.
- FastText_TT: covered 100% of the vocabulary, and 100% of all text.

Windows of writings were represented by the mean (across dimensions) of its tokens' embeddings. For example, with a text containing 3 tokens represented by [1,1,3],[4,2,1] and [1,0,5], we sum these, resulting in the vector [6,3,9], and then divide each value by the number of tokens, 3 in this case, resulting in the vector [2,1,3].

Contextualised Language Model Embeddings

As we saw, language models leverage the power of deep learning by training large NNs to achieve great performance in a variety of NLP tasks. Given the large computational costs that come with building language models from scratch, we decided to make use of pre-trained models in this case, which is known as transfer learning.

There was a large variety of models we could have chosen, but we selected two as possible candidates, both made available with Sentence-transformers, to extract sentence embeddings with. The all-MiniLM-L6-v2[23] and the all-mpnet-base-v2[24], that encode sentences into 384 and 768 dimensional vectors respectively.

This MiniLm model originated as the pre-trained 6 layered version of Microsoft's MiniLM-L12-H384-uncased (by keeping every second layer) [43], fine-tuned on a 1B sentence-pair dataset, where given a sentence from a pair, the model had to select its correct match from a set of random samples. MiniLM-L12-H384-uncased in turn was achieved by the process of knowledge distillation from BERT. Knowledge distillation is the process of compressing large models (also known as teachers in this context) by training a smaller model (known as student) to imitate their behaviour. Instead of learning by computing loss from the golden truth labels, the student model uses the output of the teacher model's layers as the true label (not necessarily the final layer, as was the case with MiniLm).

This choice resulted from the fact that despite being a very lightweight model, according to the comparison table[25] in the Sentence-transformers documentation, the all-MiniLM-L6-v2 performed rather well across 14 sentence embedding tasks, another relevant factor was that a large subset of its training data derived from Reddit comments.

The MPNet, also developed by Microsoft [44], is seen as a mix of BERT and XLNet [23], as it combines both training methods of Masked Language Modeling (MLM) in BERT and Permutation Language Modeling (PLM) (where tokens are guessed on various permutations of the original sentence as opposed to the classic left-to-right or right-to-left) from XLNet to complement each model's approaches. Despite being a larger and consequently slower model when compared to the MiniLm, the MPNet was chosen as it was the top performer in the Sentence-transformers documentation, and its paper claimed

[23] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[24] https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[25] https://www.sbert.net/docs/pretrained_models.html

state-of-the-art performance over the top performers like BERT and XLNet, which it was based on.

With these choices we also intend to see how big of a difference in performance can be achieved with a smaller vs larger language model.

3.4 Models

We divided the tasks into 2 parts, part one regarding the classification of windows of writings, to assess whether a certain number of consecutive user writings (a writing window) is likely to come from a positive subject, and part two where the results of the window classifications are converted into the subject's classification, which corresponds to our end goal.

3.4.1 Writing Window Classification

In order to make the performance of the different feature extraction methods somewhat comparable, the models used remained mostly the same across experiments, consisting of sklearn's implementations of Logistic Regression (LR) and linear Support-Vector Machine (SVM) both trained using Stochastic Gradient Descent[26] (SGD), Naive Bayes[27] (NB), ExtraTrees[28] (ET) and Perceptron[29] models. It is worth noting that Naive Bayes was only used in approach 1 due to its inability to deal with negative features, present in approach 2 and 3. The SGD version of the LR and SVM models and ET (instead of Random Forest) were chosen due to our large feature space and large amount of samples.

We chose the best model for each window size by performing 5-fold CV on the full training set (training + validation) mentioned in the dataset section, using the best average F1-score (from the 5 CV iterations) as the evaluation metric. Once the top performing models are established, we re-evaluate its performance with the inclusion of the Sentiment Analysis (SA) features, keeping them moving forward if their inclusion results in an improvement of the average F1-score.

It is essential that during the cross-validation process, no subject's writings are present in both training and testing folds simultaneously to better imitate the end goal of the models being developed of classifying completely new subjects, avoiding possible information leaks that might result from windows in training and testing folds varying by few writings. For example if a user has 5 consecutive writings A, B, C, D and E and we're using a window of size 3, there is a chance that the window containing A, B and C is present in a training fold while the window containing B, C and D is present in the testing fold. This is the information leakage we must avoid and that is the reason for our splitting being based on users.

[26] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[27] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[28] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
[29] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

Post-evaluation we select the best performing model of each feature type (with or without SA features depending on how they have performed) and perform hyperparameter optimisation using Optuna. Finally, when the optimal hyperparameters are found, we train the best models with the best parameters found, on the whole train + validation data. These models are later used in the testing stage.

BoW

Regarding the first model evaluation stage, the results are best summarized in Tables 3.5 and 3.6.

Table 3.5: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 1 with BoW features.

Window size	Model	Avg CV F1	Avg CV F1 (with SA)
1	NB	0.677	0.678
3	NB	0.753	0.759
5	NB	0.783	0.789
10	LR	0.823	0.820

Table 3.6: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 2 with BoW features.

Window size	Model	Avg CV F1	Avg CV F1 (with SA)
1	NB	0.544	0.542
3	NB	0.616	0.614
5	NB	0.642	0.641
10	NB	0.669	0.669

We selected both of the best models for window size 10 since they achieved the best performance on both tasks and optimized their hyperparameters. The resulting LR model (SGDClassifier with "log" loss) for task 1 achieved an average F1-score with 5-fold cross-validation of 0.824 (a 0.001 improvement), with the following parameters:

- Penalty: L2
- Alpha: $6e^{-5}$
- Loss: Log
- Max iterations: 2500

Since we chose a LR model to optimize, which is considered interpretable, due to its coefficients being mapped to each of the TfIdfVectorizer's features (which correspond to words in the vocabulary), we can analyse which tokens are more influential in a positive classification. For task 1, the 10 terms with the higher coefficients (influential in classifying a subject as a pathological gambler) are, from most influential to least: "gambling", "money", "ill", "bet", "gamble", "lost", "day number", "addiction", "wager", "never". Some of these are obvious, others not so much, we must also remember that in preprocessing we transformed various representations of currency like "3.00€" or "5$" into the token "money", the "day number" term is also affected by the preprocessing since numeric values that were not explicitly money related were substituted by the "number" token, which affected common mentions of numbers like dates in this case.

As for task 2, the resulting NB model achieved an average F1-score with 5-fold cross-validation of 0.669, with the following parameters:

- Alpha: 1.27
- Fit_prior: True.

For task 2, since we optimized a NB model, which is also interpretable by analysing the feature's log probabilities, we can also check on the most influential tokens. In this case the 10 most influential terms for classifying depressed users were: "number", "like", "would", "get", "one", "really", "know", "people", "think", "time".

