UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

**FACULTATEA DE INFORMATICĂ**

**A picture containing symbol, screenshot, line

Description automatically generated**

LUCRARE DE LICENȚĂ

**Detectarea semnelor de depresie din texte**

Propusă de

**Mihailescu Alessandro Ionuț**

**Sesiunea:** Iulie, 2023

Coordonator științific

**Conf. Dr. Răschip Mădălina**

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

**FACULTATEA DE INFORMATICĂ**

**Detectarea semnelor de depresie din texte**

**Mihailescu Alessandro Ionuț**

**Sesiunea:** Iulie, 2023

Coordonator științific

**Conf. Dr. Răschip Mădălina**

**Avizat,**

**Îndrumător Lucrare de Licenţă**

Conf. Dr. Răschip Mădălina

Data Semnătura

**DECLARAȚIE privind autenticitatea conţinutului lucrării de licenţă/diplomă/disertaţie/absolvire**

Subsemnatul(a) **Mihailescu Alessandro Ionuț**

domiciliat(ă) în **Horodnic de Sus**, **jud.Suceava, strada Violetelor nr. 6** născut(ă) la data de **3 Ianuarie 2002**, identificat prin CNP **5020103333498**, absolvent(ă) al(a) **Universităţii „Alexandru Ioan Cuza” din Iaşi, Facultatea de Informatică** specializarea **informatica**, promoţia **2022**

declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă/diplomă/disertaţie/absolvire cu titlul:

**Detectarea semnelor de depresie din texte** elaborată sub îndrumarea dnei **Conf. Dr. Răschip Mădălina**, este autentică, îmi aparține şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă/ diplomă/ disertaţie/absolvire să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consimțind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice în vederea facilitării falsificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diplomă sau de disertaţie şi în acest sens declar pe proprie răspundere că lucrarea de faţă nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

Data

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Detectarea semnelor de depresie din texte*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Mihailescu Alessandro Ionuț*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

**Cuprins**

[**Introduction**](#Introduction)

[**Contributions**](#Contributions)

1. [**Presentation of the problem**](#PresentationOfTheProblem)
   1. [The Data Set](#TheDataSet)
   2. [Metrics used](#MetricsUsed)
      1. [Precision](#Precision)
      2. [Sensitivity](#Sensitivity)
      3. [F1 Score](#F1Score)
2. [**Vectorial Representations**](#VectorialRepresentations)
   1. [Preprocessing](#Preprocessing)
   2. [Representations](#Representations)
      1. [Tf-Idf](#tfidf)
      2. [Word2Vec](#word2vec)
      3. [BERT](#bert)
3. [**Classifiers**](#classifiers)
   1. [Logistical Regression](#logistical_regression)
   2. [Neural Network](#neural_network)
4. [**Experimental Results**](#experimental_results)
   1. [Work Environment](#work_environment)
   2. Results
   3. Problems encountered

**Conclusions**

[**Bibliography**](#bibliography)

**Introduction**

The World Health Organization (WHO) lists depression as one of the top causes of disability globally, impacting over 264 million people. Depression is a serious global public health concern. It is characterized by recurrent depressive sensations, loss of interest or pleasure, lack of energy, attention problems, and suicidal or death thoughts. However, depression is frequently misdiagnosed and poorly treated, and many people suffer in silence as a result. Therefore, it is crucial to create new, efficient methods for early identification and intervention.

Detection of depression is vital to be done as soon as possible. Early detection and treatment have been found to change the course of depression by easing symptoms and enhancing long-term results. Traditional diagnostic techniques, such clinical interviews, might be hampered by stigma, access to care, and unreliable self-evaluation. This highlights the demand for new, non-invasive, and easily accessible techniques for diagnosing depression.

Analyzing the language people use in their written documents is one potential strategy to improve early detection; this field of research has gained popularity with the introduction of machine learning and natural language processing technology. Subtle variations in our word choice, sentence structure, and tone can provide important hints about our mental health condition because language reflects our thoughts, feelings, and mental state. Written texts are a valuable source of information for identifying depression symptoms since language is crucial for expressing and understanding emotional states.

