UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

DEVELOPMENT OF A MOBILE MOOD DETECTOR BASED ON MACHINE LEARNING TECHNIQUES

JORGE GARCÍA DE LOS HUERTOS SÁNCHEZ
2020

TRABAJO DE FIN DE MASTER

Título:	DESARROLLO DE UN DETECTOR DE ESTADOS DE ÁNIMO MOVIL BASADO EN TÉCNICAS DE MACHINE LEARNING
Título (inglés):	DEVELOPMENT OF A MOBILE MOOD DETECTOR BASED ON MACHINE LEARNING TECHNIQUES
Autor:	JORGE GARCÍA DE LOS HUERTOS SÁNCHEZ
Tutor:	CARLOS A. IGLESIAS
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	——
Vocal:	——
Secretario:	——
Suplente:	——

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

DEVELOPMENT OF A MOBILE MOOD DETECTOR BASED ON MACHINE LEARNING TECHNIQUES

2020

Resumen

Según datos publicados por la Organización Mundial de la Salud (OMS), más de 2 millones de personas sufren trastornos mentales (como por ejemplo depresión) en España, lo que supone el 5,2% de la población de nuestro país, datos que han aumentado considerablemente en los últimos años, ya que desde el 2005 al 2015 el número de personas que sufre depresión ha aumentado un 18,4%. Mostrando números a nivel global, más de 320 millones de personas sufrieron depresión en 2015. En el caso de la ansiedad, 264 millones de personas la sufrieron a nivel global en ese mismo año, lo que supone un 3,6% de la población mundial. Para concluir con cifras numéricas de la OMS, unas 788.000 personas aproximadamente, se quitan la vida cada año a causa de la depresión u otro trastorno mental.

Además, numerosos estudios científicos confirman que realizar actividades deportivas diariamente influye favorablemente en la salud y en el estado de animo de las personas. En otras palabras, la actividad física diaria es saludable ya que cura y previene enfermedades. Sin embargo, esta relación también puede entenderse como una mera interpretación subjetiva de salud, que no viene a ser otra cosa más que el bienestar personal.

En este proyecto se desarrollará una aplicación Android que realizará un seguimiento constante del estado de ánimo del usuario. Por otra parte, irá recolectando datos relacionados con las conversaciones de mensajería instantánea (como Whatsapp), así como datos relacionados con la actividad física diaria del usuario. A partir de los datos obtenidos, se aplicarán algoritmos de aprendizaje automático, con el fin de predecir y clasificar el estado de ánimo de la persona.

Palabras clave: Android, Análisis de sentimientos, Análisis de emociones, Python, Java, Firebase, Keylogger

Abstract

According to data published by the World Health Organization (WHO), more than two million people suffer from mental disorders (such as depression) in Spain, which is 5.2% of the population of our country. These data have increased considerably in recent years and, from 2005 to 2015, the number of people suffering from depression has increased by 18.4%. Showing numbers at global level, more than 320 million people suffered from depression in 2015. In the case of anxiety, 264 million people suffered from it globally in the same year, which is 3.6% of the world's population. To conclude with WHO statistics, about 788,000 people commit suicide each year because of depression or other mental disorder.

In addition, numerous scientific studies confirm that engaging in daily sports activities has a positive influence on people's health and mood. In other words, daily physical activity is healthy because it cures and prevents illness. However, this relationship can also be understood as a subjective interpretation of health, which is nothing more than personal well-being.

In this project, an Android application is developed, which constantly monitors the user's mood. In addition, the application collects data related to instant messaging conversations (such as *Whatsapp*), as well as data related to the user's daily physical activity. Based on the data obtained, machine learning algorithms will be applied in order to predict and classify the user's mood.

Keywords: Android, Sentiment analysis, Opinion mining, Python, Java, Firebase, Keylogger

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CSV	Coma Separated Value
ESM	Experience Sampling Method
EWA	Express Web Application
FP	False Positive
FN	False Negative
JSON	JavaScript Object Notation
MSD	Merck Sharp and Dohme Corporation
ML	Machine Learning
NLP	Natural Language Process
NLTK	Natural Language Toolkit
OMS	Organización Mundial de la Salud
OS	Operating System
POS	Part of Speech
PANA	Positive Activation - Negative Activation
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines
TF-IDF	Term Frecuency - Inverse Document Frecuency
TP	True Positive

TN	True Negative
UML	Unified Modeling Language
K-NN	K-Nearest Neighbors
PDF	Probability Density Function
WHO	World Health Organization

CHAPTER 1

Introduction

This chapter introduces the context of the project, including a brief overview of all the different parts that will be discussed in the project. It will also break down a series of objectives to be carried out during the realization of the project. Moreover, introduces the structure of the document with an overview of each chapter.

1.1 Context

Technology, in recent years, has become a fundamental pillar of our society. Two or three decades ago, it was unthinkable that a car could be driven autonomously or that a person could have a mobile device that recommended exercise because he or she had not achieve their daily goals. Today, no one can be surprised by such basic things as having our smart watch monitor our daily physical activity. All of this things can be done, among other things, thanks to the large amount of information that the devices handle about us. Another thing that is basic today, but a few years ago was unthinkable, is the normalization of the use of mobile devices. In other words, nowadays everyone has a mobile device with Internet connection. This has meant that anyone can create mobile applications according to their needs.

Since a few years ago, companies have realized that having information is synonymous with having power (and therefore money). One proof of this was the purchase of *Whastapp Inc.* by *Facebook*. Whastapp was a simple instant messaging application that did not reach a hundred employees. The great value that this company had and what made it sell for more than twenty billion dollars was the large number of active users (600 million users in 2016) and therefore, information (**Big Data**).

On the other hand, an infinite number of systems have been developed that make this information very useful, thanks to **Artificial Intelligence (AI)**. AI is a set of techniques and algorithms that are used for a computer to learn and be able to solve problems, based on the information it uses. Within AI there is a subarea where the computer works and solves problems related to human languages. This is known as Natural Language Process (NLP). If we combine the use of Big Data with AI techniques (such as Machine Learning (ML)) and **Natural Language Process (NLP)**, the possibilities tend to be infinite. An example can be found in the area of marketing and advertising. By studying people's behaviour, marketing departments can better target advertising campaigns.

In this project, we wanted to increase our expertise in these areas. To do this, we have created an Android application that serves to collect information from users. This information is composed of automatically collected text messages, user entered moods and physical activity performed by the user. This application will allow us to manage a large amount of user data. On the other hand, we have developed a mood classification system based on text messages. This system has been developed using ML algorithms and NLP.

With this, we can perform a **Sentiment Analysis** and then, monitor the users' moods, with the main goal of improving people's mental health.

1.2 Project goals

In relation to the previous section, this project presents a set of goals that involve several of the technologies mentioned above.

First of all, the implementation of a classification system, based on text message. This should be done using Machine Learning (ML) techniques and NLP. To do this, previous research on sentiment analysis methods and classification algorithms is necessary.

Secondly, it is necessary to create an Android application that is able to collect user information. The application will use some mechanism to collect text messages automatically and transparent for the user. In addition, the application will be able to monitor the user's mood and physical activity on a daily basis. In order to do this, it is necessary to previously do some research about keyloggers, real time databases, self-report collectors, etc.

1.3 Structure of this document

The main objective of this section is to show the reader how the document is structured, for a better understanding.

Chapter 2 is formed by the state of the art. We have divided this chapter into three sections. The first section focuses on sentimental analysis. In this section, we explain at a high level the techniques that are used to carry out these analyses. The second section focuses on ML, where the techniques are explained. For a better understanding of this section, programming skills are necessary. The third section focuses on the technologies we used to develop the Android application.

Chapter 3 deals with requirement analysis. In this chapter, we will explain the actors involved in the system. In addition, an explanation of the use cases is given. For each use case the normal flow and the alternative flow are shown.

Chapter 4 shows the methodology that has been carried out for the realization of the mood classifier. To understand this part of the project, knowledge of Python and Machine Learning is necessary.

Chapter 5 shows the structure of the whole system, including the developed Android application.

Chapter 6 is a summary of the conclusions reached after the completion of this project. In addition, possible future work is shown.

CHAPTER 2

State of Art

This chapter introduces technical concepts that have been developed during the elaboration of the thesis. The main target of this chapter is to understand, from a general point of view, technical concepts related to Artificial Intelligence (AI) and everything that it encompasses. The concept of "Sentiment Analysis" is explained in a theoretical way, as well as through a set of examples. Other concepts, such as Experience Sampling Method (ESM) or the Circumplex Model, are also explained. In addition to explaining general concepts such as sentiment analysis, different techniques that exist for analyzing texts are also explained. With this, it will be much easier to understand the next chapters of the thesis.

2.1 Sentiment Analysis

“Sentiment analysis (also known as opinion mining) is the process of determining the emotional tone of a set of words” [24]. Given a set of words and using different computer mechanisms, the purpose is to identify the feelings that the words are trying to convey and then classify the polarity.

There are three levels of classification in sentiment analysis: document level, sentence level and aspect level. The document level analysis classify the complete document with a mood, while the analysis at sentence level classify each sentence, being possible to have in a single document multitude feelings. Aspect-level sentiment analysis aims to classify the sentiment with respect to the specific aspects of entities [44]. In this project we focus on sentiment analysis at sentence level. Throughout this chapter, we are going to explain the main techniques of sentiment analysis (Figure 2.1) that exist, deepening in those used for this thesis.

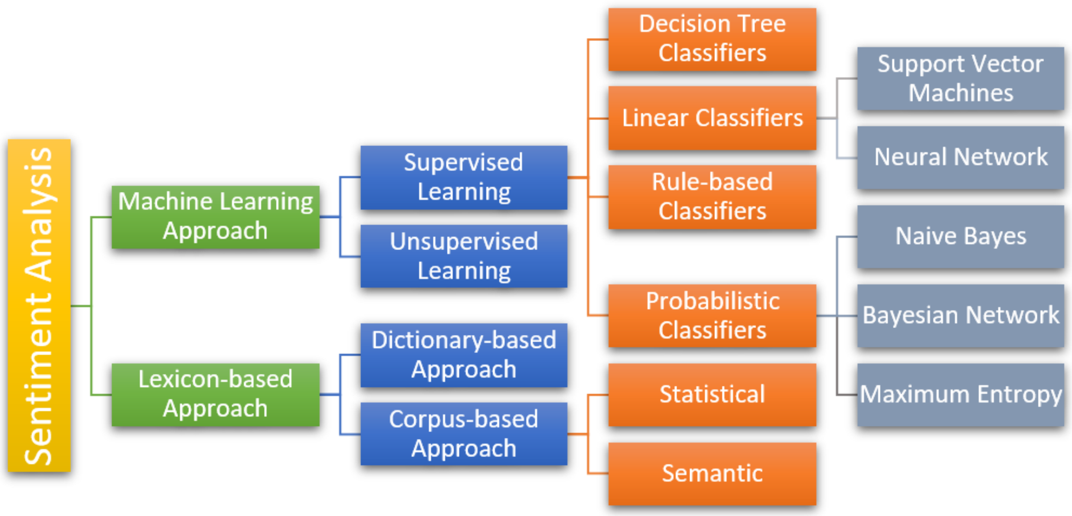


Figure 2.1: Mood Classification Techniques [26]

Sentiment analysis has many uses, and some of them could have an immense value. A well-known example of sentiment analysis was the case of Barack Obama’s Administration, which used this type of analysis to find out citizens’ opinions about political actions and their campaign messages during the 2012 elections.

Another example of sentiment analysis could be interpreting messages on social networks, with the objective of knowing the interests of the users. This can be very interesting for advertising companies, among others.

But sentiment analysis can be extremely complex, as human language is not simple.

Teaching a machine the context of different texts can sometimes be very tedious. Let's take an example to understand better what has been explained above:

"I have a test tomorrow morning at 7:00. This is so fun!"

At first, this sentence could be qualified as positive by a machine, since "fun" is a positive adjective. Obviously, nobody would qualify this sentence as positive, since the person who wrote this was trying to be sarcastic.

Prediction models can be trained to have a high success rate. Even though (as in all automated processes), you will never achieve a 100% success rate.

As we can see in Figure 2.1, there are many techniques for sentiment analysis. In the following sections, the main techniques of sentiment analysis are explained. There is another technique, called Experience Sampling Method (ESM), which is not represented in Figure 2.1 but we have also used it in this project.

2.1.1 Experience Sampling Method (ESM)

The Experience Sampling Method (ESM) [31] is a research methodology to study people's behavior. Basically it consists of asking users constantly and in random moments to obtain a report. This process should be repeated several times a day during many days or weeks. Once the method is finished, you will get a kind of diary about that person. Usually those questions are oriented to feelings and behaviors of the patient to study.

In order to do not interfere with the patient's responses, it should be ensured that users does not know at what time the questions will be asked. With this, it will be possible to take the subject off guard, and therefore, a more honest answer will be obtained.

In this thesis, a daily feelings tracking has been made to different users, using ESM methodology. It has been created through the Android application. A self-report collector has been added to the application, which ask in random time about the mood at that moment. In addition, other questions have been asked such as daily physical activity (if the user has performed any sport and how many kilometers has walked and run till that moment) or the geographical location where the user is (for example, at home, at work, on the street, in the gym, on the beach, etc).

The information collected can be useful for reporting and graphing. One of the graphs that can be created (and that has been created in this project) is the Circumplex Model. This graph can be very helpful in the sentiment analysis process. The following section explains what the Circumplex Model is all about.

2.1.2 Dimensional Models of Emotion

In the field of psychology, emotions or moods can be classified according to one or more dimensions. Most experts in psychology establish three dimensions to classify moods. Although the most accurate models usually have three dimensions, in many cases this is simplified to only two dimensions (valence and intensity). Nowadays there are many two-dimensional models. The most commonly used models are explained in the following lines:

- **Vector model:** This model consists of a pair of vectors pointing in two directions. The direction in which the mood is being measured is determined by the valence.
- **Positive Activation - Negative Activation (PANA) model:** This model argues that to classify a mood, two independent systems must be used. The way to measure emotions is similar to the vector model. The vertical axis represents positive moods and the horizontal axis represents negative moods.
- **Circumplex Model:** This circular model is divided in four parts, also named quadrants. A different kind of mood will be represented in each axis. The vertical axis represents feelings of intensity and the horizontal axis represents feelings of valence. By combining the two axes, conclusions can be obtained more visually. This is the dimensional model of emotion that has been used in this thesis [23]. In the Figure 2.2 we can see an example of a Circumplex Model.

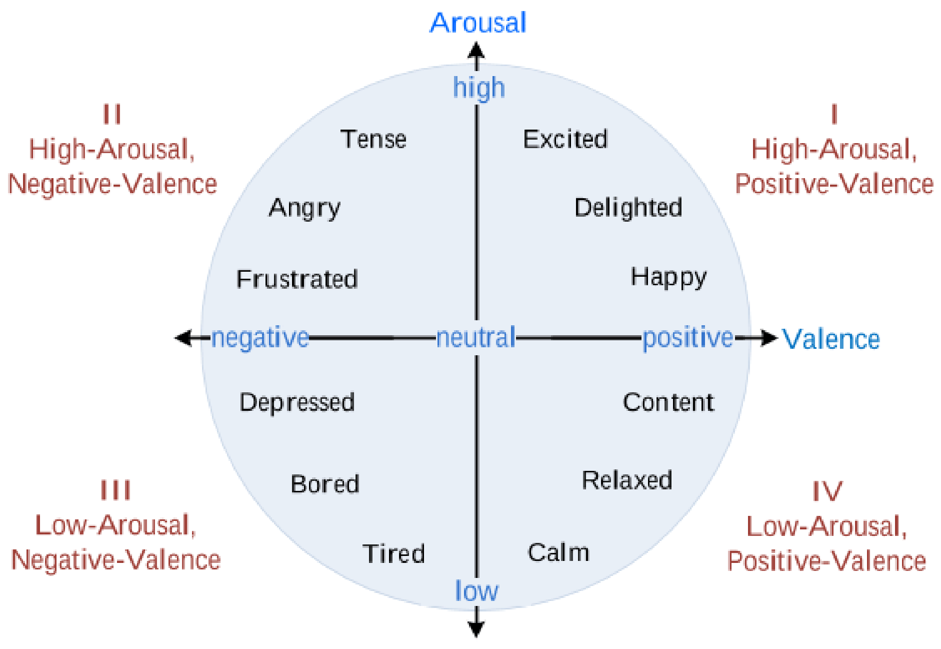


Figure 2.2: Circumplex Model of Mood [23]

In a circumplex model, the different mood state combinations are explained in the following lines:

- If the subject feels with a positive valence and a positive arousal, we can conclude that he is **happy**.
- If the subject feels with a negative valence and a negative arousal, we can conclude that he is **sad**.
- If the subject feels with a positive arousal and a negative valence, we can conclude that he is **stressed**.
- If the subject feels with a negative arousal and a positive valence, we can conclude that he is **relaxed**.

2.1.3 Lexicon-Based Approach

To carry out this type of sentiment analysis, two dictionaries are needed, a positive words dictionary and a negative words dictionary.

This method of classification can be divided into three sub-methods. The first is known as the "Manual Approach" and is practically never used because it is too slow. The other two sub-methods ("Dictionary-Based Approach" and "Corpus-Based Approach") are automatic and explained in the following lines.

On one hand, we have the *dictionary-based approach.* This is the most widely used method. The first step in carrying out this method is to collect a set of words with an associated label. Once we have a small dictionary of words, it grows through *WordNet* (by searching synonyms and antonyms). This will be done repeatedly until no new words are found. The main disadvantage of this method is the inability to find opinion words with domain and context specific orientations [44].

On the other hand, we have the *corpus-based approach.* With this method, the user not only knows the label to which each word is associated, but also the context. This can be very useful when analyzing feelings using machine learning. Within the corpus-based approach we can choose between two possibilities, the statistical approach or the semantic approach.

2.2 Machine Learning Approach

The machine learning approach refers to the use of machine learning algorithms to classify texts. This consists of, given a set of sentences, which have an assigned label (in this case a mood), managing to classify new texts whose label is unknown.

2.2.1 Reinforcement Learning

The main purpose of Reinforcement Learning (RL) (Figure 2.3) is to automatically decide the best behavior within a specific context. Rewards are used so that the agent learns from his behavior. This is known as a reinforcement signal.