Distributional Semantics Word Embeddings

The best model's performances for each window size with this type of feature are shown in Tables 3.7 and 3.8.

Table 3.7: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 1 with distributional semantics word embedding features.

Window size	Embeddings	Model	Avg CV F1	Avg CV F1 (with SA)
1	GloVe_TT	LR	0.673	0.672
	GloVe_CC	LR	0.675	0.673
	FastText_CC	SVM	0.648	0.649
3	GloVe_TT	LR	0.749	0.749
	GloVe_CC	LR	0.749	0.750
	FastText_CC	SVM	0.720	0.715
5	GloVe_TT	LR	0.776	0.780
	GloVe_CC	LR	0.777	0.783
	FastText_CC	ET	0.749	0.749
10	GloVe_TT	LR	0.809	0.809
	GloVe_CC	LR	0.817	0.818
	FastText_CC	ET	0.775	0.777

Table 3.8: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 2 with distributional semantics word embedding features.

Window size	Embeddings	Model	Avg CV F1	Avg CV F1 (with SA)
1	GloVe_TT	SVM	0.529	0.526
	GloVe_CC	LR	0.566	0.568
	FastText_CC	ET	0.509	0.509
3	GloVe_TT	SVM	0.621	0.598
	GloVe_CC	SVM	0.598	0.577
	FastText_CC	ET	0.579	0.579
5	GloVe_TT	SVM	0.654	0.656
	GloVe_CC	SVM	0.643	0.624
	FastText_CC	ET	0.606	0.604
10	GloVe_TT	LR	0.664	0.664
	GloVe_CC	LR	0.664	0.664
	FastText_CC	ET	0.634	0.633

Like before, it seems that window size 10 served the best results in this case too, with LR including SA features in task 1 and LR without SA features in task 2 (both using GloVe_CC), which we optimized. The resulting LR model for task 1 achieved an average

F1-score with 5-fold cross-validation of 0.819, with the following parameters:

- Penalty: L2
- Alpha: $5e^{-5}$
- Loss: Log
- Max iterations: 2000

As for task 2, the resulting LR model achieved an average F1-score with 5-fold cross-validation of 0.683, with the following parameters:

- Penalty: L2
- Alpha: $4e^{-5}$
- Loss: Log
- Max iterations: 1000

Contextualised Language Model Embeddings

We followed the same cross-validation approach with Scikit-learn's default models and we see in Tables 3.9 and 3.10 that window size 10 is still the best approach when using CLME as the feature type.

Table 3.9: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 1 with features extracted from language models.

Window size	Language model	Model	Avg CV F1	Avg CV F1 (with SA)
1	MiniLm	SVM	0.676	0.670
	MPNet	SVM	0.697	0.692
3	MiniLm	LR	0.757	0.756
	MPNet	LR	0.767	0.764
5	MiniLm	LR	0.787	0.787
	MPNet	LR	0.796	0.794
10	MiniLm	LR	0.819	0.819
	MPNet	LR	0.823	0.823

Table 3.10: Performance of the best models when trained with sklearn's default parameters for every window size for writings of task 2 with features extracted from language models.

Window size	Language model	Model	Avg CV F1	Avg CV F1 (with SA)
1	MiniLm	LR	0.534	0.531
	MPNet	LR	0.551	0.549
3	MiniLm	LR	0.606	0.589
	MPNet	LR	0.614	0.602
5	MiniLm	LR	0.629	0.618
	MPNet	LR	0.644	0.627
10	MiniLm	LR	0.663	0.65
	MPNet	LR	0.667	0.655

We now describe the optimal hyperparameters found and the resulting performance. The resulting LR model for task 1 achieved an average F1-score with 5-fold cross-validation of 0.826, with the following parameters:

- Penalty: L2
- Alpha: 0.0007
- Loss: Log
- Max iterations: 1000

As for task 2, the resulting LR model achieved an average F1-score with 5-fold cross-validation of 0.670, with the following parameters:

- Penalty: L1
- Alpha: 0.0001
- Loss: Log
- Max iterations: 2500

A more standard approach

Additionally, in order to test a more commonly used approach in current times, we had to perform experiments with transformer-based features of CLME while using Neural Networks as classifiers, since these tend to achieve solid performance in many NLP tasks.

We used PyTorch to create the NNs, and Optuna to find the optimal architecture and other relevant hyperparameters. With the Sentence-transformers library we essentially freeze all layers of the representation generating language models, on top of which we build our classifying NN. The language model used for both tasks was MPNet since it

outperformed the MiniLm.

We performed optimization studies to find good hyperparameters regarding the learning rate, number of hidden layers, number of nodes per layer and dropout rate at each layer. The studies consisted of 20 trials and every model was trained for 10 epochs for each of the 5-fold CV iterations. Consistent across all NN tests were the optimizer, AdamW[30] chosen due to its fast convergence times and reliable performance, the classic loss metric of binary cross entropy and a ReLu activation function on each hidden layer. In order to speed up the whole process and to allow us to validate the several parameters and architectures in a more robust way (5-fold vs train/validation split CV), we encode all windows of writings prior to any training, to avoid the unnecessary and costly vectorizations at each trial.

The resulting optimized parameters for the NN of task 1 with an average F1-score of 0.807 were:

- Number of layers: 2.
- Nodes per layer: 28 (layer 1) and 17 (layer 2).
- Dropout rate: 0.45 (layer 1) and 0.40 (layer 2).
- Learning rate: $1.04e^{-5}$

Likewise, for task 2, achieving an average F1-score of 0.694, the optimized parameters were:

- Number of layers: 3.
- Nodes per layer: 6 (layer 1), 6 (layer 2) and 5 (layer 3).
- Dropout rate: 0.50 (layer 1), 0.30 (layer 2) and 0.50 (layer 3).
- Learning rate: $4.98e^{-5}$

Taking into consideration that the tuning process adopted a 5-fold CV methodology, we considered it robust enough to train the final models on all of the training + validation data using the same seed, despite being unable to validate the resulting models prior to the final testing stage.

3.4.2 User Classification

This part of the problem consists of adopting a criterion or protocol to turn our individual classifications of windows of writings into subject classifications, since up until this point, we were classifying windows of writings and not the authors themselves. Various choices can be made here, the NLP-UNED team [41] (from which we drew inspiration for the k-rolling-window method) used a parameter n of consecutive positive windows as

[30] https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

a signal that the subject should be classified as positive. But there are alternatives like requiring a fraction or absolute value of positive window classifications up to a certain point (since writings are retrieved chronologically).

