Data for Good: predicting suicidal behavior likelihood using Deep Learning.

Sebastiano Denegri

April 25th, 2023

Happiness can be found, even in the darkest of times, if one only remembers to turn on the light.
J.K. Rowling, Harry Potter and the Prisoner of Azkaban

Deep Learning and Reinforcement Learning – IBM / Coursera.

Table of contents

1. Introduction: the project

Data for good means using Data Science and Machine Learning tools outside of the for-profit sector, to help Non-profits, NGOs, or any other organization or individual, leverage the power of data for good causes and to improve the life of others.

There are many ways to use the power of Data Science for good: data can be used to solve social issues, environmental problems, enhance community security as well as support people in need.

Nowadays, the use of social media, forums or news aggregation websites is massively widespread, with people sharing plenty of details about their life. Some people also use internet to share very serious issues, as a cry for help.

The scope of this project is to identify, amongst users in an online community, people at risk of self-harm or suicide, so that actions can be undertaken to provide Targeted Suicide Intervention in a timely-fashioned and sustained manner.

To achieve the project purpose, I've analyzed user-generated content from a social discussion website (Reddit), about mental health issues, to ultimately predict if users are struggling with severe mental problems and therefore assessing whether they are at risk of self-harm (so that support can be provided) or not.

2. Methodology

I've followed a **predictive analytic approach** aimed to **correctly classify users**, based on their posts' content, **into the correct category**:

1. **Not at risk (Negative Class)**: users don't struggle with severe mental issues and they are not at risk of self-harm.
2. **At risk (Positive Class)**: users struggle with mental health issues, they may be dealing with suicidal thoughts and/or suicidal behavior, or have already tried to commit suicide.

To deliver reliable results, I've applied the **Cross-Industry Standard Process for Data Mining (CRISP-DM)**, which consists of the following steps:

1. **Project Understanding** (see the Introduction section).
2. **Data Understanding**: data cleaning and exploratory data analysis.
3. **Data Preparation**: transform the data into a usable dataset for modeling.
4. **Modeling**: I've built 4 Deep Learning models, using the Recurrent Neural Network class.
5. **Evaluation**: I compared and evaluated models' performances using the following metrics: Accuracy Score, Confusion Matrix, Sensitivity, Specificity, F1-Score, ROC and Precision-Recall curves.

Since we are dealing with Natural Language Processing (NLP), I had to "transform" words in a way that they can be fed through a Neural Network, that is representing words with numbers. This is called **Word Embedding**: a representation of a word as a numeric vector that encodes the meaning of that word. Words that have similar meaning are (or should be) closer in the vector space.

When working with word embeddings, there are mainly 2 options: learn your own embeddings, using the data you have, or apply **Transfer Learning** concept, that is using word embeddings that have already been pre-trained on large datasets, saved, and can be re-used for solving other tasks.

For this project, I've built and trained 4 models:

- Learning the Word Embeddings from scratch, using the training data of my dataset.
- Using pre-trained GloVe Word Embeddings.
- Using pre-trained Google News Word2Vec Word Embeddings.
- Selected the best model amongst the above ones, and applied regularization techniques.

3. Data Understanding

Dataset source: www.kaggle.com.

The dataset includes posts from 500 redditors that have discussed topics about suicide and mental health illnesses, like depression. The posts have been classified into 5 categories, following a modified version of the guidelines outlined in the Columbia Suicide Severity Rating Scale (C-SSRS).

The dataset contains 3 columns for 500 observations, where each column contains text-type data. Columns are as follows:

- **User**: a string of characters representing an anonymized Reddit user ID.
- **Post**: the texts from the user (without no personal information revealed).
- **Label**: the class the users has been assigned to, based on their post contents. Classes are: **attempt, behavior, ideation, indicator, and supportive.**

See below posts from one random user per each class:

```
Posts from a user labelled as "Attempt":
user-46:
['There is nothing else to share. Nothing can change now. No matter what I
share or what people tell me, my life will be the same. I guess you could
say its time I check out a bit early', 'I really do hope you help your stu
dents. All the teachers fucking avoided me because I was socially awkward
```

and didnt know hope to act. Just make one promise to me and that is no mat
ter how bad a student is, or how much they dont appreciate your help then
just stay with them. I wish I fucking had that rather than abuse I got. Ev
en if I graduate from something. Study something I WILL NEVER HAVE THE LIF
E I WANT. before you say that I should have other goals of value other thi
ngs dont bother. Some of us want to be doctors,engineers,teachers,athletes
. Just because I want to be rich doesnt mean Im selfish. We all want to he
lp everyone in the world. but sometimes we cant. We just focus on helping
one person, even if that person is ourself. I cant be saved. Helped. Every
one I trusted, doctors, ex girlfriend, mum. Couldnt help me so whats the p
oint. Even though I respect your time to say something and want to help me
. Its time for me to check out early. I will be leaving to my destination
shortly', 'Ive just arrived at the place Im going to spend my last day. It
s peaceful and isolated. Just like my life has always been. I understand w
hat it is like to be autistic, I hope you have found out a way to help you
rself. Even though Im socially awkward having friends wouldnt change a thi
ng to my life. ', 'Its time for me to go. Earlier than I expected but I th
ere is no point living through this shit anymore. I dont expect ever to be
found or remembered, even before I go I may turn back and released that I
m a fucking idiot. But thats unlikely. Thank you for your help even it was
for a day. ', 'I dreamt of becoming a football player every since I could
kick a ball. It was a thing I did every day for at least 10 years, I gener
ally loved it more than anything. But whats the point in doing it if I can
t make a living out of it. One day if I chose to live I may end up in a ni
ce apartment, maybe a partner who actually understands me and doesnt call
me childish or shout at me when Im acting inappropriately in public. Im in
a circle which is I cant be with people if I dont know how to act socially
, and I cant learn how to act socially if I cant be with people. Its a cir
cle Ive been in for years. I may end up having a friend and a career but i
t will just mask what I want in life. The only thing that has helped me wa
s a superman comic. A fictional character. But now the affect of that has
gone. If I was good at one fucking thing then I may stay around and have h
ope that one day I can reach my goal but no. No skills in fucking anything
to help me. Its not fair. Everyone can make friends, do well at things, ha
ve a nice life but it has always been fucking me. Im sorry for all the oth
er ADHD, Autistic, Aspergers suffers, hope you all do well in life but you
have lost a team member. It was too much for me and I must go. Thanks for
your help and I appreciated it. Even if it was for a day that someone I be
lieved cared for me. Thank you   ']

Posts from a user labelled as "Behavior":
user-134:
['Thank you so much for this.', 'I dont know what to say. I dont even know
how to feel or if my emotions are justified. I know that my dad has a righ
t to be frustrated with me. After I lost my job, the bills started to pill
on and now theyre hovering over my head like a storm cloud.But its not the
fact that Ive lost my job and have no room to my own thats bothering me ri
ght now. Its the hurtful things he said to me this morning.Now Im too Anxi
ety to sleep, I feel worse about myself than I have in the past few weeks.
The last time my father hugged me or showed any visible/palpable affection
towards me was when I graduated high school five years ago. I had to gradu
ate for him to tell me he was proud of me and that he loved me.Most of the
time its static apathy or blatant resentment. There is no warmth there. Im

always on eggshells around him. Which hurts even worse because I used to b
e a real daddys girl. We used to be close, when I was a child and things w
erent so complicated.Things between us fragmented over the years and I can
count the amounts of affection Ive had that are the most memorable.High sc
hool graduation being the most recent, the one before that was when I had
a panic attack in the car when my brothers and my parents were in it. (I w
as around 15-16 years old) and I said that if my brothers werent in the ca
r I would have crashed it... I was followed around the house by both of my
parents while I looked for something to Pain myself. Finally got a hug and
an I love you from him, then... It makes me wonder what measures I have to
take to get that sort of attention... Ive been sitting here looking at my
own bloodshot eyes in the mirror and wondering if I tried to OD if my dad
would be there in the hospital when I woke up. It makes me wonder if hed t
ell me he loved me. Whats worse is that I dont know if I would believe him
at this point.I dont want to die.But I dont want to be around right now, i
f that makes sense...The closest thing to what I want would be a coma... S
omewhere in limbo where my brain can shut down for a while and the world c
ould stop turning in my mind for a second...Im scared. Im sad. And I argue
with myself over how pathetic it is to fixate myself on one emotion for ho
urs on end instead of doing something about it.But its a lot like being st
uck with your back against a wall, bracing yourself for the impact of an o
ncoming tidal wave. You want to get out, and you know that when it hits yo
u- youre going to have to fight the tide but at this point Im unsure of ju
st how much strength I have left to try and get myself to the surface when
it collides...All I can do is cry. I know it will pass but right now Im ju
st terrified and at a loss.']

Posts from a user labelled as "Ideation":
user-227:
['possibly!sorry about the empty post. its sort of a hard subject to just
jump into.basically, ive been constantly Delusional disorder at my uni for
the past couple of months, and it has been extremely detrimental to my men
tal health. its especially bad when doing anything online when i most feel
like im being spied on, making it extremely difficult to reach out for hel
p in any way. i had to stop talking with one of my good friends online bec
ause the Delusional disorder was so bad.the whole situation makes me want
to dissapear because of how much it makes me feel isolated with no chance
that things will get better. ']

Posts from a user labelled as "Indicator":
user-171:
['Ironically social work. I never really had fun with his friends, he was
just emotionally stable but really really really lazy. Our friends were mo
stly people from his program in university, so I never really had too much
to talk to them about. I feel so guilty that I dont feel sad that hes gone
. I just feel so lonely all the time now. ', 'Still got another two years
to go haha. ', 'Yeah I think we can do it, we totally got this! But it is
hard to be normal. I spend so much time and energy trying to make it app
ear that my life is groovy and drama free that everything I do these days
feels like a lie. How does temporary academic leave work? Thanks so much f
or the reply.']

Posts from a user labelled as "Supportive":

```
user-29:
['Then maybe returningt there might be the first step. You should be w/ th
e family of your choice, not w/ people who clearly dont want the best for
you. You should take care of yourself for a while and while doing that may
be also put some distance between you and that girl - it might help you to
put things into perspective. Stay strong!', 'No matter whether you believe
in fate or God or anything: that misfiring gun was a SIGN and Im really th
ankful for it. I hope you can use this extreme experience to your advantag
e!', 'Scumbag brain... Sees something beautiful and uplifting: yanks the t
ear ducts wiiide open...', 'Please dont feel guilty. You have a disease an
d it is so strong that even your partners love cant seem to stop it. Its n
ot your fault and its not an unusual thing to happen. Maybe its really for
the best if you outsource the job of taking care of you. That could take t
he feeling of guilt off your shoulders and also relieve your partner of so
me of his responsibility. You have two great things working for you at the
moment: the fact that amazingly human beings care for each other uncondit
onally and are even able to forgive the worst-seeming things. Plus, the wi
ll to live that you expressed in your last paragraph. Its there, its true.
Cling to these things with all the strength you still have, thats all you
have to do, the other things are out of reach at the moment, you can care
about them once you feel better.   Get well soon!', 'Its unfair that many
people can just leave their beds w/o problems. It may seem like a miracle
to you, it sure does to me sometimes. But everyones got special challenges
in life and even those who jump our of bed wearing a bright smile each day
might be confronted w/ difficulties once they enter a car, make a cake or
whatever. Even if not, even if their life seems perfect, they are bound to
encounter some rocks on the road eventually. So this is your challenge. It
s one of the nastier ones, no question about that. But life functions Tire
d strangely and so I think you should try to hold on, try to take on day a
fter another, try to battle against all the rocks that are in your way. I
know its worth it and Im pretty confident youll find out the same if you j
ust hang in there.']
```

3.1. Data Cleaning

Results:

- No duplicate observations were found.
- No missing values were found.
- Standard format was applied to all users' texts: deleted punctuation, lowered case all words, removed leading, trailing and consecutive white spaces.