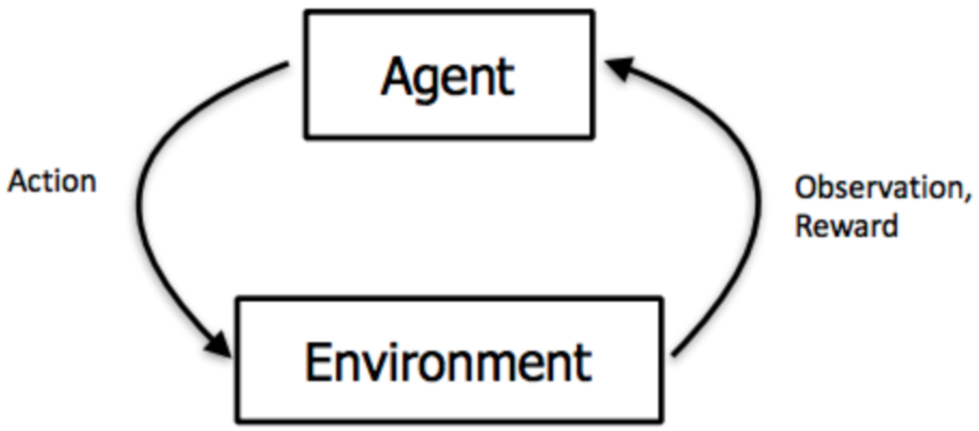


Figure 2.3: Reinforcement Learning diagram [22]

Reinforcement Learning (RL) is "concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward" [51]. RL differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

RL algorithms try to find a pattern that maps states of the world to the actions the agent ought to take in those states. This type of learning is used when the data has no labels or properties, but it is possible to get some kind of feedback. A well known example of RL is *HAL900* from the famous movie "2001: A Space Odyssey".

2.2.2 Unsupervised Learning

Unsupervised learning is a machine learning method where the model fits the observations. It is distinguished from supervised learning by the fact that there is no *a priori* knowledge [52].

Unsupervised learning is useful in cases where the data is not labeled or grouped by a property, and the algorithm must deduce or imply a relationship, as explained in Figure 2.4.

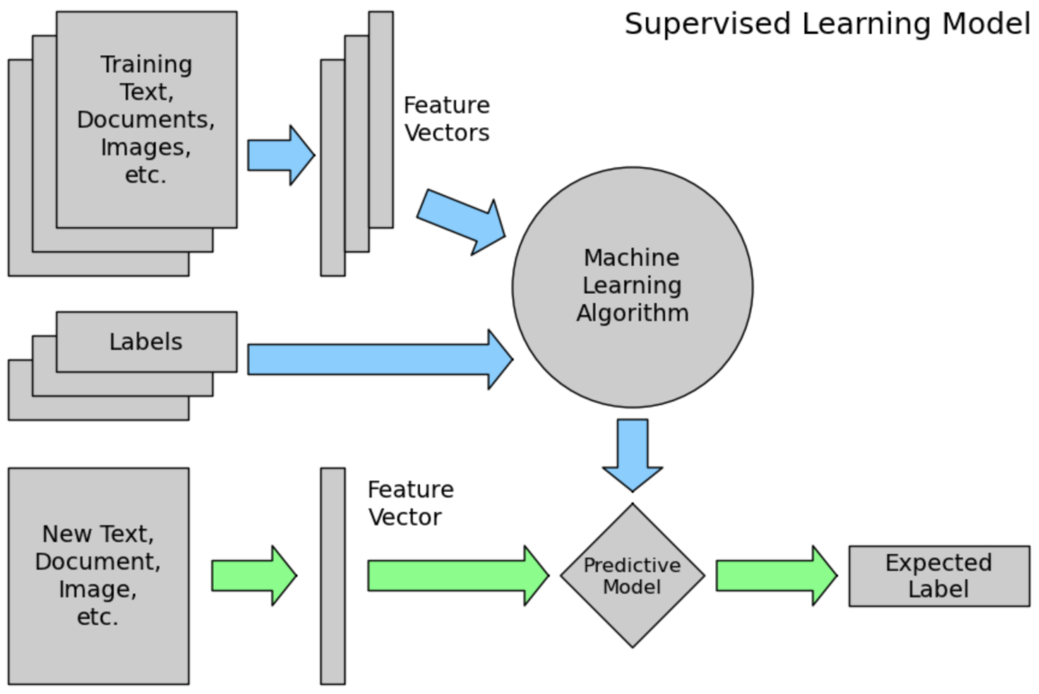

Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

Figure 2.4: Unsupervised Learning model [3]

2.2.3 Supervised Learning

“Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs” [9].

Supervised learning is used for cases where a label or property is available for the set of data that is going to be used for training the algorithm (Figure 2.5) [2].

One utility of supervised learning is to create classifiers. The most relevant ones are described below.

- **Probabilistic Classifiers:** In this category we can find the naive bayes classifier, the bayesian network or the maximum entropy classifier.

- **Rule-Based Classifiers:** Any classifier that uses the IF-THEN rule can be categorized as a rule-based classifier [5].

- **Linear Classifiers:** Here we can find support vector machines or neural network.

- **Decision Tree Classifiers:** It consists of the hierarchical decomposition of data. It is done in a recursive manner, until a conclusion is reached [53].

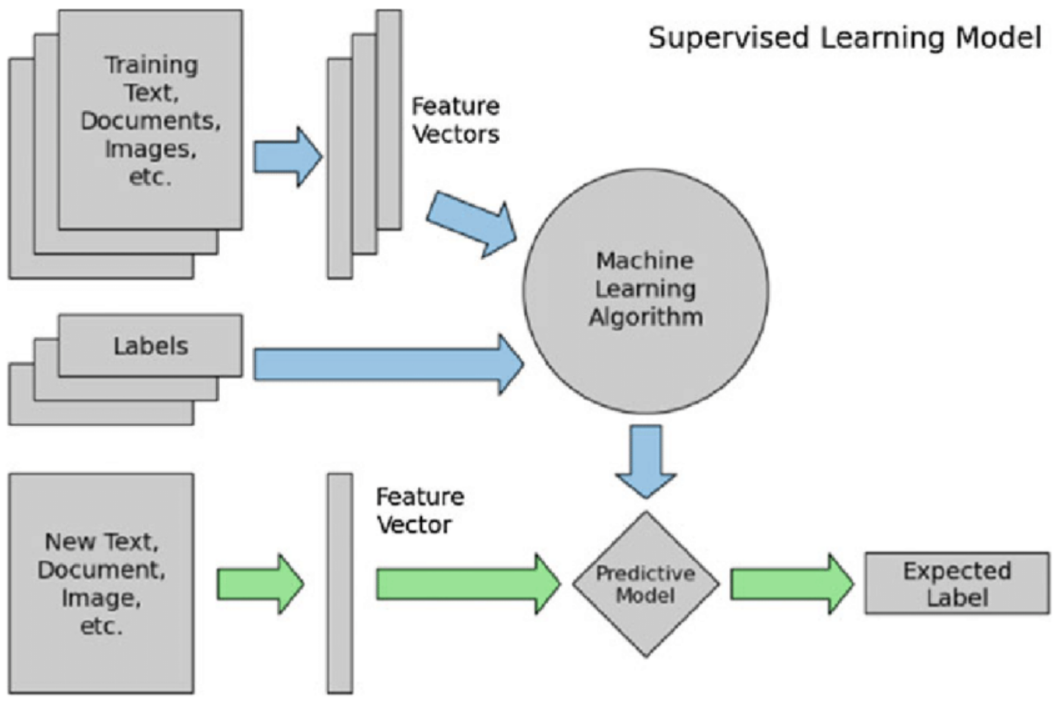

Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

Figure 2.5: Supervised Learning model [3]

2.2.4 Natural Language Process and Natural Language Toolkit

NLP is a type of AI. It consists mainly in the development of tools that can understand the different languages of the human being. Some practical examples of NLP could be voice recognition, translation of spoken sentences, comprehension and correction of texts, etc.

Natural Language Toolkit (NLTK) can be considered the most important library when it comes to NLP. It is written in Python and, like Scikit-Learn, is made up of a very large community working continuously on its development. Besides NLTK, we can find other libraries, such as Apache OpenNLP, Stanford NLP suite, Gate NLP library...

Here are some tools or functions that can be performed through this library:

- **Lemmatization:** It is a linguistic process that consists of obtaining the corresponding lemma of a word, that is to say, from the flexed form of a word, to find its dictionary form. A clear example would be, from the word “are”, to obtain the word “be”. There are many algorithms to perform this process, such as WordNetLemmatizer.

- **Stemming:** It is the process of getting the root of a word. For example, given the word “skater”, its stem would be “skat”. There are many algorithms that perform this function, such as Porter’s algorithm.

- **Tokenization:** It basically consists of splitting a phrase or text into a list of words.

- **StopWords:** Formed by different lists of words (there are different depending on the language) that do not contribute anything in NLP and should be removed. Some examples could be words such as “a”, “to”, “we”, etc.
- **Bigrams and Trigrams:** It consists of a sequence of two or three adjacent words. An example of a bigram would be as follows: Given the sentence “Peter is very intelligent”, his bigram would be “Peter is is very very intelligent”.

In this project, the whole tools mentioned above have been used, combining different algorithms and comparing the results obtained.

2.2.5 Scikit-Learn

Scikit-learn is surely the most widely used Python library in ML. It is an open source library. One of the main features is that it consists of a large number of supervised and unsupervised learning algorithms [19]. Scikit-learn is a library that is built in SciPy. It consists of the following libraries:

- **NumPy:** Library oriented to carry out operations with vectors and matrices. Formed by high-level mathematical functions.
- **Pandas:** Library whose main purpose is the analysis of data. It allows to structure the data to work in a more efficient way. Among the data structures offered, we can highlight the series and the dataframes [8].
- **SciPy:** Library formed by optimization modules, linear algebra modules, interpolation modules, special functions, etc.
- **Matplotlib:** Library to generate graphs in 2D, from lists or arrays [27].
- **Ipython:** Interactive console that allows to add extra functionalities.
- **SymPy:** Python library for symbolic mathematics [42].

This library consists of a large number of functions. Among the functions that can be performed with this library, we can highlight the supervised and unsupervised learning algorithms, cross validation, many datasets, etc..

One of the most significant advantages of this library is the community of developers that make it up. The fact that it is an open source library makes it have more than thirty-five collaborators continuously improving this library. Here are some of the most interesting features of this library:

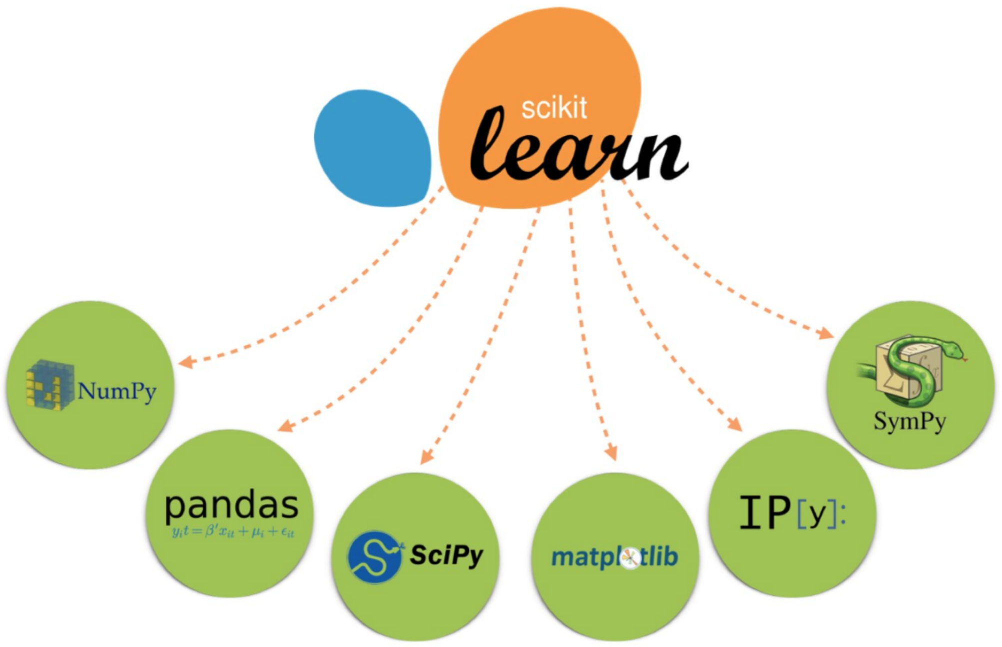


Figure 2.6: Scikit-Learn structure [35]

- ***model.fit:*** Fit training data. It is available in both supervised and unsupervised estimators. For the supervised learning case, two arguments have to be passed to it, the data (X) and the labels (Y). In the case of unsupervised learning, only one argument is passed to it, the data (X). Example: model.fit(X,Y).

- ***model.transform():*** Once you have a trained model, you can predict the labels of a new set of data. Only one parameter is passed to this function, the new data (new_X). This function is only available in supervised learning.

- ***model.predict_proba():*** This function is used in case we are making a classifier. This function returns the probability of each label in each data. The label with the highest probability will be returned. This function is only available in supervised learning.

- ***model.transform():*** Once you have a trained model, transform new data into the new basis. This function is passed a single argument (new_X) and returns the new representation of the data. It is only available in unsupervised learning.

- ***model.fit_transform():*** Performs a fit and a transform on the same input data. Only available for unsupervised learning.

2.2.6 Pandas

Pandas is a package written in Python, which allows to work with structures similar to R dataframes. Pandas depends on Numpy and the most outstanding data types of Pandas are the following ones [30]:

- Tabular data with heterogeneous type columns.
- Temporal series.

Pandas provides a great variety of tools, with which we can read and write data in multiple formats, carry out filters, manipulate and merge data, carry out different types of graphs, etc.

In this library, we can observe three types of fundamental data:

- **Series:** Equivalent to lists, in 1D.
- **DataFrame:** Equivalent to tables, in 2D.
- **Panels:** Equivalent to tables, but in 3D.

Pandas library has been used on numerous occasions in this project. Some of them are listed below:

- Read datasets that are in Coma Separated Value (CSV) format.
- Work with series and dataframes.
- Preprocessing the data.
- Viewing data and results.

2.3 Web Technologies

Once we have presented the basic concepts of sentiment analysis and machine learning, it is time to present other concepts that we have also used during the elaboration of this thesis. In this section we are going to explain the main tools for application development and management and, more in depth Firebase, since it is the tool we have used in our thesis.

Also, we are going to explain the concepts of keylogger and self-report collector, which are the two techniques we have used to obtain user data. We will use this data to predict the user's mood in the machine learning part.

2.3.1 Firebase

Firebase is a platform to develop web and mobile applications founded in 2011 and purchased by Google in 2014. This platform offers eighteen different tools and is currently present in more than one million applications [50]. For the realization of this master's thesis, the following Firebase tools have been used:

- **Real Time Database:** This tool (real time database) allows applications to synchronization and real-time data storage in the cloud. Also, users scalability can be done in a very efficient way. This NoSQL database is organized in JavaScript Object Notation (JSON) format. One of the main advantages of real time database is that users could continue use the application even when the device has no signal. This is achieved by persistent data storage when it has no signal and, once it has signal, synchronizes the data with the server.

  During the process of creating this project, into the database, a child has been created for each user, using the mobile device identification number as the unique identifier.

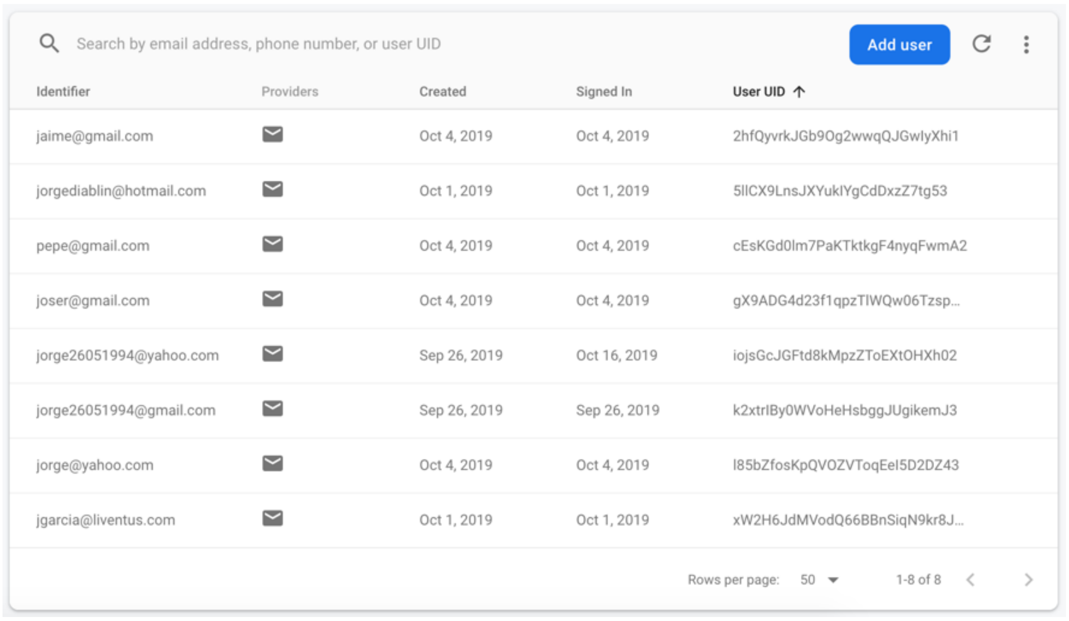
- **Firebase Authentication:** Firebase offers the possibility of adding user authentication process to mobile applications. With this tool (Firebase authentication), users can be registered directly from the Firebase console and/or through the application. This process can be used using different data, such as phone number, email, Facebook or Google credentials, etc.

  For this project, we decided to perform the authentication process through email and password, since it has been determined as the most generic. As we can see in Figure 2.7, when a user registration is made, some useful data such as registration date, last login date or the hashed password can also be obtained.

- **Analytics:** With Analytics, each user can be monitored, showing how many time each user use the application.

For this project, we have decided to use Firebase since it is the most used tool and the one that we believe has more support, particularly for Android applications (as in our case), because it belongs to Google. Despite this, there are very interesting alternatives. Here are some of Firebase competitors.