At the time of submission for the event, we followed the latter approach, a threshold on the decision confidence of the classifiers was applied and a single window positive classification was enough to make a positive final decision. That is, writing windows are evaluated individually, and if a model is really confident that at least one of a user's windows belongs to a positive subject, we classify the subject as positive. We did end up performing experiments with all the criteria initially mentioned, we will refer to these options as Single Confidence Criterion (SCC), Multi Confidence Criterion (MCC), Ratio Confidence Criterion (RCC) and Consecutive Confidence Criterion (CCC) (NLP-UNED's approach). All criterion have their own threshold on the classification confidence, besides their custom decision threshold (depending on the criterion). We will now explain each one's requirement for positive final decisions:

In the case of the SCC, three situations may occur after evaluating a window of writings:

1. the window of writings is classified as positive and the confidence in the decision meets the required threshold.

2. the window of writings is classified as positive but the confidence in the decision does not meet the required threshold.

3. the window of writings is classified as negative.

The subject is only classified with a positive final decision in case 1, where case 2 and 3 lead to a negative classification, meaning that we cannot provide a final decision and need to analyse more data.

MCC works under the same assumption as SCC, but instead of requiring 1 positive classification, we require more, with a threshold $t > 1$. The RCC works by keeping track of how many of the windows of writings seen so far were classified as positive, if at any point the ratio of positive classifications divided by the number of different individual posts seen so far meets the required threshold, the subject is classified as positive, and so, after every evaluation n, with positive window count p and threshold t, the following process occurs:

1. classify the current window.

2. add the resulting classification to p.

3. if $p/n \geq t$, output a positive final decision for this subject.

Finally, the CCC simply keeps track of consecutive positive window classifications, the counter is reset whenever the latest window is classified as negative, if the amount of consecutive positive classifications meets the required threshold, the subject is classified with a positive final decision.

As stated in the introduction, we are not only concerned with obtaining a correct classification, but also doing so in a timely manner, with this purpose in mind, we limited the number of user writings analysed to 100 (resulting in 90 windows since they are size 10). It makes sense to tune our thresholds to a smaller number of rounds (with the maximum possible being roughly 2000), since not all subjects in our dataset or in a real-life scenario have the same number of writings available. As we increase the limit of windows analysed, we also increase the variability in the amount of windows available per user, for example if we have 5 users, each with 150, 200, 90, 50 and 140, if we limit the amount of windows to 200 there is significantly more variation than if we do so with a smaller limit, in the most extreme case, limiting to 50, all users end up with the same amount of windows in this example. Naturally, thresholds tuned to a certain limit won't be as optimal for users whose maximum amount of windows falls short, but we found that 100 is a decent spot in the trade-off of minimising variability and response time while maximising the amount of data analysed.

We followed the same process used to tune the window classification models to find the optimal thresholds for each criterion. For each criterion, we split the full training (plus validation) data into 5-folds, training the best model found in the previous section with the optimized parameters on 4 of the folds and using the remaining fold at each iteration to test various threshold values. With one important detail, it is very unlikely that in a real world scenario (simulated with the official testing set), the percentage of positive to negative users will be as balanced as it is in our undersampled dataset, which makes it unreliable for finding the optimal thresholds. To address this, on the testing fold of every 5-fold CV iteration, we include the data of the subjects left out by the undersampling process (limited to the first 100 writings of each user), which consists of a very large number of control users, providing us with a more realistic distribution. Just as we did before, we used Optuna's TPEsampler to arrive at the best values, by maximising the mean of F1-score achieved in all 5 folds. The optimal thresholds and the corresponding evaluation scores are shown in Tables 3.11 and 3.12.

Table 3.11: Performance achieved in task 1, by the previously mentioned best models when tested in the user classification context with all the criteria.

Model	Features	Criterion	Thresholds		Average F1
			Confidence	Windows	
LR	Tf-Idf	SCC	0.981	1	0.748
		MCC	0.967	2	0.736
		RCC	0.942	21.7%	0.726
		CCC	0.951	7	0.715
LR	GloVe_CC + SA	SCC	0.992	1	0.312
		MCC	0.947	14	0.536
		RCC	0.901	50.1%	0.580
		CCC	0.916	19	0.522
LR	MPNet	SCC	0.981	1	0.833
		MCC	0.971	2	0.826
		RCC	0.909	35.8%	0.730
		CCC	0.963	2	0.814
NN	MPNet	SCC	0.981	1	0.641
		MCC	0.979	6	0.635
		RCC	0.977	16.8%	0.628
		CCC	0.963	10	0.615

Table 3.12: Performance achieved in task 2, by the previously mentioned best models when tested in the user classification context with all the criteria.

Model	Features	Criterion	Thresholds		Average F1
			Confidence	Windows	
NB	Tf-Idf	SCC	0.986	1	0.343
		MCC	0.984	9	0.334
		RCC	0.971	11.9%	0.330
		CCC	0.970	10	0.342
LR	GloVe_CC	SCC	0.961	1	0.239
		MCC	0.908	15	0.340
		RCC	0.833	59.6%	0.341
		CCC	0.838	20	0.328
LR	MPNet	SCC	0.984	1	0.387
		MCC	0.932	12	0.385
		RCC	0.840	51.1%	0.397
		CCC	0.871	12	0.363
NN	MPNet	SCC	0.743	1	0.254
		MCC	0.694	40	0.324
		RCC	0.694	44.8%	0.359
		CCC	0.704	14	0.369

The resulting F1-scores are lower than those seen previously but that is to be expected since the distribution of positive to negative subjects changed drastically, but we can only make an accurate assessment of the model's performance on the testing set (also since the models used to test were trained with more data due to not leaving one fold out).

The event's server also expected to receive a score estimating each user's level of risk, to be used in the rank-based evaluation. Our decision here was to keep record, for each subject, of the sum of the writing window classifier's decision confidence whenever its output was positive, if the corresponding user classifier's decision was 0, we would send 0 as the estimated risk, in the case that the user classifier's decision was 1, we would send that calculated sum as the risk estimation metric. Our goal was to focus on achieving a correct risk estimation among the subjects we deemed as positive, hence the 0 in case of negative classifications.

3.5 Summary

Here, we quickly recap the experimental methodology. The problems are divided into 2 stages, writing window classification and user classification, with the former receiving writing windows as input and providing its output to the latter, to take various windows

into account when making a final decision, in order to achieve more robust models. In the writing window classification, we begin by dividing the experiments into 3 feature types (BoW, DSWE and CLME), using classic ML algorithms of LR, SVM, NB, ET and Perceptron, and even NNs for CLME. We then perform tests using samples (windows) containing various amounts of writings (1, 3, 5 and 10) to select the best window size going forward. We ended up choosing 10 as it provided the best performance across feature types. We also tested these initial models with the addition of sentiment analysis features, though it did not have a significant impact in the resulting average F1-scores. After finding the optimal hyperparameters of the best model of each feature type, we moved on to the user classification, where we tuned and tested several decision criteria (SCC, MCC, RCC and CCC) using 5-fold CV with the inclusion of the data left out during the undersampling process in each of the testing folds.