See below posts from a random user, before and after the cleaning:

Before the cleaning:

```
user-406:
['Ive tried that. Im Exhaustion of trying and falling at everything. Im lo
oking Att gun and trying to figure why not to do it. ', 'Heh, even reddit
cant give me a reason to keep on going.']
```

After the cleaning:

```
user-406:
ive tried that im exhaustion of trying and falling at everything im lookin
g att gun and trying to figure why not to do it heh even reddit cant give
me a reason to keep on going
```

3.2. Exploratory Data Analysis

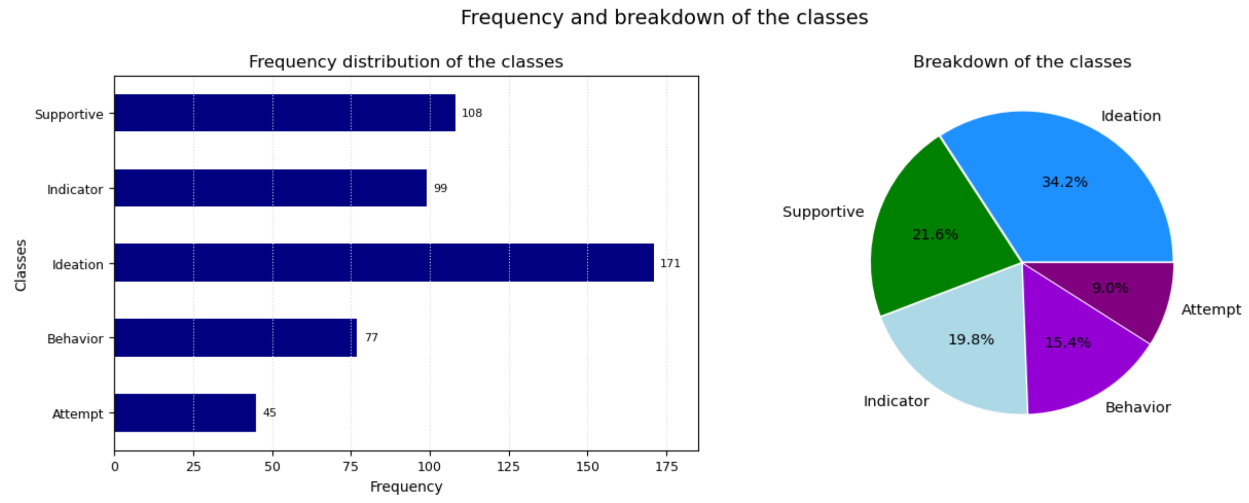

Figure 1: Frequency and breakdown of the classes

The dataset is quite imbalanced, with the biggest class, "Ideation", accounting for almost 35% of entries, and the smallest class, "Attempt", for less than 10%.

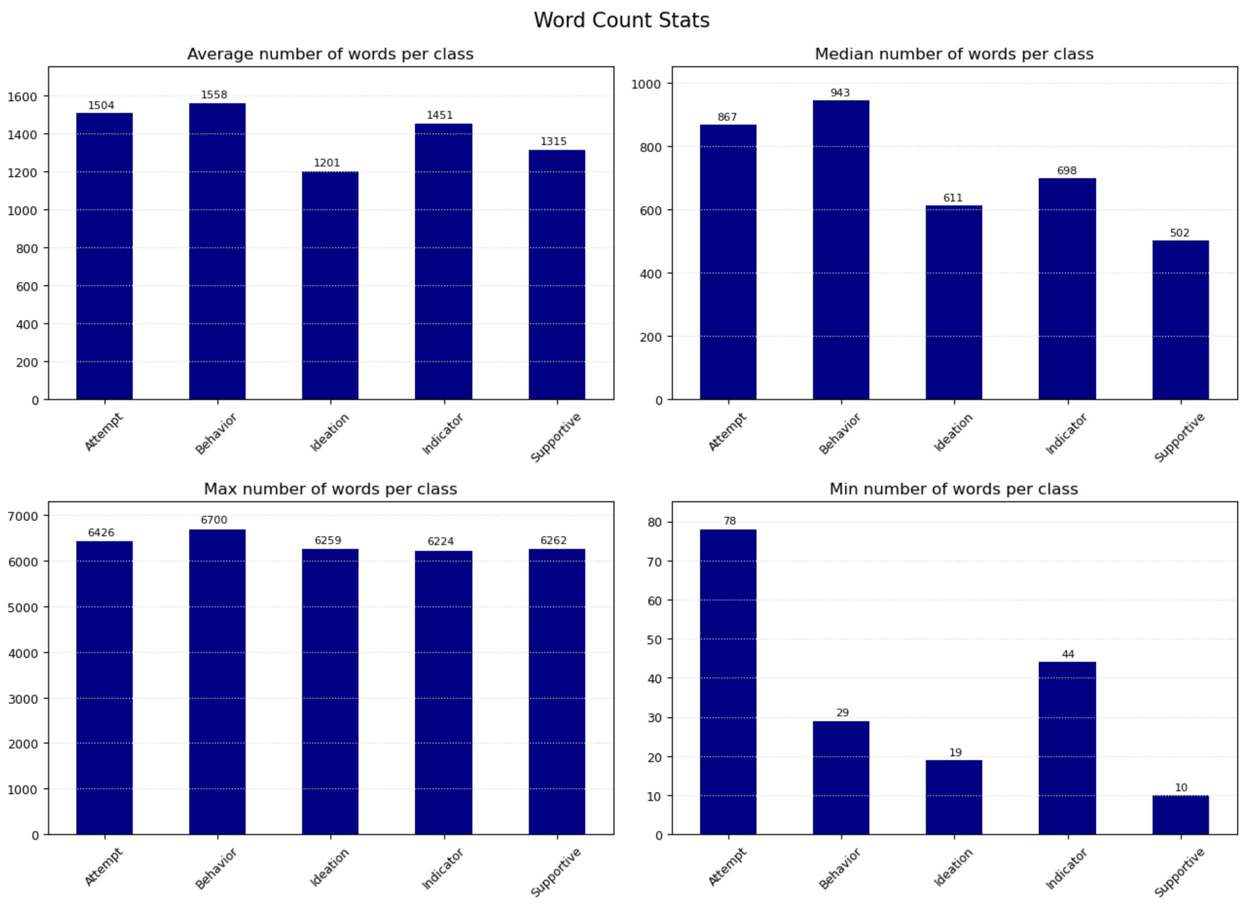

Figure 2: Word Count Statistics

Some considerations:

- The average number of words used by a user is around 1,200 - 1,500, depending on the class, whilst the median number of words is much lower: from 500 to 900 words, depending on the class. This means the mean number of words is skewed by few very long posts.
- The longest user's text in all classes is above 6,000 words: these are the texts that skew the mean.

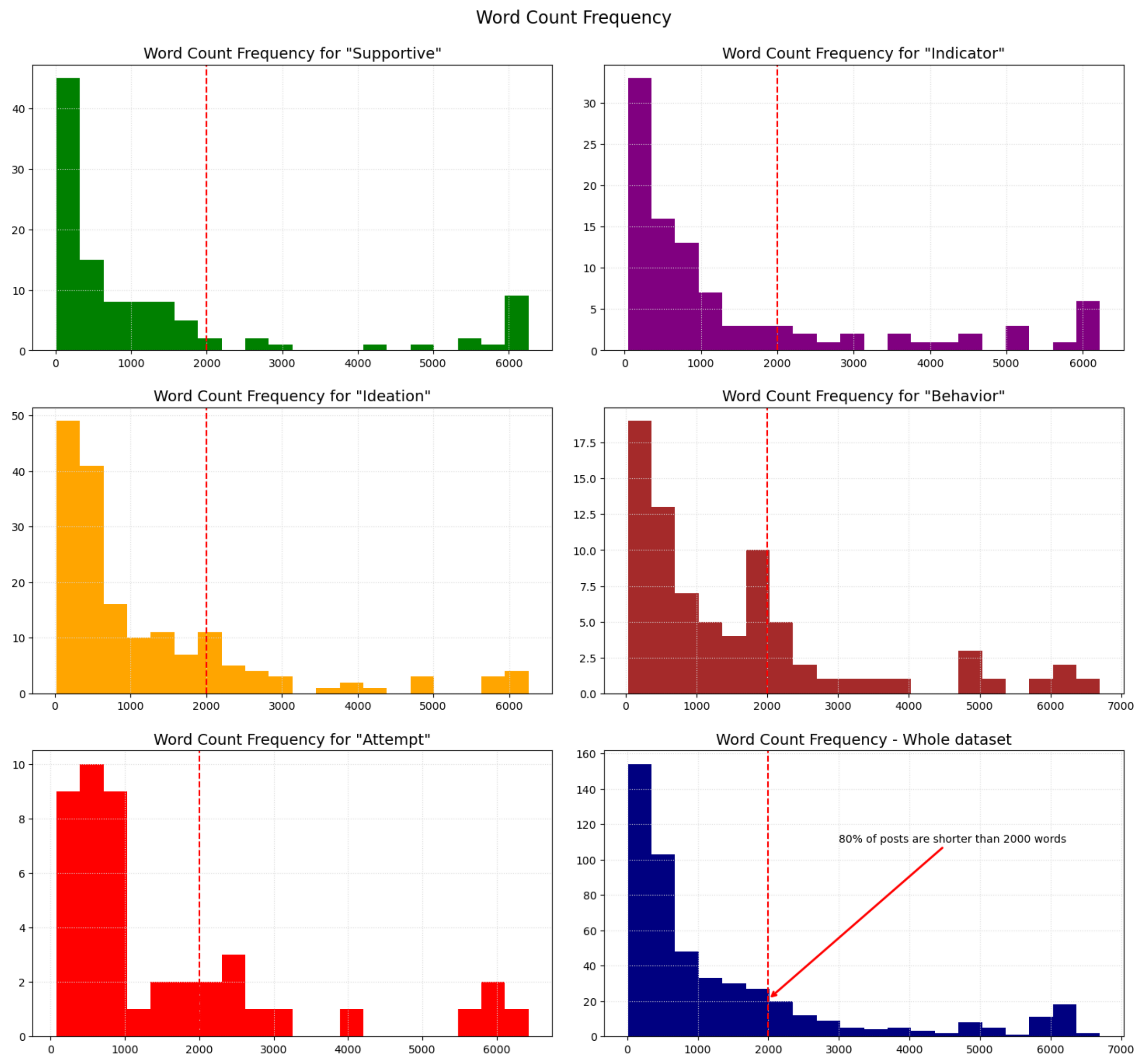

Figure 3: Word Count Frequency per Class

The Word Count Frequency is more or less the same in all classes (with users labelled as "Attempt" and "Behavior" that tend to write slightly longer texts then the others). **Almost 80% of the users' posts are shorter than 2,000 words.**

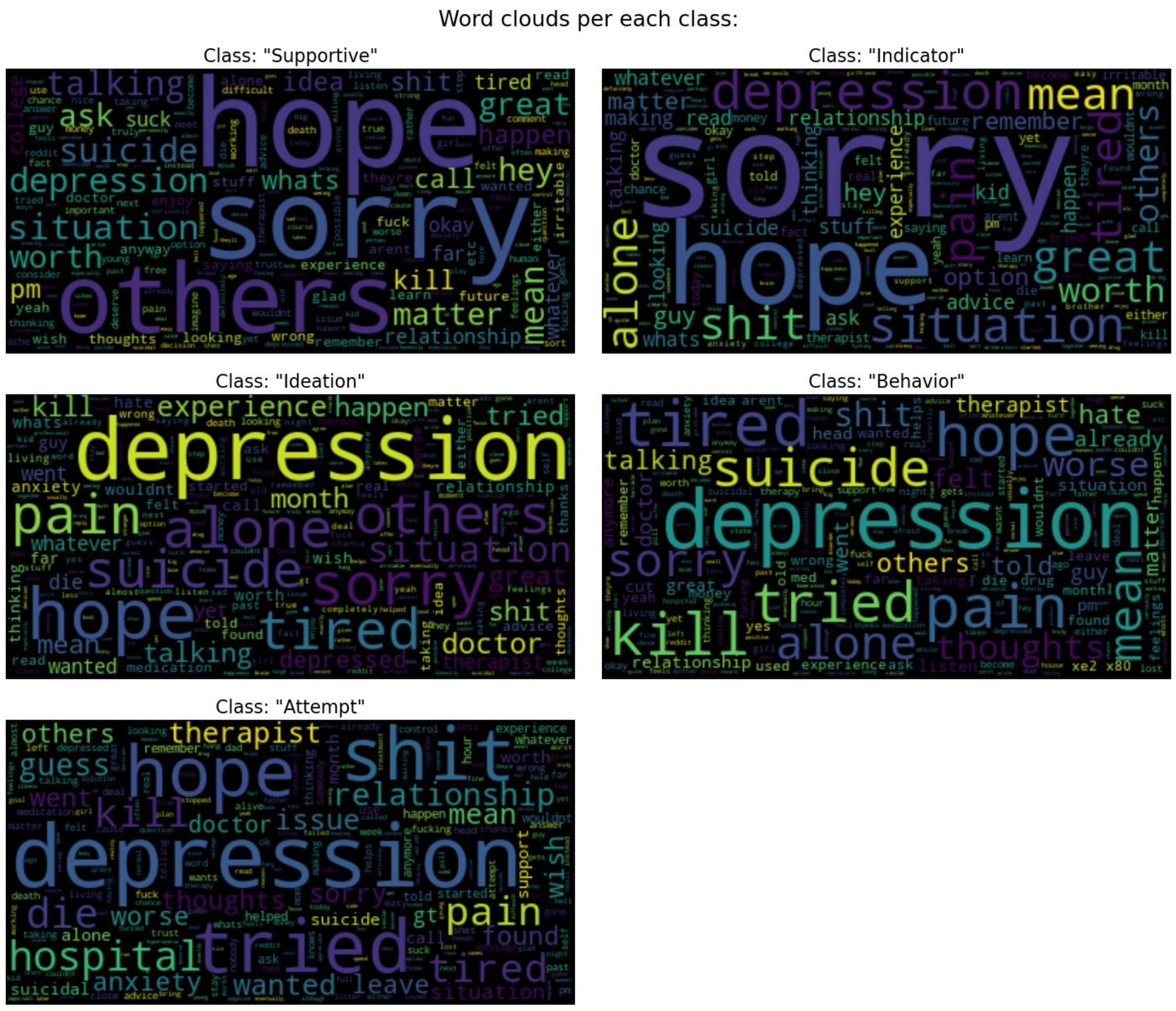

Figure 4: Word Cloud Analysis

Notes on the word clouds:

- Since 80% of users' texts are shorter than 2,000 words, to plot the word clouds I used the most frequent 2,000 words only.
- The first word clouds I plotted didn't show any type of patterns, or semantic meaning, with regard to the classes: the most common words (life, people, thing, know, want...) were, in fact, shared by all classes. Therefore, I added these "shared" words to the list of stop words, trying to show a clearer pattern, or boundary, between the classes. The word clouds included in this report are the final result, after finding and removing the most common shared words.
- The final list of stop words includes 372 words.

Thanks to the word cloud analysis, we can have a better understanding of the classes, and start defining the boundary between users "at risk" and "not at risk":

- Words like "hope" and "sorry", although used in each class, have a greater relative frequency in the texts posted by the users labelled as "Supportive" and "Indicator".
- The word "depression", although widely used in all labels, is very frequent in the texts posted by the users labelled as "Ideation", "Behavior", and "Attempt", alongside with words such as "alone", "pain", "kill", "suicide", "hospital", "tired"...

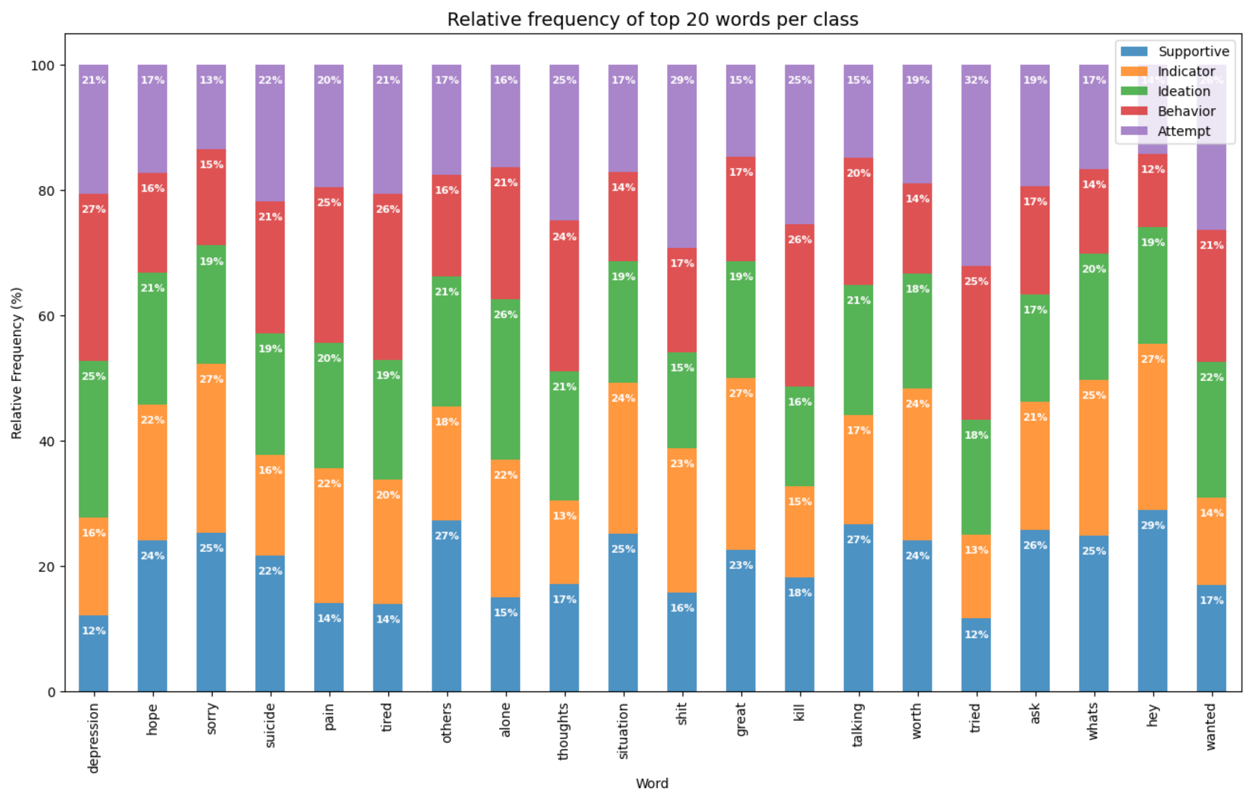

Figure 5: Top 20 words - Relative Frequency Analysis

Top 20 words - Relative Frequency Analysis results:

- Users labelled as **Supportive** and **Indicator** use, more often than the others, words such as "sorry", and "worth"; this suggests that post authors may express sympathy for someone else's problem or life situation.
- The same users also use (more often than the others) non-meaningful words such as "great", "whats", and "hey". Although these words don't carry any meaning per se (or anyway not outside the context), this difference may indicate a tendency of these users to use "lighter" words, and an overall tone of voice more friendly, easy-going, and casual.
- Users labelled as **Ideation, Behavior**, and **Attempt** use, more often than the others, words such as "depression", "thoughts", "tried", and "wanted". This suggests an ongoing

struggle with mental issues like depression, a failed attempt at something (maybe suicide, or maybe a life goal), and a general feeling of regret.

Exploratory Data Analysis results:

Classes are as follows:

- **Attempt:** users have already attempted suicide or actively planning to do it.
- **Behavior:** users struggle with mental issues, like panic attacks, have suicidal thoughts, and at times practice self-harming behavior.
- **Ideation:** users are going through rough paths, they suffer from mental issues like delusional disorder, they wish to die or disappear, and they have little hope things get better.
- **Indicator:** users show some very light signs of distress like guilt, sadness, or loneliness; they also express sympathy towards the other users.
- **Supportive:** users show empathy and support people's feeling, discouraging them from suicidal ideas.

Based on the E.D.A. results and on the class definitions, I'll classify the users as follows:

- **Supportive** and **Indicator**: these users don't seem to suffer from mental issues (or anyway not in a severe way) and they are not at risk of attempting suicide or self-harm. **I classify these users as Not-at-risk (Negative Class)**.
- **Ideation, Behavior, Attempt**: these users struggle with depression, or other mental issues, they may be having suicidal thoughts, dealing with suicidal behavior, have already tried to commit suicide, or actively planning to do it. **These are the users at-risk of attempting suicide or self-harm (Positive Class)**.

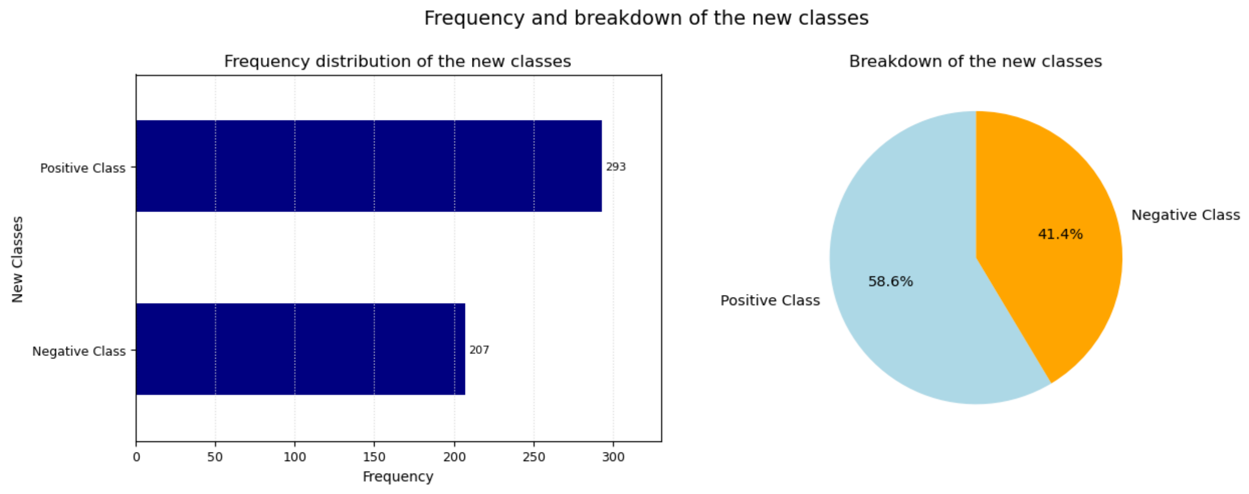

Figure 6: Frequency and breakdown of the new classes

The dataset is still quite imbalanced: almost 60% of the users belong to the positive class (at risk of attempting suicide or self-harm).