Digital communication platforms like social media, blogs, and forums are being used more often, which has resulted in an enormous and expanding supply of text data that can be analyzed. People frequently use these platforms as a means of expressing their ideas and emotions, which may reveal early signs of depression that they might not be able or willing to disclose in a clinical context. This makes text analysis a valuable tool for the early, widespread detection of depression, in addition to being a practical method.

With the creation of a method for spotting depression in written texts, we hope to advance this newly emerging subject. If it is successful, such a system would improve our capacity to recognize those who are at risk, allowing for earlier intervention and perhaps even saving lives.

Depression frequently goes unreported, despite its increasing prevalence and crippling effects. Less than 50% of people affected globally, and less than 10% in many nations, receive treatment, according to the WHO. This treatment gap is a result of misdiagnoses, a lack of resources, and the stigma associated with mental illness.

This study uses the strength of machine learning and natural language processing technology to scan written content for indications of depression in order to overcome these difficulties. By doing this, it hopes to create a proactive approach that will aid in closing the treatment gap in mental health care and serve as a beneficial resource in the larger fight against depression.

**Contributions**

In the field of Natural Language Processing (NLP) and machine learning, there is an ongoing need to understand, test, and refine these technologies to optimize their utility in real-world applications. The task of detecting depression in written texts is no different. This project contributes to this area of research by refining and applying several established methodologies, exploring various text representations, and meticulously testing and modifying model hyperparameters.

One of the primary contributions of this work is in the systematic tuning and comparison of different machine learning models. While several existing studies have explored the application of specific models, a comprehensive comparison of their performances in the context of depression detection is less common. In this project, a variety of models, including linear regression, support vector machines, and deep learning methods were thoroughly tested.

Another significant contribution lies in the exploration of different text representations. The representation of text data plays a crucial role in the success of any NLP task, and this project focused on comparing the performances of different representation methods. Various techniques, including Term Frequency-Inverse Document Frequency (TF-IDF) and more sophisticated methods such as Word2Vec­­­ were used to represent the text data.

Additionally, this project gave a lot of attention into the optimization of model hyperparameters, an essential aspect of machine learning research. A systematic approach to hyperparameter tuning was employed in this project, using techniques like grid search and cross-validation. This not only enhanced the performance of the models but also provided an understanding of the sensitivity of different models to their hyperparameters.

This study also contributes to the field through the use of real-world data. The models were trained and tested on publicly available dataset gathered mainly from Twitter. This approach ensures that the findings of the study are applicable and relevant to real-world scenarios, where the language used can be much more diverse and unstructured.

Lastly, this project contributes to the broader discourse on mental health and the role of technology in its management. By highlighting the potential of text analysis for detecting signs of depression, this work underscores the need for further research and development in this area and opens the door for future exploration.

In summary, this project contributes to the ongoing efforts to improve early detection of depression by utilizing the power of machine learning and NLP, providing a valuable resource for future researchers and practitioners in this critical field.

**Chapter 1**

**Presentation of the problem**

**1.1** **The Data Set**

The data set that was used for this project was obtained from a publicly available online source. It consists of a collection of tweets from Twitter that were scraped for the purpose of creating a substantial and diverse data set.

This data set was specifically curated for the objective of detecting signs of depression in text. Tweets were used as they offer a wide range of language use, emotions, and subject matter. Given the nature of the platform, they are also more reflective of real-world language usage, thereby enhancing the practical relevance of the project.

The classification of the data set is binary, meaning that the data is divided into two distinct categories or classes: 'depressed' (around 3500 tweets) and 'non-depressed' (around 4800). This categorization was done based on the content of the tweets. The 'depressed' class includes tweets that exhibit signs of depressive sentiment, while the 'non-depressed' class consists of tweets that do not show these signs.

This simple binary classification method provides a clear framework for training and testing the machine learning models. It facilitates a straightforward evaluation of the models' performance, as the predictions can be easily compared to the actual classifications.

The data set served as a critical resource for this project. It provided a broad and real-world basis upon which to train, test, and evaluate the various methods and models explored in the study. By using this data set, the project aimed to develop machine learning models that can accurately and effectively identify signs of depression in written text.