Chapter 4

Results

As previously mentioned, the final state of this project does not match the one at the time of submission, as a consequence, we have 2 types of results to document, the official server results achieved on submission, and the unofficial results achieved with the same testing set but with the tweaks and new experiments incorporated throughout the training stage. There are also 2 types of evaluations, those of decision-based and rank-based evaluation. Decision-based evaluation metrics are the focal point of the event and correspond to those used throughout the development process such as precision, recall and F1-score, while the organisers included others that take into account the latency involved in making a final decision for a positive user, since we limited ourselves to the first 100 writings (instead of using the full 2000 available) we decided to not focus on those as much as our responses were guaranteed to be decently fast by design (100 rounds is our highest possible latency). Despite not being the focal point of the evaluation process, the participants' runs at eRisk 2022 were also evaluated in terms of ranking with classic information retrieval metrics of Precision and Normalized Discounted Cumulative Gain (NDCG). Where the decision-based evaluation is focused on the correct classification of each individual, the rank-based evaluation serves the purpose of assessing how well the system can estimate their level of risk, by calculating those metrics on the highest scoring 10 or 100 subjects, after sorting them in descending order by their level of estimated risk.

4.1 Event Evaluation

During the event evaluation, data is supplied in rounds. In round 0 we get a list of all subjects available, and a writing for each. At each round after that, we get another writing for every subject, or nothing if no more writings are available for a given subject (we never know when a subject is going to run out of writings). At the end of each round we are asked to reply with our classifications and risk estimation scores for each subject. A single positive final decision is absolute, even if followed by negative classifications (we are asked to keep sending predictions after final decisions because of the risk estimation metrics).

Despite showing good results in our early experiments, our initial SCC approach proved ineffective in this evaluation context, mainly due to some implementation errors which we will soon discuss.

The event allowed up to 5 runs for each task. We decided to take advantage of this by running the best window classifier found at the time for each approach (BoW, DSWE and CLME) as the models of the first 3 runs, and ensemble methods reliant on the same models for runs #4 and #5. The model for run #4 was dependent on the output of the 3 models (of each textual feature type), its condition to attribute a subject with a positive final decision, is to have any of the #1, #2 and #3 models classify a given user as positive. The model for run #5 sums the decision confidence of the #1, #2 and #3 models if they had a positive final decision (contributing 0 if their classification was negative) then dividing the summed value by 3 to get an average confidence, if this average confidence passes a custom threshold, the final decision is positive. Again, these early approaches are better explained in our participation paper [34].

In summary, in order to output a positive final decision for a given user, the condition for each run's models are:

1. classify a writing window as 1 with a confidence higher than a threshold value.
2. classify a writing window as 1 with a confidence higher than a threshold value.
3. classify a writing window as 1 with a confidence higher than a threshold value.
4. have any of the previous models provide a final decision of 1.
5. sum the confidence of the original models that had 1 as a final decision and divide by 3, this averaged confidence must be higher than a threshold value.

In order to understand the degree of the discrepancy observed between our initial validation and the event's results, Tables 4.1 and 4.2 illustrate the thresholds and corresponding metrics on the held-out validation set of our early experiments, which followed the evaluation logic that a single positive window classification at any time is enough to lead to a positive final decision. It is important to note that at this time the thresholds were being optimized in a single train/validation split instead of the later adopted 5-fold CV approach, and the undersampled data was not included here either, skewing the thresholds to lower values, particularly noticeable in the case of task 1.

4.1.1 Task 1: Pathological Gambling

As mentioned, the official evaluation of the server submissions showed results significantly different from those observed during the training and testing stages.

Table 4.1: Performance of the chosen writing window models when classifying users in task 1, in validation, using the best thresholds found.

Model	Features	Threshold	Precision	Recall	F1
LR (#1)	Tf-Idf	0.80	0.91	0.97	0.94
SVM (#2)	GloVe_TT + SA	0.85	0.86	0.97	0.91
LR (#3)	MiniLm + SA	0.90	0.73	1.00	0.85
Ensemble 1 (#4)	All	None	0.61	1.00	0.76
Ensemble 2 (#5)	All	0.80	0.97	1.00	0.99

Table 4.2: Performance of the chosen writing window models when classifying users in task 2, in validation, using the best thresholds found.

Model	Features	Threshold	Precision	Recall	F1
NB (#1)	Tf-Idf	0.90	0.86	0.84	0.85
LR (#2)	GloVe_TT + SA	0.98	0.66	0.98	0.79
SVM (#3)	MiniLm + SA	0.98	0.81	0.91	0.86
Ensemble 1 (#4)	All	None	0.52	1.00	0.68
Ensemble 2 (#5)	All	0.92	0.78	0.93	0.85

Decision-based Evaluation

The left half of Table 4.3 displays the final precision, recall and F1-score obtained during the event for task 1, drastically different from those previously seen in our testing, displayed in Table 4.1.

Initially, when analysing the results, we suspected that this discrepancy may have been caused by multiple factors:

- The single confidence threshold approach might be too simple and ineffective.
- The approach may work but the thresholds selected were overfitting on the limited, and skewed (since we enforced a balance between classes) validation data and were as a result sub-optimal.
- The selected thresholds were tuned for the first 100 windows of writings of the users in our test set, a big difference from the number of windows received and analysed from the server (1002).

Immediately upon receiving the results, using the golden truth file received post-submission, we tested the exact same models in the same conditions, but evaluating only the first 100 posts. We noticed a clear improvement across the board on F1-scores by simply reducing the amount of writings evaluated, mimicking the scenario used in the tuning process. This impact can be explained due to the fact that it takes only one positive writing window classification for the server to deem the subject as positive (final decision),

Table 4.3: Comparison of the decision-based performance on task 1 upon official evaluation (processing 1002 posts) vs when mimicking the tuning scenario of only analysing the first 100 writings.