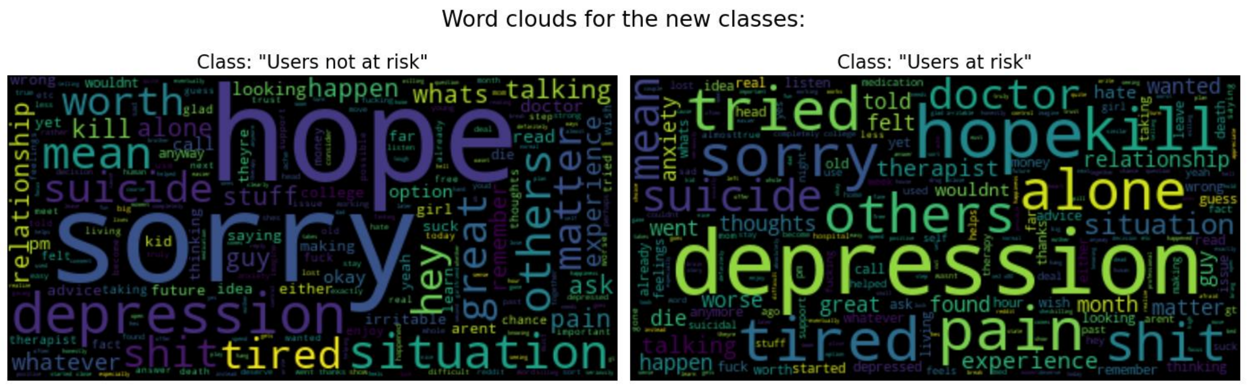

Figure 7: Word Cloud Analysis – New classes

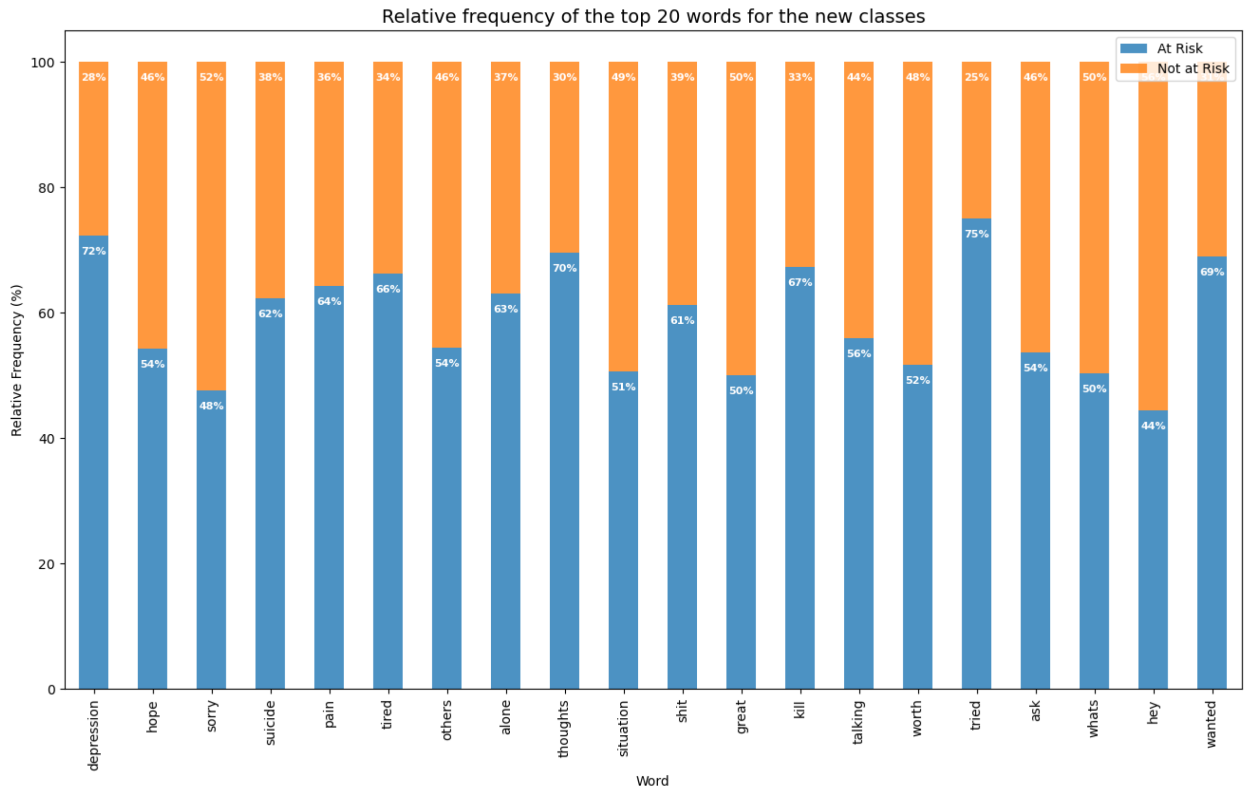

Figure 8: Top 20 words - Relative Frequency Analysis (New classes)

We can see now a clear distinction between the classes, whereas users "At Risk" use, much more than users "Not at Risk", words such as: depression, tried, thoughts, wanted, tired, pain, suicide, kill, alone, shit….

4. Data Preparation

To prepare the data for modeling, I applied the following steps:

1. Drop not-relevant features, leaving only the users' texts (Post) and the binary classes (class).
2. Remove stop words (using the stop words list from the wordcloud library) from the data.

   Text before removing the stop words:

```
'no more ideas i dont agree with live for others kind of advice i think yo
u should live for yourself and your friends and family the world isnt goin
g to be fixed any time soon so stop thinking its all on your shoulders reg
ular exercise and a lack of excessive stress is important to a good life s
o is a decent job work is now stressful yes its never done im on a long br
eak now its tired hot and humid where i now live so i cant really do anyth
ing i cant handle the heat well i want to prepare for my death before i go
back to work its not only that the career enabled me to live a certain lif
estyle and live in a certain place and not have to worry too much about mo
ney and other things why would you like that i dont think there are any ot
her kinds of job i could do in this country it has been 5 years since i lo
st my job i have tried my best the things i lost in my life i believe them
to be extremely fundamental and important things i also lost a life that h
ad little worry and stress now i have a job that gets worse every day does
nt allow me time to exercise is in a boiling hot city that saps me of ener
gy has horrible bitchy colleagues and so on this is the norm i have come t
o realize i really liked living in this country and kind of still like it
other jobs will be like this or worse we live in a world of shitty jobs i
had one of the best jobs in the world and threw it away i cant tolerate an
y job that isnt as good which is to say all of the rest of them i can move
to a different job in the same industry and city in time a less hot and hu
mid place but it wont be as good as the climate in the city where i was an
d even then ill still have lost years of my life people dont understand su
icide and arent going to understand your suicide attempt it will just be l
ooked upon as mental asthenia or a moment of madness or some kind of child
ish gesture you arent going to make people understand they dont even under
stand actual suicides and cant imagine why anyone would want to kill thems
elves i guess this lack of understanding could be a survival mechanism onl
y suicidal people are likely to understand its not going to be fixed the w
orld is fucked there are 7bn people fucking up the planet with our mere pr
esence forget about it and just enjoy your life'

Length of the post before removing the stop words: 2269
```

   Text after removing the stop words:

```
'ideas dont agree live others kind advice think live friends family world
isnt going fixed time soon stop thinking shoulders regular exercise lack e
xcessive stress important good life decent job work now stressful yes neve
r done im long break now tired hot humid now live cant really anything can
t handle heat well want prepare death go back work career enabled live cer
tain lifestyle live certain place worry much money things dont think kinds
job country 5 years lost job tried best things lost life believe extremely
```

```
fundamental important things lost life little worry stress now job gets wo
rse every day doesnt allow time exercise boiling hot city saps energy horr
ible bitchy colleagues norm come realize really liked living country kind
still jobs will worse live world shitty jobs one best jobs world threw awa
y cant tolerate job isnt good say rest move different job industry city ti
me less hot humid place wont good climate city even ill still lost years l
ife people dont understand suicide arent going understand suicide attempt
will looked upon mental asthenia moment madness kind childish gesture aren
t going make people understand dont even understand actual suicides cant i
magine anyone want kill guess lack understanding survival mechanism suicid
al people likely understand going fixed world fucked 7bn people fucking pl
anet mere presence forget enjoy life'

Length of the post after removing the stop words: 1369
```

3. Split the dataset into the training (80% of data: 400 observations) and testing (20% of data: 100 observations) sets.
4. Tokenize the posts. Tokenization means turning each user's text into a list of individual words and then convert the words into integers. The Keras Tokenizer class creates a vocabulary index based on word frequency (where the word with index 1 is the most frequent), and then it takes each word in the text and replaces it with its corresponding integer value from the created word index. To avoid data leaking, I trained the Keras Tokenizer on the training data only, and I used the trained Tokenizer to convert the words into integers for both the training and testing sets. Since there may be words in the testing set that don't appear in the training data (so the Tokenizer won't know how to convert these words) I used a token (OOV) to define those out-of-vocabulary words (Keras Tokenizer will give to the OOV token an index of 1).
   Random text from the test set:

```
'said mean thing earlier apologize shouldnt let jealousy better especially
wanted help people live tennessee im using mostly job boards businesses ar
ound refuse speak anybody regarding employment going door door ive local u
nemployment office multiple times fact response every time tried online jo
b boards try online job boards cant offer help use online job boards thing
less happened went universitys employment services office dont know econom
ic area stacks places despite efforts ive looking work past 2 months found
one job one job lasted two days job im getting unemployment benefits right
now im review past month say nothing fact ive searching work past 4 years
ive looking different job first 3 found job actually liked wound getting l
aid put situation im now basically past experiences lead believe hopeless
situation'
```

   Applied tokenization and then mapped the integers back to words to check integer meaning:

```
'said mean thing earlier apologize shouldnt let jealousy better especially
wanted help people live OOV im using mostly job boards businesses around r
efuse speak anybody regarding employment going door door ive local unemplo
yment office multiple times fact response every time tried online job boar
```

```
ds try online job boards cant offer help use online job boards thing less
happened went universitys employment services office dont know economic ar
ea OOV places despite efforts ive looking work past 2 months found one job
one job lasted two days job im getting unemployment benefits right now im
review past month say nothing fact ive searching work past 4 years ive loo
king different job first 3 found job actually liked wound getting laid put
situation im now basically past experiences lead believe hopeless situatio
n'
```

After mapping the integers back, some words are mapped to the token OOV: these are the words that the Tokenizer didn't learn, because absent from the training vocabulary. **The training data contains 15,754 words. This is the size of the vocabulary.**

5. Pad the sequences. When working with Recurrent Neural Networks, the word sequences must have the same length, so I set 100 as maximum limit of words in a sequence, and “pre” as padding and truncating strategy (that is removing values from the beginning of the sequence, in case sequences were longer than 100 words, or 0-padding from the end of sequence in case sequences were shorter than 100 words.

5. Model Development: Recurrent Neural Networks

In Natural Language Processing problems, just like in everyday language, each word can have different meanings depending on the context, that is depending on the words that have come before, and each word, as more words come in, updates the context.

The Keras **Recurrent Neural Network Class** (RNN) can help us deal with sequential data, that is when points in the data are dependent on the other points in the same data. The Recurrent Neural Networks, in fact, allow previous outputs to be used as inputs, building the so-called Hidden States. The idea is to use the notion of "recurrence":

1. The words that make a sentence (the sequence) will be input into the network one by one.
2. At each step, that is every time a word comes in, the recurrent neural network will output the (hidden) state, that is a summary of everything that happened in the past, leading up to that point (in other words the context, updated to that point).
3. This hidden state is then fed to the next layer of the network, alongside with the following word coming in.
4. The network will combine together (through matrix operations) the previous state and the new input, and pass everything combined through an activation function, to output the new state updated with the new word, to be fed to the next layer.
5. When the last word of the sequence comes in, the network will actually produce the final output.

This is how Recurrent Neural Network enable previous info in the sequence (in our case here: the previous words in the sentence) to change the network final output as each word gets fed to the model.

In Natural Language Processing (NLP), a word embedding is a representation of a word in the form of a vector (of a given dimension) that encodes the meaning of that word; words that have similar meaning should also be closer in the vector space, that is they have a similar representation. When working on an NLP problem, there are 2 options:

- Train your own word embeddings. In this case, the embeddings are learnt using the dataset for the specific problem that someone is trying to solve.
- Apply **Transfer Learning** concept, that is using pre-trained word embeddings: embeddings learnt on large datasets, saved, and then used for solving other tasks.

For this project, I built and compared performance from 3 models, using the RNN class:

1. Learning the Word Embeddings from scratch, using the training data of my dataset.
2. Using the pre-trained GloVe Word Embeddings .
3. Using the pre-trained Google News Word2Vec Word Embeddings.

The purpose of building such models is to check if different word embeddings lead to better performances. Therefore, I used the same network architecture and hyperparameters, so that the only variable affecting the models' performances is the word embeddings.

The Neural Network Architecture is as follows:

- First layer: the Embedding Layer, where each integer, representing a word, will be encoded in the form of a 300-dimensional vector.
- Second Layer: simple Recurrent Neural Network. The number of units is set to 150. This means that each layer (for both the Kernel, for each input-word, and the Recurrent part of the Network, for the hidden state) will have 150 neurons. I'll keep the other hyperparameters as the default ones, for both the kernel and recurrent initialization as well as for the activation function (which is the Hyperbolic Tangent Function, so that the output values will always be between -1 and 1).
- After the Recurrent Neural Network layer, I add only the final output layer: a fully connected layer, with only 1 node and Sigmoid as activation function, to output the final predictions: the user's probability to belong to the Positive Class.
- Weight Optimizer: Adaptive Moment Estimation (Adam); learning rate equal to 0.001.
- Since we're dealing with a binary classification problem, the cost function is the Logarithmic Loss (or Binary Cross Entropy).