**1.2** **Metrics used**

In machine learning tasks, especially those involving classification like this one, it's crucial to have a measure to evaluate how well our models are performing. For this project, we have chosen to focus on three related metrics: Precision, Sensitivity (also known as Recall), and the F1 Score. These metrics are interconnected and when considered together, provide a well-rounded view of the model's performance.

**1.2.1** **Precision**

Precision is one of the fundamental metrics used in classification tasks. It measures the proportion of correctly identified positive cases from all the cases that the model predicted as positive. In the context of this project, it represents how many of the tweets that our model flagged as 'depressed' were indeed from the 'depressed' class. A high precision indicates that when our model predicts a tweet as 'depressed', it's very likely to be correct.

Precision is calculated as the number of true positives divided by the sum of true positives and false positives.

The outcome ranges between 0 and 1, where 1 indicates perfect precision, and 0 is the worst possible score. In the context of this project, a true positive is a tweet that is correctly identified as ‘depressed’, and a false positive is a tweet incorrectly identified as ‘depressed’ when it is not.

**1.2.2** **Sensitivity**

Sensitivity, also known as Recall, is another key metric. It focuses on the model's ability to correctly identify all positive cases. In our context, it represents how well our model can detect 'depressed' tweets from all the actual 'depressed' tweets in the dataset. High sensitivity implies that our model is good at catching most of the 'depressed' tweets, and few 'depressed' tweets will go unnoticed.

Sensitivity or Recall is calculated as the number of true positives divided by the sum of true positives and false negatives.

Here, a false negative is a tweet that is incorrectly identified as 'non-depressed' when it is, in fact, 'depressed'. This metric also ranges between 0 and 1, 1 being the best possible outcome, 0 being the worst.

**1.2.3** **F1 Score**

While Precision and Sensitivity give us valuable insights, looking at them individually can be misleading as they don't provide a balanced view of the model's performance. That's where the F1 Score comes in. The F1 Score is the harmonic mean of Precision and Sensitivity. It ranges from 0 to 1, with 1 indicating perfect precision and recall, and 0 being the worst score.

What's particularly useful about the F1 Score is that, unlike a simple average, it leans towards the smaller value between Precision and Sensitivity. Therefore, a good F1 Score requires both precision and recall to be high. In the context of this project, using the F1 Score as our main metric ensures that we are not favoring either Precision or Sensitivity, but seeking a balance between them. This balance is particularly important when we aim to detect signs of depression in texts, as we want our model to correctly identify 'depressed' tweets (Precision) without missing too many of them (Sensitivity).

The F1 Score is the harmonic mean of Precision and Sensitivity.

As in its components, F1 Score also ranges between 0 and 1, 0 being the worst outcome possible, 1 being the best.

**Chapter 2**

**Vectorial Representations**

**2.1** **Preprocessing**

The process of transforming raw data into a suitable format for machine learning models is known as preprocessing. Preprocessing is a crucial step in any machine learning project as it can significantly impact the performance of the model.

In the context of NLP and this project, preprocessing involves several steps to clean and normalize the text data from tweets. The main goal is to reduce the complexity of the data and eliminate unnecessary information, allowing the model to focus on the meaningful parts of the text.

Firstly, removal of digits and non-alphanumeric characters was performed. This includes getting rid of punctuation, symbols, and numbers, which usually don't carry useful information for the task of identifying depressive signs in the text, this being the assumption that was taken when doing the preprocessing.

Usually emoticons and ‘stop words’ would be also removed in the preprocessing step, but a decision was made not to remove them, as they may carry impactful information regarding the sentiment of the tweet. Stop words are the commonly used words in a language (e.g. ‘and’, ‘the’, ‘a’, ‘not’) and are considered irrelevant when discovering the sentiment of the text. Nevertheless, when deciding if a tweet is classified ‘depressive’ or ‘non-depressive’, a simple stop word like ‘not’ may make the difference.

Additionally, all text has been converted to lowercase to maintain consistency and reduce the size of the vocabulary the model needs to understand. This step ensures that the same words in different cases are not treated as different words by the model.