Model	Features	P	R	F1	P@100	R@100	F1@100
LR	Tf-Idf	0.09	0.99	0.17	0.15	0.95	0.27
SVM	GloVe_TT + SA	0.07	1.00	0.13	0.10	0.99	0.19
LR	MiniLm + SA	0.05	1.00	0.10	0.06	1.00	0.12
Ensemble 1	All	0.05	1.00	0.10	0.06	1.00	0.11
Ensemble 2	All	0.19	0.99	0.32	0.31	0.91	0.47

and when the amount of writings rises dramatically from the one used in training (roughly 1000 vs 100), it is only natural that some positive classifications might be made where they should not (especially since the thresholds were not tuned for this amount of writing windows), resulting in a high number of false positives and high recall at the expense of a low precision and F1-score that we see here. This effect will become especially obvious soon, when we see the results of task 2, where half of the writings were processed (roughly 500 vs 1000) and the initial results were much better than those of task 1, despite the opposite being observed consistently throughout the training and validation stages (though as we will see, the choice of stricter thresholds also played a part). The results limited to the first 100 writings, shown in the right half of Table 4.3, although better, were still lacking, displaying signs that the other mentioned factors or something else entirely could be deteriorating our models' performances.

Rank-based Evaluation

In this case too, most of the rank-based evaluation results were quite poor. Despite the factors that resulted in subpar decision-based evaluation metrics, there was also a mistake in the implementation of the risk estimation calculations, that resulted in high scores (estimated levels of risk) being sent even in some cases when the user was not deemed to be a positive subject, meaning that our intended risk estimation protocol was not correctly implemented. These results are shown in Table 4.4.

Table 4.4: Rank-based evaluation results of the submitted models after analysing 100 writings of the official testing set of task 1.

Model	Features	P@10	NDCG@10	NDCG@100
LR	Tf-Idf	0.8	0.87	0.33
SVM	GloVe_TT + SA	0.0	0.00	0.00
LR	MiniLm + SA	0.4	0.30	0.29
Ensemble 1	All	0.0	0.0	0.10
Ensemble 2	All	0.0	0.00	0.03

Interestingly, despite this incorrect implementation, model 1 in particular still managed to achieve a good P@10 and NDCG@10, indicating that out of all subjects, it still managed to correctly rank a significant amount of positive users in the top 10 with highest estimated risk, the same could not be said for the top 100 though, as seen in its NDCG@100. The other models performed rather poorly across all the documented metrics.

4.1.2 Task 2: Depression

Since we followed a similar approach in task 2, the disparity between results in testing and official evaluation was also present.

Decision-based Evaluation

Despite showing worse performance in training, the combination of stricter thresholds and the reduced number of writings analysed brought better results when compared to task 1. But the factors we mentioned for task 1 still remained a cause for concern.

Rank-based Evaluation

The previously mentioned mistake was also present for this task's implementation, so naturally the rank-based evaluation was far from the best in this case too as seen in Table

Table 4.5: Comparison of the decision-based performance on task 2 upon official evaluation (processing 503 posts) vs when mimicking the tuning scenario of only analysing the first 100 writings.

Model	Features	P	R	F1	P@100	R@100	F1@100
NB	Tf-Idf	0.22	0.95	0.36	0.31	0.90	0.46
LR	GloVe_TT + SA	0.09	0.97	0.17	0.10	0.95	0.19
SVM	MiniLm + SA	0.17	0.97	0.29	0.23	0.89	0.37
Ensemble 1	All	0.09	0.99	0.17	0.10	0.97	0.18
Ensemble 2	All	0.38	0.86	0.53	0.47	0.76	0.58

4.6.

Table 4.6: Rank-based evaluation results of the submitted models after analysing 100 writings of the official testing set of task 2.

Model	Features	P@10	NDCG@10	NDCG@100
NB	Tf-Idf	0.2	0.15	0.15
LR	GloVe_TT + SA	0.2	0.25	0.14
SVM	MiniLm + SA	0.6	0.60	0.36
Ensemble 1	All	0.2	0.26	0.14
Ensemble 2	All	0.0	0.00	0.04

Though it is worth mentioning that model 3 still performed decently on the metrics for the top 10 highest estimated risk users (P@10 and NDCG@10), but it faltered when taking into account a higher range of the top estimated risk metric subjects (NDCG@100).

4.2 Post-submission Results

As we mentioned in the previous section, the performance achieved was far from the one we hoped for, so we made it our goal to address the multiple factors which we suspected to be causing issues, and to revisit the whole training/validation process while making some small tweaks and performing additional experiments. Most of these changes were already documented in previous chapter, such as the integration of Optuna's tools to optimise models along the way, the change in user classification validation (to employ 5-fold CV and to integrate left-out undersampled data) as well as tests with more pre-trained DSWE and CLME, since at the time of submission we did not include GloVe_CC nor MPNet for DSWE and CLME experiments respectively. We also performed experiments with windows of size 10, which at the time of submission we did not, due to a misunderstanding with the Sentence-transformers library sequence length limitation, where we thought that it would be inadequate to use windows larger than 5 writings, but as was shown in the previous chapter, increasing the size to 10 improved the results considerably.

In summary, the changes in the validation stage addressed the threshold tuning concerns mentioned in the previous section. To address the possible issues with the user classification protocol, we integrated the MCC, RCC and CCC as we explained in the previous chapter. And this time, we limited ourselves to 100 writings as intended, and fixed the estimated risk calculation.

4.2.1 Decision-based Evaluation

The final results achieved on the official testing sets in terms of decision-based metrics are shown in Tables 4.7 and 4.8, where we also show some of the best runs by other teams.

Table 4.7: Decision-based evaluation results of the final models after analysing 100 writings of the official testing set of task 1, compared to some of the best runs in eRisk 2022.

Model	Criterion	P	R	F1
LR (Tf-Idf)	SCC	0.938	0.741	0.828
	MCC	0.949	0.691	0.800
	RCC	0.968	0.741	0.839
	CCC	0.966	0.691	0.806
LR (GloVe_CC + SA)	SCC	0.447	0.679	0.539
	MCC	0.530	0.654	0.586
	RCC	0.610	0.617	0.614
	CCC	0.538	0.617	0.575
LR (MPNet)	SCC	0.969	0.778	0.863
	MCC	0.984	0.765	0.861
	RCC	0.969	0.778	0.863
	CCC	0.984	0.753	0.853
Ensemble (All) (LR + LR + LR)	SCC	0.953	0.753	0.841
	MCC	0.983	0.728	0.837
	RCC	0.968	0.753	0.847
	CCC	0.983	0.716	0.829
NN (MPNet)	SCC	0.971	0.815	0.886
	MCC	0.970	0.790	0.871
	RCC	0.971	0.815	0.886
	CCC	0.970	0.790	0.871
UNED-NLP Run #4		0.809	0.938	0.869
SINAI Run #2		0.908	0.728	0.808
UNSL Run #1		0.461	0.938	0.618

Table 4.8: Decision-based evaluation results of the final models after analysing 100 writings of the official testing set of task 2, compared to some of the best runs in eRisk 2022.