- **Parse Server:** This is an open source version of the Parse.com backend that can be deployed to any infrastructure that can run Node.js [29]. Before Parse Server existed, there was Parse.com, which belonged to Facebook, but in 2016 it decided to shut it down. The main difference between Parse.com and Parse Server is that Parse Server



Identifier	Providers	Created	Signed In	User UID ↑
jaime@gmail.com	✉	Oct 4, 2019	Oct 4, 2019	2hfQyvrkJGb9Og2wwqQJGwIyXhi1
jorgediablin@hotmail.com	✉	Oct 1, 2019	Oct 1, 2019	5lICX9LnsJXYukIYgCdDxzZ7tg53
pepe@gmail.com	✉	Oct 4, 2019	Oct 4, 2019	cEsKGd0lm7PaKTktkgF4nyqFwmA2
joser@gmail.com	✉	Oct 4, 2019	Oct 4, 2019	gX9ADG4d23f1qpzTlWQw06Tzsp...
jorge26051994@yahoo.com	✉	Sep 26, 2019	Oct 16, 2019	iojsGcJGFtd8kMpzZToEXtOHXh02
jorge26051994@gmail.com	✉	Sep 26, 2019	Sep 26, 2019	k2xtrIBy0WVoHeHsbggJUgikemJ3
jorge@yahoo.com	✉	Oct 4, 2019	Oct 4, 2019	l85bZfosKpQVOZVToqEeI5D2DZ43
jgarcia@liventus.com	✉	Oct 1, 2019	Oct 1, 2019	xW2H6JdMVodQ66BBnSiqN9kr8J...

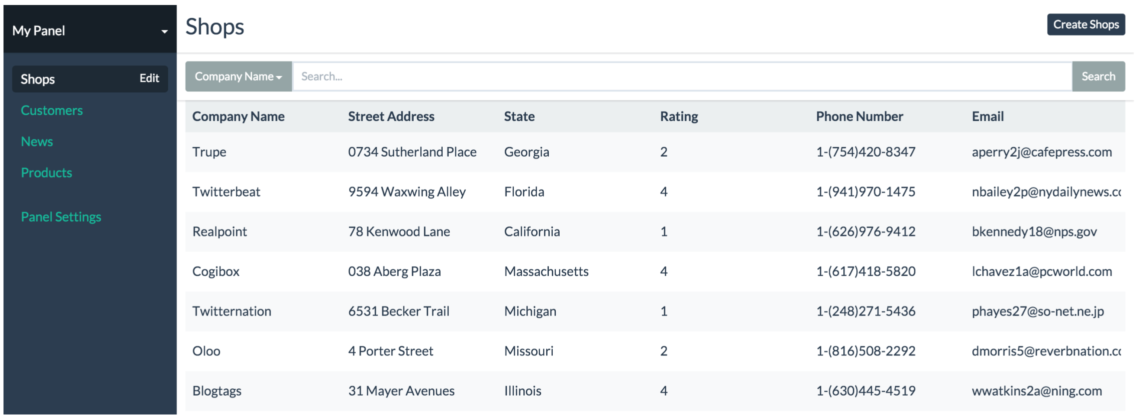
Rows per page: 50 ▾ 1-8 of 8 < >

Figure 2.7: FireBase Authentication

is open source. Parse Server could be considered Firebase's main competitor, due to its simplicity and the fact that it also offers a great number of tools. The main advantage of Parse Server is that can work with the Express Web Application (EWA) framework. As you can see in Figure 2.8, the interface is very similar to the Firebase interface [6].

- **Hoodie:** Hoodie is an off-line, self-hosted and open source JavaScript backend, that use a *noBackend* technology [16]. The main advantage of Hoodie is that the only thing the programmer needs is to work on the frontend, being able to completely forget about the backend. This is achieved through Hoodie's frontend Application Programming Interface (API), known as "Dreamcode". Another advantage of this software is that it can work without an Internet connection [6].

- **Horizon:** Horizon is an open-source backend platform, which allows you to easily create applications thanks to its JavaScript API. This software allows you to work without adding backend code. The main advantage is that if the developer wants, he can create his own backend code, loading the Horizon modules in Node.js. This software consists of three elements: a backend server, a JavaScript client library and a command-line tool [6].



Shops

Company Name	Street Address	State	Rating	Phone Number	Email
Trupe	0734 Sutherland Place	Georgia	2	1-(754)420-8347	aperry2j@cafepress.com
Twitterbeat	9594 Waxwing Alley	Florida	4	1-(941)970-1475	nbailey2p@nydailynews.c
Realpoint	78 Kenwood Lane	California	1	1-(626)976-9412	bkennedy18@nps.gov
Cogibox	038 Aberg Plaza	Massachusetts	4	1-(617)418-5820	lchavez1a@pcworld.com
Twitternation	6531 Becker Trail	Michigan	1	1-(248)271-5436	phayes27@so-net.ne.jp
Oloo	4 Porter Street	Missouri	2	1-(816)508-2292	dmorris5@reverbnation.c
Blogtags	31 Mayer Avenues	Illinois	4	1-(630)445-4519	wwatkins2a@ning.com

Figure 2.8: Parse Server interface [1]

2.3.2 KeyLogger

A keylogger is a type of software or hardware whose main function is to store each keystroke typed by the user. It is usually used as daemon malware.

In this section, we will focus on software keyloggers. As explained above, software keyloggers are small programs that are designed to work behind the main programs. Within this type of keylogger, there are many sub-types, depending on the way of creation. Here are some of software keyloggers sub-types:

- **Hypervisor-based:** This type of keylogger is theoretical. This is a type of keylogger that would work as a virtual machine [49].

- **Kernel-based:** This type of keylogger obtain root permissions. It works as if it were the keyboard driver, so it can control the functions of the keyboard to save the keystrokes [49].

- **API-based:** By using this type of keylogger you can monitor the Operating System (OS). With this, the user can detect the keystrokes [49].

There are other ways to create a keylogger, but these are the main ones. For this thesis, an API-based keylogger has been created using a feature called *AccessibilityService.*

AccessibilityService is a feature that improves user usability. This feature was conceived with the aim of helping users with disabilities or difficulties to use mobile devices. A very day-to-day example would be users who are driving and cannot use their hands and eyes to use their mobile phone [41].

	Unpleasant	**Pleasant**	**Very Pleasant**
Not Active	Sad	Calm	Relaxed
Active	Stressed	Happy	Elated
Very Active	Tense	Excited	Alert

Table 2.1: Mood combinations.

2.3.3 Self-report Collector

In addition to the data collected by the keylogger, the user's mood will also be registered daily. This mood registration will be done manually. To do this, the Experience Sampling Method (ESM) has been used, as it is the most widespread method when conducting studies on mood and behavior [41].

Self-report collector basically consists of a unique questionnaire that the user has to fill in at different times, every day. Our questionnaire will basically consist of two questions. By combining both answers, the application will obtain a conclusion about the user's mood. For example, if the user is very active and unpleasant, we can conclude by saying that the user is sad. Table 2.1 shows the result of all possible combinations that can be obtained. The questions (with their respective responses) that the user has to answer are:

1. **How pleasant do you feel?**
   (a) Not Response
   (b) Unpleasant
   (c) Pleasant
   (d) Very Pleasant

2. **How active do you feel?**
   (a) Not Response
   (b) Not Active
   (c) Active
   (d) Very Active

CHAPTER 3

Requirement Analysis

In this chapter a study of user needs and system requirements is carried out. This chapter allows to understand the main functionalities of the system.

3.1 Use Cases

In this section, we are going to present the main use cases of the system. In addition to presenting the use cases, we are going to present a list of actors involved in the system's activities. This will be done through tables and Unified Modeling Language (UML) diagrams, where actors and use cases will be shown in a more graphic way.

3.1.1 List of Actors

In this section, we will present the main actors involved in the use cases of the system. Below is a table showing the actor's id, the role, and a brief description of the actor.

Actor ID	Role	Description
ACT-1	User	End user that uses instant messaging apps and the app created in this thesis.
ACT-2	KeyLogger	Refers to the keylogger created by an Android API. This is the actor that captures every keystroke when the user is using an instant messaging application.
ACT-3	Server	Actor that performs the function of controlling the whole system. This is where the whole logical part of the system is stored.
ACT-4	EmoUPM	This is the application we have created for the user to record information every day.
ACT-5	Mood Detector	Part of the system that receives text messages as input and returns the mood to the output, using Machine Learning techniques.

Table 3.1: List of Main Actors

3.1.2 Use Cases Overview

Once the actors and use cases have been presented separately, we will proceed to present all of them in a more compact way. Next, we can see a single diagram with the use cases presented in the previous sections:

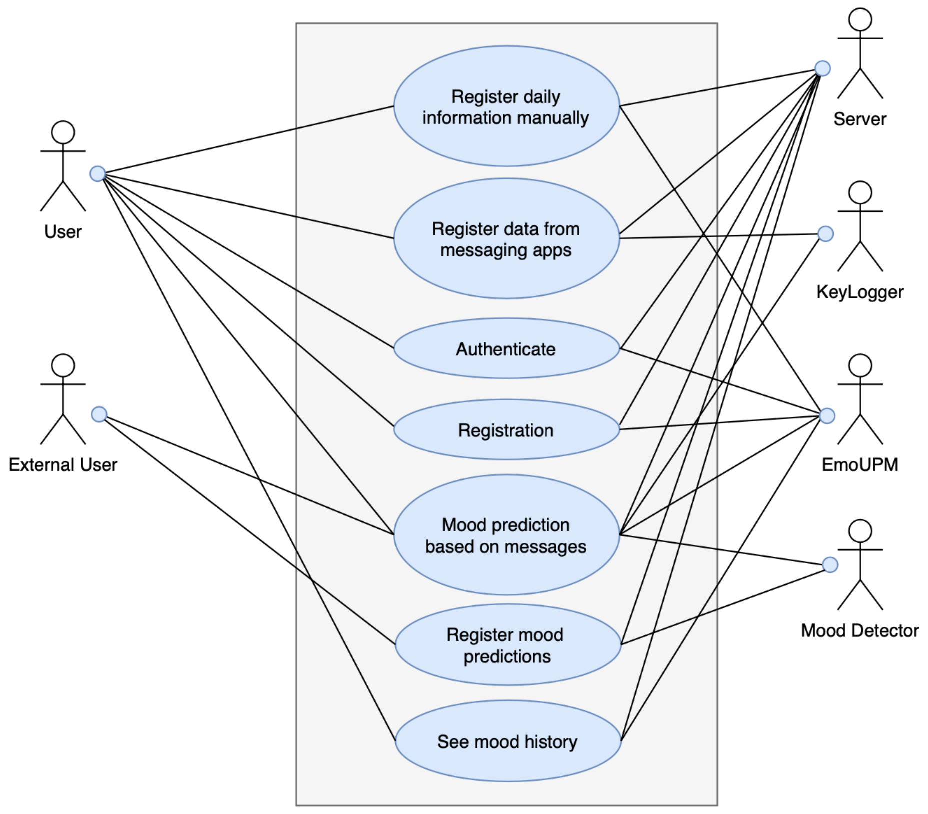

User
External User
Register daily information manually
Register data from messaging apps
Authenticate
Registration
Mood prediction based on messages
Register mood predictions
See mood history
Server
KeyLogger
EmoUPM
Mood Detector

Figure 3.1: Use Cases Diagram.

- Development and publication of an Android application where the user can register information on a daily basis. - **Mandatory** (UC-1, UC-3, UC-4, UC-5, UC-7)
- Implement keylogger for the mobile device to save keystrokes when using certain applications. - **Mandatory** (UC-2, UC-4)
- Development of a mood classification system based on text messages. - **Mandatory** (UC-4, UC-6)
- Implementation of a user search method in the developed Android application. - **Desirable** (UC-1, UC-6, UC-7)
- Evaluate and compare the results obtained with those recorded by the user manually. - **Optional** (UC-1, UC-2, UC-4)

3.1.3 UC1: Register daily information manually

Use Case Name	Register daily information manually
Use Case ID	UC-1
Actors	User, Server, EmoUPM
Pre-Condition	The user must be registered and authenticated in the application
Post-Condition	The user must have Internet connection
Normal Flow	**1.** The user introduce the data and press the upload data button. **2.** The mobile device has Internet connection, so the data is sent to the Firebase server. **3.** Once the data is on the Firebase server, it is stored based on a unique user ID.
Alternative Flow	**1.** The user introduce the data and press the upload data button. **2.** The device has no connection, so it saves the data locally. **3.** The device has connection and sends the data stored locally to the Firebase server. **4.** Once the data is on the Firebase server, it is stored based on a unique user ID.

Table 3.2: UC-1. Register daily information manually

3.1.4 UC2: Register data from instant messaging apps

Use Case Name	Register data from instant messaging apps
Use Case ID	UC-2
Actors	User, KeyLogger, Server
Pre-Condition	Go to system settings and give permissions to the keylogger
Post-Condition	The user must have Internet connection
Normal Flow	**1.** Each time the user start an application, the keylogger API check if it is a messaging app. If so, the keylogger is activated. **2.** Every time the user press the send key, in addition to the message being sent to the recipient, the message is sent to the Firebase server. **3.** Once the message is on the Firebase server, it is stored based on an unique user ID.
Alternative Flow	**1.** The user does not give permissions to EmoUPM. **2.** Text messages are not recorded.

Table 3.3: UC-2. Register data from instant messaging apps

3.1.5 UC3: User Authentication with EmoUPM

Use Case Name	User Authentication with EmoUPM
Use Case ID	UC-3
Actors	User, Server, EmoUPM
Pre-Condition	-
Post-Condition	-
Normal Flow	**1.** The user starts the application and enters the credentials. **2.** EmoUPM sends the credentials to the Firebase server for authentication. **3.** The credentials are correct, the user accesses to the main menu of the application.
Alternative Flow	**1.** The user starts the application and enters the credentials. **2.** EmoUPM sends the credentials to the Firebase server for authentication. **3.** The credentials are not correct and the app print an authentication failed message.

Table 3.4: UC-3. User Authentication with EmoUPM.

3.1.6 UC4: User Registration to EmoUPM

Use Case Name	User Registration to EmoUPM
Use Case ID	UC-4
Actors	User, Server, EmoUPM
Pre-Condition	-
Post-Condition	The mobile device needs Internet connection.
Normal Flow	**1.** The user starts the application. **2.** This is the first time the user open the app, so he press the sign up button. **3.** The user introduces the data and presses the sign up button. **4.** EmoUPM sends the data to the Firebase server to be stored in the Authn. database. **5.** The user is redirected to the login screen.
Alternative Flow	**1.** The user starts the application. **2.** This is the first time the user open the app, so he press the sign up button. **3.** The user introduces the data and presses the sign up button. **4.** The email is already registered and the user gets a registration error.

Table 3.5: UC-4. User Registration to EmoUPM.

3.1.7 UC5: Make a mood prediction based on text message

Use Case Name	Make a mood prediction based on text message.
Use Case ID	UC-5
Actors	User, External User, KeyLogger, Server, EmoUPM, MoodDetector
Pre-Condition	- The prediction model must be trained. - The mobile device needs Internet connection. - EmoUPM needs permissions.
Post-Condition	-
Normal Flow	**1.** The user writes text messages. **2.** The messages are stored on the Firebase server. **3.** An external user extracts the messages from the server. **4.** Text messages are scanned by the Mood Detector to predict the mood.
Alternative Flow	**1.** The user writes text messages. **2.** The messages are stored on the Firebase server. **3.** An external user extracts the messages from the server. **4.** The model is not trained and the prediction fails.

Table 3.6: UC-5. Mood prediction based on text messages.

3.1.8 UC6: Register mood predictions on the server

Use Case Name	Register mood predictions on the server
Use Case ID	UC-6
Actors	External User, Mood detector, Server
Pre-Condition	There must be text messages in the server database.
Post-Condition	The user must have Internet connection
Normal Flow	**1.** The external user calls the mood predictor function. **2.** The external user gets the mood based on the messages. **3.** The external user saves the results in the database.
Alternative Flow	**1.** The external user calls the mood predictor function. **2.** The external user gets null because there are no messages to be analyzed yet.

Table 3.7: UC-6. Register mood predictions on the server.

3.1.9 UC7: See mood history

Use Case Name	See mood history
Use Case ID	UC-7
Actors	User, EmoUPM, Server
Pre-Condition	There must be text messages in the server database and the user must have Internet connection
Post-Condition	-
Normal Flow	**1.** The user starts the application and enters the credentials. **2.** The user accesses to the main page of the application and press the View Mood button. **3.** The user's mood history is printed.
Alternative Flow	**1.** The user starts the application and enters the credentials. **2.** The user accesses to the main page of the application and press the view mood button. **3.** There is no user history and an empty history message is printed.

Table 3.8: UC-7. See mood history.

3.2 Conclusions

In this chapter, we have made a presentation of the actors involved in this project. As we can see in section 3.1.1, the system consists of two primary actors (User and External User). In addition, the system consists of several supporting actors, such as the Server, the Keylogger... On the other hand, we have also presented the use cases of the system (section 3.1.2 onwards). Section 3.1.2 shows five use cases, and then details each of them. Finally, we presented a list of user requirements, relating each requirement to the use cases and assigning it a label (mandatory, desirable or optional).

CHAPTER 4

Machine learning model building and Evaluation

This chapter presents the methodology used in this work. It describes the overall architecture of the project, with the connections between the different components involved on the development of the project. In order to do this, the architecture and methodology will be divided into two main parts, just like the whole project. On the one hand, we will explain the architecture and methodology of the Android application, which has been used for information gathering. On the other hand, the architecture and methodology related to sentiment analysis will be explained.

4.1 Overview

In this section, we will present the system architecture related to ML. We will present the different parts that make up the system whose objective is to classify text messages with a certain mood. In Figure 4.1, we can see a flow diagram of the whole project. In the following lines, we will go deeper into the parts related with ML, that is, from the beginning of the flow diagram to where it is marked as “EmoUPM”.