5.1. Learn the word embeddings from scratch

Model architecture and number of parameters:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 300)         4726500

 simple_rnn (SimpleRNN)      (None, 150)               67650

 dense (Dense)               (None, 1)                 151

=================================================================
Total params: 4,794,301
Trainable params: 4,794,301
Non-trainable params: 0
_________________________________________________________________
```

The network has a huge number of parameters (or weights) to learn: 4,794,301. Let's go through this number:

- The Embedding layer, since it'll map each word/integer to a 300-dimensional space, has to learn 300 parameters for each word in the training vocabulary (15,754 words) + 1 extra vector for the padding; this means that the embedding layer will train and learn 4,726,500 parameters (15,755 times 300).
- The Simple Recurrent Neural Network layer have to learn 2 weight matrices, 1 for the input words (the "Kernel"), and 1 for the states (the "Recurrent"). The Kernel must learn a number of parameters equal to the number of the input dimension + 1 (because of the bias) times the number of state nodes, that is 150. Since the input vector is 300-dimensional, the Kernel must learn (300+1) * 150 = 45,150 parameters. The Recurrent must learn a number of parameters equal to the square of the state dimension, that is 150 * 150 = 22,500 parameters. In total the Simple Recurrent Neural Network will train and learn 67,650 parameters. The weight matrix dimensions are designed this way so that the states and the input words, after the transformation with the weight matrices, will have the same dimension and can be combined together.
- The final dense layer must learn a number of parameters equal to the dimension of the input (which is 150, from the recurrent network) + 1 (because of the bias) times the number of nodes in this final layer (which is 1, since the number of classes is 2); so, the final layer will train and learn 151 parameters.

To introduce some randomness into the model, and decrease the variance (that is the tendency of the model predictions to be highly-sensitive to changes in the input data, and thus overfitting the training data), I didn't use the whole training set to determine the gradient vector and update the weights; I used the **Mini Batch Gradient Descent** technique instead, with a **batch**

size of 16 observations. Therefore, the model has to go through **25 iterations** before using the whole training set to update the weights and complete 1 epoch.

I trained the model for **25 epochs**, plotting the loss and the accuracy for both the training and validation sets to track the model's performance over the epochs.

I also applied the **data shuffling** technique after each epoch, to avoid the batches to be the same, and in the same order, in each epoch. This should also help increase model's randomness and contrast the overfitting problem.

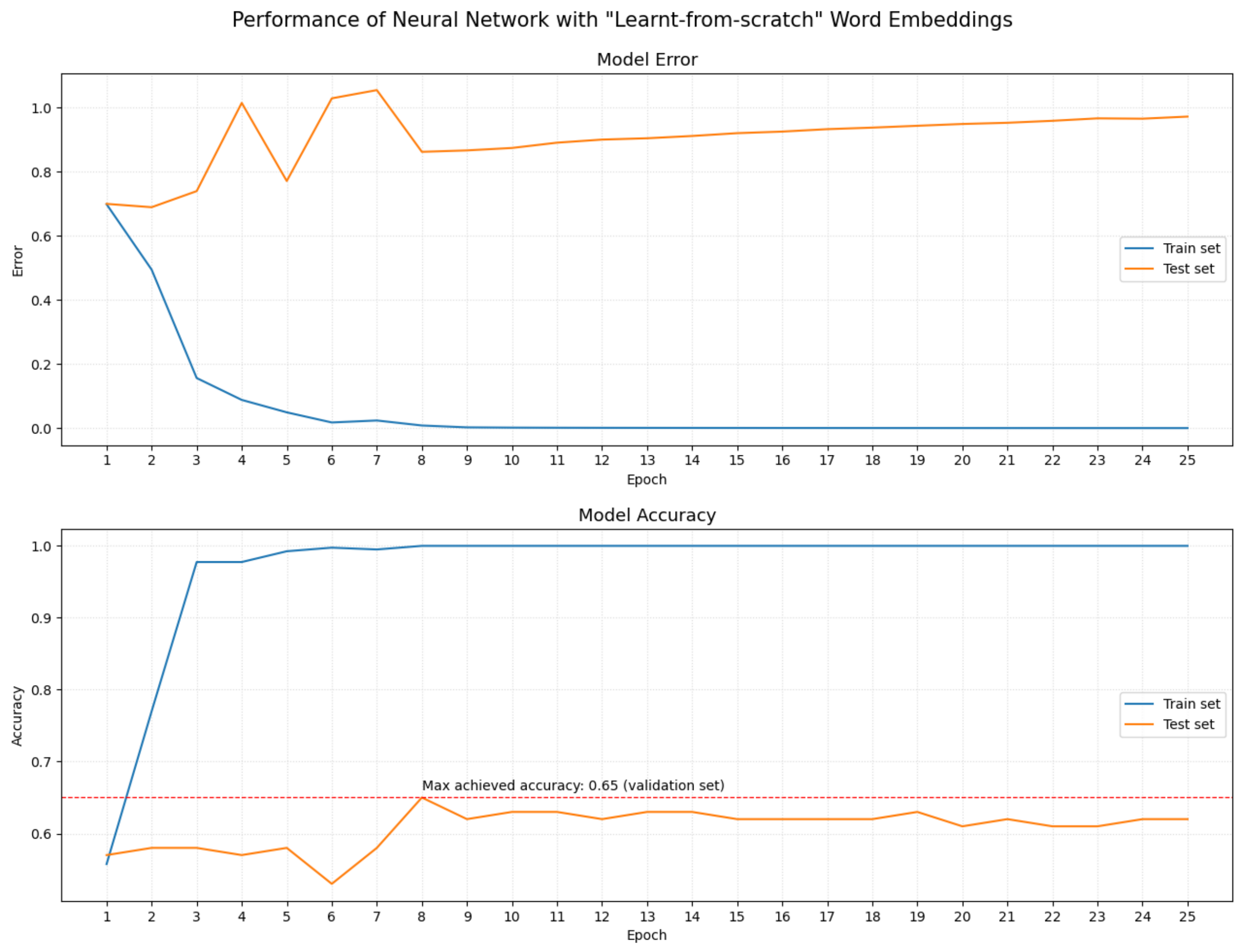

Figure 9: Loss and accuracy of the first model

Some remarks:

- The model managed to perfectly fit the training data after 8 epochs, achieving an accuracy of 100% and a loss next to 0.

- The validation loss fluctuated for the first 8 epochs, after which it kept increasing in each epoch. **This is a clear sign of overfitting.**
- The accuracy, on the validation set, reached the greatest value (65%) at the 8th epoch, after which it stays stable around 61% - 63%. **There's no further gain in keeping training the model after 8 epochs.**

5.2. Using Pre-trained Word Embeddings: GloVe (Global Vectors for Word Representation)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words, developed as an open-source project at Stanford University. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe uses several dimensions for the word vectors; the embeddings I used contains **300-dimensional vectors for 400,000 words**.

For more information on GloVe, check this link: **GloVe: Global Vectors for Word Representation**.

After loading the GloVe vectors, I matched the words with the training vocabulary (15,754 words), assigning vectors to 14,072 words; the remaining 1,682 words, which weren't found in the word vectors from GloVe, were assigned to a unique vector of all zeros.

The model architecture is the same as the first model:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 300)         4726500

 simple_rnn (SimpleRNN)      (None, 150)               67650

 dense (Dense)               (None, 1)                 151

=================================================================
Total params: 4,794,301
Trainable params: 67,801
Non-trainable params: 4,726,500
_________________________________________________________________
```

The total number of parameters is exactly the same of the first model (4,794,301), however **4,726,500 parameters are non-trainable.** Since we are not learning our own word embeddings, the Embedding layer doesn't have to learn how to represent the words. That's why the 4,726,500 parameters of the embedding layers are non-trainable: they have already been trained. Therefore, the model has to train and learn "only" 67,801 parameters (67,650 for the Recurrent Neural Network section, and 151 for the final dense layer).

To be consistent and compare model's performance based on 1 variable only (the word embeddings), I used the same **Mini Batch Gradient Descent** strategy of the first model:

- batch size: 16 observations
- iterations: 25
- epochs: 25
- data shuffle: yes

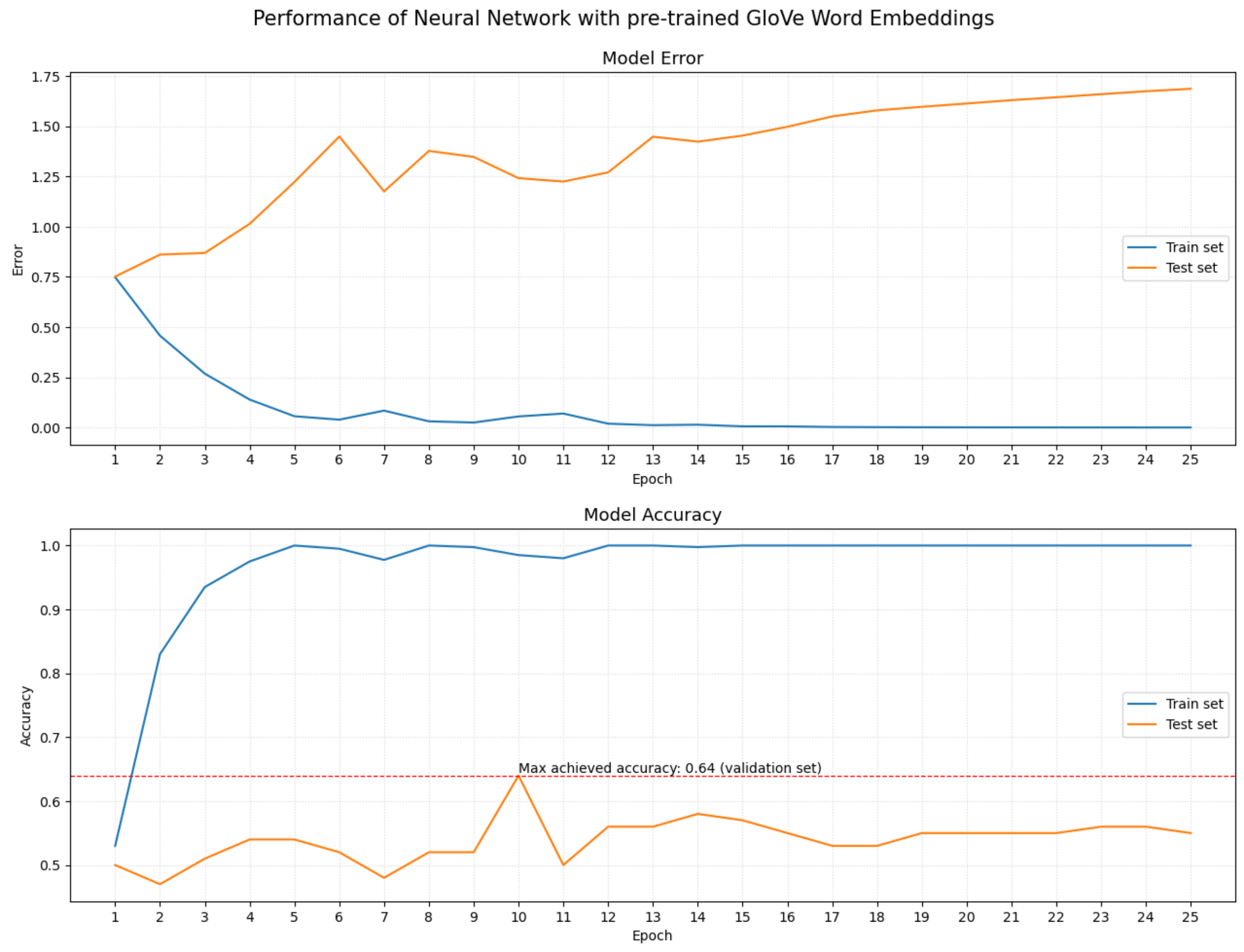

Figure 10: Loss and accuracy of the second model

Since pre-trained words embeddings are learned on very large datasets (in the case of Glove I imported 400,000 word vectors), this sometimes can help have a better representation of the words, which is not so heavily affected by the dataset at hand. However, in this case, when using GloVe pre-trained Word Embeddings, the model performs worse than when learning the word embeddings from scratch: **64% accuracy with GloVe word vectors vs 65% accuracy by training our own word embeddings.**

This means that the population (the words) the GloVe algorithm was trained on is too different from our own population (the words contained in the training data).

5.3. Using Pre-trained Word Embeddings: Google News Word2Vec model

Google News Word2Vec model has been trained on roughly 100 billion words from a Google News dataset; it includes **300-dimensional word vectors for 3 million words**.

For more information on Google News Word2Vec model, check this link: **word2vec**.

After loading the Word2Vec vectors, I matched the words with the training vocabulary (15,754 words), assigning vectors to 13,768 words; the remaining 1,986 words, which weren't found in the word vectors from Word2Vec, were assigned to a unique vector of all zeros.