Model	Criterion	P	R	F1
NB (Tf-Idf)	SCC	0.637	0.663	0.650
	MCC	0.639	0.633	0.636
	RCC	0.644	0.663	0.653
	CCC	0.659	0.592	0.624
LR (GloVe_CC)	SCC	0.193	0.398	0.26
	MCC	0.294	0.327	0.309
	RCC	0.260	0.327	0.290
	CCC	0.240	0.378	0.294
LR (MPNet)	SCC	0.714	0.255	0.376
	MCC	0.720	0.184	0.293
	RCC	0.714	0.255	0.376
	CCC	0.714	0.153	0.252
Ensemble (All) (NB + LR + LR)	SCC	0.639	0.398	0.491
	MCC	0.688	0.337	0.452
	RCC	0.660	0.357	0.464
	CCC	0.642	0.347	0.450
NN (MPNet)	SCC	0.333	0.898	0.486
	MCC	0.615	0.602	0.608
	RCC	0.446	0.806	0.575
	CCC	0.667	0.551	0.603
NLPGroup-IISERB Run #0		0.682	0.745	0.712
BLUE Run #0		0.395	0.898	0.548
SCIR2 Run #0		0.396	0.837	0.538

The additional tweaks provided significant improvements in the results, for task 1 the highest F1-score achieved was of 0.886 with the NN model. At the time of submission this model would have outperformed any other of those participating in eRisk in terms of F1-score (the best model reported had F1-score of 0.869 UNED-NLP's run #4), while only analysing the first 100 posts, resulting in fast and accurate classifications. In the case of detecting depressed users, we also see significant improvements, but in this case, the best model reported, NLPGroup-IISERB's run #0, would have still outperformed ours with an F1-score of 0.712, where our best was the BoW model with 0.653.

It seems that the solutions developed with the BoW and CLME features were a success while those relying on DSWE not so much, this could be due to our vectorization approach of transforming documents into vectors by summing each token's embedding and then dividing by the total number of tokens across each dimension. Perhaps a solution

where documents were forced to be of a constant length in terms of tokens (with padding and truncation) would work better by dealing with the variation in the length of the writing windows. It seems unlikely that the failure was due to the word embedding models themselves since they have been employed in similar tasks before to greater success, as seen in the relevant work section of the background chapter.

4.2.2 Rank-based Evaluation

In the official evaluation, P@10, NDCG@10 and NDCG@100 were measured at 1, 100, 500 and 1000 writings. Given that 1 writing is never enough for our models to make an educated guess of an individual's level of risk (since the first window is available only at the time of the 10th writing), and since we do not analyse writings past the 100th, we re-evaluated the models using the same metrics, but only at 100 writings. In this final iteration we also fixed our previous risk estimation methodology. Now, we keep a counter for each subject, at each inference, if the writing window classification output is 0 we do nothing, but if it is 1, we increment the counter with the model's confidence in its prediction. The resulting rank-based performances are displayed in Tables 4.9 and 4.10.

Table 4.9: Rank-based evaluation results of the final models after analysing 100 writings of the official testing set of task 1, compared to some of the best runs in eRisk 2022.

Model	Criterion	P@10	NDCG@10	NDCG@100
LR (Tf-Idf)	SCC	1	1	0.997
	MCC	1	1	0.995
	RCC	1	1	0.996
	CCC	1	1	0.996
LR (GloVe_CC + SA)	SCC	0.6	0.980	0.913
	MCC	0.9	1	0.938
	RCC	0.9	1	0.933
	CCC	1	1	0.943
LR (MPNet)	SCC	1	1	0.999
	MCC	1	1	0.999
	RCC	1	1	0.993
	CCC	1	1	0.999
Ensemble (All) (LR + LR + LR)	SCC	1	1	0.987
	MCC	1	1	0.984
	RCC	1	1	0.981
	CCC	1	1	0.982
NN (MPNet)	SCC	1	1	0.999
	MCC	1	1	0.999
	RCC	1	1	0.999
	CCC	1	1	1
UNSL Run #1		1.0	1.0	0.90
BLUE Run #1		1.0	1.0	0.89
UNED-NLP Run #4		1.0	1.0	0.88

Table 4.10: Rank-based evaluation results of the final models after analysing 100 writings of the official testing set of task 2, compared to some of the best runs in eRisk 2022.

Model	Criterion	P@10	NDCG@10	NDCG@100
NB (Tf-Idf)	SCC	0.9	0.950	0.928
	MCC	0.9	0.950	0.931
	RCC	0.8	0.956	0.929
	CCC	0.8	0.950	0.934
LR (GloVe_CC)	SCC	0.8	0.917	0.849
	MCC	0.6	0.879	0.850
	RCC	0.6	0.910	0.873
	CCC	0.5	0.907	0.866
LR (MPNet)	SCC	0.7	0.776	0.860
	MCC	0.8	0.993	0.945
	RCC	0.8	0.975	0.943
	CCC	0.9	0.990	0.948
Ensemble (All) (NB + LR + LR)	SCC	0.7	0.973	0.920
	MCC	0.7	0.991	0.932
	RCC	0.8	0.961	0.932
	CCC	0.7	0.972	0.926
NN (MPNet)	SCC	0.9	0.984	0.946
	MCC	0.8	0.983	0.945
	RCC	0.8	0.975	0.940
	CCC	0.9	0.977	0.942
BLUE Run #1		0.7	0.64	0.67
Sunday-Rocker2 Run #1		0.9	0.93	0.66
UNSL Run #1		0.6	0.73	0.64

These results suggest that all of our models, even those that performed worse in terms of final decisions (decision-based evaluation) are very capable of ranking the subjects in terms of estimated risk, both when looking at the top 10, and more impressively, at the top 100 highest risk subjects. Though this is not only a consequence of our models' quality, but also of the risk estimation process chosen.

4.3 Summary

Due to various tuning and implementation mistakes, the initial results were found to be subpar, but we managed to significantly improve them by introducing some changes and corrections in the training, validation and testing stages of the experimental process. In the case of the writing window classifiers:

- Added an additional DSWE pre-trained model (GLOVE_CC) for feature extraction.
- Added an additional CLME pre-trained model (MPNet) for feature extraction.
- Trained NNs alongside the initial classic ML algorithms.

As for the user classification stage:

- Used 5-fold CV (by re-training the models with the optimized parameters for every iteration) instead of a single train/validation split.
- Included the left-out (due to undersampling) control subject data.
- Tested additional decision criteria (MCC, RCC and CCC).

Regarding the final testing, we also fixed our code to work as initially intended, in terms of only analysing the initial 100 writings for each subject, and providing the correct risk estimation for the rank-based evaluation.

Regarding the decision-based metrics, the best model in our final results for task 1 outperformed all of the models used by the other teams participating in eRisk, narrowly beating UNED-NLP's run #4 in terms of F1-score. As for task 2 our final models were also among the top performers of those present in eRisk, though they still get outperformed by some of the others.