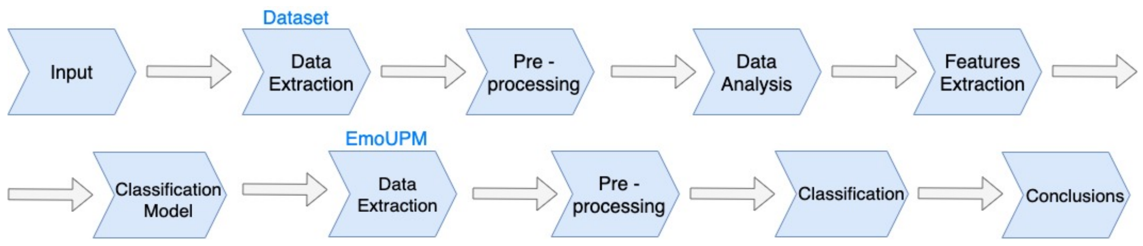


Figure 4.1: Mood Predictor phases

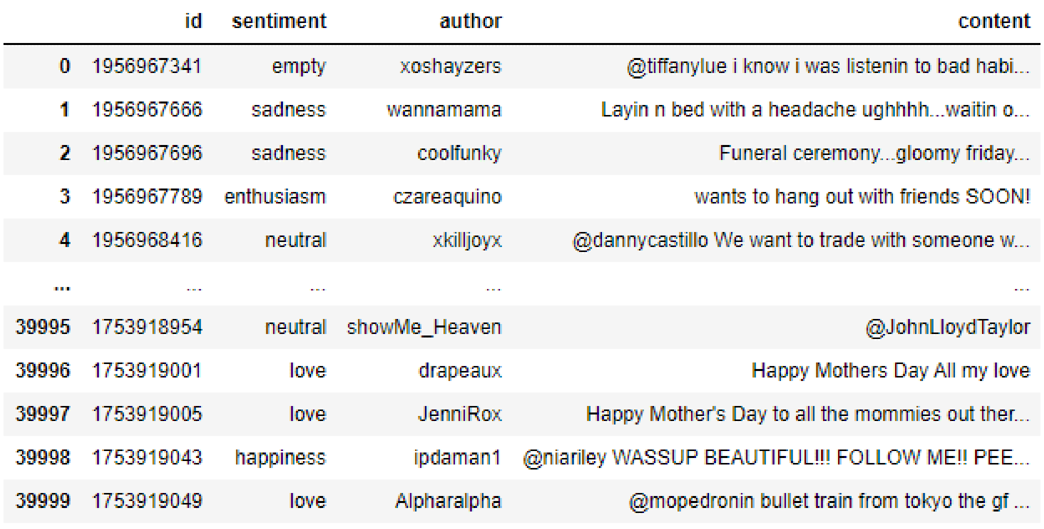
First, for the data extraction part, we have used a corpus created by a company called *Figure Eight Inc.* [10]. Once we have the corpus that we are going to use, we need to preprocess the data, to keep only the necessary information. Then, we go on to analyze the data and extract the features, testing with different techniques to obtain the best possible results. Finally, we will use different classification models with the aim of predicting the mood.

From here, the part involving the EmoUPM Android application begins. Once we have the classification model selected and trained, we can start extracting messages from the Firebase database. Once we have extracted the messages, we preprocess them and use the model to predict the user’s mood. Once this is done, we can get the relevant conclusions from our user. This part is explained in more detail below.

4.2 Data Extraction

The dataset given by *Figure Eight Inc.* is a Coma Separated Value (CSV) file formed by 40,000 rows and four columns. The first column indicates the ID, the second column indicates the mood (there are thirteen possible moods), the third column corresponds to the author of the message and the fourth column corresponds to the text message. Of the four columns that make up the dataset, the only ones that we will use in our project will be the one that indicates the mood and the column that shows the text message. In Figure 4.2 we can see how the dataset looks initially.

As we said before, in the sentiment column, we can find thirteen possible moods: empty,



	id	sentiment	author	content
0	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
4	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...
...	...	...	...	...
39995	1753918954	neutral	showMe_Heaven	@JohnLloydTaylor
39996	1753919001	love	drapeaux	Happy Mothers Day All my love
39997	1753919005	love	JenniRox	Happy Mother's Day to all the mommies out ther...
39998	1753919043	happiness	ipdaman1	@niariley WASSUP BEAUTIFUL!!! FOLLOW ME!! PEE...
39999	1753919049	love	Alpharalpha	@mopedronin bullet train from tokyo the gf ...

Figure 4.2: Dataset from *Figure Eight Inc.* [10]

sadness, enthusiasm, neutral, love, happiness, worry, surprise, fun, hate, boredom, anger and relief.

4.3 Data Analysis

In this section, we will make an analysis of the data that make up the dataset, as well as a study of how these data are distributed. This can be of great help when extracting features in the future.

First, we will perform an analysis of the data before preprocessing, and then we will do the same with the preprocessed data. As we explained before, the corpus is initially composed of 40,000 rows of data. In Figure 4.3, we can see how the data are distributed. As we can see, the distribution is not balanced. In addition, we can see in this plot that the thirteen bars are grouped in six colors. We will talk later about these groups.

Also, we are going to present some statistical data (Table 4.1), such as the mean, the standard deviation, the maximum or the minimum values.

As explained above, to simplify the classification as much as possible, we have grouped the thirteen moods into five groups. Of the 40,000 messages we had before preprocessing, we were left with 31,362 messages (the remaining 8,638 messages correspond to messages whose mood is neutral).

Next, in Figure 4.4, we can see how the distribution of the grouped moods would look like.

As presented above, we will also present some statistical parameters that may be useful. In Table 4.2, we can see some parameters such as the mean, the standard deviation, the

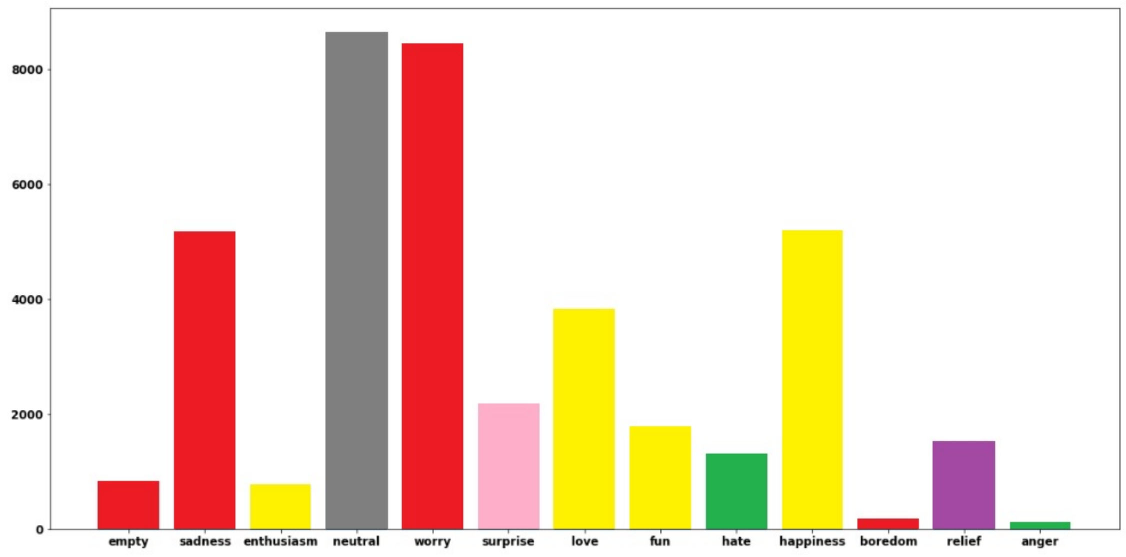


Figure 4.3: Original distribution.

Measure	**Value**
Mean	3,076.92
Std	2,841.79
Max	8638 (Neutral)
Min	110 (Anger)

Table 4.1: Initial dataset statistics

maximum or the minimum.

However, although the five groups are not balanced, they are more or less balanced in relation to their mood. That is, happy-sad and relaxed-surprised. This can be seen more clearly in Figure 4.5.

Based on the corpus created by *Figure Eight Inc.*, we have created five new datasets. We have done this, with the main goal of balancing each of the five groups. Below, in Table 4.3, we can see how each of the five datasets are structured.

On the other hand, the training process has been carried out using a part of each dataset. In this way, the part of the dataset that we have used to validate our classifier has not been used until the moment of validation. For the training process, 80% of each dataset has been used. The remaining 20% of each dataset has been used to validate each classifier. In Table 4.4, we can see the number of messages used for the training process and the number

Measure	**Happy**	**Angry**	**Relief**	**Surprise**	**Sad**
Mean	2,896.5	716.5	1,526	2,187	3,657.5
Std	1,736.81	606.5	0	0	3,370.41
Max	5,209 (Happiness)	1,323 (Hate)	1,526 (Relief)	2,187 (Surprise)	8,459 (Worry)
Min	759 (Enthusiasm)	110 (Anger)	1,526 (Relief)	2,187 (Surprise)	179 (Boredom)
Total	11,586	1,433	1,526	2,187	14,630

Table 4.2: 5 moods dataset statistics

		Mood				
		Happy	Angry	Relief	Surprise	Sad
Dataset	Happy	1,000	250	250	250	250
	Angry	250	1,000	250	250	250
	Relief	250	250	1,000	250	250
	Surprise	250	250	250	1,000	250
	Sad	250	250	250	250	1,000

Table 4.3: 5 datasets distribution

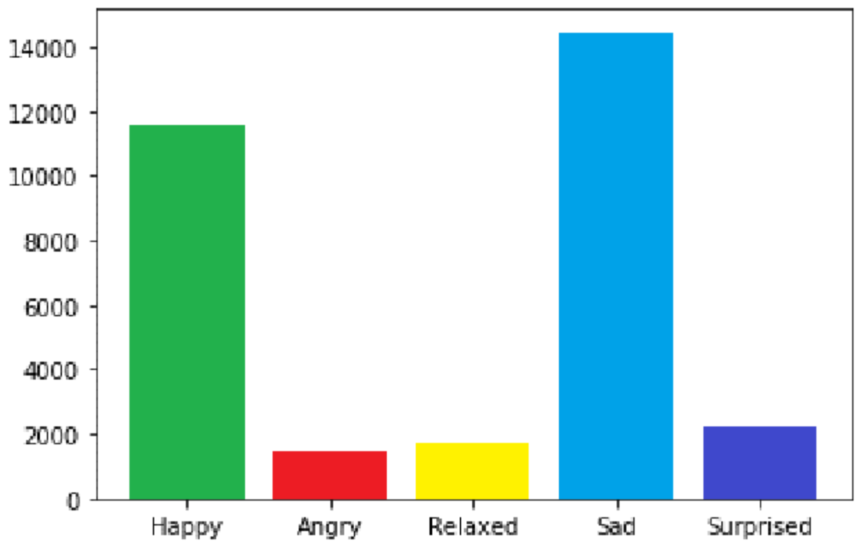


Figure 4.4: Five Moods distribution

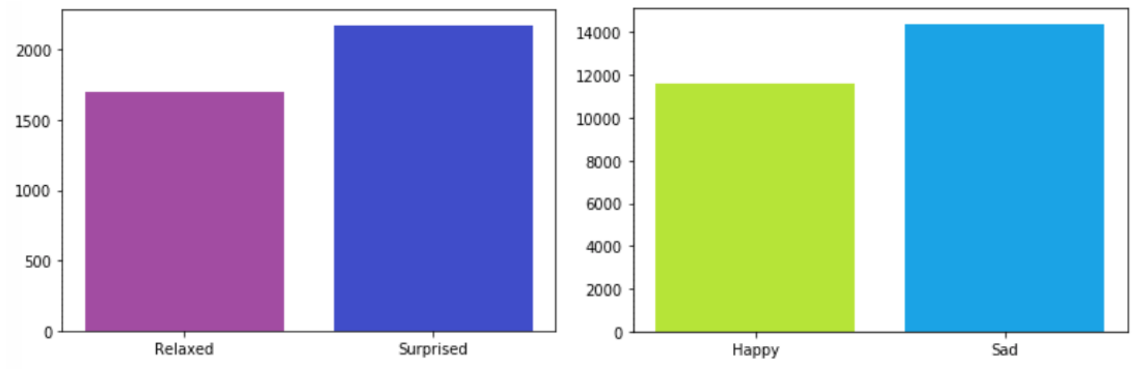


Figure 4.5: Each Mood type distribution

of messages used for the validation process.

Set	Messages
Train	1600
Test	400

Table 4.4: Training and test distribution

4.4 Preprocessing

In previous sections, we explained how we obtained the necessary data to start working with our model and how the data is structured. Once we have this data, it is necessary to modify it to facilitate the work of our model. That is, it is necessary to preprocess the data to eliminate unnecessary content. This preprocessing is done in the column containing the text messages. Within this preprocessing, we include the removal of stop words, spell check, removal of punctuation marks, etc. For this last case, the elimination of punctuation marks, will be different according to our objectives in ML. For example, in our case, where we are going to predict moods, we are interested in not eliminating some punctuation marks, as they can be decisive when classifying a text message. This whole process is carried out thanks to the toolkit provided by NLTK, which has already been explained above.

On the other hand, we have the mood column for each message. As we explained in the previous section, initially we have thirteen possible options to classify a message. These thirteen possibilities (Figure 4.3), we have grouped them in five (Figure 4.4), in order to reduce the number of moods to work with. We have made this grouping based on the circumplex model of the mood, explained in Chapter 2. This means that we have grouped the moods according to the arousal and the valence. Next, we can see how the moods have been grouped.

- **Happy:** Enthusiasm, love, happiness, fun.
- **Angry:** Hate, anger.
- **Relief:** Relief.
- **Surprise:** Surprise.
- **Sad:** Sadness, empty, worry, boredom.

As we can see from the above list, the neutral mood is not found in any of the five groups. This is because we decided to remove all messages corresponding to this mood, as they caused a lot of confusion in our model [12].

The group called Happy is made up of four moods (enthusiasm, love, happiness and fun). We have decided to group these moods because all of them have a positive valence and a slightly positive arousal (section 2.1.2). The group called Angry is formed by hate and anger moods. Both moods have negative valence and high arousal. Relief has not been grouped with any other moods, as it is the only mood that has positive valence and low arousal. We have done the same for the surprise mood, which has high arousal and slightly positive valence. Finally, the group called Sad consists of four moods (sadness, empty, worry and boredom). These four moods are found in the third quadrant of the circumplex model (negative valence and low arousal).

4.5 Feature Extraction

In this section, we will explain the features that we have extracted, to later, train our model and make predictions. Therefore, this part of the process is fundamental. It is possible to extract a large number of features, some of them with a greater interest than others, depending on the purpose of our work. Our goal is to extract the features that help us to get the highest performance in our model.

Below, we will present some of the main features we have extracted.

4.5.1 Part Of Speech (POS)

Part of Speech (POS) tagging [20] is a process of classifying words according to their grammatical category. This method of tagging also takes into account the previous word, the next word, checks if the first letter of the word is a capital letter, etc. The words can be classified according to eight possibilities: Noun, Verb, Adjective, Adverb, Preposition, Conjunction, Pronoun, Interjection. Although there are eight main categories, there are subcategories within each category. NLTK provides the necessary functionality to carry out this process. This is carried out after tokenization. An example of this function is *NLTK.POS_TAG("My name is Jacob")*. In Table 4.5 we can see the table with the tags, their meaning and examples.

4.5.2 TF-IDF

Term Frecuency - Inverse Document Frecuency (TF-IDF) is a statistical measure used to find out the relative importance of a word in a document or set of documents. This mechanism uses two types of measurement.

The first measure simply indicates the number of times a word is repeated in a document. This is known as "Term Frequency".

Tag	**Meaning**	**Examples**
NOUN	noun	table, dog, teacher, pen, city
VERB	verb	run, eat, play, live, walk
ADJ	adjective	happy, green, young, fun
ADV	adverb	never, too, well, tomorrow
PREP	preposition	on, in, from, with, near, at
CONJ	conjunction	and, or, but, because, so, yet
PRON	pronoun	I, you, we, they, he, she, it, me
INTER	interjection	Ouch! Wow! Great! Hi! Oh!

Table 4.5: POS tagging

The second measure is achieved by taking the total number of documents and dividing it by the number of documents that contain that word. To that result, the logarithm is applied. This is known as "Inverse Document Frequency". Therefore, if a word is repeated many times and also appears in many documents, its score will be close to "0". Otherwise, it will be close to "1" [7]. NLTK provides the necessary functionality to carry out this process. An example of this function would be *TfidfVectorizer()*.

4.5.3 N-GRAMS

N-gram is the process of, given a text, grouping together a set of contiguous words. 2-gram ("bigram") would be the simplest n-gram option. A simple example of a bigram would be the following: "My name", "name is", "is Jacob". In our project, we used bigrams (N = 2) and trigrams (N = 3).

The implementation of this process is the combination of *CountVecorizer* and *TF-IDF*. In other words, the data is processed by the CountVecorizer. A vectors matrix is obtained that tells us the number of times the N-gram appears. Then, TF-IDF is applied to it, obtaining a TF-IDF matrix [38].

4.5.4 Hashing

The hashing feature is a very fast and space efficient feature. Its internal operation is based on N-grams, feature explained above. This feature "lets you represent text documents of

Text	**Opinion**
I liked this bike	3
I hated this bike	1
This bike was amazing	3
I like bikes	2

Table 4.6: Text with opinion rates

Bigrams	**Frequency**
This bike	3
I liked	1
I hated	1
I like	1

Table 4.7: Bigrams with frequency rates

variable length as numeric feature vectors of equal length to reduce dimensionality" [28].

The operation of this feature is to receive as input a text and assign each sentence a punctuation. We will now explain this feature works through examples. In Table 4.6, we can see some phrases with his respective opinion rate.

Inside, what this feature does is create a dictionary of N-grams. The value of N can be set manually. In the following example, we are going to use bigrams, so the feature will create a dictionary for bigrams (Table 4.7) and another dictionary for unigrams (Table 4.8).

The next step, after creating the N-grams dictionaries, is to hash the dictionaries. Then, it check whether a feature has been used in each case. For each row, several columns of hash characteristics are generated (Table 4.9).

- If the value is 0, it means that the row does not contain that feature.
- If the value is 1, it means that the row does contain that feature.

Unigrams	**Frequency**
bike	3
I	3
bikes	1
was	1

Table 4.8: Unigrams with frequency rates

Rating	**Hashing feature 1**	**Hashing feature 2**	**Hashing feature 3**
4	1	1	0
5	0	0	0

Table 4.9: Hash features tables with rates

4.6 Classification and Evaluation

Once we have extracted the features, we can begin the classification and evaluation phase. This part consists of putting into practice everything explained in the previous sections. We will start by explaining the different models we have used to train and test our data and, then, we will explain the results obtained.

This phase can be carried out by following two paths. On the one hand, we can make a classification model that takes into account all possible moods. On the other hand, we can create multiple binary classification models, one for each mood. With the second option, we usually get a higher success rate, since each model only has predict between two options.

To begin this part of the project, it is necessary to separate the data into two parts. One part will be used to train the model, while the other part will be used to test the model once it is trained. These two parts usually do not have the same dimension. The part with which the models are trained is usually about 80% of the size of the dataset (this percentage is set manually by the user).

The Scikit-Learn library provides us with a large number of models with which we can make different classifications, passing each algorithm different parameters. Next, we're going to explain some of the algorithms we've been working with.

- **Multinominal Naive Bayes:** It is a probabilistic classifier based on Bayes' theorem and some additional hypotheses [48].

