Plataforma de medición instantánea para la prevención de conductas autolesivas en adolescentes
Platform for the prevention of self-injurious behavior in young people

Trabajo de Fin de Máster
Curso 2023–2024

Autor
Aldair Fredy Maldonado Honores

Director
Gonzalo Méndez
Pablo Gervás Gómez-Navarro

Plataforma de medición instantánea para la prevención de conductas autolesivas en adolescentes
Platform for the prevention of self-injurious behavior in young people

Trabajo de Fin de Máster en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Aldair Fredy Maldonado Honores

Director
Gonzalo Méndez
Pablo Gervás Gómez-Navarro

Convocatoria: *Septiembre 2024*
Calificación: *6 (Aprobado)*

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

25 de septiembre de 2024

Dedicatoria

A compañeros del máster y especialmente a mis padres, quienes me han brindado su total apoyo en todo momento y me han animado a seguir con el proyecto y el máster durante estos años.

Resumen

Plataforma de medición instantánea para la prevención de conductas autolesivas en adolescentes

El suicidio ha ido ganando conciencia en estos últimos años y se ha convertido en una de las principales preocupaciones de salud a nivel global, especialmente su aumento entre los jóvenes adolescentes. Es por ello que la prevención de este tipo de conductas se ha vuelto un reto entre los profesionales de la salud, debido a la gran cantidad de factores de riesgo que pueden influir, directa o indirectamente.

Bajo esta premisa, este Trabajo Fin de Máster busca diseñar y crear una aplicación para SIVARIA, un proyecto de investigación cuyo objetivo es la prevención de conductas autolesivas suicidas y no suicidas en jóvenes adolescentes de entre 12 y 21 años. Para ello, la plataforma estará compuesto por una serie de cuestionarios que proceden del sitio web del proyecto de SIVARIA que los usuarios pueden rellenar. Estos cuestionarios variarán dependiendo de si el usuario es un joven, un familiar o un profesional quien los rellena. Una vez cumplimentados estos cuestionarios, se enviarán a un Sistema Experto que tratará de predecir conductas autolesivas suicidas y no suicidas, tales como la ideación del suicidio, en la población joven. Para la realización de esta función, se diseñarán un conjunto de modelos utilizando clasificadores, entrenados con datasets generados a través de scripts artificiales y a través de un LLM (*Large Language Model*), y aplicando técnicas de ajuste de parámetros e hiperparámetros.

Gracias a estas predicciones, se pueden aplicar con anticipación medidas de monitorización y seguimiento a los pacientes, reduciendo los riesgos de posibles conductas suicidas.

Palabras clave

Inteligencia Artificial, Sistema Experto, Bayes, React Native, API, Django, Scikit-learn, Clasificador, Aplicación multiplataforma

Abstract

Platform for the prevention of self-injurious behavior in young people

Suicide has been gaining awareness in recent years and has become a major global health concern, especially its increase among young adolescents. This is why the prevention of this type of behaviour has become a challenge among mental health professionals, due to the large number of of risk factors that can influence directly or indirectly.

Under this premise, this Master's Thesis aims to design and create an application for SIVARIA, a research project focused on the prevention of suicidal and non-suicidal self-injurious behaviors. For this purpose, the platform will be composed of a set of questionnaires that come from the SIVARIA website that users can fill out. These questionnaires will vary their content depending on whether the user is a young person, a family member or a professional. Once they have been filled in, they will be sent to an Expert System which will try to predict suicidal and non-suicidal self-injurious behaviors, such as suicidal ideation, in young population. For the realization of this task, a set of models will be designed using classifiers, trained with datasets generated through artificial scripts and later, through a Large Language Model and applying parameter and hyperparameter fitting techniques.

Thanks to these predictions, monitoring and follow-up measures can be applied to patients in advance, reducing the risks of self-injurious behavior. patients in advance, reducing the risks of possible suicidal behavior.

Keywords

Artificial Intelligence, Expert System, Bayes, React Native, API, Django, Scikit-learn, Classifier, Multiplatform application

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

En estos últimos años, el suicidio se ha convertido en un problema de salud a nivel global, siendo una de las principales causa de muerte no natural entre jóvenes y adultos. Es por ello que su prevención y mitigación ha presentado un reto para los profesionales de la salud, ya que dicha problemática afecta tanto a los propios individuos, como también a los familiares y amigos cercanos.

Por esta razón que en los últimos años se están centrando los esfuerzos y los recursos económicos en la creación de programas de prevención del suicidio. En estos planes es fundamental realizar diagnósticos lo más pronto posible para detectar a tiempo síntomas mentales en la persona, y por lo tanto, poder ofrecerle un tratamiento. El problema es que muchas veces, las personas con riesgo de adoptar estas conductas suicidas y autolesivas no son conscientes de que los padecen o que pueden padecerlos, por lo que no acuden a los planes de prevención. Esto trae, como consecuencia, el agravamiento del problema, hasta acabar en un desenlace fatal. Es importante el poder informar a la persona a tiempo sobre su estado de salud mental de forma automática, rápida y eficaz, para que así sea consciente del riesgo al que está sometido, y por lo tanto, pueda pedir ayuda con anticipación.

Ante este panorama, se busca desarrollar herramientas para prevenir este tipo de conductas en la juventud. Es aquí donde comienza a jugar un papel fundamental la inteligencia artificial. De acuerdo al artículo de Fonseka et al. (2019), la inteligencia artificial se está utilizando cada vez más en el campo de la salud mental, siendo de gran utilidad para la prevención del suicidio y dar soporte a los profesionales para conseguir una evaluación exacta del riesgo de suicidio. De esta manera, se pueden tratar a los potenciales pacientes que posean estas conductas autolesivas o no autolesivas en etapas muy tempranas del desarrollo de estos comportamientos.

El motivo principal de la elección de este tema es por la relevancia que ido adquiriendo este tema a lo largo de los años. Además, es un proyecto que abarcaba el apartado de desarrollo de aplicaciones web y móviles, con el que estoy más familiarizado.

1.2. Objetivos

El objetivo principal del proyecto es el poder desarrollar una plataforma que, a través de una serie de cuestionarios, pueda realizar predicciones de posibles conductas autolesivas

suicidas y no suicidas en jóvenes adolescentes de entre 12 y 21 años, mediante consultas a un Sistema Experto. Para lograr este objetivo principal, se establecieron los siguientes objetivos de proyecto a lo largo del desarrollo del TFM.

- Diseñar un Sistema Experto para la valoración del riesgo de conductas autolesivas suicidas y no suicidas en población adolescente (12-21 años).
- Diseñar y desarrollar una aplicación web y móvil que desde donde se puedan realizar las predicciones del Sistema Experto.

1.3. Plan de trabajo

En base a los objetivos marcados en el apartado anterior, se estableció el siguiente plan de trabajo.

- Selección del tipo de Sistema Experto. Se realizará una búsqueda de los distintos tipos de Sistemas Experto y se seleccionará el más indicado para este caso.
- Entendimiento del proyecto de SIVARIA. Acceder a la página del proyecto, investigar y entender los cuestionarios de la página, que serán de utilidad para diseñar los modelos del Sistema Experto.
- Selección de herramientas y librerías para implementar los modelos para el Sistema Experto seleccionado.
- Diseño e implementación de los modelos mediante las herramientas y librerías seleccionadas.
- Entrenar y validar los modelos creados a través de varios ciclos o iteraciones. Se interpretarán los resultados en las iteraciones y se mejorarán los parámetros y datos de entrada hasta conseguir un rendimiento aceptable en los modelos del Sistema Experto.
- Estudio de herramientas y tecnologías para la creación de la aplicación móvil.
- Creación de las casos de uso para el desarrollo de la aplicación.
- Desarrollo: se lleva a cabo el desarrollo de la aplicación en base a los casos de uso especificados mediante la utilización de las herramientas estudiadas.

Todo el código realizado durante el desarrollo del proyecto se encuentra en el siguiente repositorio de Github.

`https://github.com/NILGroup/TFM-2324-SIVARIA`

Cuenta con licencia MIT, y el repositorio es público.

1.4. Estructura de la Memoria

El contenido de la memoria se compone de los siguientes apartados.

1.4.1. Estado del Arte

En dicho apartado se expondrá el estado actual del suicidio juvenil, y de los planes y estrategias creados para prevenir el riesgo de suicidio. Se mostrarán algunos ejemplos de planes físicos aplicados a los colegios, ya que son las zonas donde más se puede interactuar con los alumnos. Más adelante, se citarán más ejemplos, pero en este caso, de aplicaciones web y móviles que buscan aplicar sus respectivos planes de prevención, intervención y formación, mediante el acercamiento a los jóvenes con el uso de dispositivos móviles. Finalmente se realiza un análisis comparativo entre las aplicaciones y se evalúan los puntos fuertes y débiles de cada uno. De esta manera, podemos obtener una visión global de las funciones que debe realizar la aplicación.

1.4.2. Descripción del Trabajo

En esta sección se explicará todo lo relacionado con el diseño, creación y entrenamiento de los modelos. Se muestra el trabajo realizado para diseñar los modelos iniciales del Sistema Experto. Asimismo, se muestran las versiones de los entrenamientos realizados, junto con la interpretación de los resultados. También se incluyen una serie de análisis, y sus correspondientes conclusiones, sobre cada versión de los entrenamientos.

1.4.3. Aplicación

En este apartado se mostrará todo el trabajo relacionado con el tema de la aplicación. Se explicarán los dos servidores principales de la aplicación: el frontend y backend, y los frameworks utilizados para crearlos, además de cómo se realiza la conexión entre los dos proyectos, dependiendo de si se hace desde la máquina local o el dispositivo móvil.

Adicionalmente se enseñará la estructura y diseño de la base de datos, los diagramas de casos de uso, y el funcionamiento de la aplicación.

Finalmente, se ahondará en las medidas de seguridad que hay implementadas en la aplicación, tanto por parte del cliente, como del servidor y la base de datos.

1.4.4. Conclusiones y Trabajo a Futuro

Al final de la memoria, se realizará una serie de conclusiones sobre el trabajo realizado durante la fase de entrenamiento de los modelos del Sistema Experto, así como la fase del desarrollo de la aplicación. Posteriormente, se proponen una serie de mejoras que se pueden añadir al proyecto en el futuro, como son el uso de datasets reales, validación del Sistema Experto con otros usuarios, o la posible migración a otras alternativas, entre otros.

1.4.5. Apéndice A

El primer apéndice incluye un manual de instrucción con los pasos a realizar para poder ejecutar una versión de la aplicación, en caso de que quiera probarlo en la máquina local. Se explican la creación del entorno virtual, la descarga de los paquetes necesarios del proyecto frontend y backend, y la creación y configuración de la base de datos.

1.4.6. Apéndice B

En ella se incluyen una serie de figuras y tablas adicionales a la memoria.

Capítulo 2

Estado del Arte

En este capítulo se el estado actual de la salud mental en la población juvenil y la situación de las aplicaciones centradas en abordar estos temas.

2.1. Salud mental juvenil

La salud mental es un estado en el que una persona se encuentra en un bienestar emocional, social y psicológico. Este bienestar influye directamente en nuestra vida diaria, permitiéndonos o no el poder realizar actividades cotidianas, interactuar con otras personas, entre otros. Es por ello que, la falta de salud mental puede acarrear problemas y trastornos mentales a las personas, y por consiguiente, en la adopción de conductas autolesivas suicidas y no suicidas.

Las conductas autolesivas son comportamientos que adoptan las personas para inflingirse autolesiones. Dentro de los comportamientos autolesivos, están los comportamientos no suicidas, que son conductas autolesivas que no buscan causar la muerte, sino sólo el inflingir daño a sí mismos. Estas conductas puede incluir cortarse, quemarse o golpearse. Por otro lado están los comportamientos suicidas, que son más graves ya que en estos casos sí que se busca poner fin a la vida, por lo que representan una emergencia de salud mental.

2.2. Programas de prevención

A lo largo de los últimos años, se han lanzado programas de prevención del riesgo de suicidio para jóvenes adolescentes, tanto en España como en otros países. Entre ellos, algunos ejemplos son:

- ARSUIC (Atención al Riesgo Suicida)[1] (Sola et al. (2019)): es un programa de prevención creado por la Comunidad de Madrid, y cuyo objetivo es reducir el riesgo posterior al intento de suicidio, facilitando la posibilidad de sacar cita ambulatoria después del alta hospitalaria. Es una estrategia de enfoque general, en lugar de centrarse específicamente en la población joven.

[1] `https://www.comunidad.madrid/transparencia/informacion-institucional/planes-programas/programa-prevencion-del-suicidio`

- Surviving the Teens (Bustamante y Florenzano (2013) y King et al. (2011)): se creó en el Cincinatti Children's Hospital, e incluye estrategias de psicoeducación a padres, profesores y alumnos. De esta manera, se intenta incentivar conductas positivas entre los adolescentes, como guías para aumentar la autoestima, gestionar la ira o comunicarse mejor con los padres, entre otros. Se realiza en sesiones y se caracteriza por ser un plan corto de 3 meses, por lo que existe el riesgo de recaer.

- Adolescents Depression Awareness Program (ADAP) (Bustamante y Florenzano (2013)): es un programa creado en el Johns Hopkins University School of Medicine con el objetivo de educar a los alumnos, profesores y familias sobre la depresión.

- TeenScreen (Bustamante y Florenzano (2013)): era un programa diseñado para detectar indicios de ideación suicida entre la población juvenil, redirigiéndolo a los servicios de salud mental si finalmente se da el caso.

- National Youth Suicide Prevention Strategy (Bustamante y Florenzano (2013)): es un programa creado por el gobierno australiano, en colaboración con los centros de salud y los colegios. Consistía en el impulso de formación psicoeducativo a profesionales de la salud, además del fomento de programas de protección para los adolescentes.

También se han propuesto otros programas de prevención nuevos e innovadores para poder detectar conductas autolesivas:

- El artículo de Agüero-Nate et al. (2023) presenta un enfoque innovador para abordar la problemática de la depresión en adolescentes mediante estrategias educativas de prevención. Dichas estrategias constan de programas de orientación educativa para ayudar a los adolescentes a manejar mejor sus emociones y fortalecer sus habilidades para enfrentar factores de riesgo asociados a la depresión.

- El trabajo presentado por Monteagudo Rodenas (2022) propone estrategias educativas y de intervención temprana para reducir el riesgo de suicidio en adolescentes, subrayando la importancia de la capacitación de profesionales en áreas como la psicología y la pedagogía para implementar programas de prevención efectivos.

- Un estudio de Kurniawan et al. (2024) realiza un análisis de posibles estrategias para reducir los comportamientos autolesivos en adolescentes. En él, se explica que una de las posibles soluciones es el uso de aplicaciones web y móviles para la gestión de conductas autolesivas no suicidas, ofreciendo asistencia adicional. De acuerdo a un artículo de Kruzan et al. (2022), el uso de este tipo de aplicaciones en estos casos supone un descenso en la cantidad de autolesiones de los jóvenes, así como un aumento en sus niveles de confianza para poder cambiar sus conductas autolesivas.

- En el siguiente artículo de Burn et al. (2024) se explica el diseño y desarrollo de un programa web para entrenar al personal escolar en detectar conductas autolesivas y en aplicar técnicas y estrategias gestión.

Como se puede apreciar, la mayoría de estas estrategias van dirigidas a jóvenes y familiares, a los que se les aplican métodos de educación sobre las conductas autolesivas y fomento de actitudes positivas. Con esto, se busca detectar rápidamente cualquier factor de riesgo, y aplicar la estrategia correspondiente.

Cabe mencionar que el propio SIVARIA[2] también tiene publicado en su sitio web guías y recursos a jóvenes, familias y profesionales para la detección y prevención de conductas suicidas y no suicidas. En ella, se distinguen 6 niveles de alerta, dependiendo del riesgo del paciente: ausente, leve, moderado, grave, crisis y emergencia.

2.3. Aplicaciones

Además de programas físicos, también se han desarrollado aplicaciones web y móviles que buscan tratar el tema de la salud mental en las personas. A continuación, mostraremos algunos ejemplos y se realizará un análsis comparativo entre todas ellas.