In the case of the rank-based metrics, our models seem to be very good at ranking the subjects in terms of risk estimation, achieving the best results across all metrics considered for both tasks.

Chapter 5

Conclusions

This project's goal was to determine how effective machine learning methods can be in detecting subjects at risk of suffering from mental illnesses, using their writings extracted from social media platforms. To this end, we participated and published a paper [34] in 2022's edition of CLEF eRisk, on tasks 1 and 2, concerned with detecting pathological gambling and depression respectively.

In the experimental section, we made it a goal of ours to compare the effectiveness of the 3 types of textual features most commonly found in the literature, though we also performed some experiments including sentiment analysis features. We saw that despite the different degrees of complexity regarding the various textual features, models trained with Bag-of-Words and contextualised language model features performed best while those using features of distributional semantics word embeddings faltered, perhaps due to our implementation method. The inclusion of sentiment analysis features did not prove to be very useful in our experiments, despite the fact that it has been employed in the literature to great success, perhaps more experiments with different sentiment analysis tools or different integration methods would be required to fully take advantage of the information they provide.

The results varied significantly for task 1 and 2, with task 2 solutions achieving worse F1-scores across the board. This may be due to a variety of factors, either originating from the quality of dataset used, our methods, or perhaps due to an inherent difficulty in detecting depression from social media speech. It may be the case that depressed subjects are more subtle in their cues than pathological gamblers, this hypothesis is supported by the quality of the results achieved in task 1 with BoW features, showing us that the use of certain specific terms like "gambling", "bet", "wager" or "money" is very relevant when making the distinction from the control group. While in the case of task 2, the most influential tokens in the model with BoW features were mostly common terms that would not necessarily be associated with depression.

There are a lot of different ways to further improve on this work. We could gather more data to form a new dataset, consider more feature types besides our main 3, by including descriptive features for instance, since the statistical analysis of our datasets

matched those of works that got good results employing such features. We could also further invest in the main 3 by experimenting with other pre-trained models or even by training or fine-tuning some of those ourselves. And last but not least, we could try to tackle the decision criterion section of the problem in a more ML-based manner rather than using pre-determined rules, by letting models such as NNs come up with their own decision protocols learned by analysing large amounts of the writing window models' output (turning the several confidence values into a classification).

Nevertheless, our experiments proved that ML is a powerful tool to be associated with more standard/clinical procedures for an early detection of mental health problems, with varying degrees of success partly depending on the mental issue being addressed.

References

[1] *Health and Well-Being*. Online; accessed January, 2022.
URL: https://www.who.int/data/gho/data/major-themes/health-and-well-being (cit. on p. 1).

[2] *Suicide - key facts*. Online; accessed January, 2022. 2021.
URL: https://www.who.int/news-room/fact-sheets/detail/suicide (cit. on p. 1).

[3] Michael J. Paul and Mark Dredze. "Social Monitoring for Public Health." In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 9 (5 Aug. 2017), pp. 1–183. ISSN: 1947-945X. DOI: 10.2200/S00791ED1V01Y201707ICR060. (Cit. on p. 3).

[4] Fabio Crestani, David E. Losada, and Javier Parapar. "Early Detection of Mental Health Disorders by Social Media Monitoring." In: *Studies in Computational Intelligence* 1018 (2022). Ed. by Fabio Crestani, David E. Losada, and Javier Parapar. DOI: 10.1007/978-3-031-04431-1. (Cit. on p. 4).

[5] *Roc Curve*. Online; accessed January, 2022. 2018.
URL: https://commons.wikimedia.org/wiki/File:Roc_curve.svg (cit. on p. 10).

[6] Susel Góngora Alonso, Isabel de la Torre-Díez, Sofiane Hamrioui, Miguel López-Coronado, Diego Calvo Barreno, Lola Morón Nozaleda, and Manuel Franco. "Data Mining Algorithms and Techniques in Mental Health: A Systematic Review." In: *Journal of Medical Systems* 42 (9 Sept. 2018), p. 161. ISSN: 0148-5598. DOI: 10.1007/s10916-018-1018-2. (Cit. on p. 10).

[7] Zhijun Yin, Lina M. Sulieman, and Bradley A. Malin. "A systematic literature review of machine learning in online personal health data." In: *Journal of the American Medical Informatics Association* 26 (6 Mar. 2019), pp. 561–576. ISSN: 1527974X. DOI: 10.1093/JAMIA/OCZ009. (Cit. on p. 10).

[8] *Decision Tree - survival of passengers on the Titanic*. [Online; accessed January, 2022]. 2020.
URL: https://commons.wikimedia.org/wiki/File:Decision_Tree_-_survival_of_passengers_on_the_Titanic.jpg (cit. on p. 13).

[9] *Non-linear SVM*. Online; accessed January, 2022. 2015.
URL: https://www.researchgate.net/figure/4-Non-linear-SVM-Image-from-19_fig7_334784703 (cit. on p. 14).

[10] *Fundamentals of Neural Networks*. Online; accessed January, 2022. 2019.
URL: https://wandb.ai/site/articles/fundamentals-of-neural-networks (cit. on p. 16).

[11] *Creating Word Embeddings: Coding the Word2Vec Algorithm in Python using Deep Learning*. Online; accessed January, 2022. 2020.
URL: https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8 (cit. on p. 20).

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* (Jan. 2013).
URL: https://arxiv.org/abs/1301.3781v3 (cit. on p. 20).

[13] *A Complete Guide To Understand Evolution of Word to Vector*. Online; accessed January, 2022. 2021.
URL: https://medium.com/co-learning-lounge/nlp-word-embedding-tfidf-bert-word2vec-d7f04340af7f (cit. on p. 21).

[14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.
URL: https://aclanthology.org/D14-1162 (cit. on pp. 21, 42).

[15] *Implementing Deep Learning Methods and Feature Engineering for Text Data: The GloVe Model*. Online; accessed January, 2022. 2018.
URL: https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-glove.html (cit. on p. 21).

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5 (July 2016), pp. 135–146. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00051.
URL: https://arxiv.org/abs/1607.04606v2 (cit. on p. 22).

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. *Deep contextualized word representations*. 2018.
DOI: 10.48550/ARXIV.1802.05365.
URL: https://arxiv.org/abs/1802.05365 (cit. on p. 22).

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners." In: (2020). arXiv: 2005.14165 [cs.CL]. (Cit. on p. 22).

[19] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1 (Oct. 2018), pp. 4171–4186.
URL: https://arxiv.org/abs/1810.04805v2 (cit. on pp. 22, 23).