- **Stochastic Gradient Descent (SGD):** SGDClassifier is a linear classification model implemented through the use of regularized linear models and SGD [36].
- **Logistic Regression:** It is a statistical form of regression analysis that is used to predict a categorical variable based on independent variables [34].
- **Support Vector Machines (SVM):** This is a supervised learning method that uses a multiple algorithms for classification, regression and outliers detection. It is provided by Scikit-Learn under the name SVC [37].
- **Bernoulli Naive Bayes:** This is a method based on a Naive Bayes classifier with a Bernoulli event.
- **Ridge Regression:** This is a variation on Linear Regression. It is provided by Scikit-Learn under the name RidgeClassifier [25].
- **Passive Agressive Classifier:** These are a family of algorithms for large-scale learning, that is, learn from massive streams of data [43].
- **Random Forest:** In this method, multiple trees are created and each tree generates a classification. The final result will be the classification with more votes [21].
- **Extra-Tree:** Also known as “Extremely Randomized Trees”. This algorithm is really similar to Random Forest.
- **K-Nearest Neighbors (K-NN):** It is a supervised classification method, which is based on the Probability Density Function (PDF). This algorithm can become extremely slow as the size of the data increases. It is provided by Scikit-Learn under the name KNeighborsClassifier [47].

4.6.1 Evaluation

In this section of the chapter we will show the results obtained once we have trained our models and made the predictions. Once the results are presented, we will make a comparison, in which we will be able to check which models are better, based on objective parameters such as confusion matrix, accuracy, precision, etc. Below, we will show the parameters we have used to measure the quality of our models.

- **Confusion Matrix:** This is a “table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known” [33].

  - **True Positive (TP):** This is when the prediction is “yes” and the correct result is also “yes”.
  - **True Negative (TN):** This is when the prediction is “no” and the correct result is also “no”.
  - **False Positive (FP):** This is when the prediction is “yes” and the correct result is “no”.
  - **False Negative (FN):** This is when the prediction is “no” and the correct result is “yes”.

- **Precision:** This is a measure of how close two or more values are to each other. That is, how scattered the values are.

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

- **Accuracy:** This measure indicates how close the values obtained are compared to the real values.

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

- **Recall:** This measure is calculated by dividing the number of true positives by the sum of true positives and false negatives. This measure is used less than precision or accuracy.

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

- **F1 Score:** It is a measurement based on precision and recall.

$$F1Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Once we have presented the parameters that we will use to measure the quality of the results obtained, it is time to present these results. As we have already mentioned, we are going to make multiple binary models to predict the moods. This way, one model will make “Happy-Not Happy” predictions, another model will make “Surprise-Not Surprised” predictions, etc.

We have done this by creating a different dataset for each predictive model. Next, we are going to present the dataset (from now on "Happy dataset") we have used to make "Happy-Not Happy" predictions. The other datasets will be equivalent.

The "Happy dataset" is a dataset that is balanced. This means that it consists of 1000 rows of messages whose mood is "happy", 250 rows whose mood is "angry", 250 rows whose mood is "relaxed", 250 rows whose mood is "sad" and 250 rows whose mood is "surprised". Thus, half of the dataset will be made up of messages whose mood is "happy" and the other half will be made up of messages whose mood is "unhappy". In Figure 4.6, we can see what was explained in the previous lines in a more graphic way.

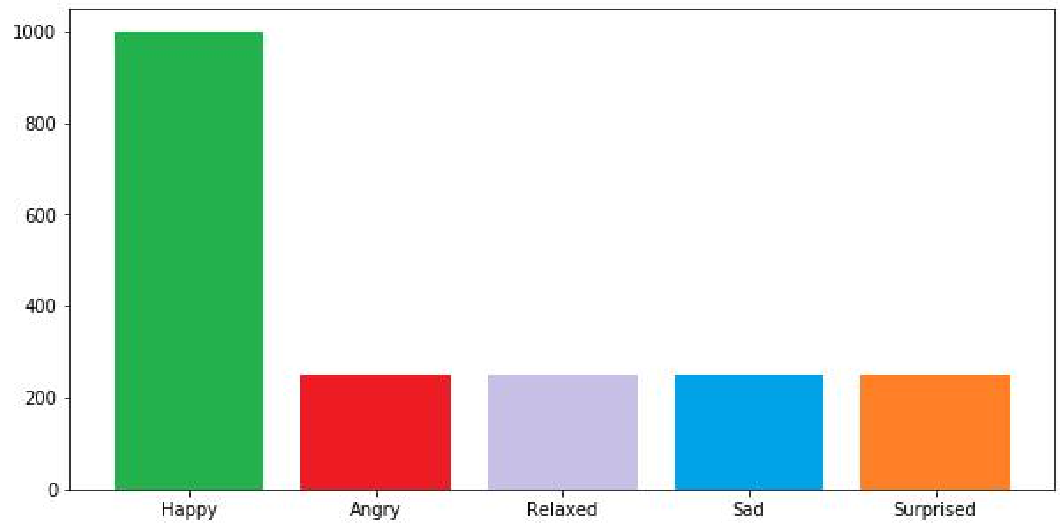


Figure 4.6: Happy dataset Structure

For the Happy-Unhappy classifier, we have started applying CountVectorizer as the only feature. The result of using this feature is to convert a collection of text to a matrix of token counts. The results are shown in Table 4.10.

In Table 4.10, we can see the best results written in green. The best result was obtained using BernoulliNB's model. An accuracy of 0.6875 has been obtained.

Next, we are going to do the same but applying the TF-IDF feature instead of the CountVectorizer. These results are shown in Table 4.11. In this case, the best results have been obtained using the LogisticRegression's model. An accuracy of 0.69 has been obtained, slightly higher than that obtained using the CL-1 classifier.

Once the predictions using the TF-IDF feature are finished, we have made the same predictions but adding the POS tagging feature and the bigrams feature, both explained in previous chapters. These results are shown in Table 4.12. As expected, the probability of success have increased. Using this classifier, the best results have been obtained using the RidgeClassifier model. An accuracy of 0.71 has been obtained.

ID	Features	Model	Precision	Accuracy	Recall	F1 Score
CL-1	CountVectorizer	MultinominalNB	0.5648	0.645	0.6489	0.6453
		SGDClassifier	0.5492	0.625	0.6436	0,6253
		LogisticRegression	**0.5836**	**0.6675**	**0.6755**	**0.6678**
		SVC	0.47	0.47	1	0.3005
		BernoulliNB	**0.6018**	**0.6875**	**0.6702**	**0.6875**
		LinearSVC	0.5731	0.655	0.6596	0.6553
		RidgeClassifier	0.5710	0.6525	0.6596	0.6528
		PassiveAgressive	0.5377	0.61	0.6064	0.6103
		ExtraTree	**0.5838**	**0.6675**	**0.6596**	**0.6677**
		DecisionTree	0.5608	0.64	0.633	0.6402
		RandomForest	0.4977	0.5275	0.9681	0.4338
		KNeighbors	0.5217	0.5925	0.3032	0.5582
		NuSVC	0.5669	0.6475	0.6649	0.6478

Table 4.10: Results for CL-1

After several tests using the TF-IDF feature, we decided to change the feature to check the effectiveness of the Hashing feature. Below, in Table 4.13, we show the results obtained using the Hashing feature. If we compare this classifier with CL-1 and CL-2, this is the classifier that provides the worst results, as the highest accuracy obtained is 0.675, using the RidgeClassifier model.

As in the case of TF-IDF, we have also combined different features. In this case, we have combined the Hashing feature with POS tagging and Bigrams. In Table 4.14, we show the results obtained. Once we have combined some features, the results are much better. In this case, the best results are achieved using LinearSVC and RidgeClassifier, with an accuracy of 0.705.

Once we have finished with the “Happy-Not Happy” classifier, we proceed to show the results obtained after performing the same for the other four classifiers: “Angry-Not Angry”, “Relief-Not Relief”, “Surprised-Not Surprised” and “Sad-Not Sad”. In Table 4.15, we can see the results obtained.

As we see in Table 4.15, some results are still a little low. To improve these percentages,

ID	Features	Model	Precision	Accuracy	Recall	F1 Score
CL-2	TF-IDF	MultinominalNB	0.5455	0.62	0.6436	0.6202
		SGDClassifier	0.5710	0.6525	0.6968	0,6524
		LogisticRegression	**0.6031**	**0.69**	**0.7181**	**0.6902**
		SVC	0.47	0.47	1	0.3005
		BernoulliNB	0.5794	0.6625	0.6649	0.6628
		LinearSVC	0.5772	0.66	0.6915	0.6602
		RidgeClassifier	**0.5941**	**0.68**	**0.7234**	**0.68**
		PassiveAgressive	0.5441	0.6175	0.6702	0.6171
		ExtraTree	**0.5975**	**0.6825**	**0.6489**	**0.6822**
		DecisionTree	0.5498	0.625	0.6808	0.6246
		RandomForest	0.4893	0.5125	0.9468	0.4186
		KNeighbors	0.5522	0.6275	0.3936	0.6072
		NuSVC	0.5611	0.64	0.6808	0.64

Table 4.11: Results for CL-2

we will train the model without using the five mood groupings. For the "Happy-Not Happy" classifier, we are going to use only messages whose mood is happiness (we are not going to use either enthusiastic, love or fun). For the "Angry - Not Angry" classifier, we will use only messages whose mood is hate. Finally, for the "Sad - Not Sad" classifier, we will use only messages whose mood is sadness. For the "Relief - Not Relief" and "Surprise - Not Surprise" classifiers, we have already done it. In Table 4.17 we can see the results improve significantly when we do not use the mood groupings. The "Happy - Not Happy" classifier is the least improved compared to Table 4.15 (about 2%). "Angry - Not Angry" classifier has improved by almost 4%. Finally, the "Sad - Not Sad" classifier has improved the most (more than 6%), even though it has the worst percentages (it has 68% accuracy).

ID	Features	Model	Precision	Accuracy	Recall	F1 Score
CL-3	TF-IDF POS Bigrams	MultinominalNB	0.5573	0.635	0.6915	0.6346
		SGDClassifier	0.5984	0.685	0.7287	0.685
		LogisticRegression	**0.6046**	**0.6925**	**0.7659**	**0.6917**
		SVC	0.47	0.47	1	0.3005
		BernoulliNB	0.6099	0.6925	0.5904	0.6892
		LinearSVC	**0.6113**	**0.7**	**0.7713**	**0.6994**
		RidgeClassifier	**0.6206**	**0.71**	**0.7659**	**0.7098**
		PassiveAgressive	0.5962	0.6825	0.7287	0.6824
		ExtraTree	0.5986	0.6825	0.6170	0.6811
		DecisionTree	0.5450	0.6175	0.7181	0.6149
		RandomForest	0.4896	0.5125	0.9574	0.4130
		KNeighbors	0.5431	0.6175	0.3776	0.5957
		NuSVC	0.5950	0.6775	0.5798	0.6743

Table 4.12: Results for CL-3

ID	Features	Model	Precision	Accuracy	Recall	F1 Score
CL-4	Hashing	MultinominalNB	-	-	-	-
		SGDClassifier	0.5835	0.6675	0.6862	0.6677
		LogisticRegression	**0.5836**	**0.6675**	**0.6808**	**0.6678**
		SVC	0.47	0.47	1	0.3005
		BernoulliNB	0.47	0.47	1	0.3005
		LinearSVC	0.5814	0.665	0.6808	0.6653
		RidgeClassifier	**0.59**	**0.675**	**0.6915**	**0.6753**
		PassiveAgressive	0.5570	0.635	0.6596	0.6352
		ExtraTree	**0.5848**	**0.6675**	**0.5957**	**0.6658**
		DecisionTree	0.5813	0.665	0.7447	0.6639
		RandomForest	0.4934	0.5175	1	0.3978
		KNeighbors	0.5681	0.6425	0.3883	0.6191
		NuSVC	0.5586	0.6225	0.25	0.5661

Table 4.13: Results for CL-4

ID	Features	Model	Precision	Accuracy	Recall	F1 Score
CL-5	Hashing POS Bigrams	MultinominalNB	-	-	-	-
		SGDClassifier	0.6038	0.69	0.6862	0.6902
		LogisticRegression	**0.6108**	**0.6975**	**0.6915**	**0.6977**
		SVC	0.47	0.47	1	0.3005
		BernoulliNB	0.47	0.47	1	0.3005
		LinearSVC	**0.6174**	**0.705**	**0.7128**	**0.7052**
		RidgeClassifier	**0.6177**	**0.705**	**0.7021**	**0.7052**
		PassiveAgressive	**0.6102**	**0.6975**	**0.7128**	**0.6978**
		ExtraTree	0.6037	0.685	0.5585	0.6798
		DecisionTree	0.5690	0.65	0.6223	0.6499
		RandomForest	0.4722	0.475	0.9947	0.3157
		KNeighbors	0.5536	0.6275	0.3617	0.6009
		NuSVC	0.5420	0.6075	0.2234	0.5450

Table 4.14: Results for CL-5

Classifier	**Features**	**Model**	**Precision**	**Accuracy**	**F1 Score**
Angry	TF-IDF	MultinominalNB	0.5971	0.6825	0.6826
	POS	**SGD**	**0.6024**	**0.6875**	**0.6871**
	Bigrams	PassiveAgressive	0.5957	0.68	0.6793
Relief	Hashing	**LogisticRegression**	**0.6096**	**0.695**	**0.6945**
		Ridge	0.5843	0.6675	0.6669
		ExtraTree	0.5735	0.655	0.6536
Surprised	TF-IDF	**LogisticRegression**	**0.53**	**0.6**	**0.6**
		Ridge	0.5207	0.5875	0.5869
		DecisionTree	0.5173	0.5825	0.5819
Sad	Hashing	**LogisticRegression**	**0.5463**	**0.6225**	**0.6199**
		Ridge	0.5229	0.59	0.59
		LinearSVC	0.5176	0.5825	0.5824

Table 4.15: Results four classifiers

Classifier	Features	Model	Precision	Accuracy	F1 Score
Happy	Hashing	**ExtraTree**	**0.6383**	**0.7225**	**0.7213**
	POS	Ridge	0.6254	0.715	0.7149
	Bigrams	LogisticRegression	0.6182	0.7075	0.7071
Angry	TF-IDF	**SGD**	**0.6352**	**0.725**	**0.7251**
	POS	PassiveAgressive	0.6238	0.7125	0.7127
	Bigrams	LinearSVC	0.6219	0.71	0.71
Sad	Hashing	**LogisticRegression**	**0.5946**	**0.68**	**0.6802**
		Ridge	0.5751	0.6575	0.6575
		LinearSVC	0.5649	0.645	0.6452
Relief	Hashing	**LogisticRegression**	**0.6096**	**0.695**	**0.6945**
		Ridge	0.5843	0.6675	0.6669
		ExtraTree	0.5735	0.655	0.6536
Surprised	TF-IDF	**LogisticRegression**	**0.53**	**0.6**	**0.6**
		Ridge	0.5207	0.5875	0.5869
		DecisionTree	0.5173	0.5825	0.5819

Table 4.16: Results classifiers without grouping

4.7 Conclusions

Once we have finished working with the models and classifiers, it is time to draw conclusions about this chapter. To do this, we are going to compare the results obtained after preprocessing (Table 4.17) and the results obtained if we had not done any kind of preprocessing, just balancing the data (Table 4.18).

Table 4.18 shows the results obtained for each classifier, without any preprocessing. As we can see in this table, the results in most cases are significantly worse. The happy classifier has improved by almost 10% in the case of using the ExtraTree algorithm. The angry and sad classifiers have also improved by almost 10%. The worst differences were achieved for the relief and surprised classifiers. For the first one, the improvement was less than 5%. For the surprised classifier the improvement was less than 1%.

We can conclude this chapter by stating that messages whose moods have a neutral valence (surprised or relief) are more difficult to classify and to improve their prediction. On the other hand, messages with a very positive or very negative valence (happy or sad) are much easier to classify and to improve their results.

Classifier	Features	Model	Precision	Accuracy	F1 Score
Happy - Not Happy	-	ExtraTree	0.5731	0.655	0.655
		Ridge	0.5840	0.6675	0.6673
		LogisticRegression	**0.5928**	**0.6775**	**0.6774**
Angry - Not Angry	-	SGD	0.5648	0.645	0.6451
		PassiveAgressive	0.5754	0.6575	0.6570
		LinearSVC	**0.5839**	**0.6675**	**0.6674**
Sad - Not Sad	-	**LogisticRegression**	**0.5314**	**0.6**	**0.5998**
		Ridge	0.5190	0.5825	0.5826
		LinearSVC	0.5227	0.5875	0.5874
Relief - Not Relief	-	**LogisticRegression**	**0.5730**	**0.655**	**0.6552**
		Ridge	0.5508	0.6275	0.6277
		ExtraTree	0.5351	0.6075	0.6069
Surprised - Not Surprised	-	**LogisticRegression**	**0.5239**	**0.59**	**0.5902**
		Ridge	0.4948	0.5375	0.5284
		DecisionTree	0.4997	0.5525	0.5527

Table 4.18: Results classifiers without preprocessing

CHAPTER 5

Architecture and System Operation

In this chapter, we will explain the structure and operation of the entire system. In addition, we will explain how we created the Android application, called EmoUPM.

5.1 Introduction

According to the Merck Sharp and Dohme Corporation (MSD) manual, mood disorders are "mental health disorders, consisting of prolonged periods of excessive sadness (depression), excessive excitement or euphoria (mania), or both" [54]. The causes of these disorders are multiple, but the remedies to cure or even prevent them are in many cases very similar. There are many scientific studies that confirm that daily physical activity and mood disorders are closely related.

This chapter describes a tool that can help experts when dealing with patients who have mood disorders or patients who are at risk for such disorders.

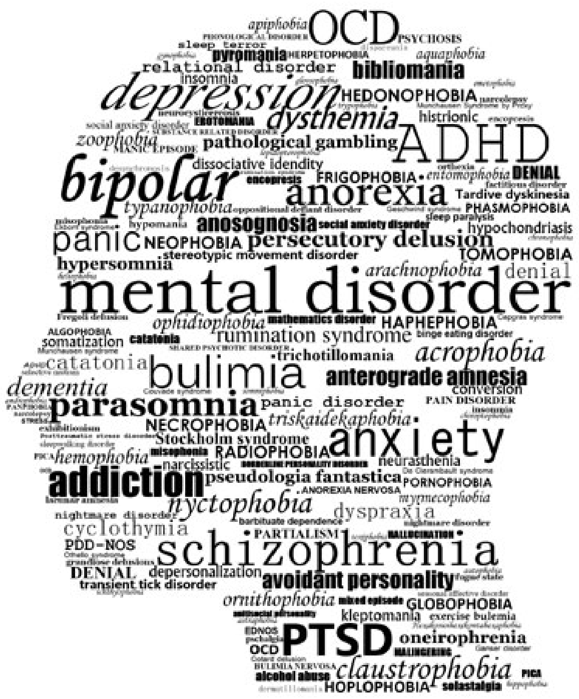


Figure 5.1: Mental disorders [46].