2.3.1. CalmHarm

CalmHarm[3] es una aplicación móvil desarrollada por la Dra. Nihara Krause, psicóloga clínica, para la organización benéfica Stem4 en Reino Unido. Se centra en ofrecer una serie de actividades generales y personalizadas basadas en la Terapia Dialéctiva Conductual (DBT) (Salsman y Linehan (2006)) para regular las emociones fuertes que empujan a la autolesión y impulsando que se adopten conductas más sanas. Menciona que se puede usar para jóvenes de 13 años en adelante.

2.3.2. Prevensuic

Prevensuic[4] es un programa creado por la *Fundación Española para la Prevención del Suicidio*, destinado a la prevención, divulgación y la formación acerca del suicidio. Dicho programa cuenta con un sitio web y una aplicación móvil (disponible tanto en Android como en iOS) para ser accesible a un mayor número de personas, tanto adultos como jóvenes.

2.3.3. Suicide Safety Plan

Suicide Safety Plan[5] es una aplicación móvil desarrollada por Inquiry Health LLC y está disponible tanto en Android como en iOS. La aplicación funciona como un recordatorio para que el usuario pueda establecer qué cosas pueden afectar a su salud mental, e inluye información para ayudar a la persona a evaluar su propio estado mental. Además, proporciona una serie de números de teléfono para contactar con profesionales en caso de necesitar ayuda urgente.

2.3.4. ISNISS

En este caso, ISNISS[6] es un sitio web,que se encarga de dar información sobre cómo realizar una evaluación correcta del riesgo de suicidio en un paciente, aunque no la realiza el propio sitio web automáticamente. Aparte incluye otra serie de servicios, como terapias psicológicas.

[2] https://blogs.uned.es/investigacioninfantojuvenil/sivaria-gestion-del-suicidio/
[3] https://calmharm.stem4.org.uk/
[4] https://www.prevensuic.org/
[5] https://play.google.com/store/apps/details?id=com.moodtools.crisis.app&hl=en_US
[6] https://www.isniss.es/

2.3.5. PAPAGENO

PAPAGENO[7] es un sitio web que ofrece información, documentos científicos y materiales de autoyuda para la prevención, investigación, tratamiento y postvención de las conductas suicidas. Destaca por proporcionar enlaces y números de teléfono de urgencias contra el suicidio, guías de autoayuda, información sobre asociaciones enfocadas en esta temática, etc.

2.4. Tabla comparativa de las aplicaciones

Se creó una tabla comparativa (Tabla 2.1), en donde se evaluan las aplicaciones seleccionadas mediante una serie de criterios, como:

- Tipo de aplicación, que puede ser web y/o móvil.
- Interacción directa: indica si la aplicación interactúa con los usuarios, en el sentido de incluir actividades personalizadas e interactivas, seguimiento de cada caso y retroalimentación en tiempo real.
- Enfoque educativo: indica si la aplicación proporciona información educativa y materiales de formación para los usuarios.
- Plan de seguridad: se refiere a si la aplicación tiene la posibilidad de crear o gestionar un plan de emergencia en caso de crisis.
- Recursos de emergencia: se refiere a si la aplicación proporciona acceso rápido a contactos de emergencia, como números de teléfono de emergencia y ayuda.

En base a estos criterios, se elaboraron las tablas comparativas 2.1.

	Prevensuic	**Suicide Safety Plan**	**CalmHarm**	**ISNISS**	**PAPAGENO**
Tipo de aplicación	Web y móvil	Móvil	Móvil	Web	Web
Interacción directa	✓	✓	1	✗	✗
Enfoque educativo	✓	✗	✓	✓	✓
Plan de seguridad	✓	✓	✗	✓	✗
Recursos de emergencia	✓	✓	✗	✓	✓

Tabla 2.1: Tabla comparativa de aplicaciones de prevención contra el suicidio. Última actualización: mayo de 2024

[7] `https://papageno.es/about/quienes-somos`

2.5. Análisis comparativo de las aplicaciones

Al analizar las aplicaciones, pude observar que son aplicaciones diversas, en el que cada una tiene puntos fuertes y débiles, y se enfocan en aspectos concretos. Desde el punto de vista de los usuarios adolescentes que buscan aplicaciones interactivas, CalmHarm es una opción destacada. Por otro lado, Prevensuic y Suicide Safety Plan son útiles para ofrecer planes de seguridad y accesos rápidos a recursos de emergencia. Finalmente, ISNISS y PAPAGENO se especializan en proporcionar información educativa para el seguimiento, evaluación y prevención de conductas autolesivas, que pueden ser de ayuda para profesionales y padres interesados.

Sin embargo, una cosa en común entre todos ellos es que no ofrecen ningún algoritmo ni ningún sistema para poder detectar y pronosticar en tiempo real las posibles conductas autolesivas en las personas, sino que ofrecen herramientas para poder prevenirlo, tales como las actividades de CalmHarm o los planes de seguridad de Suicide Safety Plan.

Otro punto a destacar es Prevensuic, que es considerada una aplicación o programa muy completo y psicoeducativo, es decir, que es capaz de brindar información a las personas que están sufriendo de un trastorno psicológico. No obstante, presenta un gran inconveniente, y es que no está actualizado para las nuevas versiones de sistemas operativos (Android 14, por ejemplo). Su última actualización data del 11 de abril de 2019, lo que supone una problema de accesibilidad de la aplicación y, por lo tanto, del programa en general.

2.5.1. Conclusión

En base a las ventajas y desventajas de las aplicaciones analizadas, se buscará diseñar Sistema Experto y una aplicación multiplataforma que pueda ser accesible en las versiones más recientes de iOS y Android, que además tenga versión web, y que pueda proporcionar una evaluación automática del riesgo de suicidio en base a las respuestas de los cuestionarios de SIVARIA. Dicha aplicación estará dirigida principalmente a jóvenes, pero también pueden acceder profesionales, y familiares y amigos del paciente, para monitorizar la situación de su paciente o familiar.

2.6. Sistema Experto

Teniendo en cuenta los objetivos del proyecto, el primer paso fue diseñar la infraestructura del Sistema Experto. Para lograr dicho propósito, era necesario realizar una investigación para aprender los fundamentos de un Sistema Experto, estudiar qué tipos de sistemas pueden haber y cuál sería el tipo elegido para los modelos, que se verá a continuación.

Un sistema experto (Wikipedia (2024)) es un programa informático diseñado para simular el conocimiento y las habilidades analíticas de un especialista en un campo concreto, pudiendo tomar decisiones complejas respecto a un problema.

Dentro del concepto de sistema experto, existen los siguientes tipos:

- **Reglas previamente establecidas o RBR (*Rule Based Reasoning*)**: en base a la definición proporcionado por Grosan y Abraham (2011), son sistemas en el que el conocimiento adquirido es representado mediante reglas IF. Por ejemplo, dado una condición A y las conclusiones B y C, la estructura sería:

$$\text{if } A \text{ then } B \text{ else } C$$

Un gran inconveniente de este tipo de sistema requiere de una coincidencia exacta entre el problema planteado y las precondiciones establecidas para poder predecir las conclusiones, por lo que puede llegar a ser un factor limitante en grandes proyectos.

- **Basados en casos o CBR (*Case Based Reasoning*)**: de acuerdo a la definición del artículo Kolodner (1992), un sistema experto basado en CBR aprende de experiencias anteriores, en lugar de reglas, como sucedía con los sistemas RBR. De esta manera, cuando se le presenta un problema al modelo, este empleará las experiencias con las que se le ha entrenado para elaborar una solución nueva.
- **Basados en redes bayesianas**: es un sistema cuyas decisiones estarán basadas en un modelo probabílistico conformado por variables y relaciones de dependencia entre ellas. De esta manera, cuando se le presenta un problema, el sistema realizará su predicción en base a las probabilidades de las variables que se habrán calculado previamente en base a los entrenamientos que el sistema haya recibido.

En nuestro caso, se ha decidido implementar el Sistema Experto mediante el uso de redes bayesianas, ya que presentan mayor flexibilidad que los otros tipos. En el caso de los modelos RBR y CBR presentan demasiada rigidez, ya que es necesario especificar las reglas y los casos explícitamente para que realice las predicciones correctamente.

Durante esta fase de investigación, se realizó un proceso de aprendizaje sobre los sistemas expertos y las redes bayesianas. Se leyó documentación para aprender y entender el concepto de redes bayesianas y sus componentes.

Con respecto al proyecto, el objetivo de este sistema es:

- Identificar y, por lo tanto, predecir, aquellos jóvenes que cumplen con las condiciones para ser considerados de alto riesgo (pronóstico).
- Monitorizar sus conductas de riesgo mediante evaluación momentánea (Ecological Momentary Assessment) (monitorización).
- Prevenir que se produzcan conductas autolesivas futuras (prevención), mediante la provisión de indicaciones y actuaciones personalizadas y ajustadas a los niveles de riesgo bajo la supervisión de profesionales.

Por otro lado, se ha propuesto que la estructura del sistema sea de tal forma que reciba una serie de datos de entrada, como se muestra en la Figura 2.1.

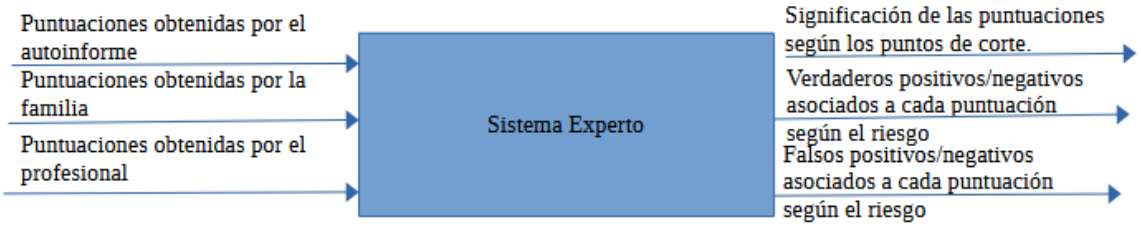


Figura 2.1: Estructura del Sistema Experto

El sistema recibirá una serie de puntuaciones obtenidas de los cuestionarios realizados a los usuarios por el autoinforme, a las familias de los usuarios y a los profesionales de la salud (médicos, piscólogos, etc.). Estos formularios provienen del sitio web de SIVARIA (Figuras 2.2 y 2.3).

Figura 2.2: Página inicial del sitio web de SIVARIA

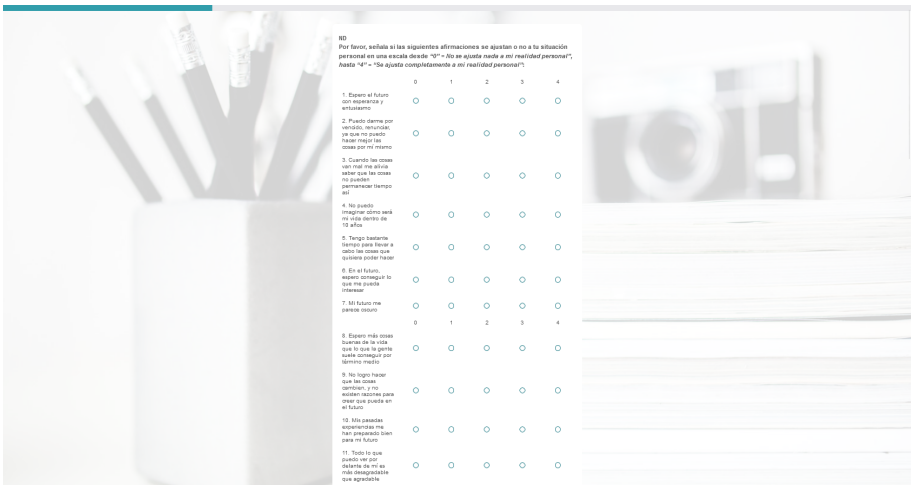


Figura 2.3: Encuesta a jóvenes de más de 16 años de SIVARIA

En base a estos datos y al entrenamiento realizado, el sistema los procesará y devolverá el número de verdaderos y falsos positivos y negativos. Esto será interpretado por la aplicación para determinar si existen actuales o potenciales conductas autolesivas suicidas y no suicidas.

2.7. Redes de Bayes

Como se ha explicado anteriormente, se ha optado por implementar el Sistema Experto mediante una red de Bayes (Sucar y Tonantzintla (2006)).

Las redes bayesianas, propuesto inicialmente por Judea Pearl en 1988, son un modelo probabilístico en el que se representan las relaciones probabilísticas entre las variables (nodos) y las dependencias condicionales que hay entre ellas, formando de esta manera un grafo acíclico dirigido (DAG, *Directed Acyclic Graph*).

Para que un grafo pueda ser considerado un DAG, todos los arcos (*edges*) del grafo

deben ser dirigidos, es decir, que indiquen una sola dirección. Además, el grafo no debe tener ciclos, en el que dado un vértice v, no hay un camino directo que empiece y termine en v.

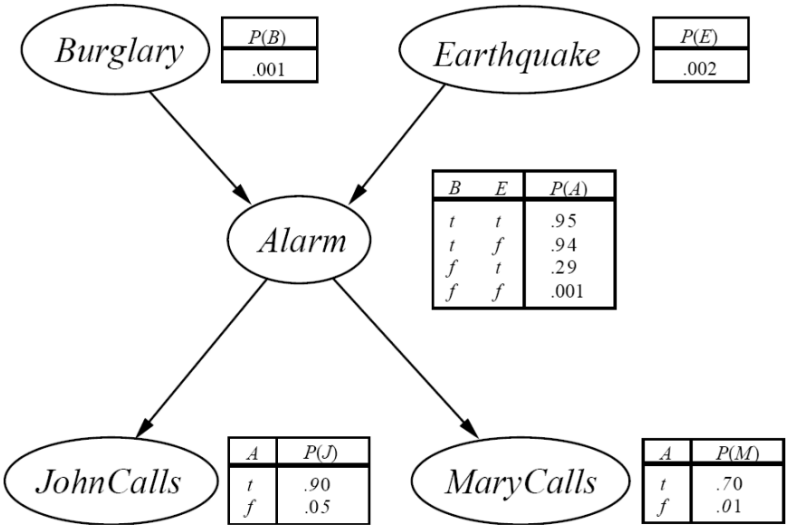


Figura 2.4: Ejemplo de un Diagrama de Bayes[8].

Dentro de una red bayesiana, destacan los conceptos de probabilidad conjunta y probabilidad condicional.

2.7.1. Probabilidad conjunta

La probabilidad conjunta es la probabilidad de que n eventos o variables ocurran simultáneamente.

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{p(i)}) \tag{2.1}$$

Donde $X_{p(i)}$ es el padre de X_i.
Por ejemplo, dado:

- J = *John Calls.*
- M = *Mary Calls.*
- A = *Alarm.*
- B = *Burglary.*
- E = *Earthquake.*

La probabilidad conjunta de la Figura 2.4 sería:

[8]Fuente: `https://www.cs.ubc.ca/~hkhosrav/ai/slides/chapter14.pdf`

$$\begin{aligned} P(J,M,A,B,E) &= P(J|A)P(M,A,B,E) \\ &= P(J|A)P(M|A)P(A,B,E) \\ &= P(J|A)P(M|A)P(A|B,E)P(B,E) \\ &= P(J|A)P(M|A)P(A|B,E)P(B)P(E) \end{aligned} \tag{2.2}$$

2.7.2. Probabilidad condicional

La probabilidad condicional es la probabilidad de que un evento A ocurra dado otro evento B, $P(A|B)$.

Para calcular cualquier probabilidad condicional, se utiliza el Teorema de Bayes:

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(A_i) \cdot P(B|A_i)}{P(B)} \tag{2.3}$$

Donde $P(A_i)$ son las probabilidades a priori del espacio muestral, es decir, las probabilidades iniciales que ya sabemos su valor, y $P(A_i|B)$ son las probabilidades a posteriori, es decir, posibilidad de que suceda un resultado B dado un conjunto de eventos $A = \{A_1, A_2, ..., A_i\}$.