Model architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 300)         4726500

 simple_rnn (SimpleRNN)      (None, 150)               67650

 dense (Dense)               (None, 1)                 151

=================================================================
Total params: 4,794,301
Trainable params: 67,801
Non-trainable params: 4,726,500
_________________________________________________________________
```

Since the model architecture is the same, the number of trainable and non-trainable parameters is exactly the same of the previous model.

I trained this 3rd model using the same **Mini Batch Gradient Descent** strategy:

- batch size: 16 observations
- iterations: 25
- epochs: 25
- data shuffle: yes

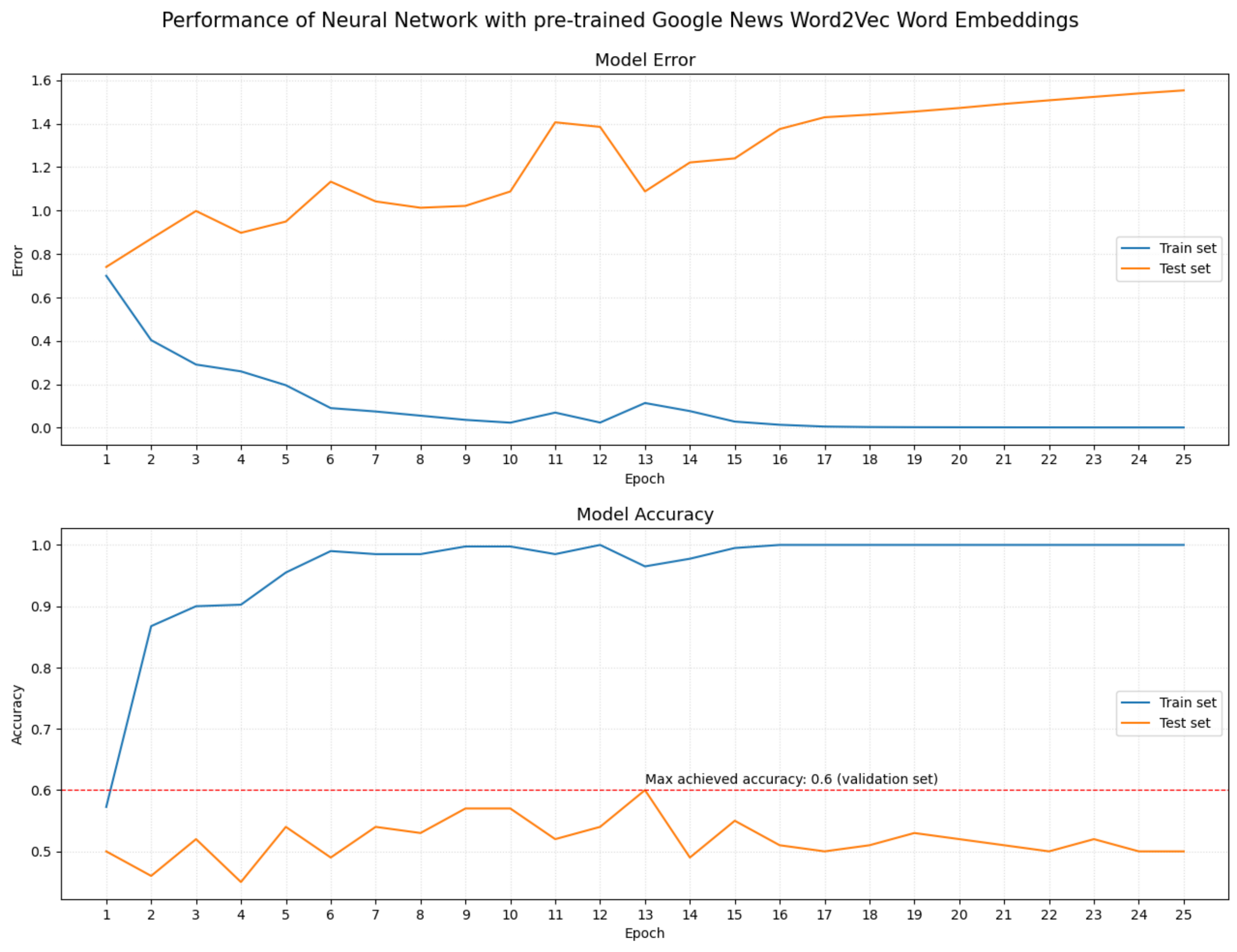

Figure 11: Loss and accuracy of the third model

Using Google News Word2Vec Word Embeddings produced the worst model in terms of out-of-sample accuracy: **60% accuracy with Google News Word2Vec vs 65% with our own word embeddings.**

Again, it appears that the population the Word2Vec model was trained on is too different from our own population.

5.4. Recurrent Neural Network with Regularization Techniques

Learning the word embeddings from scratch proved to be the best option in terms of model's accuracy, calculated on unseen data.

All trained models, however, have a common issue: the overfitting. All networks, regardless of the used word embeddings, fits the training data very well, achieving 100% training accuracy after few epochs; however, the accuracy calculated on the validation set, seems unable to exceed the 65% threshold:

- Learn the word embeddings from scratch: 65% accuracy
- Pre-trained GloVe Word Embeddings: 64% accuracy
- Pre-trained Google News Word2Vec Word Embeddings: 60% accuracy

I tried to increase the out-of-sample accuracy with the last model: learning the word embeddings from scratch and using some **regularization techniques**, in order to decrease the variance of the model.

Regularization techniques should lead to a "simpler" model, where the predictions are less sensitive to changes in the input data, and so, hopefully, increase the ability of the model to "generalize" better, that is to better capture the underlying relationships between the target and the predictor, without following too much the noise present in the training data; this will, almost certainly, lead to a lower training accuracy, but the hope is that the validation accuracy will increase.

I used the following regularization techniques:

- **L2 Regularization Penalty** for the Embedding and the Recurrent Neural Network layers. The sum of the squares of the parameters (for the Recurrent Neural Network, the penalty is considered for both the bias, the Kernel, and the Recurrent weights) will be added to the model loss, thus forcing the model to keep the parameter values smaller (and so reducing the sensitivity to the input data).
- **Keras constraint Max Norm** for all model layers. The Max Norm constraint forces the model weights to have a magnitude less than or equal to a certain value, which will be set to 1 for all layers.

Model architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 300)         4726500

 simple_rnn (SimpleRNN)      (None, 150)               67650

 dense (Dense)               (None, 1)                 151

=================================================================
Total params: 4,794,301
Trainable params: 4,794,301
Non-trainable params: 0
_________________________________________________________________
```

As expected, the number of trainable parameters is the same as the first model.

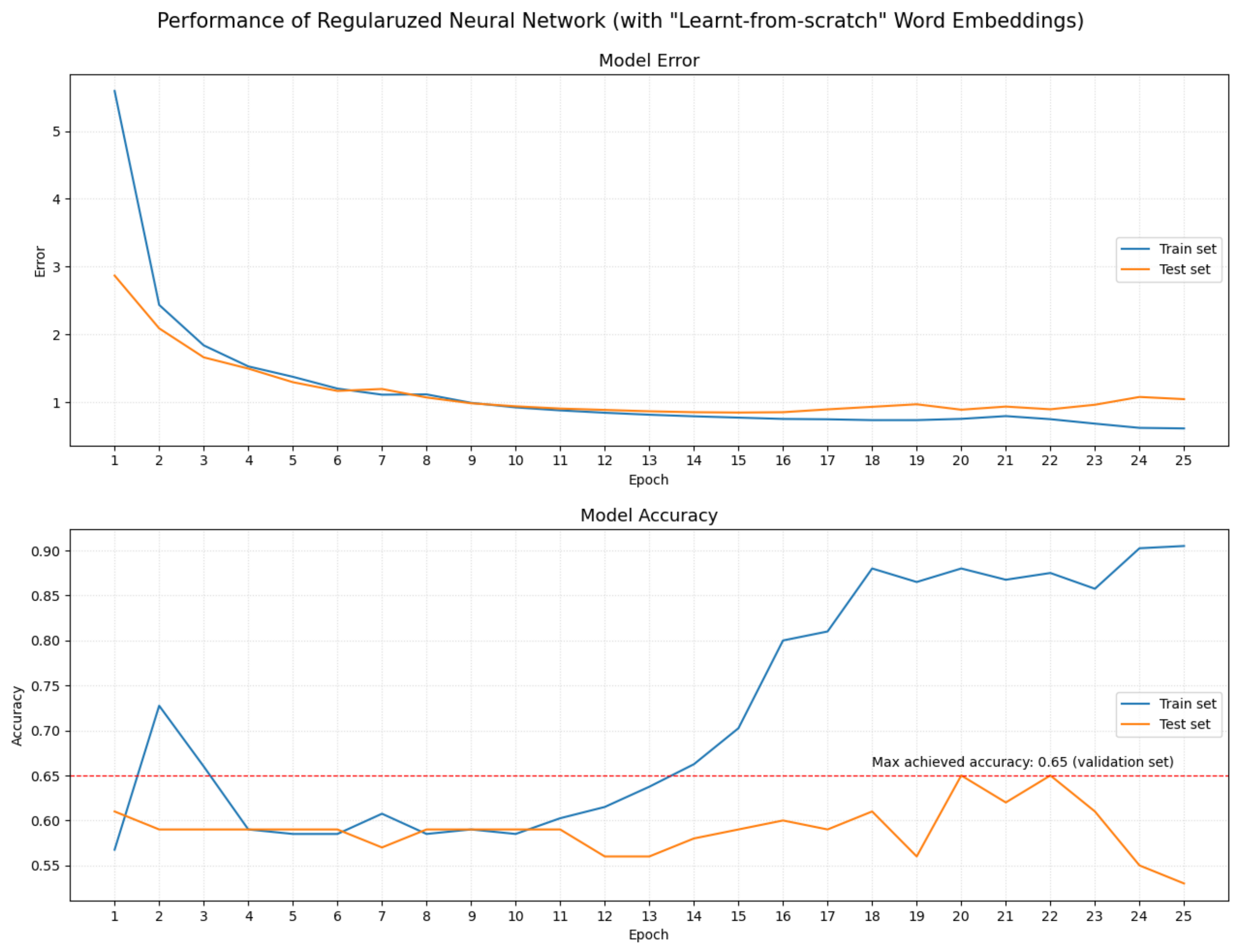

Figure 12: Loss and accuracy of the fourth model

Some remarks:

- The model achieved, again, a max accuracy of 65%, after 20 epochs. Even regularized model couldn't exceed that threshold, and it took longer for the regularized model to achieve that maximum value.
- As expected, the training accuracy is less than when not using regularization techniques. On the 20th epoch, that is when the model achieved the best result, the training accuracy is equal to 88%.

6. Results.

6.1. Non-regularized model

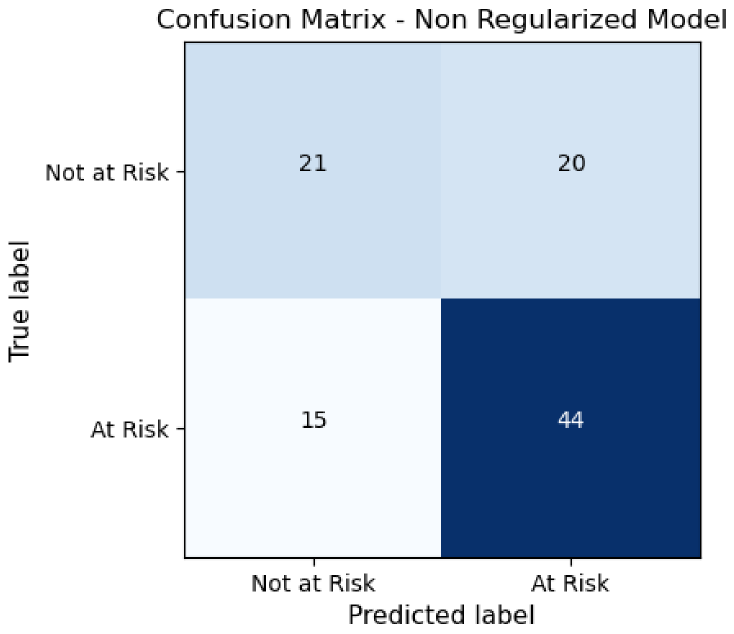

Figure 13: Confusion Matrix – Non-Regularized Model

```
Classification Report - Non-Regularized Model
              precision    recall  f1-score   support

 Not at Risk       0.58      0.51      0.55        41
     At Risk       0.69      0.75      0.72        59

    accuracy                           0.65       100
   macro avg       0.64      0.63      0.63       100
weighted avg       0.64      0.65      0.65       100
```

Remarks:

- The model did a decent job at correctly predicting the users belonging to the positive class, with a Sensitivity (the recall on the positive class) of 75%: 3 out of 4 users "At Risk" of self-harm have been correctly classified.
- The Specificity (the recall on the negative class) is not very high though: 51%; only 1 out of 2 "Not-at-Risk" users were correctly classified.
- Overall, the model struggles to identify the "Not-at-Risk" users, and achieved an overall F1 Score (the weighted average of each class F1 Score (which is the Precision and Recall harmonic average)) of 65%.