The tool consists of an Android mobile application (called *EmoUPM*), which performs two main functions. Firstly, it allows the manual recording of the user's mood and the daily physical activity performed so far. Secondly, it automatically records all text messages (prior authorization from the user is required). All these data, both those recorded manually and those recorded automatically, are sent to an external database. These data can be processed and studied by experts, with the aim of drawing beneficial conclusions for the patient.

In the following sections we will describe in detail how the Android application is structured and its possible functionalities.

5.2 Structure & Operation

In this section, we will explain the architecture of the Android application, *EmoUPM*, from a global perspective, including the parts that interact in the system.

From a global perspective, we can divide the system into three parts. The first part, would be formed by the *EmoUPM* application, the second part would be the Firebase server and the third part would be formed by the external agent. The complete global architecture of the system is shown in Figure 5.2.

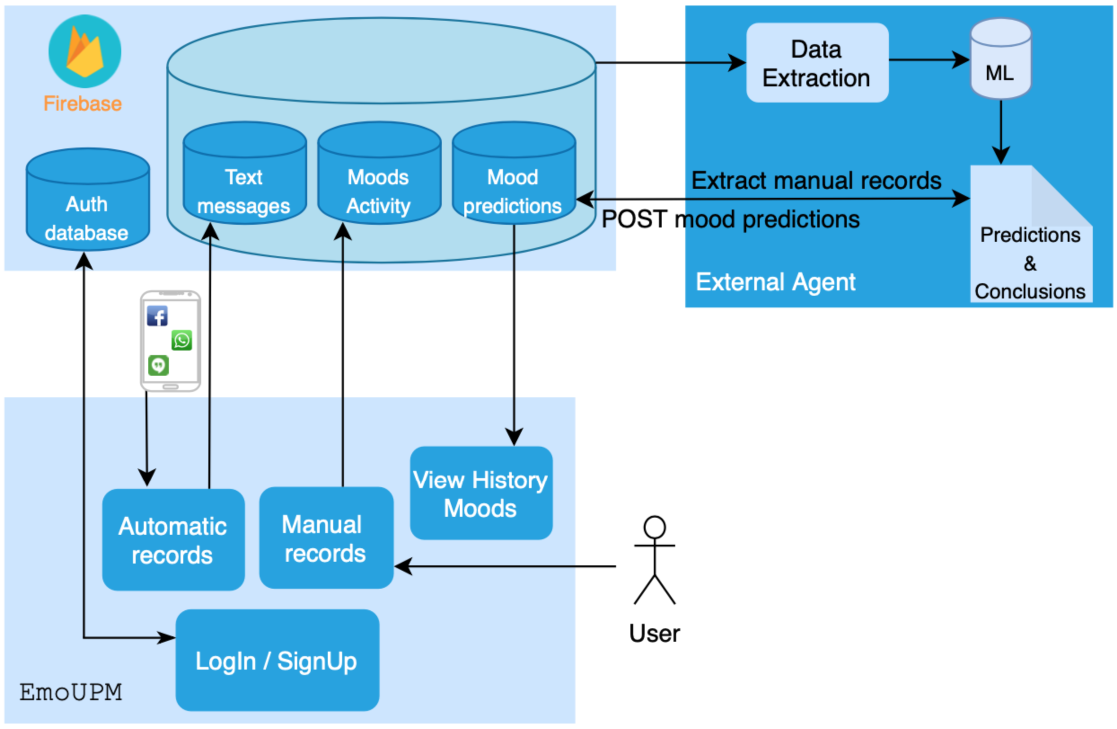


Figure 5.2: Global architecture of the system developed.

5.2.1 EmoUPM Module

EmoUPM is the Android application we have developed to monitor the user's mood. This application has seven screens and its diagram is shown in Figure 5.3.

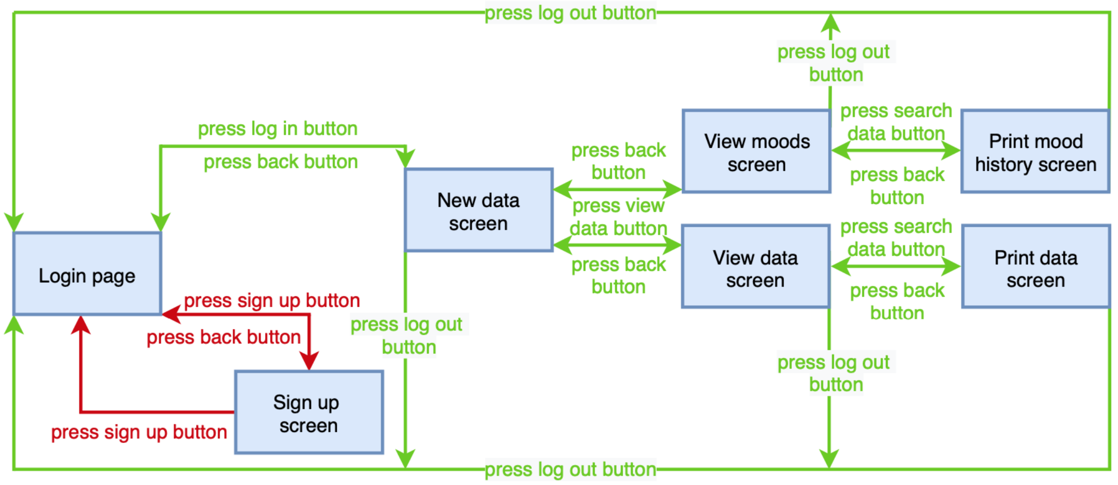


Figure 5.3: Screen flow diagram.

Also, *EmoUPM* performs five main functions. The first function, which takes place as soon as we run the application, is to login or register (Figure 5.4). If the user is already registered, they simply need to enter their credentials and, once they have authenticated themselves with the Firebase server, they will be allowed to access the application. On the contrary, if the user is not registered yet, he will have the option to register as a new user, introducing his personal data. Once the user is registered, the application will redirect him to the login screen.

The second function, which takes place as soon as you log in to the application, is to register your data manually (Figure 5.5). Recording the data consists of entering our mood and our daily physical activity performed so far. In the area of psychology, this is known as ESM, and is explained in section 2.1.1. The mood is registered based on two parameters: the level of pleasant and the level of activation (section 2.1.2). Regarding the level of pleasant, the user can choose between four possibilities (No response, Very Pleasante, Pleasante or Unpleasante). Regarding the level of activation, the user can choose between four possibilities (No response, Very Active, Active or Not Active). The recording of daily physical activity consists of entering the distance walked and ran so far and the sports performed. Once the parameters have been entered and the "upload data" button pressed, two situations may occur. If we have an Internet connection, the data will be sent to the Firebase server. On the other hand, if we do not have a Internet connection, the data will be stored locally and will be sent as soon as our device has connection. The manual

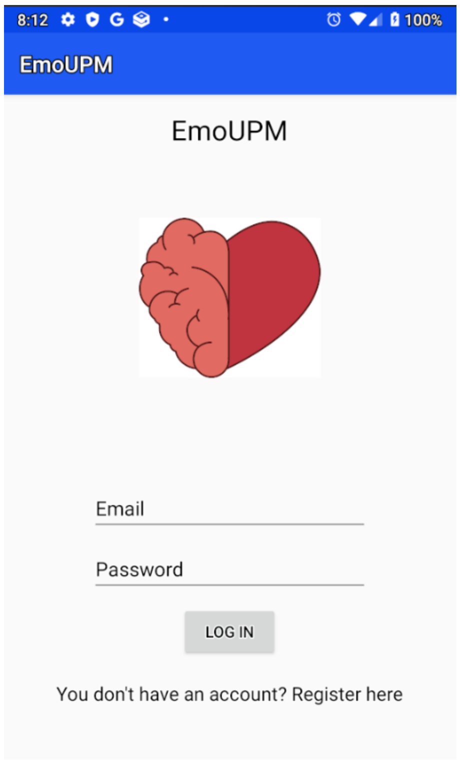


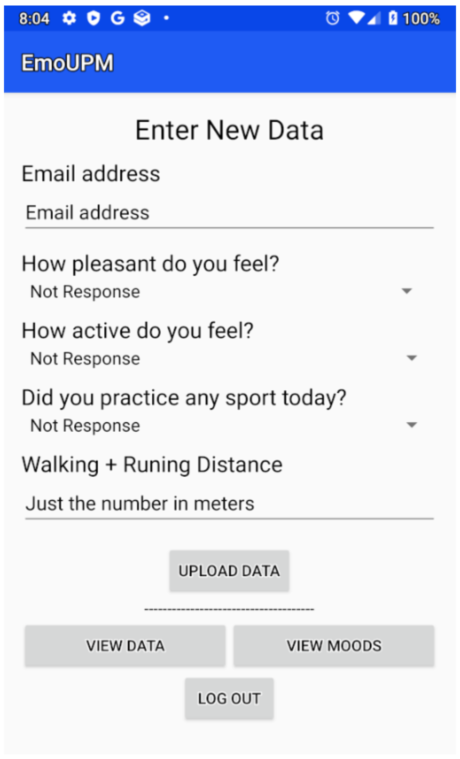
Figure 5.4: Login screen.

data recording process can be done as many times as necessary or with the periodicity recommended by the external agent.

The third function is to record all text messages that are sent from our device. This process will be invisible to the user. That is, the user will not know that this is happening. To activate this functionality, it is necessary that we give the necessary permissions. This is done in our device settings. Once the device has the necessary permissions, the automatic recording function is activated. This function consists of sending every text message, sent by the user, to the Firebase server. Like the manual registration, the automatic recording requires Internet connection to send the data. In case you don't have a connection, it will save the data locally and send it once the device is connected.

The fourth function is that the user can see the data history that he has recorded manually. Finally, the fifth function is that the user can see the history of moods that the classifier has been performing based on text messages. This function will only be available if the external agent sends the information to the Firebase server. We have done it this way because in some cases the external agent may consider that it is not appropriate for the patient to see his moods history. An example of moods history is shown in Figure 5.6.

To implement the automatic message recording function, a keylogger (explained in section 2.3.2) has been used. The keylogger has been introduced on the Android device through


8:04 100%

EmoUPM

Enter New Data

Email address

Email address

How pleasant do you feel?

Not Response

How active do you feel?

Not Response

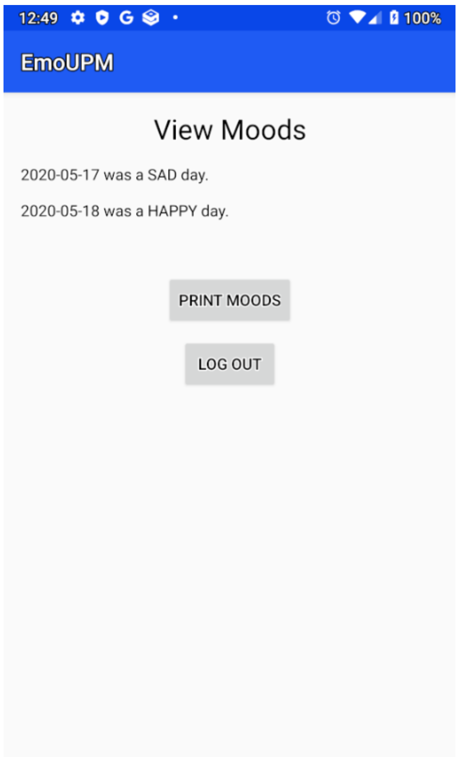
Did you practice any sport today?

Not Response

Walking + Runing Distance

Just the number in meters

Figure 5.5: Register data.


12:49 100%

EmoUPM

View Moods

2020-05-17 was a SAD day.

2020-05-18 was a HAPPY day.

Figure 5.6: User mood history.

the *EmoUPM* application.

Figure 5.7 shows the UML class diagram that we have developed in Android Studio, along with the relationships between each class. As we can see in Figure 5.7, *MyAccesibilityService* class is not related to any other class. This is because this class is the one we used to create the keylogger, and therefore has no relation to the EmoUPM functions.

The *MainActivity* class corresponds to the log in screen of the EmoUPM application. From here the user can access the upload data screen (corresponding to the class *subirDatos*) or the registration screen for a new user (corresponding to the *register* class). From the *register* class, it is possible to return to the *MainActivity* class in two ways. The first way is by registering as a new user (the user will be redirected to *MainActivity*). The second way is by pressing the back button. If we move to *subirDatos* class, we can also return to *MainActivity* class by pressing the logout button. From the *subirDatos* class the user can access to *verDatos* class, that corresponds to the screen to see user data. The *verDatos* class will use the class called *userPojo*, to print the data. From the *subirDatos* class the user also can access to *verMoods* class, that corresponds to the screen to see user moods history.

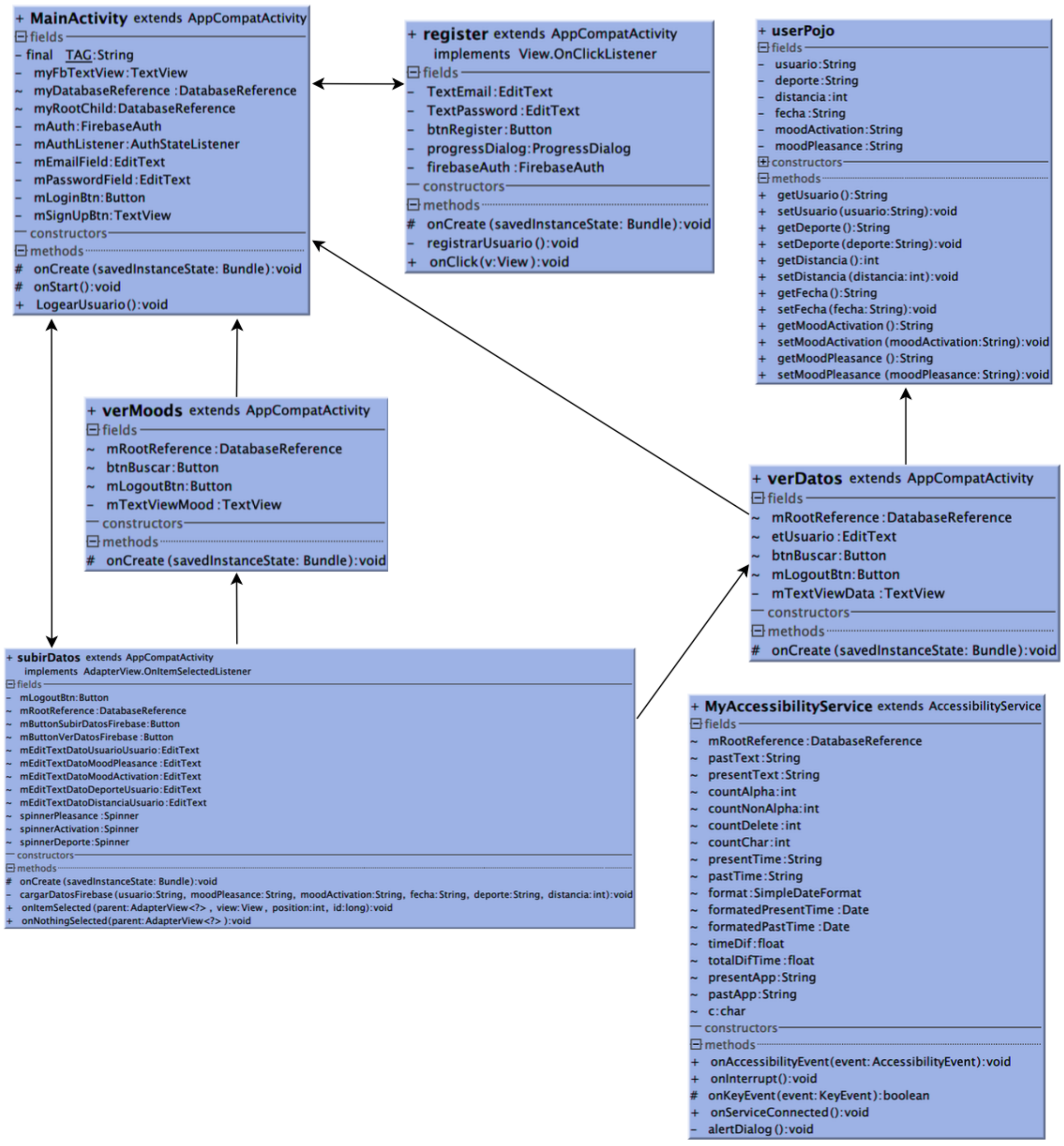

+ MainActivity extends AppCompatActivity
fields
- final TAG:String
- myFbTextView:TextView
~ myDatabaseReference :DatabaseReference
~ myRootChild:DatabaseReference
- mAuth:FirebaseAuth
- mAuthListener:AuthStateListener
- mEmailField:EditText
- mPasswordField:EditText
- mLoginBtn:Button
- mSignUpBtn:TextView
constructors
methods
# onCreate (savedInstanceState: Bundle):void
# onStart():void
+ LogearUsuario():void
+ register extends AppCompatActivity implements View.OnClickListener
fields
- TextEmail:EditText
- TextPassword:EditText
- btnRegister:Button
- progressDialog:ProgressDialog
- firebaseAuth :FirebaseAuth
constructors
methods
# onCreate (savedInstanceState: Bundle):void
- registrarUsuario ():void
+ onClick(v:View):void
+ userPojo
fields
- usuario:String
- deporte:String
- distancia:int
- fecha:String
- moodActivation:String
- moodPleasance:String
constructors
methods
+ getUsuario():String
+ setUsuario(usuario:String):void
+ getDeporte():String
+ setDeporte(deporte:String):void
+ getDistancia():int
+ setDistancia(distancia:int):void
+ getFecha():String
+ setFecha(fecha:String):void
+ getMoodActivation():String
+ setMoodActivation(moodActivation:String):void
+ getMoodPleasance():String
+ setMoodPleasance(moodPleasance:String):void
+ verMoods extends AppCompatActivity
fields
~ mRootReference:DatabaseReference
~ btnBuscar:Button
~ mLogoutBtn:Button
~ mTextViewMood:TextView
constructors
methods
# onCreate (savedInstanceState: Bundle):void
+ verDatos extends AppCompatActivity
fields
~ mRootReference:DatabaseReference
~ etUsuario:EditText
~ btnBuscar:Button
~ mLogoutBtn:Button
- mTextViewData :TextView
constructors
methods
# onCreate (savedInstanceState: Bundle):void
+ subirDatos extends AppCompatActivity implements AdapterView.OnItemSelectedListener
fields
- mLogoutBtn:Button
~ mRootReference:DatabaseReference
~ mButtonSubirDatosFirebase:Button
~ mButtonVerDatosFirebase :Button
~ mEditTextDatoUsuarioUsuario:EditText
~ mEditTextDatoMoodPleasance :EditText
~ mEditTextDatoMoodActivation:EditText
~ mEditTextDatoDeporteUsuario:EditText
~ mEditTextDatoDistanciaUsuario:EditText
~ spinnerPleasance :Spinner
~ spinnerActivation:Spinner
~ spinnerDeporte:Spinner
constructors
methods
# onCreate (savedInstanceState: Bundle):void
- cargarDatosFirebase (usuario:String, moodPleasance:String, moodActivation:String, fecha:String, deporte:String, distancia:int):void
+ onItemSelected (parent:AdapterView<?> , view:View , position:int, id:long):void
+ onNothingSelected(parent:AdapterView<?>):void
+ MyAccessibilityService extends AccessibilityService
fields
~ mRootReference:DatabaseReference
~ pastText:String
~ presentText:String
~ countAlpha:int
~ countNonAlpha:int
~ countDelete:int
~ countChar:int
~ presentTime:String
~ pastTime:String
~ format:SimpleDateFormat
~ formatedPresentTime :Date
~ formatedPastTime :Date
~ timeDif:float
~ totalDifTime:float
~ presentApp:String
~ pastApp:String
~ c:char
constructors
methods
+ onAccessibilityEvent(event:AccessibilityEvent):void
+ onInterrupt():void
# onKeyEvent(event:KeyEvent):boolean
+ onServiceConnected():void
- alertDialog ():void

Figure 5.7: EmoUPM class diagram.