2.7.3. Distribuciones de probabilidad condicional

Los conceptos de probabilidad conjunta y condicional son de utilidad para entender el funcionamiento de las tablas de distrbución de las redes bayesianas.

Una tabla de distribución de la probabilidad condicional (*Conditional Probability Distribution, CPD*) es una de las características fundamentales de las redes bayesianas. En ella, se especifica la probabilidad de cada posible valor de una variable dada una combinación específica de los valores de sus variables padres.

Tomando como ejemplo la red de la Figura 2.4, una CPD sería la tabla que se encuentra al lado de la variable *Alarm*. En ella, se puede observar que es dependiente de *Burglary* y *Earthquake*, por lo que su probabilidad queda condicionada por la probabilidad conjunta de estas dos variables.

Burglary	**Earthquake**	**P(Alarm=T)**	**P(Alarm=F)**
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

Tabla 2.2: Ejemplo de CPD de la variable *Alarm*

2.7.4. Inferencias

Una inferencia es un razonamiento probabilístico que consiste en asignar valores a ciertas variables (evidencia), y se obtiene la probabilidad posterior de las demás variables dadas las variables conocidas.

2.8. Aprendizaje supervisado

El aprendizaje supervisado o *machine learning supervisado* es una subcategoría de la Inteligencia Artificial. Se caracteriza por el uso de conjuntos de datos etiquetados para que el modelo pueda detectar patrones dentro del entrenamiento, y así realizar predicciones con alta precisión. Se ha decidido por este tipo de aprendizaje dentro de scikit learn debido a la naturaleza de los datasets, ya que los datos que contienen ya vienen etiquetados, tanto los de entrada como los de salida. Dentro de la librería scikit learn se ofrecen diversos algoritmos de clasificación que permiten prededcir un resultado. Aún así, tiene la desventaja de que requiere realizar modificaciones manuales dentro del dataset para categorizar los datos o corregir datos que no estén categorizados.

2.9. Testeo del entrenamiento del modelo

Para medir el rendimiento de nuestro modelo, se debe comparar las respuestas predichas con las respuestas reales del conjunto de testeo.

Esto nos devuelve el número de verdaderos y falsos positivos, y verdaderos y falsos negativos, que se usarán internamente para calcular un ratio de acierto comprendido entre 0 y 1. Existen cuatro métricas principales para calcular dicho ratio. Estos son:

- **Exactitud** (*Accuracy*): mide la proporción de predicciones correctas (tanto verdaderos positivos como verdaderos negativos) entre el total de predicciones realizadas.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

  Donde:

  - TP: Verdaderos Positivos (*True Positives*)
  - TN: Verdaderos Negativos (*True Negatives*)
  - FP: Falsos Positivos (*False Positives*)
  - FN: Falsos Negativos (*False Negatives*)

- **Precisión**: mide la proporción de verdaderos positivos entre todas las predicciones positivas hechas por el modelo. Es decir, de todas las veces que el modelo predijo que una instancia es positiva, cuántas veces acertó.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

- **Sensibilidad** (*Recall*): mide la proporción de verdaderos positivos entre todas las instancias que son realmente positivas. Es decir, de todas las veces que una instancia es realmente positiva, cuántas veces el modelo la detectó correctamente.

$$Sensibilidad = \frac{TP}{TP + FN} \tag{2.6}$$

- **F1 Score**: es la media armónica entre la precisión y la sensibilidad. Es bastante útil en contextos donde se necesita un equilibrio entre la precisión y la sensibilidad, o cuando hay una distribución desequilibrada de clases.

$$F1 = 2 \cdot \frac{Precision \cdot Sensibilidad}{Precision + Sensibilidad} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.7}$$

Todas estas métricas ya están predefinidos dentro del módulo scikit-learn (*sklearn.metrics*), así que no hace falta calcularlos de forma manual. Dentro de la función de cálculo de las métricas predomina el parámetro *average*, que sirve para especificar la forma en la que se debe calcular la media de cada categoría. Dentro de este parámetro, destacan dos valores para este proyecto:

- *micro*: con este enfoque, se suman todos los verdaderos y falsos positivos y negativos a nivel global, sin ponderar la cantidad que haya en cada clase. Esto puede provocar que los valores de todas las métricas coincidan.

$$Precision_{micro} = \frac{\sum TP}{\sum (TP + FP)} \tag{2.8}$$

$$Recall_{micro} = \frac{\sum TP}{\sum (TP + FN)} \tag{2.9}$$

$$F1_{micro} = 2 \cdot \frac{Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}} \tag{2.10}$$

- *weighted*: en este caso, las métricas se calculan para cada clase individualmente, y luego se ponderan con respecto a un soporte (número de verdaderos ejemplos en cada clase). Esto significa que las métricas de clases más grandes tienen más influencia en el promedio final.

$$Precision_{weighted} = \sum (\frac{Soporte_i}{Soportetotal} \cdot Precision_i) \tag{2.11}$$

$$Recall_{weighted} = \sum (\frac{Soporte_i}{Soportetotal} \cdot Recall_i) \tag{2.12}$$

$$F1_{weighted} = \sum (\frac{Soporte_i}{Soportetotal} \cdot F1_i) \tag{2.13}$$

Existen otros valores aparte de los dos ya mencionados. Estos son *macro*, *binary* y *samples*. Como son valores que no encajaban con la naturaleza ni la distribución de los datos con los que trabajamos, se optó por descartarlos. Este parámetro se irá ajustando a lo largo de las iteraciones del entrenamiento.

2.10. Validación cruzada

Una forma de poder validar el entrenamiento de un modelo es usando la validaciónn cruzada de la librería scikit-learn. La validación cruzada es una técnica para asegurar que el modelo no solo se ajusta a los datos de entrenamiento, sino también a los datos no vistos, evitando así una situación de **overfitting**, es decir, la posibilidad de que el modelo sólo sea capaz predecir datos que ya ha visto. Consiste en dividir el conjunto de datos en varios subconjuntos o *folds*, que se van combinando entre ellos para entrenar al modelo y evaluarlo con los otros restantes. Este proceso se repite varias veces entre todos los candidatos, obteniendo de esta manera, un ratio con mayor robustez en el rendimiento del modelo. Este método también se utilizará para calcular el margen de error de los entrenamientos.

[9] Fuente: `https://qu4nt.github.io/sklearn-doc-es/modules/cross_validation.html`

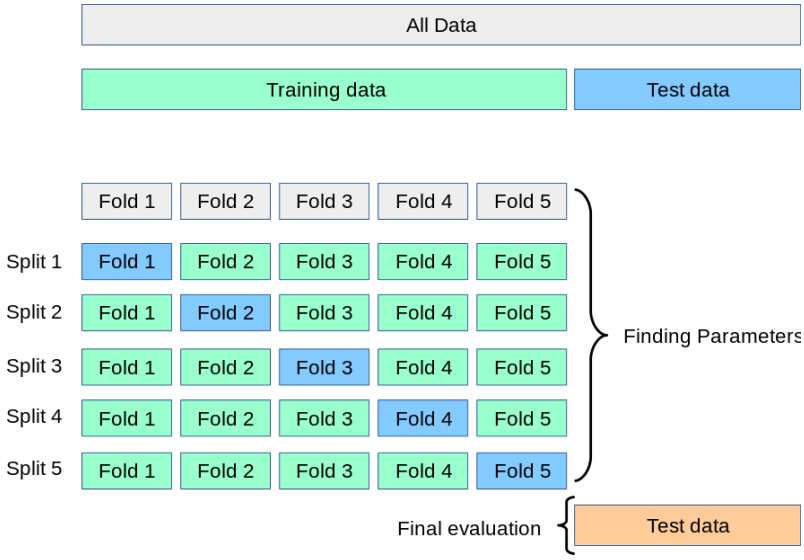


Figura 2.5: Validación cruzada de la librería Scikit-learn[9].

Capítulo 3

Descripción del Trabajo

3.1. Diseño del modelo

En esta sección se explicará el proceso que se siguió para diseñar los diagramas de Bayes. En un principio, tenía pensado diseñar un sólo diagrama común para los usuarios de los jóvenes, familias y profesionales. Sin embargo, el problema era que en la página de SIVARIA[1] habían cuestionarios que eran únicos para cada tipo de usuario, como el cuestionario PARQ, que sólo estaba disponible para las familias. Por lo tanto, las variables que necesitaba cada tipo de usuario eran ligeramente distintos, y para que un modelo pueda predecir correctamente, es necesario que todos tengan las mismas variables. De lo contrario, habrían variables presentes en unos y en otros no, lo que daría errores a la hora de predecir un resultado.

Ante esta dificultad, opté por la creación de tres modelos separados: modelo de los jóvenes, familias y profesionales. De esta manera, las variables estarían separadas y diferenciadas entre sí, haciendo que el entrenamiento fuera correcto para todos los modelos. A continuación, se van a mostrar los diagramas resultantes, así como las variables y valores de cada uno. Cabe resaltar que todas las variables y sus valores de los modelos se han basado en la documentación inicial proporcionada por el director del proyecto. Asimismo, los valores de las variables, además de la documentación, se han escogido teniendo en cuenta las preguntas de los cuestionarios de los jóvenes, familias y profesionales del sitio web de SIVARIA.

A continuación se van a mostrar el diseño inicial de los modelos para los jóvenes (Autoinforme), el de las familias y el de los profesionales. De todas formas, estos modelos están sujetas a cambios, como se podrá observar posteriormente en el entrenamiento de los modelos.

3.1.1. Modelo del Autoinforme

El modelo va a constar de los siguientes grupos de variables:

- Variables de identificación:
    - Curso: (PRIMARIA, ESO, BACHILLERATO,UNIVERSIDAD, FORMACION PROFESIONAL).

[1] `https://blogs.uned.es/investigacioninfantojuvenil/sivaria-gestion-del-suicidio/`

- Variables sociodemográficas:
  - Sexo asignado: (HOMBRE, MUJER, OTRO).
  - Transgénero: (SÍ, NO, NO ESTOY SEGURO DE SER TRANS, NO ESTOY SEGURO DE LO QUE SE PREGUNTA).
  - Edad: (MENOR DE 12, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, MAYOR DE 21).
  - Situación laboral padre: (NO TRABAJA, TRABAJA, PENSIONADO).
  - Situación laboral madre: (NO TRABAJA, TRABAJA, PENSIONADO).
  - Nivel profesional padre: (BAJO, MEDIO, ALTO).
  - Nivel profesional madre: (BAJO, MEDIO, ALTO).
  - Nivel promedio del rendimiento académico: (INSUFICIENTE, SUFICIENTE, NOTABLE, SOBRESALIENTE, EXTRAORDINARIO).
  - Nivel de autopercepción masculina: (0, 1, 2, 3, 4, 5, 6). Esta variable indica cuánto se percibe el joven como hombre, yendo de 0 (nada) a 6 (completamente).
  - Nivel de autopercepción femenina: (0, 1, 2, 3, 4, 5, 6). Esta variable indica cuánto se percibe el joven como mujer, yendo de 0 (nada) a 6 (completamente).
  - Nivel de heteropercepción masculina: (0, 1, 2, 3, 4, 5, 6). Esta variable indica cuánto es percibida como hombre por otras personas, yendo de 0 (nada) a 6 (completamente).
  - Nivel de heteropercepción femenina: (0, 1, 2, 3, 4, 5, 6). Esta variable indica cuánto es percibida como mujer por otras personas, yendo de 0 (nada) a 6 (completamente).
- Variables de salud:
  - Altura (cm): (MENOS DE 149, 150-159, 160-169, 170-179, 180-189, MAS DE 190).
  - Peso (kg): (MENOS DE 49, 50-59, 60-69, 70-79, 80-89, MAS DE 90).
  - Tratamiento psiquiátrico previo: (SÍ, NO).
  - Presenta enfermedad crónica: (SÍ, NO).
- Variables de bullying: del cuestionario EBIPQ/ECIPQ (*European Bullying Intervention Project Questionnaire*) y ECIPQ (*European Cyberbullying Intervention Project Questionnaire*). Ortega-Ruiz et al. (2016).
  - Víctima de bullying: (SÍ, NO).
  - Perpetrador de bullying: (SÍ, NO).
  - Víctima de Cyberbullying: (SÍ, NO).
  - Perpetrador de Cyberbullying: (SÍ, NO).
- Variables de adicción o abuso: perteneciente al cuestionario MULTICAGE CAD-4. Pérez et al. (2007).
  - Adicción o abuso alcohol: (SÍ, NO).
  - Adicción o abuso sustancias: (SÍ, NO).
  - Adicción o abuso Internet: (SÍ, NO).

- Psicopatología: corresponde con el cuestionario SENA (*Sistema de Evaluación de Niños y Adolescentes*). Fernández-Pinto et al. (2015)
  - Problemas interiorizados: (SÍ, NO).
  - Problemas exteriorizados: (SÍ, NO).
  - Problemas de contexto: (SÍ, NO).
  - Problemas de recursos psicológicos: (SÍ, NO).
- Variables del constructo del comportamiento suicida:
  - Percepcion de discriminación: (SÍ, NO).
  - Fuente de discriminación: (EDAD, RAZA, DISCAPACIDAD, GENERO, ORIENTACION SEXUAL).
- Variables de regulación y resistencia: cuestionario CERQS (*Cognitive Emotion Regulation Questionnaire*). Garnefski y Kraaij (2006)
  - Nivel de resistencia o resiliencia: (1, 2, 3, 4, 5): cuestionario ER (*Escala de Resiliencia*). Sánchez et al. (2016). Se refiere a la capacidad de soportar los contratiempos, adaptarse positivamente y recuperarse de la adversidad.
  - Nivel de regulación positiva: (1, 2, 3, 4, 5). Consiste en la capacidad de controlar nuestras emociones positivas.
  - Nivel de regulación negativa: (1, 2, 3, 4, 5). Consiste en la capacidad de controlar nuestras emociones negativas.
- Variables volitivo-motivacionales para el suicidio
  - Atrapamiento interno: (SÍ, NO): escala ATI. Gilbert y Allan (1998). Hace referencia al nivel de percepción que tiene el joven de estar atrapado en su propia mente, pensamientos o emociones negativas.
  - Atrapamiento externo: (SÍ, NO): escala ATE. Gilbert y Allan (1998). Hace referencia al nivel de percepción que tiene el joven de estar atrapado en circunstancias externas o situaciones de la vida de las que una persona siente que no puede escapar.
  - Nivel percibido de derrota o fracaso: (BAJO, MEDIO, ALTO): escala ED (*Escala de Derrota*).
  - Sentimiento de pertenencia frustada: (SÍ, NO): cuestionario INQ-15 (*Interpersonal Needs Questionnaire - 15*). Hill et al. (2015)
  - Percepción de ser una carga: (SÍ, NO): cuestionario INQ-15 (*Interpersonal Needs Questionnaire - 15*). Hill et al. (2015)
  - Autoeficiencia para el suicidio: (SÍ, NO). Basado en el cuestionario FASM.
- Perfil de riesgo psicosocial
  - Madre adolescente: (SÍ, NO)
  - Padre adolescente: (SÍ, NO)
  - Padres divorciados: (SÍ, NO)
  - Familia monoparental: (SÍ, NO)
  - Tratamiento psicológico de padre o madre: (SÍ, NO)

- Adicción de padre o madre: (SÍ, NO)
- Relaciones conflictivas del hijo con padre o madre: (SÍ, NO)
- Familia reconstruida: (SÍ, NO)

- Tecnologías y RRSS
  - Búsqueda información autolesión: (SÍ, NO)
  - Petición de ayuda en Internet: (SÍ, NO)
  - Compartir en RRSS pensamientos sobre autolesión: (SÍ, NO)
  - Realización de autolesión después de ver un contenido en Internet: (SÍ, NO)
  - Conocidos que comparten autolesión en Internet: (SÍ, NO)
  - Contacto con información sobre autolesión: (SÍ, NO)
  - Denuncia a otra persona cometiendo autolesión en Internet: (SÍ, NO)
- Desenlace: (NINGUNO, AUTOLESIÓN, COMUNICACIÓN DE SUICIDIO, DESEO DE SUICIDIO, IDEACIÓN DE SUICIDIO, PLANIFICACIÓN DE SUICIDIO, INTENCIÓN DE SUICIDIO)

Debido al elevado número de variables que hay en el diagrama del Autoinforme, se ha dividido en tres partes para que sea más fácilmente visible en la memoria. De todas formas, la imagen completa de este diagrama, al igual que los de los otros modelos, también se encuentran en el repositorio de Github, dentro de la carpeta *DiagramBayesSketches/v1*.