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need." In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 5999–6009. ISSN: 10495258.
URL: https://arxiv.org/abs/1706.03762v5 (cit. on p. 22).

[21] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2019. DOI: 10.48550/ARXIV.1909.11942.
URL: https://arxiv.org/abs/1909.11942 (cit. on p. 22).

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692.
URL: https://arxiv.org/abs/1907.11692 (cit. on p. 22).

[23] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2019. DOI: 10.48550/ARXIV.1906.08237.
URL: https://arxiv.org/abs/1906.08237 (cit. on pp. 23, 44).

[24] Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. "Natural Language Processing of Social Media as Screening for Suicide Risk." In: *Biomedical Informatics Insights* 10 (Jan. 2018). ISSN: 1178-2226. DOI: 10.1177/1178222618792860. (Cit. on pp. 23, 30).

[25] Robert Thorstad and Phillip Wolff. "Predicting future mental illness from social media: A big-data approach." In: *Behavior Research Methods* 51 (4 Aug. 2019). ISSN: 1554-3528. DOI: 10.3758/s13428-019-01235-z. (Cit. on pp. 24, 30).

[26] Raymond Chiong, Gregorius Satia Budhi, Sandeep Dhakal, and Fabian Chiong. "A textual-based featuring approach for depression detection using machine learning classifiers and social media texts." In: *Computers in Biology and Medicine* 135 (Aug. 2021), p. 104499. ISSN: 0010-4825. DOI: 10.1016/J.COMPBIOMED.2021.104499. (Cit. on pp. 25, 30, 34).

[27] Xuetong Chen, Martin D. Sykora, Thomas W. Jackson, and Suzanne Elayan. "What about Mood Swings: Identifying Depression on Twitter with Temporal Measures of Emotions." In: *The Web Conference 2018 - Companion of the World Wide Web Conference, WWW 2018* (Apr. 2018), pp. 1653–1660. DOI: 10.1145/3184558.3191624. (Cit. on pp. 25, 30).

[28] Martin D. Sykora, Thomas W. Jackson, Ann O'Brien, and Suzanne Elayan. "Emotive ontology: Extracting fine-grained emotions from terse, informal messages." In: *Proceedings of the IADIS International Conference Intelligent Systems and Agents 2013, ISA 2013, Proceedings of the IADIS European Conference on Data Mining 2013, ECDM 2013* (2013), pp. 19–26. ISSN: 1646-3692. (Cit. on p. 26).

[29] Yla R. Tausczik and James W. Pennebaker. "The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods:" in: *http://dx.doi.org/10.1177/0261927X09351676* 29 (1 Dec. 2009), pp. 24–54. ISSN: 0261927X. DOI: 10.1177/0261927X09351676.
URL: https://journals.sagepub.com/doi/10.1177/0261927X09351676 (cit. on p. 26).

[30] Ahmet Emre Aladag, Serra Muderrisoglu, Naz Berfu Akbas, Oguzhan Zahmacioglu, and Haluk O. Bingol. "Detecting Suicidal Ideation on Forums: Proof-of-Concept Study." In: *J Med Internet Res 2018;20(6):e215 https://www.jmir.org/2018/6/e215* 20 (6 June 2018), e9840. ISSN: 14388871. DOI: 10.2196/JMIR.9840.
URL: https://www.jmir.org/2018/6/e215 (cit. on pp. 26, 30).

[31] Amir Hossein Yazdavar, Hussein S. Al-Olimat, Monireh Ebrahimi, Goonmeet Bajaj, Tanvi Banerjee, Krishnaprasad Thirunarayan, Jyotishman Pathak, and Amit Sheth. "Semi-Supervised approach to monitoring clinical depressive symptoms in social media." In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2017* (July 2017), pp. 1191–1198. DOI: 10.1145/3110025.3123028.
URL: http://dx.doi.org/10.1145/3110025.3123028 (cit. on pp. 27, 30).

[32] Fidel Cacheda, Diego Fernandez, Francisco J. Novoa, and Victor Carneiro. "Early detection of depression: Social network analysis and random forest techniques." In: *Journal of Medical Internet Research* 21 (6 June 2019). ISSN: 14388871. DOI: 10.2196/12554. (Cit. on pp. 27, 30, 35).

[33] David E. Losada and Fabio Crestani. "A test collection for research on depression and language use." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9822 LNCS (2016), pp. 28–39. ISSN: 16113349. DOI: 10.1007/978-3-319-44564-9_3. (Cit. on p. 29).

[34] Rodrigo Ferreira, Alina Trifan, and José Luís Oliveira. "Early risk detection of mental illnesses using various types of textual features." In: *Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*. 2022. (Cit. on pp. 31, 58, 71).

[35] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019. (Cit. on p. 32).

[36] Javier Parapar, Patricia Martín-Rodilla, David E. Losada, and Fabio Crestani. "Overview of eRisk 2021: Early Risk Prediction on the Internet." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12880 LNCS (2021), pp. 324–344. ISSN: 16113349. DOI: 10.1007/978-3-030-85251-1_22. (Cit. on p. 33).

[37] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. "Bag of Tricks for Efficient Text Classification." In: *arXiv preprint arXiv:1607.01759* (2016). (Cit. on p. 34).

[38] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. "FastText.zip: Compressing text classification models." In: *arXiv preprint arXiv:1612.03651* (2016). (Cit. on p. 34).

[39] David E. Losada, Fabio Crestani, and Javier Parapar. "eRISK 2017: CLEF lab on early risk prediction on the internet: Experimental foundations." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10456 LNCS (2017), pp. 346–360. ISSN: 16113349. DOI: 10.1007/978-3-319-65813-1_30. (Cit. on p. 36).

[40] David E. Losada, Fabio Crestani, and Javier Parapar. "Overview of eRisk: Early risk prediction on the internet." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11018 LNCS (2018), pp. 343–361. ISSN: 16113349. DOI: 10.1007/978-3-319-98932-7_30. (Cit. on p. 36).

[41] E. Campillo-Ageitos, H. Fabregat, L. Araujo, and J. Martinez-Romo. "NLP-UNED at eRisk 2021: self-harm early risk detection with TF-IDF and linguistic features." In: *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum*. 2021. (Cit. on pp. 40, 51).

[42] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. "Advances in Pre-Training Distributed Word Representations." In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018. (Cit. on p. 42).

[43] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. DOI: 10.48550/ARXIV.2002.10957.
URL: https://arxiv.org/abs/2002.10957 (cit. on p. 44).

[44] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. *MPNet: Masked and Permuted Pre-training for Language Understanding*. 2020. DOI: 10.48550/ARXIV.2004.09297.
URL: https://arxiv.org/abs/2004.09297 (cit. on p. 44).