5.2.2 Server Module

In this section we are going to explain in depth the structure where all the user data is stored. This module is the link between the EmoUPM module and the External Agent module. The databases diagram is shown in Figure 5.8.

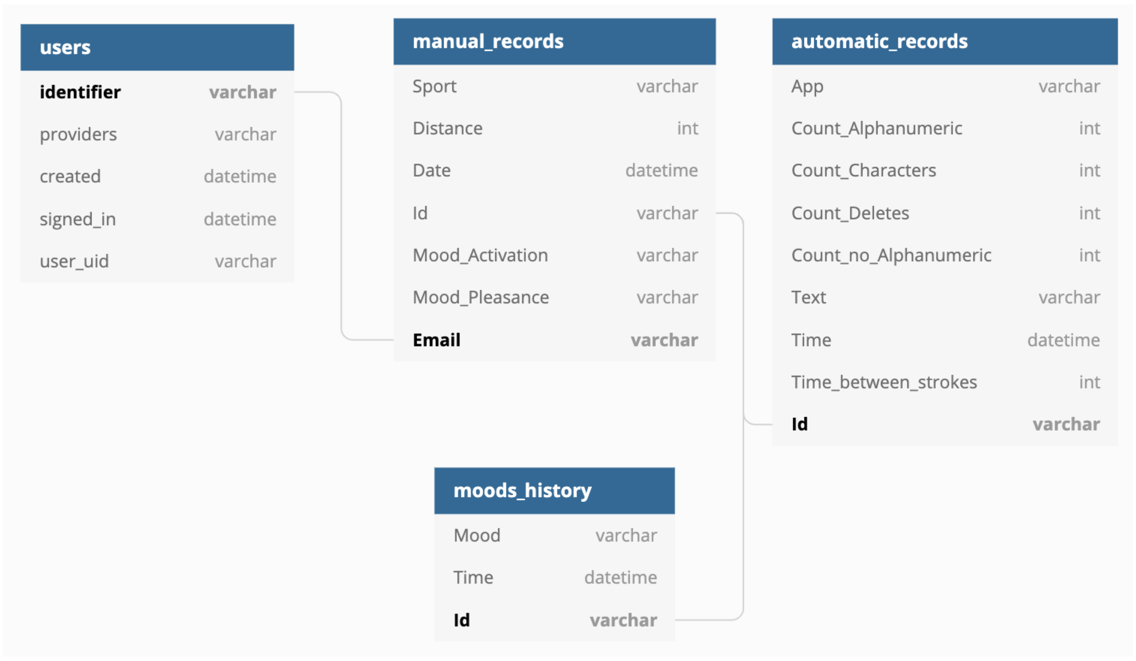
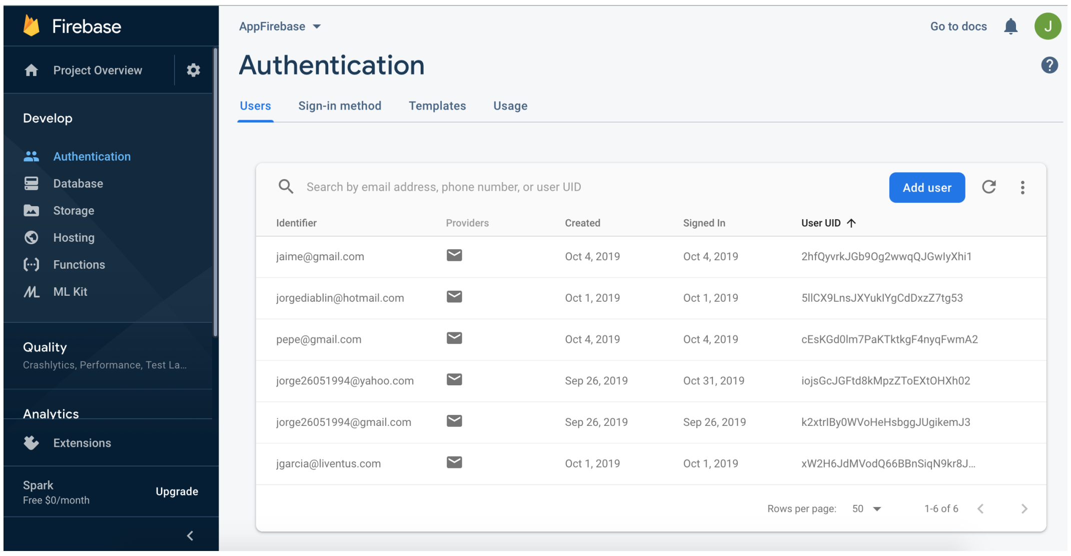


Figure 5.8: Database diagram.

On the one hand, all data collected by the *EmoUPM* application is sent to this Firebase server (section 2.3.1). On the other hand, all these data stored in the server are extracted by the External Agent to process them and get conclusions about the user's mood. Once the external agent obtains the conclusions and predictions, he will send the data to the server, so that it can be seen by the patient.

The server module is divided in three parts. The first part contains the credentials database, which is used in the process of authentication and registration of new users. This would be users in Figure 5.8. The second part contains the database with the user's data, and the third part contains the mood predictions. This third part would be moods_history in Figure 5.8.

The authentication database stores five parameters, even though the user only enters two of those five parameters. The first parameter is called the identifier. This parameter is entered by the user and in this case is the email address. The second parameter indicates the provider. We have decided that it should be the email address, but Firebase gives you the possibility to make it another one, such as phone number, Facebook, Twitter, etc. The third and fourth parameters correspond to the date of registration and the date of the last login, respectively. Finally, the fifth parameter corresponds to the password (user_uid in Figure 5.8), and like the email, is also entered by the user. For security reasons, the password is shown hashed. In Figure 5.9, the structure of our authentication database is shown.



Identifier	Providers	Created	Signed In	User UID ↑
jaime@gmail.com	✉	Oct 4, 2019	Oct 4, 2019	2hfQyvrkJGb9Og2wwqQJGwlyXhi1
jorgediablin@hotmail.com	✉	Oct 1, 2019	Oct 1, 2019	5llCX9LnsJXYuklYgCdDxzZ7tg53
pepe@gmail.com	✉	Oct 4, 2019	Oct 4, 2019	cEsKGd0lm7PaKTktkgF4nyqFwmA2
jorge26051994@yahoo.com	✉	Sep 26, 2019	Oct 31, 2019	iojsGcJGFtd8kMpzZToEXtOHXh02
jorge26051994@gmail.com	✉	Sep 26, 2019	Sep 26, 2019	k2xtrIBy0WVoHeHsbggJUgikemJ3
jgarcia@liventus.com	✉	Oct 1, 2019	Oct 1, 2019	xW2H6JdMVodQ66BBnSiqN9kr8J...

Rows per page: 50 1-6 of 6

Figure 5.9: Authentication database.

In Figure 5.10, we can see an UML sequence diagram. This diagram shows three possible events in the authentication process. On the one hand (in green) we can see the authentication process performed successfully. Secondly (in red), we can see the failed authentication process. Finally (in blue), we can see the registration process of a new user.

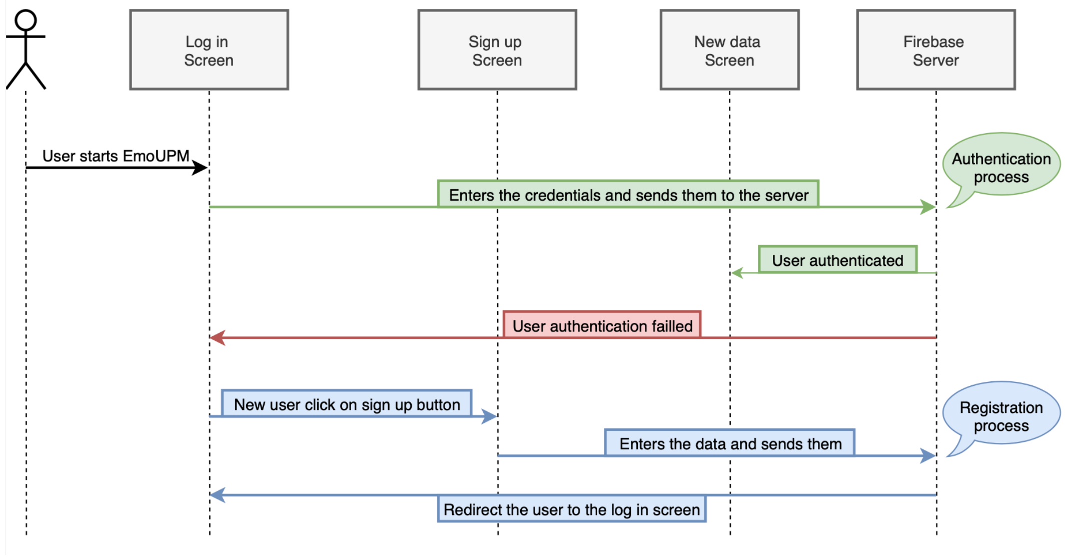
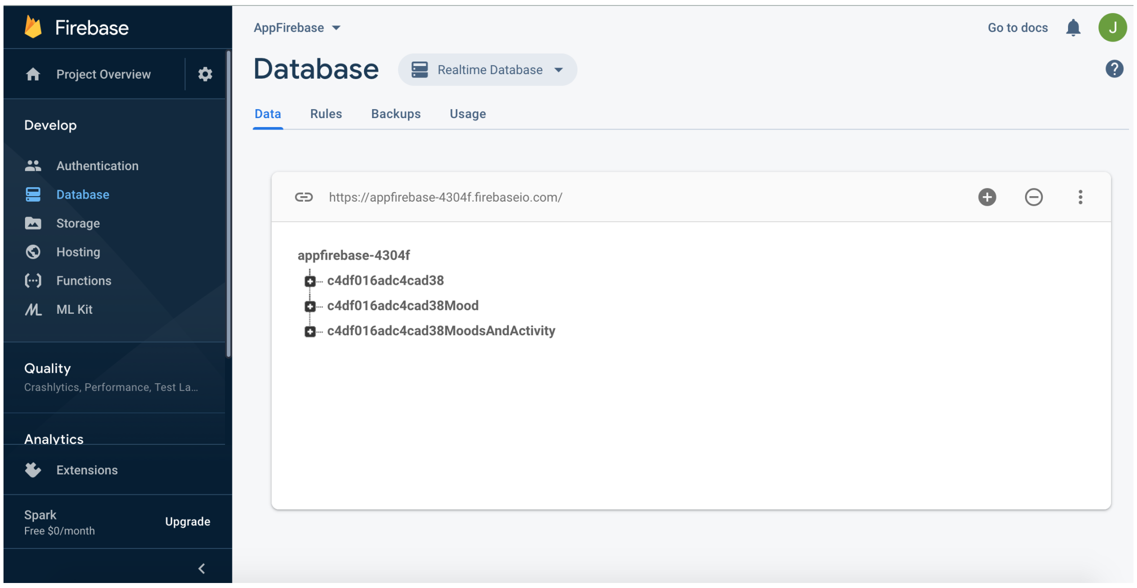

Log in Screen
Sign up Screen
New data Screen
Firebase Server
User starts EmoUPM
Authentication process
Enters the credentials and sends them to the server
User authenticated
User authentication failled
New user click on sign up button
Registration process
Enters the data and sends them
Redirect the user to the log in screen

Figure 5.10: Authentication UML sequence diagram.

The second part of the server module consists of the database that contains the user information (Figure 5.11). As we can see in Figure 5.2, we have divided this database, at the same time, in three databases (manual_records, automatic_records and moods_history in Figure 5.8). We have done this to indicate that the manual records (containing the moods and daily physical activity) will be save separately from the automatic records (containing the text messages) and from the mood predictions (containing a daily history of the patient's moods) made by the external agent.



https://appfirebase-4304f.firebaseio.com/

appfirebase-4304f
- c4df016adc4cad38
- c4df016adc4cad38Mood
- c4df016adc4cad38MoodsAndActivity

Figure 5.11: Moods & Activity database.

Regarding the messages database, as we have already commented in the lines above, this database contains all the messages sent by the user. This database consists of the following columns:

- **App:** Indicates the application from which the message was sent.
- **Count_Alphanumeric:** Indicates the number of letters and numbers that make up the message.
- **Count_Deletes:** Indicates the number of times the user has pressed the "delete" key before sending the message.
- **Count_no_Alphanumeric:** Indicates the number of characters in the message that are neither letters nor numbers.
- **Text:** In this column you will find the text message.
- **Time:** Indicates the time the message was sent.

- **Time_between_strokes:** Indicates the average time (in milliseconds) it has taken to write each letter of the message.
- **Id:** Unique user identifier.

As we can see, this database stores much more data than we need to predict the mood, since we are only going to use the text message column. The rest of the columns, can be very useful for the external agent when it comes to draw conclusions, once he has all the information available.

Regarding the database that contains moods and daily physical activity, it consists of the following columns:

- **Sport:** Indicates the daily sport the user have performed up to the time of registration.
- **Distance:** Indicates the daily distance (running and walking) up to the time of registration.
- **Date:** Indicates the date and time of the record.
- **Id:** Unique user identifier.
- **Mood_Activation:** Indicates the degree of user activation at the time of registration.
- **Mood_Pleasance:** Indicates the degree of pleasant of the user at the time of registration.
- **Email:** Indicates the email address.

Finally, regarding the database containing the history of mood predictions, it is made up of the following columns:

- **Mood:** Indicates the mood predicted by the external agent.
- **Time:** Indicates the date and time of the prediction.
- **Id:** Unique user identifier.

5.2.3 External Agent Module

In this part of the chapter, we are going to explain in depth the parts involved in the External Agent. This is the part of the system where the expert that is going to monitor the user, participates. Furthermore, this is the part of the system where all the elements of the system are gathered, both the ML part, explained in Chapter 4, and all the data collected from the EmoUPM Module and the Server Module.

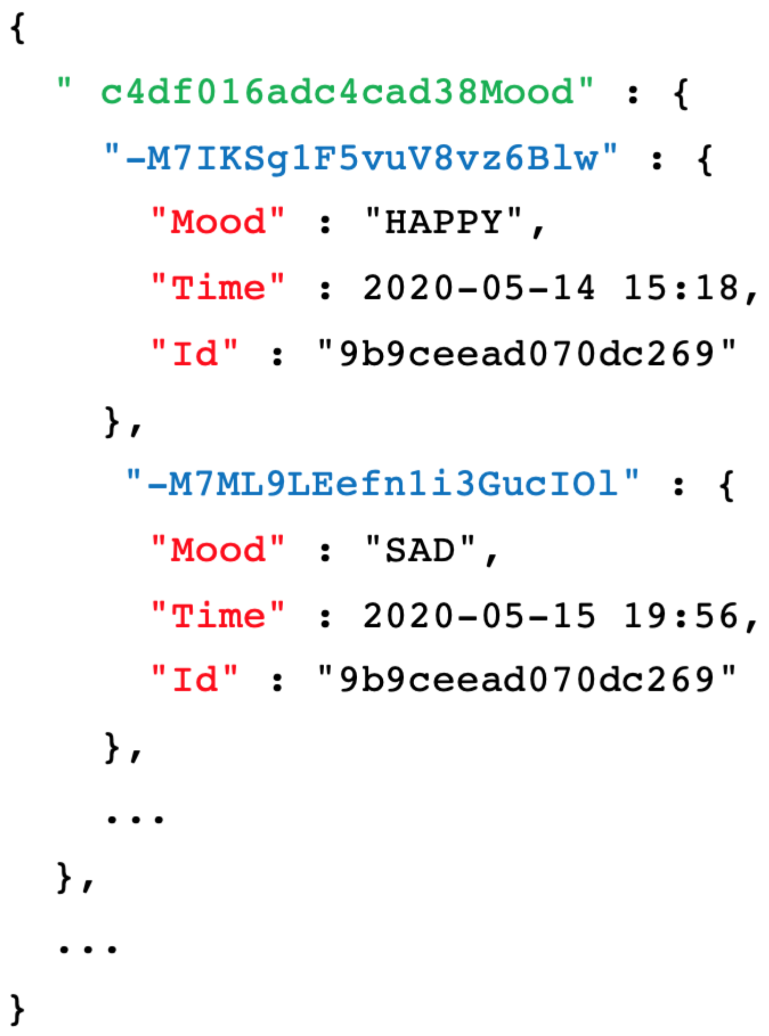
The structure and operation of this module is divided into three very well differentiated parts. The first part consists of extracting the data from the database of the Firebase server. This information can be extracted by clicking on the three points at the top right of screen (Figure 5.11) and selecting the option "Export JSON". This will download a JSON file with all the data. In Figure 5.12, we can see the result of a JSON file created, which contains both, an automatic (text message) and a manual record (mood & activity). This JSON file has three parts. The first part is the unique user identifier (green), the second part is the record identifier (blue) and the third part is the columns of each record (red).

Once we have the JSON file, we have to preprocess the data and then make a classification of each message, assigning it a mood (second part of Figure 4.1). All this will be done using the classifiers created in Section 4.6. Finally, with all the classified messages and the information obtained from the database of moods and daily activity, the expert will be able to make a report with conclusions.