Los diagramas se encuentran en las Figuras 3.1, 3.2 y 3.3.

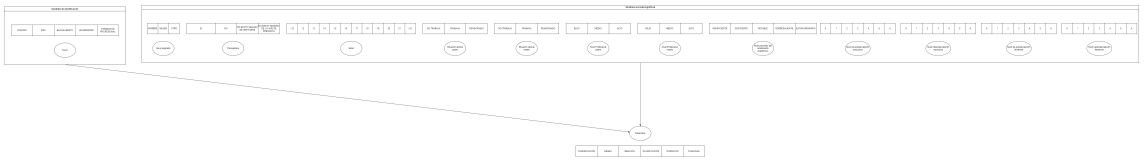


Figura 3.1: Red de Bayes del modelo del Autoinforme (I)

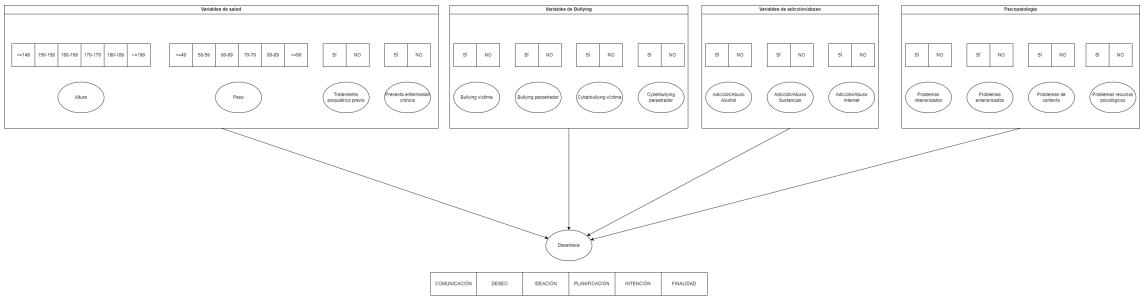


Figura 3.2: Red de Bayes del modelo del Autoinforme (II)

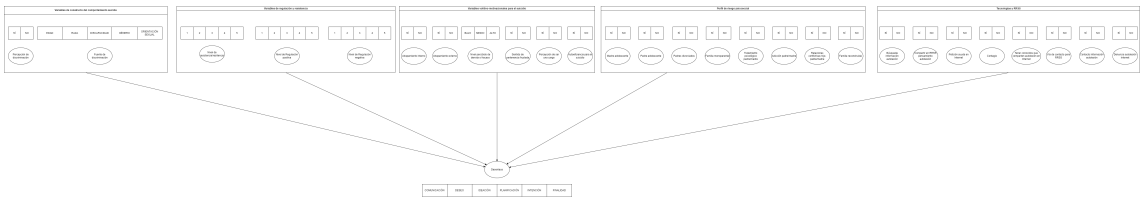


Figura 3.3: Red de Bayes del modelo del Autoinforme (III)

3.1.2. Modelo de las familias

El modelo va a constar de los siguientes grupos de variables con sus posibles valores:

- Perfil de riesgo psicosocial
- Psicopatología
- Variables de las familias: cuestionario PARQ-S (*Child-Parental Acceptance-Rejection Questionnaire - Short*). Del Barrio et al. (2014).
    - Aceptación/Rechazo parental: (ACEPTACIÓN, RECHAZO)
    - Control parental: (SÍ , NO).
- Desenlace

El diagrama se encuentra representado en la Figura 3.4.

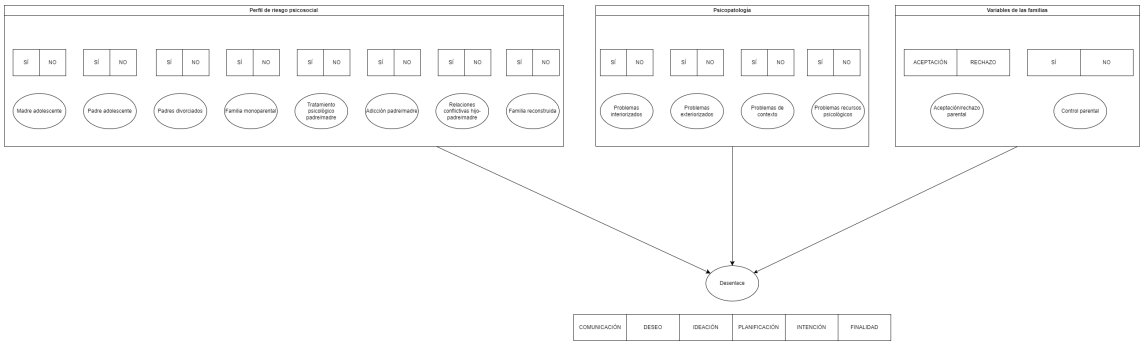


Figura 3.4: Red de Bayes del modelo de las familias

3.1.3. Modelo de los profesionales

El modelo va a constar de los siguientes grupos de variables

- Perfil de riesgo psicosocial
- Variables de los profesionales
    - Situación económica familiar precaria: (SÍ, NO)
    - Estudios de la madre: (SÍ, NO)
    - Estudios del padre: (SÍ, NO)

- Supervisión parental insuficiente: (SÍ, NO)
- Maltrato al adolescente: (SÍ, NO)
- Duelo: (SÍ, NO)
- Ingreso familiar mensual: (<=499, 500-999, 1000-1499, 1500-1999, 2000-2499, >=2500)

- Desenlace

El diagrama se encuentra representado en la Figura 3.5.

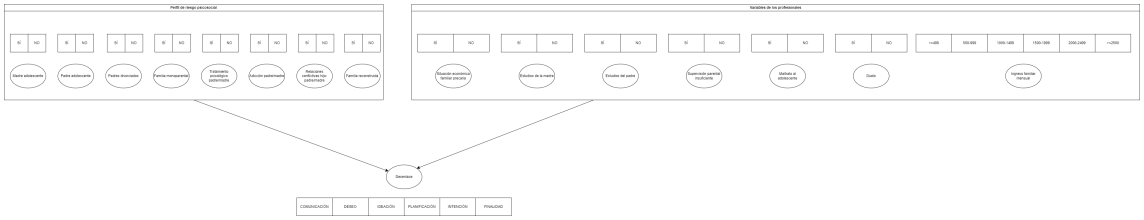


Figura 3.5: Red de Bayes del modelo de los profesionales

3.2. Desarrollo de los modelos

Para el desarrollo de los modelos, decidí utilizar Python como lenguaje de programación principal, ya que ofrece varias librerías de código abierto para la creación, entrenamiento y testeo de redes bayesianas, como *pybnesian* o *pomegranate*. No obstante, los descarté por mi falta de familiaridad en estos módulos, además de su complejidad a la hora de crear y entrenar los modelos. Finalmente, me decanté por emplear una librería llamada *pgmpy* (Ankan y Panda (2015)), ya que es una librería creada específicamente para el desarrollo de diagramas de Bayes. Además, permitía poder especificar con mayor exactitud las relaciones entre las variables y, por lo tanto, crear redes de Bayes de todo tipo.

Sin embargo, al final lo tuve que descartar debido a que me daba errores de memoria a la hora de crear el diagrama, y las pocas veces que lograba crearse, luego tardaba demasiado tiempo en entrenarlo y guardar el modelo, a menudo llegando incluso a congelarse durante la ejecución. Esto se debía al tamaño del diagrama, que ocupaba demasiado espacio de memoria. Es por ello, que tuve que buscar otras alternativas. Finalmente, decidí implementarlo con scikit-learn (Pedregosa et al. (2011)), ya que, a pesar de que no es una librería dedicada a las redes de Bayes, sí que contiene un apartado para entrenar modelos bayesianos de forma dinámica, que es el uso del modelo Naive Bayes.

Naive-Bayes es un modelo de clasificación de aprendizaje automático que asume la independencia de las características (variables) para predecir a qué categoría (desenlace) pertenece un conjunto de datos. Este algoritmo utiliza internamente el Teorema de Bayes (2.3) para llevar a cabo las predicciones.

El modelo Naive-Bayes de scikit-learn funciona de la siguiente manera:

1. Durante el entrenamiento, se crea una tabla de frecuencia y, posteriormente, una tabla de probabilidad, utilizando los datos del dataset. Dichas tablas contiene un conteo de las veces que ha sucedido un desenlace final, así como el conteo de las variables que han provocado ese desenlace.

2. Una vez creada la tabla, durante el periodo de prueba, el modelo debe realizar predicciones para compararlas con los datos reales y así saber su nivel de acierto. Para ello, debe realizar una serie de predicciones, que las hará calculando la probabilidad posterior de cada categoría mediante el Teorema de Bayes, asumiendo la independiencia de todas las características. Aquella categoría del modelo que obtenga mayor probabilidad de todas, es la que se selecciona finalmente.

En la Figura (3.6) se puede observar un ejemplo de las tablas que se generarían siguiendo el proceso mencionado anteriormente.

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Figura 3.6: Ejemplo de la creación de una tabla de frecuencia (segunda tabla) y tabla de probabilidad (tercera tabla) dado un dataset (primera tabla)[2].

3.2.1. Datasets

Antes de comenzar con el desarrollo de los scripts, hubo un inconveniente con respecto a los datasets que se tenían que usar para el entrenamiento. En un principio se iba a contar con una serie de datasets proporcionados por los profesionales de la salud mental que estaban involucrados en este proyecto. Sin embargo, dichos datasets finalmente no fueron proporcionados. Por lo tanto, ante esta falta importante de datos para tomarlos de referencia, tuve que buscar otras alternativas que se podrán ver a lo largo de las iteraciones de los entrenamientos. En un primer lugar, opté por generar yo mismo mis propios datasets de forma artificial y aleatoria, creándome scripts de Python para su creación. Sin embargo, los ficheros resultantes tenían presentes una alta aleatoriedad en los datos, lo que deriva en información poco realista y con potenciales incoherencias. Como consecuencia, el entrenamiento se veía muy perjudicado, y por consiguiente, las predicciones y la evaluación del modelo no serían correctas.

Es por ello, que a partir de la segunda iteración, opté por buscar otras soluciones. Finalmente, me decanté por tratar de generar estos datasets de entrenamiento mediante el uso de un LLM (*Large Language Model*). Utilicé ChatGPT para generar los datos,

[2]Fuente del ejemplo: `https://www.javatpoint.com/machine-learning-naive-bayes-classifier`

más concretamente, el modelo GPT-4o, ya que era la versión más moderna y veloz que puede ofrecer OpenAI a día de hoy. Para ello, se crearon tres versiones de GPT para los jóvenes, familias y profesionales. Posteriormente, se intrdujeron prompts iniciales en cada una de ellas para generar las primeras versiones de los datasets, y se iban ajustando progresivamente a través de los continuos chats que se enviaban. Es importante mencionar que, a pesar de que esta solución es más sofisticada y podría dar mejor rendimiento, se presentaron ciertas limitaciones, ya que ChatGPT tiene restringido el uso de GPT-4 y GPT-4o para 40 mensajes. Si se superaba esa cantidad, OpenAI bloqueaba el modelo durante 5 horas o más, dependiendo del tráfico que había en la herramienta. Esto presentaba una desventaja a la hora de generar de datasets con un elevado número de filas.

Finalmente, tanto los primeros datasets generados por el script como los que han sido generados por el LLM, también están a disposición pública en el repositorio de Github. Al ser datos artificiales, no se requiere aplicar medidas de protección de datos, al no haber información sensible que proteger. Se ubicarán dentro de la carpeta *scripts/datasets* y estarán distribuidos dependiendo de cada versión del entrenamiento y del tipo de modelo que se pretende entrenar. Del mismo modo, todas las versiones de los entrenamientos realizados se ubican en cuadernos de Jupyter, también dentro de la carpeta *scripts*. De esta manera, por ejemplo, si se quiere llevar a cabo el entrenamiento del dataset de los familiares de la primera versión, habría que acceder al siguiente cuaderno de Jupyter: *scripts/first_version_model_training_scikit_learn.ipynb*. Y si se quiere acceder al contenido de dicho dataset de las familias para saber que dataset se pueden utilizar, se puede acceder al siguiente path: *scripts/datasets/familia/dataset1.csv*.

3.2.2. Procesos de entrenamiento y testeo del modelo

El entrenamiento se dividirá en versiones, y dentro de cada una, en iteraciones. Esta organización se hizo para diferenciar los entrenamientos cada vez que había un cambio relevante en el diseño del modelo o en la manera de generar los datasets. Cada iteración seguirá una serie de pasos específicos siguiendo una estructura definida:

1. Los datos procedentes del dataset se dividirán en dos subconjuntos: los datos de entrenamiento y de prueba. Para saber qué porcentaje de los datos corresponden a un proceso u otro, se especificará un parámetro, que será un ratio de 0 a 1 e irá variando a lo largo de las iteraciones.

2. Se crea el modelo y se entrena con los datos de entrenamiento. Al principio, el modelo se creará con los hiperparámetros por defecto, aunque es recomendable que posteriormente se haga un ajuste de hiperparámetros para encontrar los valores más adecuados para obtener un mejor entrenamiento.

3. Se realiza el testeo del modelo, introduciendo los datos de prueba para obtener una predicción del posible desenlace de cada fila.

4. Se genera la matriz de confusión normalizada, comparando los desenlaces obtenidos de la predicción con los que ya habían en los datos de prueba.

5. Se calcula el número de verdaderos y falsos positivos y negativos, para evaluar de forma numérica el número de fallos y aciertos por cada clase.

6. Se calculan las métricas *accuracy*, *precision*, *recall* y *f1_score* para obtener diversas puntuaciones del rendimiento del modelo.

7. Se valida el modelo y se calcula el error medio de los resultados mediante la validación cruzada.

8. Se guarda el modelo en un fichero .SAV que se ubicará en la carpeta correspondiente del *scripts/configFiles*. Dentro habrá una carpeta por cada tipo de modelo: autoinforme, familia y profesional. El formato del nombre del fichero de guardado será el siguente (Figura 3.7):

   model_<TIPO_DE_MODELO>_<VERSION>_<FECHA_Y_HORA_DE_CREACION>.sav.

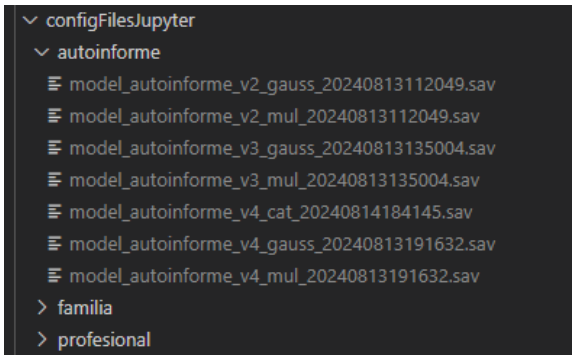


Figura 3.7: Ejemplo del directorio de archivos de guardado con el modelo del Autoinforme

   De esta manera, se consigue que haya un histórico de todos los entrenamientos del modelo, permitiendo así un control de versiones y pudiendo borrar la versión del modelo que se considere necesario.

3.2.2.1. Clasificadores utilizados

A lo largo de las versiones, se han utilizado distintos tipos de clasificadores Naive-Bayes pertenecientes a la librería *scikit-learn*, dependiendo de la naturaleza de los datos y su distribución en los datasets. Estos son:

- **Clasificador gausiano** (*GaussianNB*): en este clasificador, se asume que los datos siguen una distribución normal y que son características continuas, es decir, variables que no tienen un conjunto finito de posibles valores. Por ejemplo, una variable de llamado *Víctima de bullying = {Sí, No}* es una variable discreta, porque su conjunto de posibles valores es finito, mientras que una variable *Peso = {50, 50.5, 51, 51.5, 52, 53, ...}* es una variable continua, porque la variable puede tener infinitos valores. Por lo tanto, la probabilidad en este clasificador, se calcula internamente de la siguiente manera:

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

  Donde:

  - x_i es la categoría posterior.
  - y es el conjunto de condiciones para que se dé la categoría.