One of the important steps in the preprocessing of text data is tokenization. Tokenization is the process of splitting the text into individual words or 'tokens'. This helps the model to analyze the text on a word-by-word basis.

Finally, lemmatization or stemming is sometimes applied to reduce words to their base or root form. For example, 'running' would be reduced to 'run'. This can help the model generalize better.

In this project, preprocessing was carried out using Python programming language with the help of several libraries. Libraries like NLTK (Natural Language Toolkit) were used for tokenization, stop word removal, and lemmatization. Regular expressions (a feature of Python and many other programming languages) were used for the removal of digits, emoticons, and other non-alphanumeric characters.

In conclusion, preprocessing is a vital part of this project. It helps to simplify the text data, making it easier for the model to learn from it and thus improving the overall performance of the depression detection task.

**2.2** **Representations**

**2.2.1** **Tf-Idf**

In order to feed text data to machine learning algorithms, we need to convert the text into numerical or vector form. One such method is Term Frequency-Inverse Document Frequency (TF-IDF). This technique is widely used in information retrieval and text mining to represent text data.

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure used to evaluate the importance of a word in a document, which is part of a larger corpus. This technique transforms text into meaningful numerical representations that machine learning algorithms can understand and learn from.

The TF-IDF score of a word is the product of two different metrics: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF) is a measure of how frequently a term appears in a document. It's calculated as:

Inverse Document Frequency (IDF) measures the importance of a term within the entire corpus. It's calculated as:

Where *NoOfInstances* is the number of times the word has appeared in the documents, and *Fd* is the number of documents in which the word has appeared.

The TF-IDF score is then calculated as:

The key benefit of the TF-IDF approach is that it gives more weight to terms that are more unique and meaningful to a document, and less weight to terms that are used frequently across all documents, like common words ('and', 'is', 'the', etc.). This reduces the noise in the data and helps the model focus on the significant parts of the text.

In the context of this project, TF-IDF can help in identifying words or terms that are particularly associated with depressive sentiment. For instance, a word that frequently appears in 'depressed' tweets but rarely appears in 'non-depressed' ones will get a high TF-IDF score. This makes it a useful feature for the machine learning model to learn from.

Another advantage of TF-IDF is its simplicity and ease of implementation. Libraries such as Scikit-learn provide built-in functions for calculating TF-IDF, making it a straightforward and efficient choice for text representation.

In conclusion, the TF-IDF representation was chosen for this project due to its ability to transform text into meaningful numerical representations, its focus on important terms over common ones, and its ease of implementation. This makes it a powerful and efficient method for preparing the text data for machine learning models.

**2.2.2** **Word2Vec**

While TF-IDF provides an efficient method of vector representation, it does have some limitations, primarily its lack of capturing semantic meanings and relationships between words. To address these limitations, another vector representation called Word2Vec is used in this project.

Word2Vec, developed by researchers at Google, is a more sophisticated method of representing words as vectors. It uses a shallow neural network to learn vector representations in a way that similar words have similar vector representations in the vector space, thereby capturing the semantic meaning and relationship between words.

Word2Vec model is trained on a large corpus of text and then each word is represented by a dense vector which is learned as the model is trained. This vector captures much more information about the word, specifically about how it is used and what words it is used with.

There are two primary training algorithms for Word2Vec:

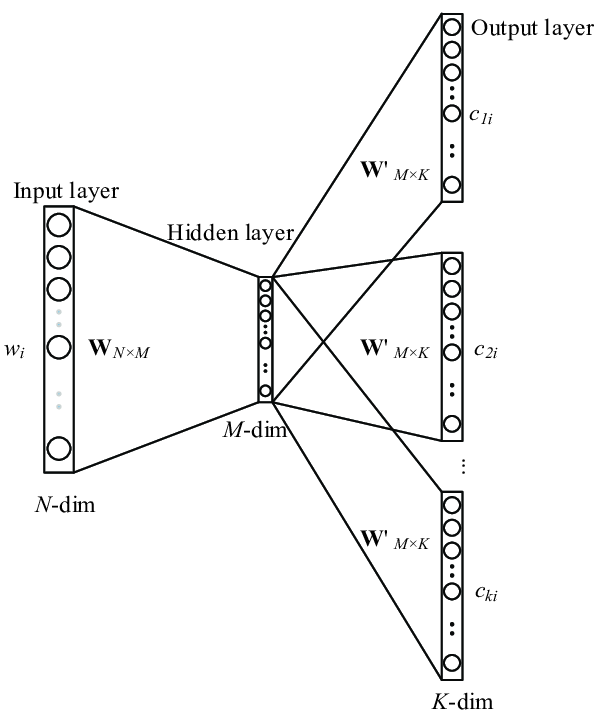
* **Continuous Bag of Words (CBOW)**, where the aim is to predict a word given the surrounding words (context).
* **Skip-Gram**, where the goal is the opposite: to predict the surrounding words given a word (context).