6.2. Regularized model

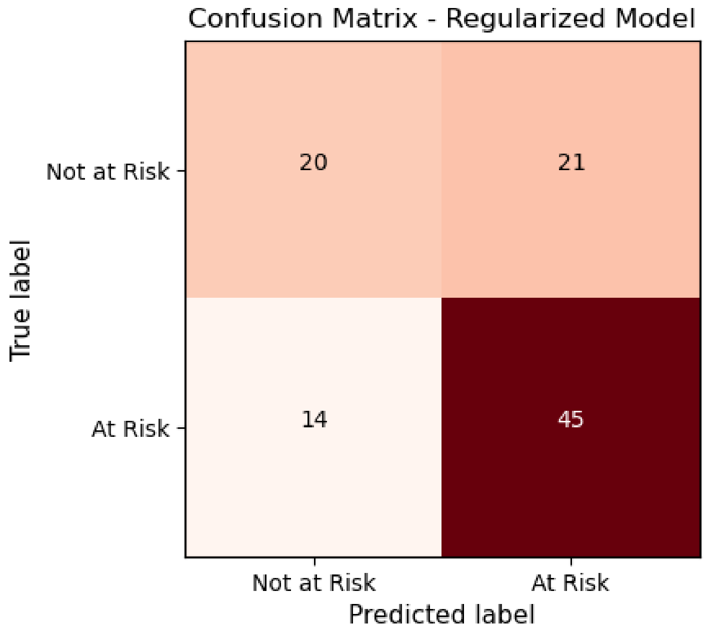

Figure 14: Confusion Matrix – Regularized Model

```
Classification Report - Regularized Model
              precision    recall  f1-score   support

 Not at Risk       0.59      0.49      0.53        41
     At Risk       0.68      0.76      0.72        59

    accuracy                           0.65       100
   macro avg       0.64      0.63      0.63       100
weighted avg       0.64      0.65      0.64       100
```

Remarks:

- The performance of the regularized model is very similar to the non-regularized one, with the network still struggling to identify the "Not-at-Risk" users.
- The regularized model predicted 2 more users as "At Risk", compared to the non-regularized model, 1 correctly and 1 not correctly; this led to a higher Sensitivity (the recall on the positive class or the True Positive Rate), but a lower Specificity (the recall on the negative class), so the False Positive Rate also increased (thus reducing the precision of the positive class predictions).
- Overall, the regularized model achieved an overall F1 Score of 64%, slightly less than the non-regularized model.

6.3. Moving the discrimination threshold

The performance of the 2 models is very similar, and they both struggle with a high False Positive Rate ("not-at-risk" users predicted as "at risk") around 50%. However, the networks output the probabilities of the observations to belong to the positive class, and not directly the class values. I inferred the class value by applying the "default" probability threshold of 50%, that is if an example has a predicted probability of belonging to the positive class equal or greater than 50%, it'll be classified as "at-risk"; if less than 50%, then "not at risk". I try now to calibrate this probability threshold, to either decrease the number of false positive or the number of false negatives. To look for the best probability threshold for both the regularized and non-regularized model, I used the Receiver Operating Characteristic (ROC) as well as the Precision-Recall curves:

- **Receiver Operating Characteristic (ROC) curve**: a graphical plot that illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied. The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.
- **Precision-Recall curve:** a graphical plot that shows the tradeoff between precision and recall for different probability thresholds. A high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

To find out the best decision threshold, I calculated the best trade-off between:

- The True Positive Rate and the False Positive Rate, using as metric the **Youden's J statistic** (which is the difference between the True Positive Rate and the False Positive Rate).
- The Precision and Recall of the positive class, using as metric the **F1 Score** (the harmonic mean between Precision and Recall).

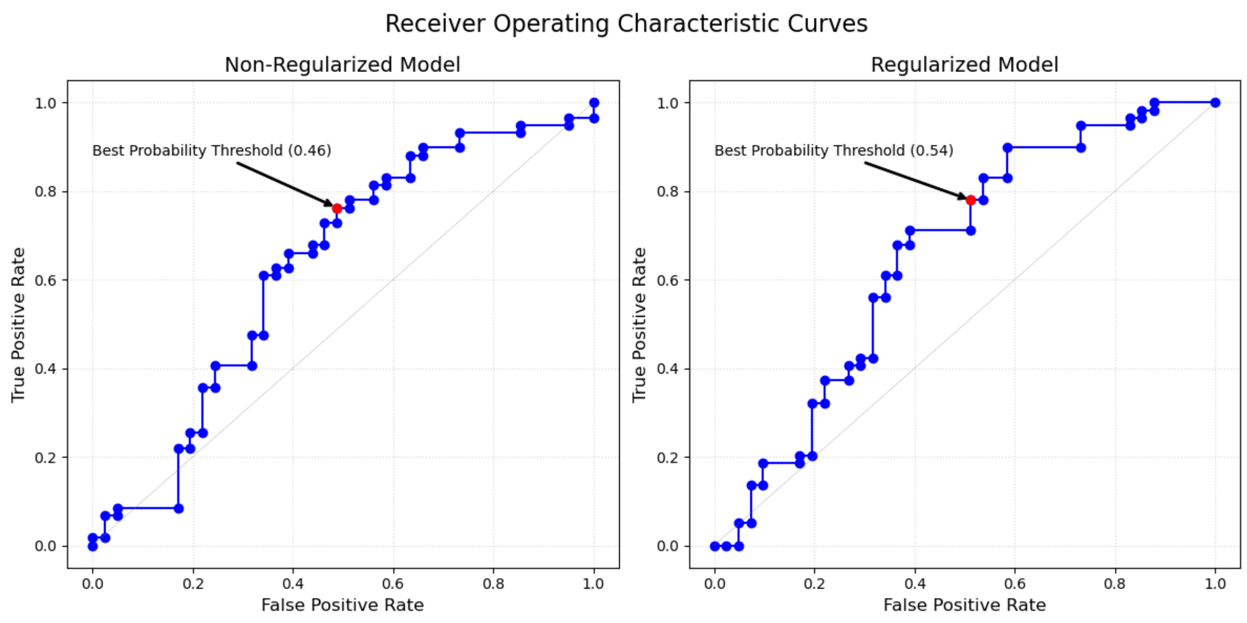

Figure 15: ROC Curve for both Regularized and Non-Regularized Models

```
ROC Area Under the Curve - Non-Regularized Model:  0.63
ROC Area Under the Curve - Regularized Model: 0.65
```

Both models are above the line of random guessing for almost every threshold, with an overall Area Under the Curve slightly greater for the regularized model (0.65 vs 0.63). The non-regularized model achieved the best Youden's J statistic (0.27) at the 46% discrimination threshold. The regularized model achieved the best Youden's J statistic (0.32) at the 54% discrimination threshold.

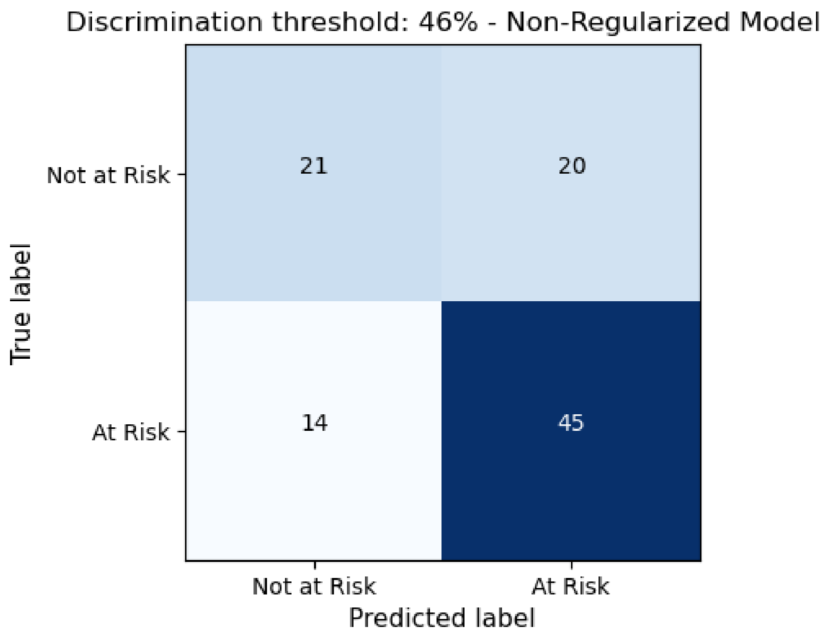

Figure 16: Confusion Matrix – Non-Regularized Model with probability threshold at 46%

```
Discrimination Threshold: 46%. Non-Regularized Model Classification Report
               precision    recall  f1-score   support

 Not at Risk       0.60      0.51      0.55        41
     At Risk       0.69      0.76      0.73        59

    accuracy                           0.66       100
   macro avg       0.65      0.64      0.64       100
weighted avg       0.65      0.66      0.65       100
```

Applying a discrimination threshold equal to 46% (as in to make it slightly easier for the model to predict users belonging to the positive class) on the non-regularized model produced better results (compared to the 50% threshold):

- The model sensitivity, that is the True Positive Rate, increased from 75% to 76%, without negatively affecting any other metric: the model managed to correctly predict one more user classified as "At Risk", keeping the same specificity (or True Negative Rate).
- The model precision, on the Negative Class, also increased (from 58% to 60%) as well as the F1 Score (Macro Average: from 63% to 64%) and the overall accuracy, from 65% to 66%.

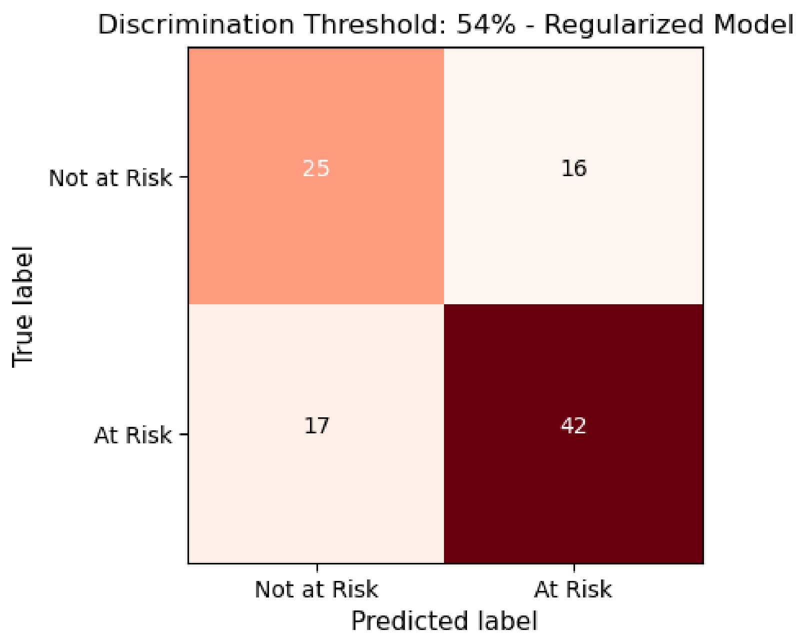

Figure 17: Confusion Matrix – Regularized Model with probability threshold at 54%

```
Discrimination Threshold: 54% - Regularized Model Classification Report
              precision    recall  f1-score   support

 Not at Risk       0.60      0.61      0.60        41
     At Risk       0.72      0.71      0.72        59

    accuracy                           0.67       100
   macro avg       0.66      0.66      0.66       100
weighted avg       0.67      0.67      0.67       100
```

Applying a discrimination threshold equal to 54% (as in to make it slightly more difficult for the model to predict users belonging to the positive class) on the Regularized model produced a more accurate algorithm, with one drawback:

- The model specificity increased from 49% to 61%: the model is now able to predict way more accurately users that are "Not-At-Risk".
- Overall, the model precision increased for both classes, from 59% to 60% for the Negative Class, and from 68% to 72% for the Positive Class leading to a better F1 Score (weighted average: from 64% to 67%) and Accuracy: from 65% to 67%.
- However, the model sensitivity decreased from 76% to 71%; the model has indeed become more accurate, but it has now more difficulty predicting the users "at-risk", which is the main purpose of the project, since these users are the ones that need help.