- μ la media de la sumatoria de todos los valores w_i de esa columna.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{3.2}$$

- σ es la desviación estándar, cuyo cálculo es el siguiente:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(w_i - \mu)^2} \tag{3.3}$$

- e es la constante de Euler.

- **Clasificador multinomial** (*MultinomialNB*): el clasificador asume que los datos siguen una distribución multinomial, es decir, que cada celda del fichero representa la frecuencia con la que se repite el valor de esa columna para cada fila del dataset. Puede ser de utilidad con datasets que tengan características discretas y numéricas.
- **Clasificador categórico** (*CategoricalNB*): en este caso, se asume que los datos siguen una distribución categórica, en donde todas las características son variables discretas con valores fijos. En un dataset, cada fila representa un valor en concreto, a diferencia del clasificador multinomial, que cada columna de una fila guarda un conteo numérico. que provocan un desenlace. Cabe destacar que existe una variante de este clasificador, que es el Bernoulli (*BernoulliNB*). Sin embargo, es necesario que todos los datos tengan que ser binarios, es decir, que cada columna del dataset sólo puede tener dos posibles valores: (Sí, No), (0,1), etc.

3.3. Versiones del entrenamiento

En esta sección se llevará un registro de los entrenamientos realizados en las distintas versiones. Cada versión se compone de un número de iteraciones. Las iteraciones se iban realizando en base a los resultados que se obtenían de la anterior.

3.3.1. Primera versión: datasets artificiales y aleatorios

En esta primera versión empecé con un entrenamiento simple, utilizando los datasets generados de forma artificial a través de tres scripts de Python, uno por cada modelo. Estos archivos se encuentran dentro de la carpeta *scripts/csvV1Creator*. De esta manera, se generaron un total de cinco datasets para cada modelo, con 10000 filas cada una. Esta elevada cantidad de datos se debe a que no se tenía en cuenta la coherencia de los mismos, y porque el código desarrollado permitía añadir el número de filas que uno consideraba oportuno.

El código del entrenamiento se encuentra en el cuaderno de Jupyter:

scripts/first_version_model_training_scikit_learn.ipynb

Durante esta primera versión, se empezó utilizando el clasificador gausiano para realizar las predicciones, ya que al principio asumí que era la única variante de clasificador bayesiano que tenía *scikit-learn*, además de que dentro del dataset se encontraban datos que consideré continuos, como la edad, el peso o la altura. Sin embargo, como había otras variables que era discretas, entonces se tuvo que estandarizar todo el dataset. Es por ello que, internamente, se codifican los datos recibidos para que todos sean de tipo numérico, y así se pueda aplicar el modelo gausiano.

3.3.1.1. Iteración 1

En esta primera iteración, para tener un primer punto de partida, se inició con los siguientes parámetros por defecto:

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1.*
- **Balance del dataset**: 75 % - 25 %.
- ***average***: micro.

En base a estos parámetros, se obtuvieron las siguientes métricas para cada tipo de modelo, reflejados en la Tabla 3.1 y en las matrices de confusión de las Figuras 3.8, 3.9 y 3.10.

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.160	0.166	0.178	0.174	0.172
	Precision	0.182	0.160	0.166	0.178	0.174	0.172
	Recall	0.182	0.160	0.166	0.178	0.174	0.172
	F1 Score	0.182	0.160	0.166	0.178	0.174	0.172
Familia	Accuracy	0.174	0.168	0.175	0.176	0.182	0.175
	Precision	0.174	0.168	0.175	0.176	0.182	0.175
	Recall	0.174	0.168	0.175	0.176	0.182	0.175
	F1 Score	0.174	0.168	0.175	0.176	0.182	0.175
Profesional	Accuracy	0.166	0.176	0.159	0.153	0.171	0.165
	Precision	0.166	0.176	0.159	0.153	0.171	0.165
	Recall	0.166	0.176	0.159	0.153	0.171	0.165
	F1 Score	0.166	0.176	0.159	0.153	0.171	0.165

Tabla 3.1: Comparativa de las métricas del modelo. Iteración 1

Figura 3.8: Matriz de confusión del Autoinforme al final de la primera iteración

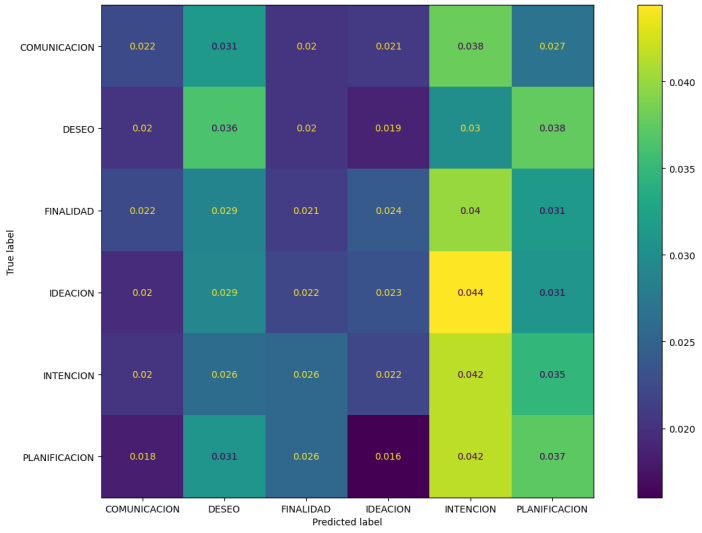


Figura 3.9: Matriz de confusión de las familias al final de la primera iteración

Figura 3.10: Matriz de confusión de los profesionales al final de la primera iteración

En estos resultados, se pueden observar que las matrices de confusión arrojan resultados dispares, con unas métricas considerablemente bajas. En el caso del Autoinforme, se puede observar que la mayoría de las filas se predijeron como COMUNICACIÓN, pero el nivel de error es alto, ya que muchos de los datos son realmente IDEACIÓN, INTENCIÓN o PLANIFICACIÓN. Lo mismo se aplica al resto de las clases. Acierta mucho por el elevado número de filas utilizados para entrenar, pero también falla bastante. Ocurre de la misma manera con el modelo de las familias, con la clase INTENCIÓN como la más predicha, y el de los profesionales, con la clase IDEACIÓN.

3.3.1.2. Iteración 2

Viendo que los resultados en la primera iteración fueron bajos, se consideró que se podía deber a un cambio en el cálculo de las métricas. Es por eso que en esta segunda iteración de entrenamiento se realizó un cambio en el cálculo del average de las métricas. En vez de usar la opción micro, se va a probar con la opción *weighted*[6], para darle más peso a las clases con mayor número de aciertos.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1.*
- **Balance del dataset**: 75 % - 25 %.
- ***average***: weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.160	0.166	0.178	0.74	0.172
	Precision	0.182	0.160	0.163	0.178	0.171	0.171
	Recall	0.182	0.160	0.166	0.178	0.174	0.172
	F1 Score	0.182	0.157	0.160	0.174	0.170	0.169
Familia	Accuracy	0.174	0.168	0.175	0.176	0.182	0.175
	Precision	0.175	0.172	0.173	0.175	0.181	0.175
	Recall	0.174	0.168	0.175	0.176	0.182	0.175
	F1 Score	0.173	0.159	0.168	0.172	0.179	0.170
Profesional	Accuracy	0.166	0.176	0.159	0.153	0.171	0.165
	Precision	0.153	0.179	0.161	0.159	0.173	0.165
	Recall	0.166	0.176	0.159	0.153	0.171	0.165
	F1 Score	0.130	0.163	0.147	0.148	0.159	0.149

Tabla 3.2: Comparativa de las métricas del modelo. Iteración 2

Los resultados no han sido los esperados, y se siguen obteniendo ratios bajos. Ante esto, una posible solución sería un cambio en el balance del dataset, para comprobar si el modelo necesita un mayor o menor número de filas de entrenamento o evaluación.

[6]Cabe resaltar que este parámetro influye sobre las métricas durante el testeo, pero no sobre el entrenamiento. Esto implica que los resultados del entrenamiento serán los mismos, y por lo tanto, las matrices de confusión no varían.

3.3.1.3. Iteración 3

En este tercer ciclo de entrenamiento, se ha decidido modificar el porcentaje de división del dataset. En este escenario, el 70 % del dataset será utilizado en el entrenamiento, y el 30 % restante para testeo. Esto se ha hecho con el objetivo de darle más datos para poder testear, y así obtener mejores métricas. Como el balance del dataset ha cambiado, los resultados del entrenamiento van a variar, por lo que las matrices de confusión también serán diferentes. Por otro lado, el valor del parámetro *average* para el cálculo de las métricas vuelve a ser *micro*.

Se espera que con este cambio, haya un aumento relativamente alto de los entrenamientos.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1*.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: micro.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.183	0.168	0.160	0.178	0.169	0.172
	Precision	0.183	0.168	0.160	0.178	0.169	0.172
	Recall	0.183	0.168	0.160	0.178	0.169	0.172
	F1 Score	0.183	0.168	0.160	0.178	0.169	0.172
Familia	Accuracy	0.172	0.165	0.166	0.175	0.183	0.172
	Precision	0.172	0.165	0.166	0.175	0.183	0.172
	Recall	0.172	0.165	0.166	0.175	0.183	0.172
	F1 Score	0.172	0.165	0.166	0.175	0.183	0.172
Profesional	Accuracy	0.164	0.169	0.169	0.159	0.174	0.167
	Precision	0.164	0.169	0.169	0.159	0.174	0.167
	Recall	0.164	0.169	0.169	0.159	0.174	0.167
	F1 Score	0.164	0.169	0.169	0.159	0.174	0.167

Tabla 3.3: Comparativa de las métricas del modelo. Iteración 3

Figura 3.11: Matriz de confusión del Autoinforme al final de la tercera iteración

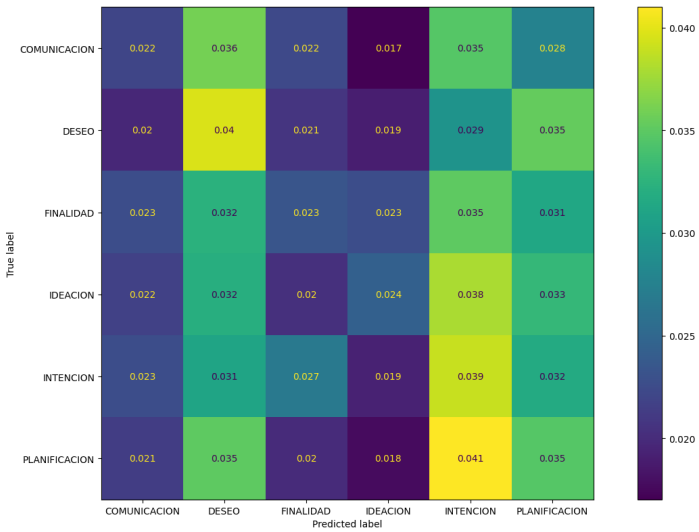


Figura 3.12: Matriz de confusión de las familias al final de la tercera iteración

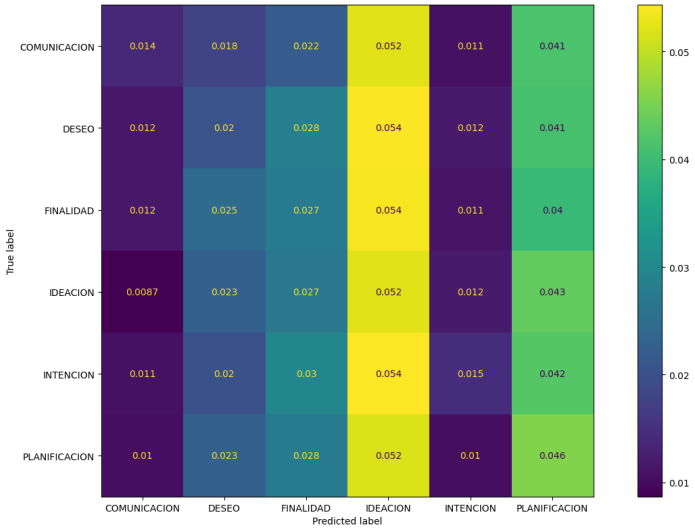


Figura 3.13: Matriz de confusión de los profesionales al final de la tercera iteración

Cambiando los parámetros se ha visto que no ha dado el resultado esperado. Las métricas continúan siendo bajas y las predicciones siguen arrojando resultados poco coherentes. La matriz de confusión de los jóvenes muestra de nuevo una mayor cantidad de predicciones en la clase PLANIFICACIÓN, pero falla al predecir muchas de ellas como PLANIFICACIÓN, al igual que con el resto de las clases de la matriz, y también con las matrices de los otros modelos. El modelo de las familias predice la mayoría de las filas como INTENCIÓN, pero las predice como PLANIFICACIÓN. Finalmente, el de los profesionales predice los datos a la clase IDEACIÓN, y falla al predecirlos como DESEO, FINALIDAD o INTENCIÓN.

3.3.1.4. Iteración 4

En este caso, se ha cambiado la forma de calcular las métricas para observar si había una variación en los ratios, y podría haber un ratio que devuelva un valor considerablemente mayor.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1*.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.183	0.168	0.160	0.178	0.169	0.172
	Precision	0.181	0.167	0.156	0.180	0.166	0.170
	Recall	0.183	0.168	0.160	0.178	0.169	0.172
	F1 Score	0.181	0.164	0.154	0.176	0.166	0.168
Familia	Accuracy	0.172	0.165	0.166	0.175	0.183	0.172
	Precision	0.172	0.164	0.164	0.176	0.183	0.172
	Recall	0.172	0.165	0.166	0.175	0.183	0.172
	F1 Score	0.171	0.155	0.159	0.171	0.181	0.167
Profesional	Accuracy	0.164	0.166	0.169	0.159	0.174	0.166
	Precision	0.144	0.176	0.172	0.156	0.181	0.166
	Recall	0.164	0.166	0.169	0.159	0.174	0.166
	F1 Score	0.132	0.155	0.159	0.154	0.164	0.153

Tabla 3.4: Comparativa de las métricas del modelo. Iteración 4

3.3.1.5. Iteración 5

En este ciclo, viendo los resultados de la iteración anterior, se decidió probar con otra división diferente del dataset. Esta vez la división será un 80 % para entrenamiento y un 20 % para testeo. Esto implica un cambio en el rendimiento del entrenamiento.