One of the major advantages of Word2Vec over methods like TF-IDF is its ability to capture semantic meaning. For instance, Word2Vec can understand that words like 'happy' and 'joyful' are similar in meaning and can understand the usage of a word based on its context.

In the context of this project, this ability to capture semantic meaning can be beneficial in identifying signs of depression in text. Words expressing depressive sentiments may not always be the same; however, they often convey similar semantic meanings. By understanding these similarities, Word2Vec can help in identifying depressive signs more effectively.

Furthermore, Word2Vec creates dense vectors, which means all elements in the vector contribute to defining the position of the vector in the vector space. This allows for a richer representation of words, capturing more nuanced relationships between them.

A general visual representation can be seen in the below illustration:



***Neural network of the Skip-Gram model***

Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FThe-architecture-of-Skip-gram-model-20\_fig1\_322905432&psig=AOvVaw2qyo87j3p7-ncJjHXLc59v&ust=1687523651186000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCKjYkdDx1v8CFQAAAAAdAAAAABAJ

Lastly, Word2Vec vectors are usually of fixed length irrespective of the size of the vocabulary, making them more computationally efficient than sparse representations (like TF-IDF) for large vocabularies.

In conclusion, Word2Vec was chosen as a method for vector representation in this project due to its ability to capture semantic meanings and relationships, its efficient representation for large vocabularies, and its potential for identifying signs of depression in text more effectively.

**2.2.3** **BERT**

To further improve our model's performance and capture even more complex relationships in the text, we also employ BERT (Bidirectional Encoder Representations from Transformers) for text representation. Developed by Google, BERT is a state-of-the-art model that has revolutionized the field of Natural Language Processing (NLP).

BERT stands out from previous methods due to its bidirectional nature. Traditional models like Word2Vec or GloVe learn representations by predicting a word based on its surrounding words, either from left-to-right (forward direction) or right-to-left (backward direction). In contrast, BERT takes into account the context from both sides (bidirectionally), leading to a more comprehensive understanding of the text.

BERT employs the Transformer, an attention mechanism that weighs the significance of different words when understanding the context. The Transformer, which was introduced in the paper "Attention is All You Need" by Vaswani et al., has shown to effectively capture the dependencies between words, even if they are far apart in the sentence.

Furthermore, BERT has been pre-trained on a large corpus of text, including the entire Wikipedia and a large book corpus. This pre-training phase allows BERT to understand the complex semantics of language and capture intricate patterns that would not be evident from smaller corpora.

One of the key advantages of BERT is its ability to understand the context of words in a way that many previous models can't. By analyzing text in both directions, BERT can more accurately capture the meanings of words based on their surrounding context.

Moreover, BERT is capable of understanding more nuanced aspects of language, such as sarcasm or irony, that can be particularly challenging for other models. This capability can be crucial in tasks like detecting signs of depression in text, where the emotional context is key.

Finally, BERT can be fine-tuned with just one additional output layer to create state-of-the-art models for a variety of NLP tasks, without substantial task-specific architecture modifications.

In this project, BERT was employed due to its high capability in capturing context and semantic meanings, its adaptability to various NLP tasks, and its promise in improving the model's ability to detect signs of depression in text. Using BERT as part of our pipeline can potentially lead to more accurate and reliable results in depression detection.

**Chapter 3**

**Classifiers**

**3.1** **Logistical Regression**

Logistic Regression is one of the most straightforward and commonly used classifiers in Machine Learning, known for its simplicity, efficiency, and robustness. While its name might imply that it is a regression algorithm, it is actually used for binary classification tasks.