We have also created an automatic way to get text messages. This way, the user only has to execute one function, passing the user's unique identifier as a parameter, and the function returns the user's mood based on the text messages. At the same time, this function sends the mood predicted to the Firebase server, along with the date. In Figure X we can see how the mood predictions database looks like. As we have explained before, we have done this in this way in case the external agent does not want to send the data to be seen by the patient.

```
{
  "9b9ceead070dc269" : {
    "-LsntHd_axxrsqNBd1DK" : {
      "App" : "com.google.android.talk",
      "Count_Alphanumeric" : 11,
      "Count_Characters" : 14,
      "Count_Deletes" : 0,
      "Count_no_Alphanumeric" : 3,
      "Text" : "This is James!",
      "Time" : "18:51:25.737",
      "Time_between_strokes" : 0,
      "Id" : "9b9ceead070dc269"
    },
    ...
  },
  "c4df016adc4cad38" : {
    "-M5utR4MySXb5DnqBshg" : {
      "Sport" : "Soccer",
      "Distance" : 9000,
      "Date" : "2020-04-27T11:09:11.540",
      "Id" : "c4df016adc4cad38",
      "mood_Activation" : "Very Active",
      "mood_Pleasance" : "Very Pleasant",
      "Email" : "jorge@yahoo.com"
    },
    ...
  }
}
```

Figure 5.12: Output JSON file generated.



```
{
  " c4df016adc4cad38Mood" : {
    "-M7IKSg1F5vuV8vz6Blw" : {
      "Mood" : "HAPPY",
      "Time" : 2020-05-14 15:18,
      "Id" : "9b9ceead070dc269"
    },
     "-M7ML9LEefn1i3GucIOl" : {
      "Mood" : "SAD",
      "Time" : 2020-05-15 19:56,
      "Id" : "9b9ceead070dc269"
    },
    ...
  },
  ...
}
```

Figure 5.13: JSON file for mood prediction database.

CHAPTER 6

Conclusions

This chapter will describe the achieved goals done by the master thesis following some the key points developed in the project.

6.1 Conclusions

As we explained at the beginning of this thesis, mood disorders are a disease suffered by more than 450 million people. This type of psychological illness does not only affect adults. In 2011, mood disorders were the leading cause of hospitalization for people aged 1 to 17 years in the United States, with more than 110,000 cases [55]. Many of the cases can be avoided and others can be treated without the need for medication, simply by changing patients' habits.

In this thesis, a system for monitoring people's moods has been developed, with the aim of preventing or curing, in the future, the greatest number of mood disorders. To do this, an Android application has been developed that allows two types of monitoring. First, it allows manual recording of moods (based on activation and pleasant) and daily physical activity. Secondly, we implemented a mechanism to register all the text messages sent by the user. The latter is done automatically and transparently for the user (the user has to give permissions when downloading the application), so that it does not disturb him when communicating with other users. Thanks to these two registration mechanisms, an expert, in the field of psychology, will be able to monitor the user mood (patient) quite accurately.

On the other hand, we have developed a mood prediction system based on text messages. This system has been developed using NLP and ML algorithms. To develop this system, we have used a corpus, from a company called *Figure Eight Inc*, which is made up of 40,000 messages. Each message has an associated mood (thirteen possible moods). We have tested using the main moods (happy, angry, relief, surprise and sad). In addition, we have also tested using mood groups. We have grouped the thirteen moods according to proximity within the Circumplex Model. We have performed numerous tests by grouping the moods and not grouping the moods. In addition, we have used more than ten algorithms, combining them with some features.

With the results obtained, we can draw several conclusions. Firstly, based on the results, we can say that better results are obtained when we do not group the moods. The fact of making groups of moods has made us get almost 10% less accuracy (in the worst situation). On the other hand, we can conclude that some moods are easier to predict than others. For example, moods with a very positive or negative valence are easier to predict than those with a neutral valence, even though the arousal is positive or negative. In other words, this means that we have obtained notably better results for moods such as happy or angry than for moods such as surprised.

6.2 Achieved Goals

The development of this thesis has successfully reached the goals proposed at the beginning. Firstly, we have gained experience in the development of Android applications. In addition, we have obtained experience in the development of classification systems using AI.

The goals achieved for this project are shown below:

- We have developed an Android application where the user can register and log in.
- We have developed a keylogger from scratch. This has allowed us to automatically record all user conversations.
- We have developed a system that uses machine learning algorithms to predict moods based on text messages.
- We have carried out an evaluation of the results obtained. We have created several classifiers, using various algorithms and features and we have compared the results obtained.
- We have developed an application where the user can monitor his mood history.
- We have developed an application where the user can monitor physical activity history.

6.3 Future work

Once the project is finished, it is time to present some ideas on how this thesis could be continued.

For this thesis, we have developed a mood monitoring system. To do this, we have created an Android application that collects all text messages sent by the user. Once the text messages are in the database, a psychology expert extracts them and uses them as input in the classification system we have created. The expert will get a set of moods with which to write a report. A possible future work is to create a script (the script will run the mood detector) that runs automatically every day. This would limit the dependency on an expert, who is responsible for reporting on the patient. In no case we could eliminate the functions of the expert, since a classification system will never be perfect and the supervision of a human being is always necessary.

Also, we could create a web application for the external agent. In this web application, the psychology expert could configure specific parameters for each user. For example, we could create a parameter to allow or not the patient to see his mood history.

Another possible future work is to use other techniques for the mood classification system, such as resampling. For this project, we have used a corpus with 40,000 unbalanced messages. In other words, there were not the same number of messages whose mood is happy as messages whose mood is relaxed. To solve this problem, what we have done is to create a classifier for each mood. That is, we have created a corpus with 50% of happy messages and the other 50% of not-happy messages, 12.5% of each mood (sad, angry, relief, surprised). We've done this for every one of the five moods we've worked with. But instead of doing this procedure, we could have done resampling. This is a technique that is often used in situations where the corpus is not balanced. This technique consists of removing rows of messages from the majority mood (undersampling) and/or adding rows of messages from the minority mood (oversampling) [32]. This technique does not guarantee better results. We would have to test using this technique and see if the results are better or worse.

In addition to the improvements or alternatives explained in the previous lines, we could add improvements in the application's interface, without adding functionalities to our project. Currently, the application displays the user's mood history in list format. An improvement could be adding a calendar to the EmoUPM application. This way, the user can see his mood in the calendar every day. This would make the application more user friendly.

APPENDIX A

Social, Ethical, Economical and Environmental Impact

This appendix shows the social, economic and environmental impact of carrying out this project. It also explains the ethical and professional responsibility.

A.1 Social Impact

This project is mainly focused on people with mood disorders and their psychologists. This project serves as a tool for psychologists to help during the patient's recovery process. It can also serve for patients who do not suffer from any mood disorder, but are at risk of suffering from it.

The method of accessing this tool is different for patients and for psychologists. On the one hand, psychologists must pay a monthly fee, depending on the number of patients. Once they have paid the first fee, the tool will be installed on their computer and they will be provided with credentials to access the Firebase server. On the other hand, patients will simply have to download the application to their mobile phone, through Google Play.

Finally, regarding the patient's privacy, it should be noted that the psychologist will be able to access all of his messages, whether from *Whatsapp*, *Hangouts* or *Facebook Messenger*. This is necessary in order to monitor the patient's mood.

A.2 Economical Impact

Economically speaking, this tool has a variable price depending on the number of patients the psychologist has. This will be discussed in more detail in the next appendix. On the other hand, it is important to mention that assistance to psychologists is quite expensive in Spain and that it is generally not covered by the public health system. This tool serves to speed up the recovery process for patients. In this way, patients will be able to recover their mental health sooner and save money indirectly.

A.3 Environmental Impact

The development of this project has practically no environmental impact. The biggest environmental impact is caused by the Firebase servers we use. On the other hand, we also have to take into account the use of the patients' smartphones and their batteries. These are all indirect impacts.

A.4 Ethic Impact

The most important question we asked ourselves during the development of this project was what data to collect from the user's smartphone. At first, we considered collecting only some data such as the number of characters in each message, the number of times the user presses the delete key, etc. But finally, we also decided to collect the content of each message, with the aim of maximizing the probability of success. We made this decision taking into account that a person who attends a psychologist, usually, does not hide anything. Therefore, you should have no problem with your psychologist being able to access your messages.

On the other hand, we assume that the psychologist will have a professional attitude and, even though he has access to his patient's messages, the purpose will be only professional.

APPENDIX B

Project Budget

In this appendix we will explain, in a general way, the cost of development and implementation of this project.

B.1 Project's development

For the development of this project, we have used several utilities. The cost of these devices can be approximate. The company called *Liventus Inc.* has provided me with the network, the computer and the monitors. All the development and testing has been done using a Lenovo T480 computer with the following features: 500GB SSD, Intel Core i7-8550U at 1.80GHz and 24GB of RAM. Below are the approximate costs:

- Firebase Server using Spark plan **$0/month**.
- Laptop **$1,500**.
- Two monitors **$150**.
- Network cost **$80/month**.
- Labor **$2,200/month**.

Assuming the duration of this project would be approximately two months, we can conclude that the cost would be around $6,210.

B.2 Deployment of a real-life case

In this case, we will show an estimate of the costs that would be involved in deploying this project. We have not taken into account costs such as the laptop (since the psychologist is supposed to have one already), the patient's mobile phone, the psychologist's or the patient's data network, etc. We have only taken into account the costs involved in the deployment of the project. Below we show the approximate costs for a supposed case of a psychologist and twenty-five patients:

- Firebase Server using Blaze plan **$60/month**.
- Software maintenance and support **$700/year**.

We have also created a priority support option. This option provides 24/7 support to the client, to solve any kind of problem they may have. This costs an additional $99/year.

Therefore, the total cost (including priority support) for a psychologist to hire this tool for the twenty-five patient scenario would be approximately $126.59/month.

Bibliography

[1] Adminca. The interface for your parse (server) data. http://adminca.com/, 2016. [Online; accessed 5-March-2020].

[2] Ajanta Das. Supervised learning model. https://www.researchgate.net/figure/Supervised-learning-model-22_fig11_312025091, 2019. [Online; accessed 18-December-2019].

[3] Amit Kumar. Introduction to machine learning. https://www.allprogrammingtutorials.com/tutorials/introduction-to-machine-learning.php, 2018. [Online; accessed 17-March-2020].

[4] AndroidDev. Accessibilityservice. https://developer.android.com/guide/topics/ui/accessibility/service?hl=en, 2020. [Online; accessed 27-January-2020].

[5] Anthony K. H. Tung. Rulebasedclassifier. https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_559, 2009. [Online; accessed 24-February-2020].

[6] Brenda Clark. Firebase alternative. https://medium.com/@brenda.clark/firebase-alternative-3-open-source-ways-to-follow-e45d9347bc8c, 2017. [Online; accessed 5-March-2020].

[7] Bruno Stecanella. What is tf idf. https://monkeylearn.com/blog/what-is-tf-idf/, 2019. [Online; accessed 1-April-2020].

[8] CodeAcademy. Pandas. https://www.codecademy.com/learn/data-processing-pandas/modules/dspath-intro-pandas, 2019. [Online; accessed 18-December-2019].

[9] Wikipedia Contributors. Supervised learning. https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=920525193, 2019. [Online; accessed 18-December-2019].

[10] Crowdflower. Sentiment analysis in text. https://data.world/crowdflower/sentiment-analysis-in-text, 2017. [Online; accessed 27-March-2020].

[11] CrowdFlower. The emotion in text dataset. https://www.crowdflower.com/wp-content/uploads/2016/07/text_emotion.csv, 2019. [Online; accessed 16-January-2020].

[12] Daphne Rietvelt. Influence of neutral word removal on sentiment analysis. http://essay.utwente.nl/78791/1/Rietvelt_BA_EEMCS.pdf, 2018. [Online; accessed 3-May-2020].

[13] Jocelyn D'Souza. Pos. https://medium.com/greyatom/learning-pos-tagging-chunking-in-nlp-85f7f811a8cb, 2018. [Online; accessed 21-January-2020].

[14] Brian Granger Fernando Pérez. Jupyter. https://jupyter.org, 2019. [Online; accessed 16-January-2020].

[15] Gaines Arnold. Circumplexmodelmood. https://study.com/academy/lesson/circumplex-model-of-emotion.html, 2019. [Online; accessed 14-January-2020].

[16] Hoodie. Hoodie intro. http://hood.ie/intro/, 2020. [Online; accessed 5-March-2020].

[17] Hafsa Jabeen. Stemmingandlemmatization. https://www.datacamp.com/community/tutorials/stemming-lemmatization-python, 2018. [Online; accessed 21-January-2020].

[18] Jason Brownlee. Vectorizermodels. https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/, 2017. [Online; accessed 23-January-2020].

[19] Jason Brownlee. Scikit-learn. https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/, 2019. [Online; accessed 18-December-2019].

[20] Jocelyn D Souza. Learning pos tagging and chunking in nlp. https://medium.com/greyatom/learning-pos-tagging-chunking-in-nlp-85f7f811a8cb, 2018. [Online; accessed 1-April-2020].

[21] Johanna Orellana Alvear. Decission trees and random forest. https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html, 2018. [Online; accessed 7-April-2020].

[22] John Schulman and Pieter Abbeel. Berkeley cs294 deep reinforcement learning. https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265, 2017. [Online; accessed 17-March-2020].

[23] Jonathan Posner, James A. Russell, and Bradley S. Peterson. Circumplexmodel. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2367156/, 2019. [Online; accessed 14-January-2020].

[24] Kristian Bannister. Sentimentanalysis. https://www.brandwatch.com/es/blog/analisis-de-sentimiento/, 2015. [Online; accessed 29-January-2020].

[25] Kyoosik Kim. Ridge regression for better usage. https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db, 2019. [Online; accessed 7-April-2020].

[26] lohithnv, mmbhatk, Varsha2018. Sentiment analysis. https://devopedia.org/sentiment-analysis, 2019. [Online; accessed 7-May-2020].

[27] Matplotlib. Matplotlib. `https://matplotlib.org/`, 2019. [Online; accessed 18-December-2019].

[28] Microsoft. Feature hashing. `https://docs.microsoft.com/es-es/azure/machine-learning/algorithm-module-reference/feature-hashing`, 2020. [Online; accessed 3-April-2020].

[29] Parse. Parse server getting started. `https://docs.parseplatform.org/parse-server/guide/`, 2020. [Online; accessed 5-March-2020].

[30] Pydata. Pandas pydata. `https://pandas.pydata.org/`, 2019. [Online; accessed 18-December-2019].

[31] Qolty. Esm. `https://qolty.com/experience-sampling-methods/`, 2019. [Online; accessed 14-January-2020].

[32] Rafael Alencar. Resampling strategies for imbalanced datasets. `https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets`, 2018. [Online; accessed 5-May-2020].

[33] Data School. Confusionmatrix. `https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/`, 2014. [Online; accessed 22-January-2020].

[34] Scikit-Learn. Logistic regression classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`, 2020. [Online; accessed 6-April-2020].

[35] Scikit Learn. Scikitlearn structure. `https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html`, 2020. [Online; accessed 17-March-2020].

[36] Scikit-Learn. Sgd classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html`, 2020. [Online; accessed 6-April-2020].

[37] Scikit-Learn. Svc classifier. `https://scikit-learn.org/stable/modules/svm.html`, 2020. [Online; accessed 6-April-2020].

[38] Shashank Kapadia. Language models ngram. `https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9`, 2019. [Online; accessed 1-April-2020].

[39] Srishti M. Random sampling methods for handling class imbalance. `https://stats.stackexchange.com/questions/351638/random-sampling-methods-for-handling-class-imbalance`, 2018. [Online; accessed 5-May-2020].

[40] Bivas Mitra Pradipta Det Surjya Ghosh, Niloy Ganguly. Towards designing an intelligent experience sampling method for emotion detection. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7983143`, 2019. [Online; accessed 18-December-2019].

[41] Niloy Ganguly Bivas Mitra Pradipta De Surjya Ghosh, Sumit Sahu. Emokey. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8711078`, 2019. [Online; accessed 18-December-2019].

[42] Sympy. Sympy. `https://www.sympy.org/en/index.html`, 2019. [Online; accessed 18-December-2019].

[43] Venali Sonone. Passive aggressive algorithms. `https://medium.com/@venali/conventional-guide-to-supervised-learning-with-scikit-learn-passive-aggressive-algo`
2018. [Online; accessed 7-April-2020].

[44] Walaa Medhat, Ahmed Hassan, Hoda Korashy. Sentiment analysis algorithms and applications. `https://reader.elsevier.com/reader/sd/pii/S2090447914000550?token=D5C248F0CF6377C3E0BE3B816E59C4D9617823D144A72010A617A9E1090F49CDE51EA4A7285DE11203F7`
2014. [Online; accessed 17-February-2020].

[45] Walaa Medhat, Ahmed Hassan, Hoda Korashy. Sentiment analysis algorithms and applications: A survey. `https://www.sciencedirect.com/science/article/pii/S2090447914000550`, 2014. [Online; accessed 17-March-2020].

[46] Wikipedia. Mental disorder. `https://www.wikiwand.com/en/Mental_disorder`, 2018. [Online; accessed 26-April-2020].

[47] Wikipedia. K nearest neighbors algorithm. `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`, 2020. [Online; accessed 7-April-2020].

[48] Wikipedia. Naive bayes classifier. `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`, 2020. [Online; accessed 6-April-2020].

[49] Wikipedia Comunity. Keylogger types. `https://en.wikipedia.org/wiki/Keystroke_logging`, 2020. [Online; accessed 10-March-2020].

[50] Wikipedia contributors. Firebase. `https://en.wikipedia.org/w/index.php?title=Firebase&oldid=928231722`, 2019. [Online; accessed 18-December-2019].

[51] Wikipedia contributors. Reinforcement learning. `https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=929834114`, 2019. [Online; accessed 18-December-2019].

[52] Wikipedia contributors. Unsupervised learning. `https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=928831374`, 2019. [Online; accessed 18-December-2019].

[53] Wikipedia contributors. Decisiontreeclassifier. `https://es.wikipedia.org/wiki/Aprendizaje_basado_en_%C3%A1rboles_de_decisi%C3%B3n`, 2020. [Online; accessed 24-February-2020].

[54] William Coryell. Introduction to mental disorders. `https://www.msdmanuals.com/es/hogar/trastornos-de-la-salud-mental/trastornos-del-estado-de-%C3%A1nimo/introducci%C3%B3n-a-los-trastornos-del-estado-de-%C3%A1nimo`, 2018. [Online; accessed 26-April-2020].

[55] World Health Organization. Mood disorders. `https://www.who.int/es/news-room/fact-sheets/detail/mental-disorders`, 2019. [Online; accessed 4-May-2020].

[56] Albert Au Yeung. Bigrams. `http://www.albertauyeung.com/post/generating-ngrams-python/`, 2018. [Online; accessed 21-January-2020].