Este será el último balance del dataset, porque ya se está observando en iteraciones anteriores que los resultados apenas varían y no se puede ver un aumento claro en ellos. Por lo tanto, este balance servirá para probar si finalmente es necesario cambiar de enfoque o continuar con esta estrategia.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1.*
- **Balance del dataset**: 80 % - 20 %.
- ***average***: micro.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.164	0.168	0.180	0.177	0.174
	Precision	0.182	0.164	0.168	0.180	0.177	0.174
	Recall	0.182	0.164	0.168	0.180	0.177	0.174
	F1 Score	0.182	0.164	0.168	0.180	0.177	0.174
Familia	Accuracy	0.161	0.168	0.171	0.172	0.188	0.172
	Precision	0.161	0.168	0.171	0.172	0.188	0.172
	Recall	0.161	0.168	0.171	0.172	0.188	0.172
	F1 Score	0.161	0.168	0.171	0.172	0.188	0.172
Profesional	Accuracy	0.168	0.186	0.163	0.159	0.174	0.170
	Precision	0.168	0.186	0.163	0.159	0.174	0.170
	Recall	0.168	0.186	0.163	0.159	0.174	0.170
	F1 Score	0.168	0.186	0.163	0.159	0.174	0.170

Tabla 3.5: Comparativa de las métricas del modelo. Iteración 5

Figura 3.14: Matriz de confusión del Autoinforme al final de la quinta iteración

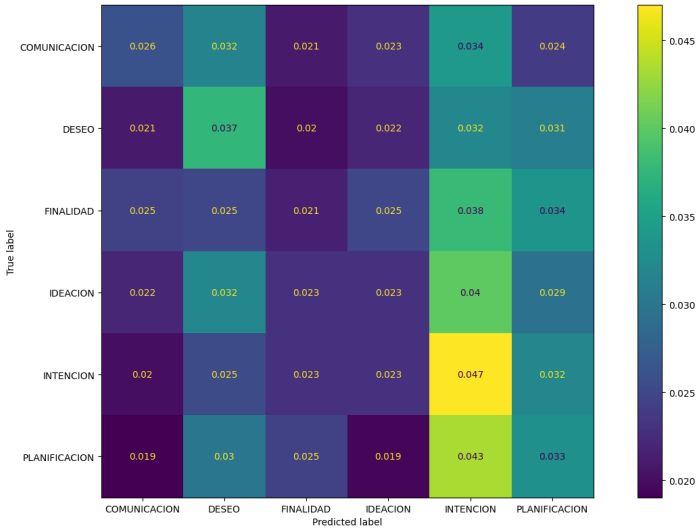


Figura 3.15: Matriz de confusión de las familias al final de la quinta iteración

Figura 3.16: Matriz de confusión de los profesionales al final de la quinta iteración

Finalmente, los resultados permanecen prácticamente iguales, con unas diferencias mínimas con respecto al resto de iteraciones. La aleatoriedad de los datos se mantiene visible en los resultados obtenidos hasta este momento.

3.3.1.6. Iteración 6

Finalmente, en esta iteración el balance se mantiene con respecto al ciclo anterior, con la diferencia de que el parámetro de cálculo de las métricas se vuelve a modificar de *micro* a *weighted*.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/<TIPO_DE_MODELO>/v1*.
- **Balance del dataset**: 80 % - 20 %.
- ***average***: weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.164	0.168	0.180	0.177	0.174
	Precision	0.180	0.165	0.168	0.179	0.174	0.173
	Recall	0.182	0.164	0.168	0.180	0.177	0.174
	F1 Score	0.179	0.162	0.162	0.177	0.172	0.170
Familia	Accuracy	0.159	0.168	0.175	0.175	0.185	0.172
	Precision	0.157	0.171	0.178	0.175	0.179	0.172
	Recall	0.159	0.168	0.175	0.175	0.185	0.172
	F1 Score	0.155	0.159	0.168	0.170	0.177	0.166
Profesional	Accuracy	0.168	0.186	0.163	0.159	0.174	0.170
	Precision	0.152	0.185	0.168	0.160	0.185	0.168
	Recall	0.168	0.186	0.163	0.159	0.174	0.170
	F1 Score	0.148	0.174	0.151	0.150	0.160	0.157

Tabla 3.6: Comparativa de las métricas del modelo. Iteración 6

Viendo los bajos resultados que se han obtenido de las seis primeras iteraciones, se decidió parar el entrenamiento en este punto, realizar una serie de conclusiones y buscar otras alternativas para mejorar estos resultados en próximas versiones.

3.3.1.7. Análisis inicial de los resultados

En esta sección, analizamos los resultados obtenidos de todas las iteraciones de entrenamiento, y se decide qué tipo de métrica se va a utilizar para el cálculo de la fiabilidad del Sistema Experto, el tipo de parámetro *average* (*micro* o *weighted*) y el porcentaje en la división del dataset en el futuro. Para llevar a cabo esta tarea, realicé una tabla comparativa de las medias aritméticas de las métricas de cada iteración para cada tipo de modelo (Autoinforme, familia y profesional). Se puede ver en la siguiente Tabla 3.7.

Analizando la tabla, concluí que la iteración que ofrece mejores resultados, es la quinta. El motivo principal es que en líneas generales, es el que devuelve un ratio más balanceado

Iteraciones	Modelo	Accuracy	Precision	Recall	F1 Score
Iteración 1	Autoinforme	0.172	0.172	0.172	0.172
	Familia	0.175	0.175	0.175	0.175
	Profesional	0.165	0.165	0.165	0.165
Iteración 2	Autoinforme	0.172	0.171	0.172	0.169
	Familia	0.175	0.175	0.175	0.170
	Profesional	0.165	0.165	0.165	0.149
Iteración 3	Autoinforme	0.172	0.172	0.172	0.172
	Familia	0.172	0.172	0.172	0.172
	Profesional	0.167	0.167	0.167	0.167
Iteración 4	Autoinforme	0.172	0.170	0.172	0.168
	Familia	0.172	0.172	0.172	0.167
	Profesional	0.166	0.166	0.166	0.153
Iteración 5	Autoinforme	0.174	0.174	0.174	0.174
	Familia	0.172	0.172	0.172	0.172
	Profesional	0.170	0.170	0.170	0.170
Iteración 6	Autoinforme	0.172	0.172	0.172	0.166
	Familia	0.172	0.172	0.172	0.166
	Profesional	0.170	0.168	0.170	0.157

Tabla 3.7: Tabla comparativa de las métricas. Primera Versión

para cada uno de los modelos. El resto de iteraciones devuelven un ratio ligeramente alto para un modelo, pero luego flaquean en el resto.

- En el modelo del Autoinforme, la mejor iteración es la quinta, ya que las métricas *accuracy* y *recall* ofrecen un mejora de un 0.002 con respecto al resto de iteraciones. En la *precision*, la diferencia está entre un 0.002 y un 0.004. Finalmente, con la métrica *F1 Score*, la mejora es de entre un 0.002 y 0.008 con respecto al resto.

- En el modelo de las familias, la iteración 1 es la que ofrece el mejor rendimiento, ya que ofrece un ratio de 0.175 en todas las métricas, con una diferencia del 0.003 con respecto al resto de iteraciones. También logra una diferencia del 0.005 con respecto a la iteración 2 en el *F1 Score*, haciendo que sea el mejor escenario para los modelos de las familias. En la iteración 1, las características fue una división del dataset del 75 (Entrenamiento) - 25 (Testeo), y el uso del valor *micro* en el parámetro *average* en el cálculo de las métricas.

- En el modelo de los profesionales, la tabla muestra que la iteración que ofrece mejor rendimiento en todas las métricas es la quinta, ya que ofrece un ratio de 0.170 en todas las métricas, con una diferencia de entre 0.002 y 0.005 con respecto al resto de iteraciones. Además, si se compara con la iteración 6, que es la segunda mejor iteración, vemos que logra una diferencia del 0.002 en la métrica *precision* y un 0.13 con el *F1 Score*.

Estos resultados muestran que la diferencia entre todas las métricas es muy pequeña, pudiendo ser insignificante. Por lo tanto, no hay prácticamente diferencias entre las métricas de las 6 iteraciones.

3.3.1.8. Elección de la métrica y los parámetros

Partiendo de que la diferencia entre todas las métricas de todas las iteraciones es mínima, en general, la que ofrece mejores resultados y más consistentes, en todas las iteraciones es el *accuracy*, a diferencia de las otras, que varían dependiendo de la división del dataset o por la forma en la se calcula el promedio de los verdaderos/falsos positivos y verdaderos/falsos negativos. Por consiguiente, el *accuracy* será la métrica utilizada para determinar la fiabilidad del Sistema Experto.

Por otro lado, se puede observar que la iteración 5 es el que ofrece el *accuracy* más elevado en las tablas del Autoinforme, con una diferencia del 0.002; y el de los profesionales, con un diferencia de entre un 0.003 y un 0.005 en comparación con el resto de iteraciones. Considero que estas diferencias compensan la ligera pérdida de rendimiento en el modelo de las familias, ya que en este escenario, la iteración 1 es la que da mejores resultados.

Recordando las características de la iteración 5, se utiliza el valor *micro* en el parámetro *average* y se utiliza una división del 80 % (Entrenamiento) - 20 %(Testeo).

3.3.1.9. Conclusión final de la primera versión del Sistema Experto

Teniendo en cuenta los resultados obtenidos y el análisis posterior, se puede concluir que la fiabilidad en las predicciones del modelo del autoinforme es del 0.174 (17.4 %), 0.172 (17.2 %) en el modelo de las familias, y un 0.172 (17.2 %) en el modelo de los profesionales.

Para mostrar la fiabilidad del Sistema Experto al completo, se debía tener en cuenta a los 3 modelos creados. Para ello, opté por calcular la media aritmética de las tres mediciones de fiabilidad F:

$$F_{SistemaExperto} = \frac{F_{autoinforme}+F_{familias}+F_{profesionales}}{NumeroModelos} = \frac{0{,}174+0{,}172+0{,}170}{3} = 0{,}172$$

Podemos afirmar que hemos logrado crear un Sistema Experto con una fiabilidad general del 0.173 (17.3 %).

Este porcentaje de acierto general lo consideré demasiado bajo como para realizar predicciones reales en ella. Esto puede deberse principalmente a dos motivos:

1. **La naturaleza de los datasets**. Cómo bien se explicó en la sección 3.2.1, los datasets utilizados para el entrenamiento de los modelos se generaron de manera aleatoria y artificial, debido a que los originales no fueron proporcionados por los profesionales involucrados. Esto afectó negativamente a los procesos de entrenamiento, predicción y evaluación.

2. **La estructura de los modelos bayesianos**. Los modelos se pueden refinar más, aumentando o reduciendo el número de variables, especificando mejor las existentes.

3.3.2. Segunda versión: nuevos datasets con LLM

Viendo los resultados presentados en la primera versión, traté de buscar otra forma de generar datasets más coherentes y realistas. Traté de buscar datasets públicos en Internet, pero ninguno se ajustaba medianamente a la problemática de este proyecto. Es por ello, que me decanté por generar mis propios datasets, utilizando el modelo GPT-4o de OpenAI, a través de ChatGPT. Cabe resaltar que también se intentó utilizar Gemini, la IA de Google, pero presentaba varias limitaciones. Las respuestas que generaban eran muy limitadas, los

datos generados carecían de variedad y coherencia, y no permitía generar suficientes filas de datos como para formar un dataset.

El prompt inicial fue el siguiente:

Eres un experto psicólogo español. Genera un dataset que contenga los siguientes campos:

- *Variables de identificación*
  - *Curso: (NINGUNO, PRIMARIA, ESO, BACHILLERATO, UNIVERSIDAD, FORMACION PROFESIONAL).*
- *Variables de sociodemográficas*
  - *Sexo asignado: (HOMBRE, MUJER, OTRO).*
  - *Transgénero: (SÍ, NO, NO ESTOY SEGURO DE SER TRANS, NO ESTOY SEGURO DE LO QUE SE PREGUNTA).*
  - *Edad: (<12, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, >21).*
  - *Situación laboral padre: (NO TRABAJA, TRABAJA, PENSIONADO).*
  - *Situación laboral madre: (NO TRABAJA, TRABAJA, PENSIONADO).*
  - *Nivel profesional padre: (NINGUNO, PRIMARIA, ESO, BACHILLERATO, UNIVERSIDAD, LICENCIA).*
  - *Nivel profesional madre: (NINGUNO, PRIMARIA, ESO, BACHILLERATO, UNIVERSIDAD, LICENCIA).*
  - *Nivel promedio del rendimiento académico: (INSUFICIENTE, SUFICIENTE, NOTABLE, SOBRESALIENTE, EXTRAORDINARIO).*
  - *Nivel de autopercepción masculina: (0, 1, 2, 3, 4, 5, 6).*
  - *Nivel de autopercepción femenina: (0, 1, 2, 3, 4, 5, 6).*
  - *Nivel de percepción masculina externa: (0, 1, 2, 3, 4, 5, 6).*
  - *Nivel de percepción femenina externa: (0, 1, 2, 3, 4, 5, 6).*
- *Variables de salud*
  - *Altura: (<=149, 150-159, 160-169, 170-179, 180-189, >=190).*
  - *Peso: (<=49, 50-59, 60-69, 70-79, 80-89, >=90).*
  - *Tratamiento psiquiátrico previo: (SÍ, NO).*
  - *Presenta enfermedad crónica: (SÍ, NO).*
- *Variables de bullying*
  - *Víctima de bullying: (SÍ, NO).*
  - *Perpetrador de bullying: (SÍ, NO).*
  - *Víctima de Cyberbullying: (SÍ, NO).*
  - *Perpetrador de Cyberbullying: (SÍ, NO).*
- *Variables de adicción o abuso*
  - *Adicción o abuso alcohol: (SÍ, NO).*

- *Adicción o abuso sustancias: (SÍ, NO).*
- *Adicción o abuso Internet: (SÍ, NO).*

- *Psicopatología*
  - *Problemas interiorizados: (SÍ, NO).*
  - *Problemas exteriorizados: (SÍ, NO).*
  - *Problemas de contexto: (SÍ, NO).*
  - *Problemas de recursos psicológicos: (SÍ, NO).*
- *Variables del constructo del comportamiento suicida*
  - *Fuente de discriminación: (NINGUNA, EDAD, RAZA, DISCAPACIDAD, GÉNERO, RELIGIÓN, ORIENTACIÓN SEXUAL).*
- *Variables de regulación y resistencia*
  - *Nivel de resistencia o resiliencia del joven: (1, 2, 3, 4, 5).*
  - *Nivel de regulación positiva del joven: (1, 2, 3, 4, 5).*
  - *Nivel de regulación negativa del joven: (1, 2, 3, 4, 5).*
- *Variables volitivo-motivacionales para el suicidio*
  - *Sensación de atrapamiento interno: (SÍ, NO).*
  - *Sensación de atrapamiento externo: (SÍ, NO).*
  - *Nivel percibido de derrota o fracaso: (BAJO, MEDIO, ALTO).*
  - *Sentimiento de pertenencia frustada: (SÍ, NO).*
  - *Percepción de ser una carga: (SÍ, NO).*
  - *Autoeficiencia para el suicidio: (SÍ, NO).*
- *Perfil de riesgo psicosocial*
  - *Madre adolescente: (SÍ, NO).*
  - *Padre adolescente: (SÍ, NO).*
  - *Padres divorciados: (SÍ, NO).*
  - *Familia monoparental: (SÍ, NO).*
  - *Tratamiento psicológico de padre o madre: (SÍ, NO).*
  - *Adicción de padre o madre: (SÍ, NO).*
  - *Relaciones conflictivas del hijo con padre o madre: (SÍ, NO).*
  - *Familia reconstruida: (SÍ, NO).*
- *Tecnología y RRSS*
  - *Búsqueda información autolesión: (SÍ, NO).*
  - *Petición de ayuda en Internet: (SÍ, NO).*
  - *Compartir en RRSS pensamientos sobre autolesión: (SÍ, NO).*
  - *Realización de autolesión después de ver un contenido en Internet: (SÍ, NO).*
  - *Conocidos que comparten autolesión en Internet: (SÍ, NO).*

  - *Contacto con información sobre autolesión: (SÍ, NO).*
  - *Denuncia a otra persona cometiendo autolesión en Internet: (SÍ, NO).*

- *Desenlace: (NINGUNO, AUTOLESIÓN, COMUNICACIÓN DE SUICIDIO, DESEO DE SUICIDIO, IDEACIÓN DE SUICIDIO, PLANIFICACIÓN DE SUICIDIO, INTENCIÓN DE SUICIDIO).*

Todas las variables tienen relación directa con la variable Desenlace. Es decir, las variables sociodemográficas tienen relación directa con Desenlace, uno a uno.

En esta versión, decidí cambiar el enfoque del entrenamiento. A diferencia de la anterior versión, donde en cada iteración se entrenaban simúltaneamente los tres modelos con 5 datasets distintos para cada uno, a partir de esta nueva versión, se entrenaría cada modelo de manera individual en cada iteración hasta conseguir un resultado lo suficientemente bueno como para pasar al siguiente modelo. Mediante esta estrategia, se puede enfocar con mayor facilidad el entrenamiento del modelo en concreto, y así poder identificar más errores. Por otro lado, también se hizo ciertas modificaciones en el diseño de los jóvenes para especificar ciertos valores de las variables. El nuevo contenido del dataset se encuentra en la Figura B.1.