Logistic Regression, in its essence, uses a logistic function to model a binary dependent variable. This logistic function, also known as the sigmoid function, maps any real-valued number into a range between 0 and 1, giving an output that can be interpreted as a probability.

The general form of the logistic model is:

, *where*

* P(Y=1) is the probability of the class with label 1.
* e is the base of natural logarithms.
* z is the input to the function (the prediction of the algorithm).

z is calculated as the weighted sum of the feature (x) plus a bias (b):

In the context of binary classification, the output of the logistic regression model can be interpreted as the probability of a particular sample belonging to the positive class. A threshold, typically 0.5, can then be applied to this probability to determine the final class label.

One of the key strengths of Logistic Regression is its simplicity and efficiency. It's relatively easy to implement and understand, making it a good baseline model for binary classification tasks.

Logistic Regression can also provide insight into the relationship between the features and the predicted outcome. The weights learned by the model can be interpreted as the importance of the corresponding features.

Furthermore, Logistic Regression works well with high-dimensional datasets, making it a good fit for text classification tasks, where after the text representation phase, we often deal with a large number of features.

Despite its simplicity, Logistic Regression can perform exceptionally well in many scenarios, especially when the data is linearly separable.

In the context of this project, Logistic Regression serves as a baseline model. It provides a benchmark for the performance of more complex models and allows for easy interpretation of the model's decisions. The performance of Logistic Regression on this task can also help to validate the quality of the feature extraction and preprocessing steps, given its sensitivity to the input feature space.

While Logistic Regression is a powerful tool, it's not without its limitations. Here are a few key ones to consider:

* **Linear Decision Boundary:** Logistic Regression assumes a linear decision boundary. It works well when the classes are linearly separable (i.e., you can draw a straight line or a hyperplane to separate the classes). However, if this isn't the case (which often happens in real-world datasets), the logistic regression classifier may underperform.
* **Sensitive to Feature Scale:** Logistic Regression is sensitive to the scale of input features. If features are on different scales, they can impact the model's performance, making feature scaling a necessary preprocessing step.
* **Outliers:** Logistic Regression can be sensitive to outliers in the dataset. Outliers can affect the estimation of the regression coefficients, which can reduce the model's accuracy.
* **Multicollinearity:** Logistic Regression assumes that the input features are not highly correlated with each other (a condition known as multicollinearity). If this condition is violated, it can cause the model to become unstable and produce unreliable and hard-to-interpret results.

In the context of this project, we need to be mindful of these limitations when applying Logistic Regression. The decision to use Logistic Regression was primarily driven by its simplicity and interpretability, and it serves as an effective baseline to compare with more complex models.

Given the nature of text data, we have ensured that the text representation methods provide vectors that limit multicollinearity. Outliers in terms of overly long or short texts have been managed during preprocessing, and feature scaling has been applied during the text vectorization phase.

Thus, while it might not be the most sophisticated model available, Logistic Regression provides a valuable first step and benchmark in the detection of signs of depression in text data. The results obtained from Logistic Regression can inform and guide the application of more complex models, helping us build an effective and robust system for depression detection.

**3.2** **Neural Network**

Neural Networks represent a core component of modern Machine Learning, providing powerful capabilities in tasks that involve complex pattern recognition, such as text analysis. They're inspired by the biological neural networks that constitute our brains, hence the name.

A Neural Network is made up of an input layer, one or more hidden layers, and an output layer. Each layer consists of multiple nodes, also known as neurons, which are interconnected. Here's what each layer does:

* **Input Layer:** This layer receives the input data. The number of nodes in this layer usually corresponds to the number of features in the data.
* **Hidden Layer(s):** These layers perform computations on the inputs received and pass the result to the next layer. The complexity of a neural network often lies in these layers, as they can transform the input in non-linear ways that can help solve complex tasks. The number of hidden layers and the number of nodes in each hidden layer are important hyperparameters that can significantly influence the model's performance.
* **Output Layer:** This layer produces the final output. The number of nodes in this layer corresponds to the number of classes in the case of classification tasks.