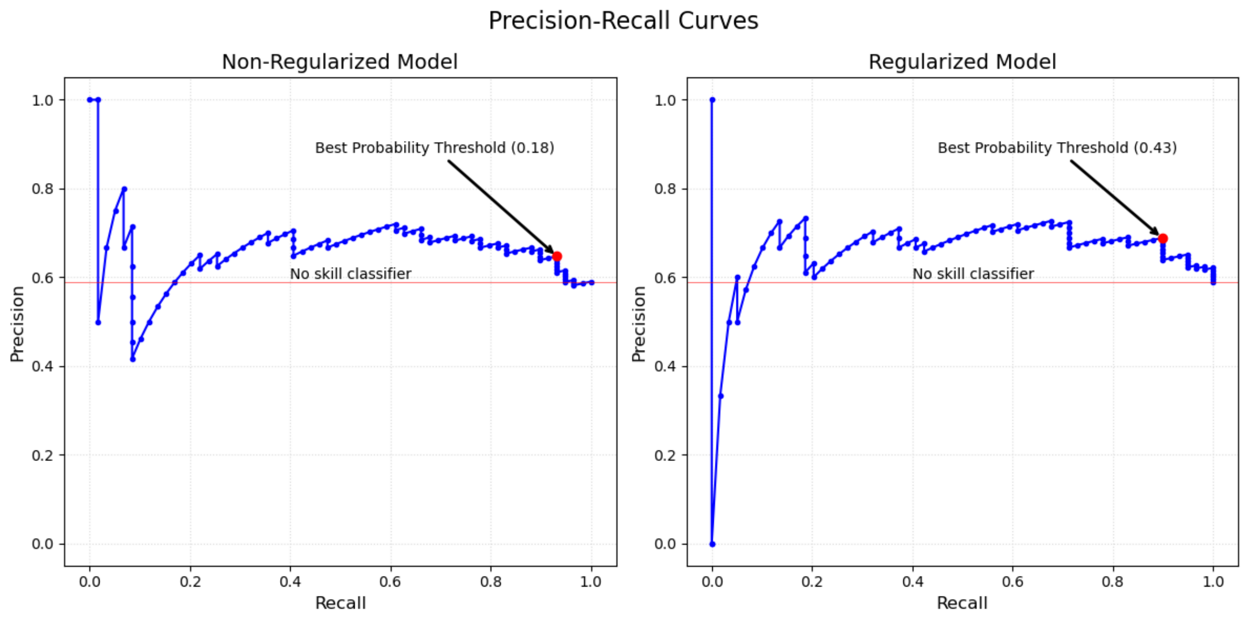

Figure 18: Precision-Recall Curve for both Regularized and Non-Regularized Models

Both models are above the line of "No-Skill Classifier" (that is the precision we would get if the model predicted all observations as members of the positive class) for the vast majority of the threshold, with an overall Area Under the Curve of 66% for both models. The non-regularized model achieved the best F1 Score (76%) at the 18% discrimination threshold. The regularized model achieved the best F1 Score (78%) at the 43% discrimination threshold.

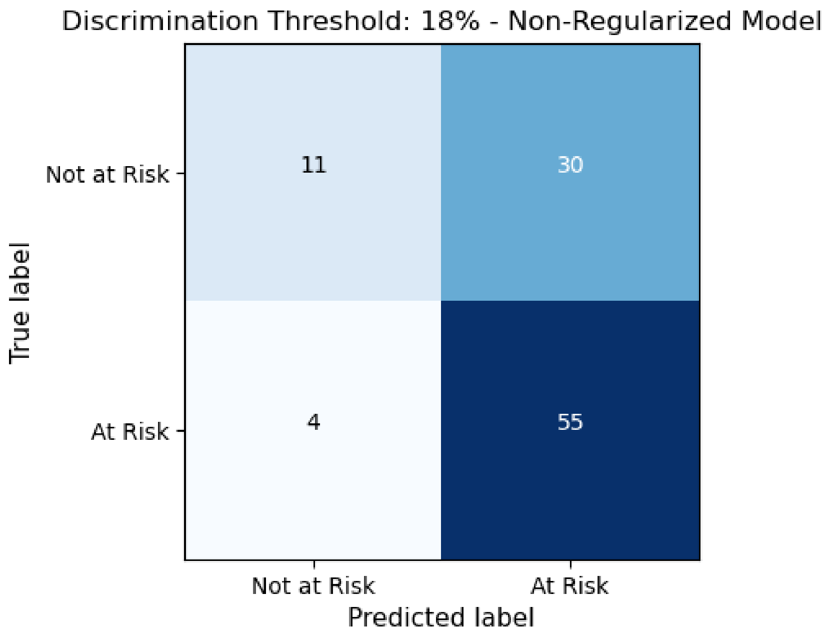

Figure 19: Confusion Matrix – Non-Regularized Model with probability threshold at 18%

```
Discrimination Threshold: 18%. Non-Regularized Model Classification Report
              precision    recall  f1-score   support

 Not at Risk       0.73      0.27      0.39        41
     At Risk       0.65      0.93      0.76        59

    accuracy                           0.66       100
   macro avg       0.69      0.60      0.58       100
weighted avg       0.68      0.66      0.61       100
```

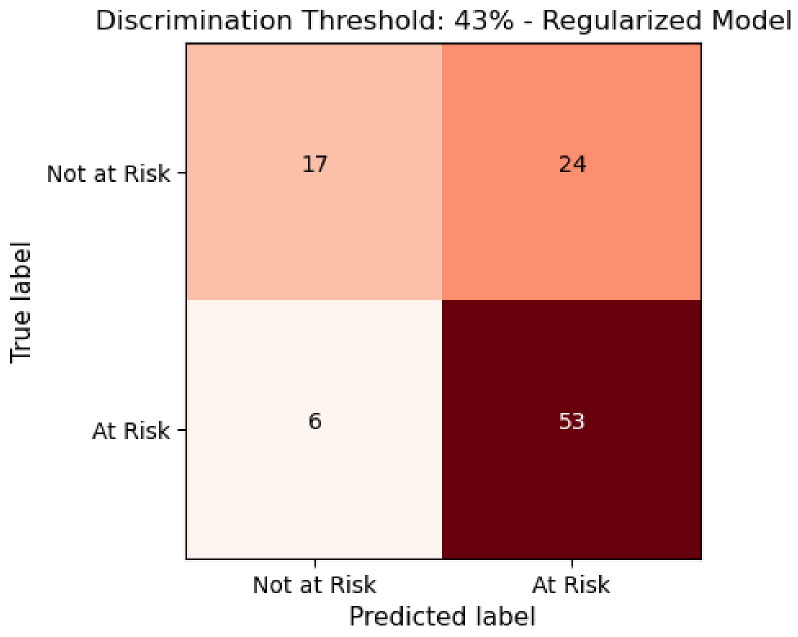

Figure 20: Confusion Matrix – Regularized Model with probability threshold at 43%

```
Discrimination Threshold: 43% - Regularized Model Classification Report
              precision    recall  f1-score   support

 Not at Risk       0.74      0.41      0.53        41
     At Risk       0.69      0.90      0.78        59

    accuracy                           0.70       100
   macro avg       0.71      0.66      0.66       100
weighted avg       0.71      0.70      0.68       100
```

Results:

- Setting a threshold as low as 18% (that is making it very easy for the model to predict the positive class), not surprisingly, increased massively the model sensitivity (up to 93%), but the specificity, consequently, fell down to a very low 27%. The precision of the positive class predictions also fell to 65%. Overall, the model performed very well in predicting almost every true positive (the users "At-Risk") correctly, but the model exactness suffered, leading to the lowest F1 Score so far: 61%, calculated as the weighted average of each class F1 Score, and 58%, as the macro average of each class F1 Score.

- **Setting a decision threshold to 43% produced the best model** in terms of Accuracy (70%), average Precision (71%) as well as in terms of the trade-off between model's sensitivity and specificity, with the greatest F1 Score: 68% (calculated as the weighted average of each class F1 Score).

7. Discussion

After several tries, **I managed to increase the model accuracy from 65% to 70%,** measured on the out-of-sample set. The final model produced the best results in terms of accuracy (7 out of 10 predictions were correct) and trade-off between correctly predicting as many "at-risk" users as possible, without excessively compromising the model precision:

- The neural network achieved a sensitivity of 90%; this means that, if this model was to be implemented, **9 out of 10 users at risk of self-harm could be correctly predicted, and the needed support actions could be triggered.**
- The model precision on predicting users at risk is almost 70%: 3 out of 10 users predicted as at risk of self-harm are actually not at risk.

The model needs to improve on recognizing the users that are actually not at risk of self-harm. This will lead to a better precision and accuracy, reducing the numbers of false positives (users not at risk of self-harm classified as at risk), thus enabling more targeted focus on those cases that are actually in need of help. At this purpose, I reckon that **increasing the model complexity won't achieve any better results**, considering that the model already perfectly fits the training data. The model should actually learn how to "generalize" better, that is learning the patterns in the data, without "being distracted" by the noise that is inevitably present in the training data. For this other regularization techniques (like dropout, for instance) could be tried and tested; however, the best approach here would be collecting way more data to train the model on, for the network to better understand the hidden patterns within the data itself.

8. Conclusion

8.1. Project summary

In this project I trained 4 neural networks in order to correctly predict whether online users, amongst redditors who have discussed topics about suicide and mental health issues, are at risk of self-harm. The scope of the project is to enable Support Services and Help Centers to provide targeted suicide intervention in a timely-fashioned and sustained manner.

Project stages:

- The dataset has been cleaned and analyzed; neither duplicates nor missing values were found; users' texts were cleaned and a standard format applied (deleted punctuation, lowered case all words, removed leading and trailing spaces…). I've performed Word Cloud and Word-Relative-Frequency analysis to understand better the class meaning, and separate users at risk of self-harm from the ones not at risk.
- I prepared the data for modeling by removing the stop words, splitting the data into training (80% of data) and test (20%) sets, tokenizing the texts, and padding the sequences.
- I trained 3 Recurrent Neural Networks:
    - Learning the Word Embeddings from scratch.
    - Using the pre-trained GloVe Word Embeddings
    - Using the pre-trained Google News Word2Vec Word Embeddings
- Learning the Word Embeddings from scratch proved to be the best performing approach. I then trained another neural net, with the same architecture, applying regularization techniques (L2 penalty and Max Norm constraint) to contrast the overfitting problem; eventually, I tuned the discrimination threshold looking for the best trade-off between the model recall and precision.

8.2. Outcome of the analysis

The model that produced the best results has the below architecture:

- 1 Embedding layer to encode words in 300-dimensional vectors, **learning the embeddings from scratch.**
- A (simple) Recurrent Neural Network, with a 150-dimensional state and the hyperbolic tangent function as the activation function.
- A final output layer, with one node and Sigmoid as the activation function to output a binary classification.
- L2 Regularization Penalty applied to the Embedding layer and the Recurrent Neural Network section.
- The Max Norm constraint applied to all model layers.
- A discrimination threshold set to 43%.

The above model correctly classified 90% of the users at risk of self-harm and achieved an overall accuracy of 70%.

8.3. Potential developments

As mentioned in the discussion, **way more data is needed** in order for the model to capture the underlying patterns in the data and increase the predictions' accuracy. Another limitation of the chosen neural network is that it's **dependent on the train/test split** at hand.
The below is a suggested plan for revisiting the model in order to achieve better results:

1. To avoid the train/test split dependency, the model could be trained with **cross-validation technique**, using the GridSearchCV class from the scikit-learn library.
2. When it comes to deep learning models, there are lots of hyperparameters and different architectures to set: layers, activation functions, number of neurons, regularization techniques…; all these hyperparameters can affect the model's results. Again, the GridSearchCV class could be used for **hyperparameters optimization**, that is to search for those hyperparameters that yield the best model performance. This is a great article on how to Grid Search Hyperparameters for Deep Learning Models in Python with Keras.
3. I implemented some NLP techniques, like tokenization, word stop removals, and word embeddings. However, other NLP techniques like **stemming** (a technique used to extract the base form of the words by removing affixes from them) **and lemmatization** (the process of grouping together the inflected forms of a word so they can be analyzed as a single item) could be implemented in the pre-processing stage, and tested to check if more data transformation would enable the model to achieve better results.
4. Different types of recurrent networks, like **Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM) networks** could be added to the model, in place of the Simple Recurrent Neural Network, and the difference in the model's performance should be measured for model's comparison.

9. Appendix

- Link to the first notebook: Data for Good: predicting suicidal behavior likelihood using Deep Learning (Part 1)
- Link to the second notebook: Data for Good: predicting suicidal behavior likelihood using Deep Learning (Part 2)
- Link to the third notebook: Data for Good: predicting suicidal behavior likelihood using Deep Learning (Part 3)
- Link to the fourth notebook: Data for Good: predicting suicidal behavior likelihood using Deep Learning (Part 4)