Además, en este punto, había descubierto nuevos tipos de clasificadores más allá del gausiano, que es el modelo multinomial. El clasificador multinomial es otro tipo de modelo Naive-Bayes, también presente en *scikit-learn*, que es utilizado en contextos donde los datos sean discretos, en lugar de continuos, en los que encajaría mejor el clasificador GaussianNB. En este nuevo clasificador, los datos deben seguir una distribución multinomial, es decir, que los datos representan la frecuencia con la que se repiten esa característica.

El script del entrenamiento corresponde con el cuaderno de Jupyter:

scripts/second_version_model_training_scikit_learn.ipynb

3.3.2.1. Iteración 7

Comenzamos esta segunda versión del entrenamiento con el modelo de los jóvenes, es decir, el del Autoinforme. En esta iteración se va a utilizar un primer dataset de prueba de 40 filas. Además, se ha añadido un nuevo tipo de clasificador para las pruebas, que es el clasificador multinomial. Este nuevo tipo de clasificación tiene el nombre de *MultinomialNB* en la librería *scikit learn*.

Se utilizarán los siguientes parámetros para iniciar esta nueva versión del entrenamiento.

- **Modelo**: Autoinforme.
- **Clasificador**: GaussianNB, MultinomialNB.
- **Dataset**: *scripts/datasets/autoinforme/v2/dataset_v2_1.csv* - 40 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: micro.

En este caso se devolvió los siguientes resultados con la siguiente matriz de confusión (Figura 3.17):

	GaussianNB	MultinomialNB
Accuracy	0.583	0.417
Precision	0.583	0.417
Recall	0.583	0.417
F1	0.583	0.417
Error medio	0.122	0.0

Tabla 3.8: Resultados de las métricas de la iteración 7

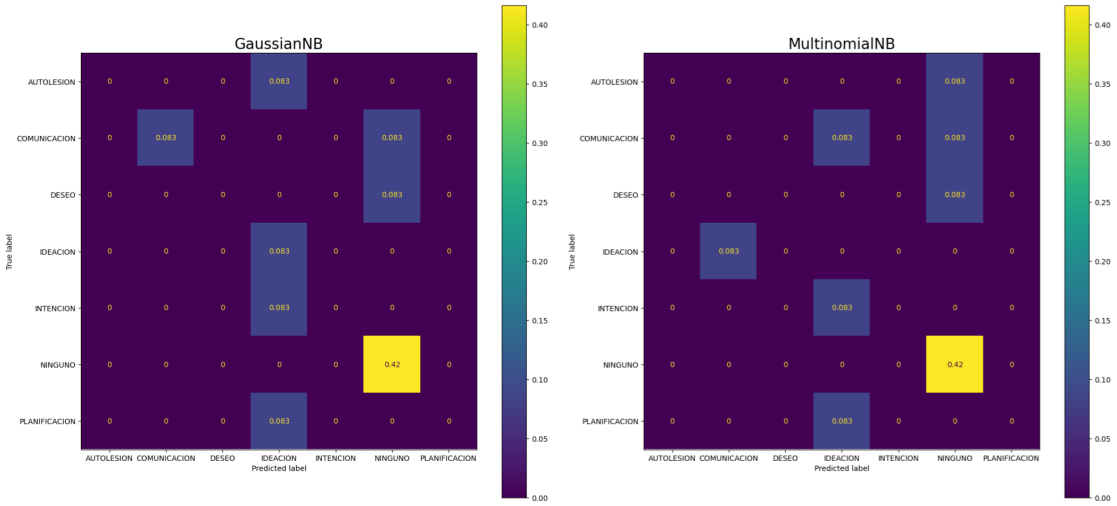


Figura 3.17: Matrices de confusión. Autoinforme. Iteración 7

Se puede observar que se ha aumentado el resultado del entrenamiento con este primer dataset. Sin embargo, el error medio en el modelo gausiano es relativamente alto, y en el multinomial es directamente 0. Esto puede deberse a los pocos datos que hay en el dataset, por lo que las predicciones son más dispares y erróneas. De hecho, gran parte del aumento de la métrica se debe a los aciertos en el desenlace de NINGUNO en ambas matrices.

3.3.2.2. Iteración 8

Ante los problemas de la iteración anterior, se optó por incrementar el tamaño del dataset previo, pidiendo al LLM que genere más filas para el dataset, se le pidió que generé datos más diversos, para que la gran mayoría de filas no sean con el desenlace NINGUNO. El resultado final fue un nuevo dataset con 68 filas más aparte de las 40 ya existentes, con variedad de desenlaces dependiendo de las variables del dataset.

En esta iteración, no se ha modificado ninguno de los parámetros establecidos. Esto es debido a que se quiso dar más enfoque a la generación del dataset, ya que es la principal forma de mejorar el rendimiento de un modelo.

- **Modelo**: Autoinforme.
- **Clasificador**: GaussianNB, MultinomialNB.
- **Dataset**: *scripts/datasets/autoinforme/v2/dataset_v2_2.csv* - 108 filas.
- **Balance del dataset**: 70 % - 30 %.

- ***average***: micro.

Se volvió a entrenar el modelo de los jóvenes y se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB
Accuracy	0.727	0.636
Precision	0.727	0.636
Recall	0.727	0.636
F1	0.727	0.636
Error medio	0.149	0.0774

Tabla 3.9: Resultados de las métricas de la iteración 8

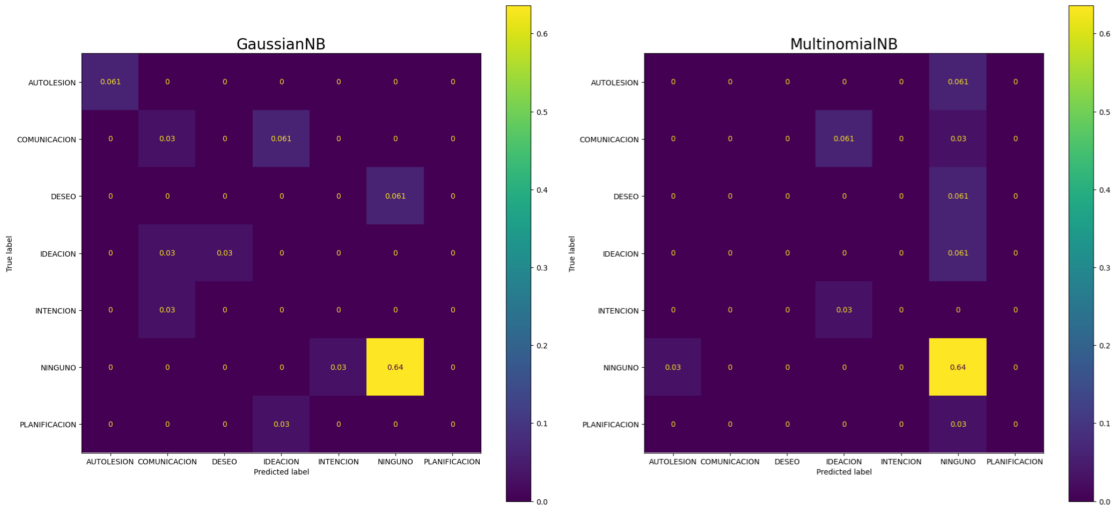


Figura 3.18: Matrices de confusión. Autoinforme. Iteración 8

Se percibe una mejora en las métricas y en las matrices de confusión, especialmente en el gausiano, porque los datos del dataset se ajustan más a una distribución gausiana en lugar de una multinomial. Se puede observar que comienza a acertar ligeramente las predicciones de AUTOLESION y algunas de COMUNICACION, además de NINGUNO. No obstante, siguen sin ser suficientes para el resto de desenlaces, por lo que es necesario seguir ampliando el dataset.

3.3.2.3. Iteración 9

Se optó por seguir ampliando el dataset, generando 4 filas más de ejemplo de COMUNICACION, DESEO, etc., resultando en un dataset con un total de 112 filas de entrenamiento. Al igual que en el resto de iteraciones, de momento no se han ajustado los parámetros de entrenamiento, con el fin de enfocarse en el desarrollo del dataset.

- **Modelo**: Autoinforme.
- **Clasificador**: GaussianNB, MultinomialNB.
- **Dataset**: *scripts/datasets/autoinforme/v2/dataset_v2_3.csv* - 112 filas.

- **Balance del dataset**: 70 % - 30 %.
- ***average***: micro.

	GaussianNB	MultinomialNB
Accuracy	0.588	0.647
Precision	0.588	0.647
Recall	0.588	0.647
F1	0.588	0.647
Error medio	0.185	0.124

Tabla 3.10: Resultados de las métricas de la iteración 9

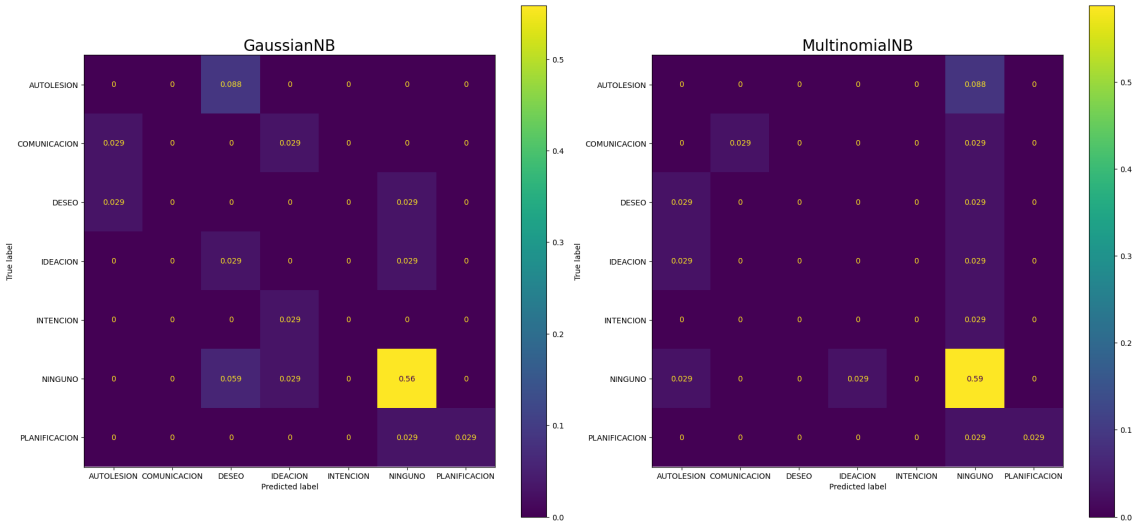


Figura 3.19: Matrices de confusión. Autoinforme. Iteración 9

Como resultado, se ha obtenido un peor rendimiento con respecto a la octava iteración en el modelo gausiano, aunque hay una ligera mejora en el categórico, en donde acierta en algunas predicciones con el desenlace COMUNICACION y PLANIFICACION. Además, observando la matriz de confusión, las predicciones han empeorado en el modelo gausiano. Ahora falla en el desenlace de AUTOLESION y COMUNICACION, valores que anteriormente acertaba. Esto puede ser debido a que las filas añadidas no siguen los patrones previamente entrenados, lo que deriva en una mayor cantidad de fallos en las predicciones. Por otro lado, en el modelo multinomial, ha comenzado a acertar en el desenlace de COMUNICACION y en el de PLANIFICACION, pero sigue errando considerablemente en el resto.

3.3.2.4. Conclusiones de la segunda versión de entrenamiento

Después de tres iteraciones de entrenamiento, se ha visto una mejora en las métricas con respecto a la primera versión de entrenamiento, pero analizando las matrices de confusión, se ha podido observar que el aumento de las puntuaciones de las métricas se deben a una mayor tasa de acierto en las predicciones de NINGUNO. Sin embargo, no hay mejora con el resto de predicciones. Esto se produce porque el modelo no percibe diferencias claras entre

las distintas categorías de desenlace. Por ejemplo, no hay un patrón que diferencie una fila con desenlace de PLANIFICACION o IDEACION. Esto provoca un peor rendimiento en el entrenamiento y posteriormente en el testeo del modelo. Además, los márgenes de error calculados en estas tres iteraciones se consideraron que eran demasiado altos como para continuar con el enfoque de la segunda versión. Es por eso que se llevó a cabo una modificación de los posibles desenlaces, para que el modelo pueda diferenciar mejor los distintos tipos que hay.

3.3.3. Tercera versión: cambio en el diseño del modelo

Los resultados de la segunda versión habían mejorado con respecto a la primera, pero seguían siendo relativamente bajos. Además, sus matrices de confusión demostraban que las predicciones realizadas por el modelo del Autoinforme no eran acertadas en la mayoría de las ocasiones.

Es por ello que se decidió rediseñar el modelo de los jóvenes, modificando algunas variables y sus valores. Esto también implicaba realizar cambios en el prompt utilizado en ChatGPT para generar los datasets de entrenamiento, para que se ajusten al nuevo tipo de modelo. Las nuevas estructuras de los datasets están disponibles en las Figuras B.2, B.3 y B.4.

En este caso, el script del entrenamiento corresponde con el cuaderno de Jupyter:

scripts/third_version_model_training_scikit_learn.ipynb

3.3.3.1. Iteración 10

Se le pidió al LLM generar un nuevo dataset con un nuevo formato diferente a la de la anterior versión.

El prompt inicial para generar el dataset fue el siguiente:

Eres un experto psicólogo español. Genera un dataset que contenga los siguientes campos:

- *Edad: (12-16,17-18,19-21).*
- *Curso: (Ninguno, Primaria, ESO, Bachillerato, Universidad, Otro).*
- *Peso (en kg): (x <= 49, 50-69, 70-89, x >= 90).*
- *Altura (en cm): (x <= 149, 150-169, 170-189, x >= 190).*
- *Sexo asignado: (Hombre, Mujer, Otro).*
- *Transgénero (Si, No, No se).*
- *Nivel promedio del rendimiento academico ->Nivel promedio acad.*
- *Situación laboral madre: (Trabaja, No trabaja, Pensionado).*
- *Situación laboral padre: (Trabaja, No trabaja, Pensionado).*
- *Nivel profesional madre: (Ninguno, Bachillerato, Formacion Profesional, Universidad, Otro).*

- *Nivel profesional padre: (Ninguno, Bachillerato, Formacion Profesional, Universidad, Otro).*
- *Nivel de autopercepción masculina: (0,1,2,3,4,5,6).*
- *Nivel de autopercepción femenina: (0,1,2,3,4,5,6).*
- *Nivel de percepción masculina externa: (0,1,2,3,4,5,6).*
- *Nivel de percepción femenina externa: (0,1,2,3,4,5,6).*
- *Tratamiento psicológico previo: (Si, No).*
- *Presenta enfermedad crónica: (Si, No).*
- *Víctima de bullying: (Si, No).*
- *Perpetrador de bullying: (Si, No).*
- *Víctima de cyberbullying: (Si, No).*
- *Perpetrador de cyberbullying: (Si, No).*
- *Adicción o abuso alcohol: (Si, No).*
- *Adicción o abuso sustancias: (Si, No).*
- *Adicción o abuso de Internet: (Si, No).*
- *Problemas interiorizados: (Si, No).*
- *Problemas exteriorizados: (Si, No).*
- *Problemas de contexto: (Si, No).*
- *Problemas de recursos psicológicos: (Si, No).*
- *Fuente de discriminación: (Ninguno, Genero, Raza, Orientacion sexual, Otro).*
- *Nivel de resistencia o resiliencia: (1,2,3,4,5).*
- *Nivel de regulacion positiva: (1,2,3,4,5).*
- *Nivel de regulación negativa: (1,2,3,4,5).*
- *Sensación de atrapamiento interno: (Si, No).*
- *Sensación de atrapamiento externo: (Si, No).*
- *Nivel percibido de derrota o fracaso: (Bajo, Medio, Alto).*
- *Sentido de pertenencia frustada: (Si, No).*
- *Percepcion de ser una carga: (Si, No).*
- *Autoeficiencia para el suicidio: (Si, No).*
- *Madre adolescente: (Si, No).*
- *Padre adolescente: (Si, No).*

- *Padres divorciados: (Si, No).*
- *Familia monoparental: (Si, No).*
- *Tratamiento psicologico padre o madre: (Si, No).*
- *Adiccion padre/madre: (Si, No).*
- *Relaciones conflictivas hijo-padre/madre: (Si, No).*
- *Familia reconstruida: (Si, No).*
- *Busqueda de informacion sobre autolesion: (Si, No).*
- *Compartir en RRSS pensamiento de autolesion: (Si, No).*
- *Peticion de ayuda en Internet: (Si, No).*
- *Realizacion de autolesion despues de ver contenido: (Si, No).*
- *Tener conocidos que comparten autolesion en Internet: (Si, No).*
- *Contacto con informacion sobre autolesion: (Si, No).*
- *Denuncia de autolesion en Internet: (Si, No).*
- *Desenlace: (Ninguno, Autolesion, Ideacion, Planificacion, Intencion).*

Ten en cuenta que todos son factores de riesgo que conducen a un Desenlace, que están ordenados del menos grave al más grave.