Different types of Neural Networks exist, each suited to a specific type of task. Here are some commonly used ones:

* **Dense (Fully Connected) Layers:** In these layers, every node is connected to every node in the previous layer. These are the most traditional layers in neural networks.
* **Embedding Layers:** These are used for converting categorical data or text data into continuous vectors that can be input to a neural network. They are often used in natural language processing tasks.
* **LSTM (Long Short Term Memory) Layers:** LSTM is a type of recurrent neural network that can remember patterns over time, making them very effective for tasks involving sequential data, like time series analysis or text.
* **Dropout Layers:** These are a regularization technique that randomly sets a fraction of the input units to 0 at each update during training, which helps prevent overfitting.

Activation functions introduce non-linearity into the network, allowing it to learn complex patterns. Some commonly used ones are:

* **Softmax:** This function converts a vector of numbers into a probability distribution, making it useful for the output layer in multi-class classification problems.

Where exp(x) is the exponential function and denotes the summation.

* **ReLU (Rectified Linear Unit):** This function outputs the input directly if it's positive, else it outputs zero. It's the most widely used activation function for hidden layers.
* **Leaky ReLU:** This is a variant of ReLU that allows small negative values when the input is less than zero, mitigating the "dying ReLU" problem where neurons can become inactive and only output 0.

This function return x if x is greater than 0, otherwise.

* **ELU (Exponential Linear Unit):** This function is similar to Leaky ReLU, but it takes on exponential values for inputs less than zero, which can help the network learn more complex patterns.

Several hyperparameters can be adjusted in neural networks:

* **Number of Epochs:** An epoch is one complete pass through the entire training dataset. The number of epochs is how many times the learning algorithm will work through the entire training dataset.
* **Learning Rate:** This parameter determines how much the weights of the network will change in each step of learning. A smaller learning rate might lead to more accurate learning, but the process will be slower.
* **Initial Weights and Biases:** The initial values of the weights and biases can have a big effect on the final accuracy of the model. Often, these are set randomly, but specific techniques can also be used.

**Chapter 4**

**Experimental Results**

**4.1** **Work Environment**

The primary device used for developing and running the code was a MacBook Pro 2019 model. The specific specifications are as follows:

* **Processor:** 2.4 GHz 8-Core Intel Core i9
* **Memory:** 32 GB 2667 MHz DDR4
* **Storage:** 1TB SSD

This device was chosen for its reliable performance and processing capabilities, which were crucial for handling the intensive computations often required in machine learning tasks.

The project was implemented in Python, a language widely favored in the data science community for its readability, simplicity, and extensive library support.

Here are the key Python libraries and APIs used:

* **NumPy and Pandas:** These libraries were used for data manipulation and analysis. NumPy provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Pandas is used for data manipulation and analysis, providing data structures and functions needed to manipulate structured data.
* **Scikit-learn:** This is one of the most widely used machine learning libraries in Python. It provides a range of supervised and unsupervised learning algorithms. For this project, it was used for implementing Logistic Regression and for various tasks like splitting the dataset and computing the accuracy metrics.
* **TensorFlow and Keras:** TensorFlow is an open-source library developed by Google for building and training neural networks. Keras is a user-friendly neural network library written in Python that runs on top of TensorFlow. Keras was used to build the neural network models due to its simplicity and high-level APIs.
* **Natural Language Toolkit (NLTK):** This library was used for the text preprocessing tasks. NLTK is a leading platform for building Python programs to work with human language data and provides easy-to-use interfaces to many corpora and lexical resources.
* **Gensim:** This library was used for unsupervised semantic modelling from plain text. It can handle large text collections using data streaming and incremental online algorithms, which is important for scalability.

The entire project was developed using PyCharm, an integrated development environment (IDE) used for programming in Python. PyCharm offers a wide array of features such as smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, automated code refactoring, and rich navigation capabilities. These features make it a highly productive tool for Python developers.

In this particular project, PyCharm's capabilities for code editing, running, debugging, and testing were leveraged extensively. The IDE's support for web technologies also aided in the process of fetching and processing data from the web.

**4.2 Results**

**Bibliography**