Este nuevo dataset estaría constituido por 140 filas. De nuevo, se ha intentado que los datos sean los más diversos y realistas posibles.

- **Clasificador**: GaussianNB, MultinomialNB.
- **Dataset**: *scripts/datasets/autoinforme/v3/dataset_v3_1.csv* - 140 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: micro.

En base a este primer dataset y a los parámetros establecidos, se obtuvo los siguientes resultados:

	GaussianNB	**MultinomialNB**
Accuracy	0.595	0.857
Precision	0.595	0.857
Recall	0.595	0.857
F1	0.595	0.857
Error medio	0.0698	0.0544

Tabla 3.11: Resultados de las métricas de la iteración 10

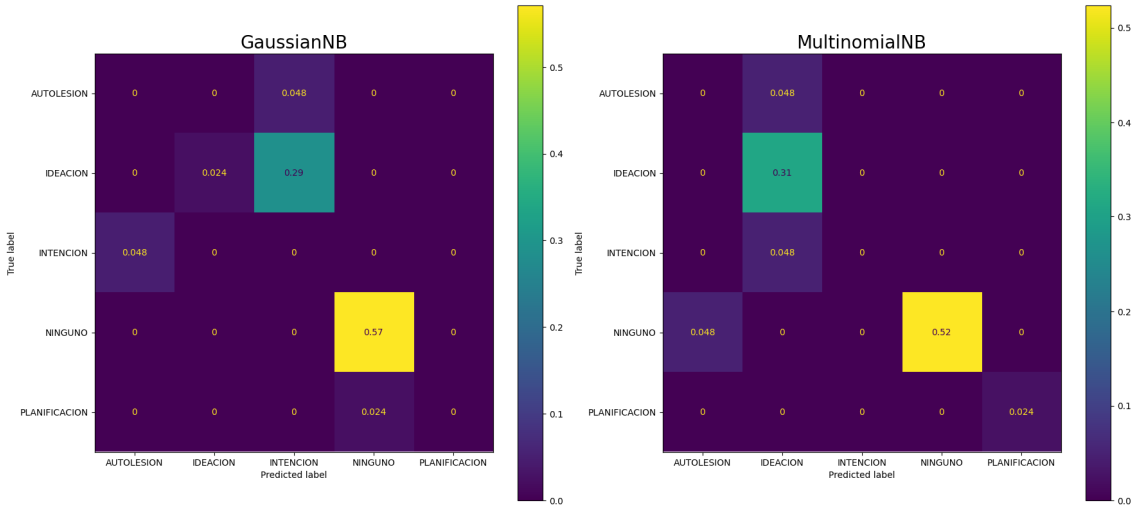


Figura 3.20: Matrices de confusión. Autoinforme. Iteración 10

Con este dataset ha habido una reducción del margen de error medio en comparación con la iteración 9, convirtiéndose en algo residual que no influye de forma drástica en la variación las métricas, como ocurría en la versión anterior.

Con la reducción de número de desenlaces ha permitido un aumento en el rendimiento, como se puede ver en la matriz de confusión del modelo gausiano y, especialmente, del modelo multinomial, en donde ha acertado correctamente los desenlaces de NINGUNO, IDEACION y PLANIFICACION.

3.3.3.2. Iteración 11

Se decidió ampliar el dataset de la iteración 10, añadiendo más ejemplos de AUTOLESION, ya que en la anterior no había suficientes como para conseguir un mejor rendimiento para esa clase, como se pudo observar en la matriz de confusión. El número total resultante del dataset son de 258 filas.

Un punto a destacar es que a partir de esta iteración 11, se ha añadido un nuevo clasificador, que es el categórico (CategoricalNB). Decidí incluir este nuevo tipo porque consideré que se ajusta más al tipo de datos con los que se estaban trabajando. Esto es debido a que, analizando los datasets, se ha comprobado que los datos siguen una distribución categórica, en donde cada dato es una categoría, es decir, un valor de una variable, en lugar de la distribución normal o multinomial.

De todas formas, opté por mantener el entrenamiento de todos los clasificadores, para no omitir posibles resultados que puedan ser mejores que el categórico. Este clasificador se intentó incluir en la iteración anterior. Sin embargo, no se pudo ya que en el dataset generado no estaban presentes todos los valores de las variables del diagrama, lo que generaba un error durante la fase de testeo.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/autoinforme/v3/dataset_v3_2.csv* - 258 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: micro.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.910	0.885	0.910
Precision	0.910	0.885	0.910
Recall	0.910	0.885	0.910
F1	0.910	0.885	0.910
Error medio	0.0555	0.0693	-

Tabla 3.12: Resultados de las métricas de la iteración 11

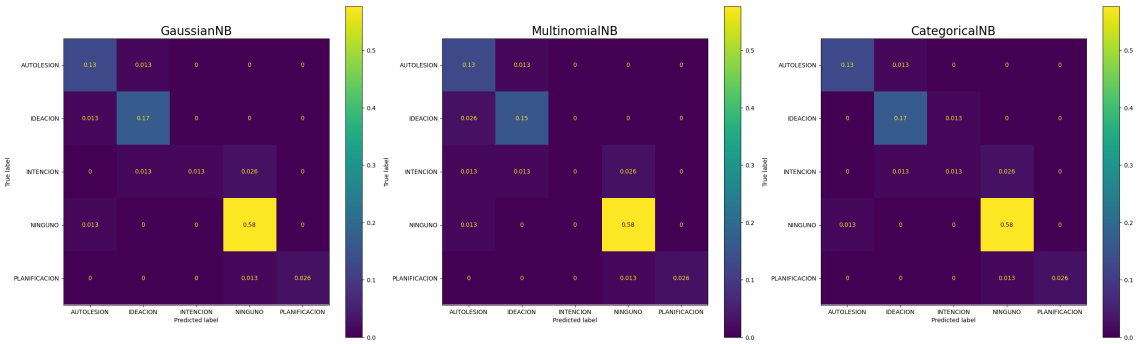


Figura 3.21: Matrices de confusión. Autoinforme. Iteración 11

En este caso se han conseguido resultados similares en todos los clasificadores. Hay una mejora positiva de las predicciones de AUTOLESION, IDEACION y PLANIFICACION. Sin embargo, todavía comete ciertos fallos con INTENCION. Por lo tanto, se realizarán una ampliación del dataset y repetir el entrenamiento para probar si se puede mejorar ese aspecto. El rendimiento del modelo multinomial sube ligeramente con respecto a la décima iteración, y el modelo categórico presenta unas métricas relativamente altas, similares al del modelo gausiano. Sin embargo, hubo un problema porque el error medio del clasificador categórico no se ha podido calcular. Esto se debe a que durante la validación cruzada, en alguna de las posibles combinaciones se habrá utilizado un conjunto que no contenga todos los posibles desenlaces (AUTOLESION, IDEACION, PLANIFICACION, INTENCION o NINGUNO), por lo que provoca un error en la validación. Para solucionarlo, sería necesario generar un nuevo conjunto de datos o aumentar el existente para añadir mayor variedad en los datos y una mejor distribución.

3.3.3.3. Iteración 12

En este caso, se ha vuelto a ampliar el dataset generando más filas y llegando hasta las 295 filas en total. Se han añadido más filas con desenlace de PLANIFICACION e INTENCION, con el objetivo de mejorar el rendimiento de las predicciones en esas clases. Los parámetros se mantienen intactos.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/autoinforme/v3/dataset_v3_3.csv* - 295 filas.
- **Balance del dataset**: 70 % - 30 %.

- ***average***: micro.

Se obtuvieron los siguientes resultados:

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.854	0.820	0.831
Precision	0.854	0.820	0.831
Recall	0.854	0.820	0.831
F1	0.854	0.820	0.831
Error medio	0.0573	0.084	0.0723

Tabla 3.13: Resultados de las métricas de la iteración 12

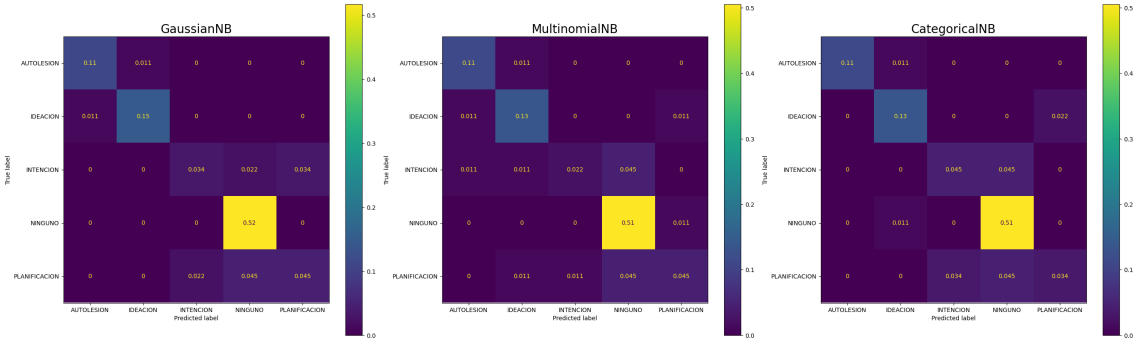


Figura 3.22: Matrices de confusión. Autoinforme. Iteración 12

En este caso, el modelo categórico sí que devuelve un cálculo del error medio, debido al aumento de datos en el dataset, lo que implica mayor variedad y distribución de los datos. Se puede percibir un ligero descenso del rendimiento en todos los clasificadores, pero teniendo en cuenta las matrices de confusión, las predicciones se siguen manteniendo similares en líneas generales. En este caso, el desenlace INTENCION ha empezado a ofrecer unos resultados ligeramente mejores, en donde la mitad de las filas de INTENCION del dataset se están prediciendo correctamente, en el caso de los clasificadores gausiano y categórico. En esta iteración, se observa que el que ha ofrecido mejor rendimiento ha sido el gausiano, pero las diferencias entre los tres clasificadores es mínima.

3.3.3.4. Iteración 13

Decidí cambiar la forma de calcular las métricas para observar si puede haber un aumento en la evaluación del modelo. Como este parámetro sólo modifica esta parte del proceso, la matriz de confusión no varía.

- **Clasificador**: GaussianNB, MultinomialNB.
- **Dataset**: *scripts/datasets/autoinforme/v3/dataset_v3_3.csv* - 295 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Los resultados del entrenmiento se encuentran reflejados en la Tabla 3.14, en donde se mostrarán todas las métricas del entrenamiento para los tres tipos de clasificadores gaussiano, multinomial y categórico, junto con el error medio de cada uno.

El cambio de métrica nos permite obtener resultados más variados, en lugar de recibir el mismo resultado en todas las métricas. Es por ello, que a partir de ahora se calcularán las métricas teniendo en cuenta el soporte o la importancia de cada clase dentro del dataset, en lugar de calcularlos de forma global.

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.854	0.820	0.831
Precision	0.834	0.803	0.816
Recall	0.854	0.820	0.831
F1	0.838	0.796	0.815
Error medio	0.0573	0.084	0.0723

Tabla 3.14: Resultados de las métricas de la iteración 13

3.3.3.5. Iteración 14

En esta iteración, consideré que el rendimiento podría mejorar si se modifica el balance del dataset, ajustándolo a un nuevo del 80 % - 20 %, en lugar del 70 % - 30 %.

- **Clasificador**: GaussianNB.
- **Dataset**: *scripts/datasets/autoinforme/v2/dataset_v2_3.csv* - 295 filas.
- **Balance del dataset**: 80 % - 20 %.
- ***average***: weighted.

Se obtuvieron los siguientes resultados:

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.881	0.847	0.898
Precision	0.861	0.785	0.893
Recall	0.881	0.847	0.898
F1	0.866	0.813	0.893
Error medio	0.0573	0.084	0.0723

Tabla 3.15: Resultados de las métricas de la iteración 14

Vemos que el modelo ha mejorado ligeramente, y el error medio se mantiene el mismo, por lo que no habría ninguna caída de rendimiento ni de puntuación en las métricas. Sin embargo, hay un ligero empeoramiento en el rendimiento de PLANIFICACION en el clasificador de Gauss. En este caso, las filas de esta clase se están prediciendo por lo general, como NINGUNO. Por otro lado, en el modelo multinomial se mantiene relativamente igual, aunque ha empezado a acertar las predicciones de INTENCION. Finalmente, el modelo categórico es el que se ha visto más beneficiado con esta modificación, ya que su puntuación es la más alta de los tres clasificadores.

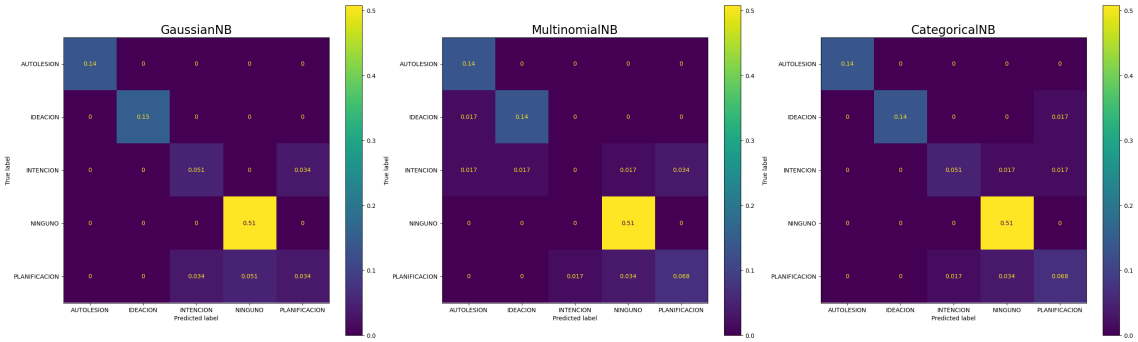


Figura 3.23: Matrices de confusión. Autoinforme. Iteración 14

3.3.3.6. Iteración 15

A lo largo de todas estas iteraciones, se ha ido ampliando el dataset utilizado, agregando cada vez más filas y redistribuyendolas a lo largo del dataset para obtener mejores predicciones. Es por ello, que en esta última iteración del modelo del Autoinforme, se quiso probar el entrenamiento con un dataset completamente diferente al de los anteriores. Este dataset consta de 123 filas totalmente diferentes a las del dataset anterior, para probar si el rendimiento se replica si recibe un dataset distinto. Es por ello que seleccioné el entrenamiento de la Iteración 13 como la indicada para utilizarlo en la aplicación.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/autoinforme/v3/dataset_v3_4.csv* - 123 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Se obtuvieron los siguientes resultados:

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.811	0.595	0.622
Precision	0.873	0.637	0.650
Recall	0.811	0.595	0.622
F1	0.794	0.603	0.626
Error medio	0.03908	0.0578	0.0623

Tabla 3.16: Resultados de las métricas de la iteración 15

El resultado final ofrece una matriz de confusión en donde se puede observar que realizan las predicciones correctamente, pero que ofrece peores resultados en comparación al entrenamiento de la iteración anterior. Es por ello, y además por la necesidad de entrenar a los otros 2 modelos restantes, por lo que decidí concluir el entrenamiento del modelo de los jóvenes.

3.3.3.7. Iteración 16

Esta es la primera iteración del modelo de las familias. Empecé con una primera versión de un dataset generado por ChatGPT.

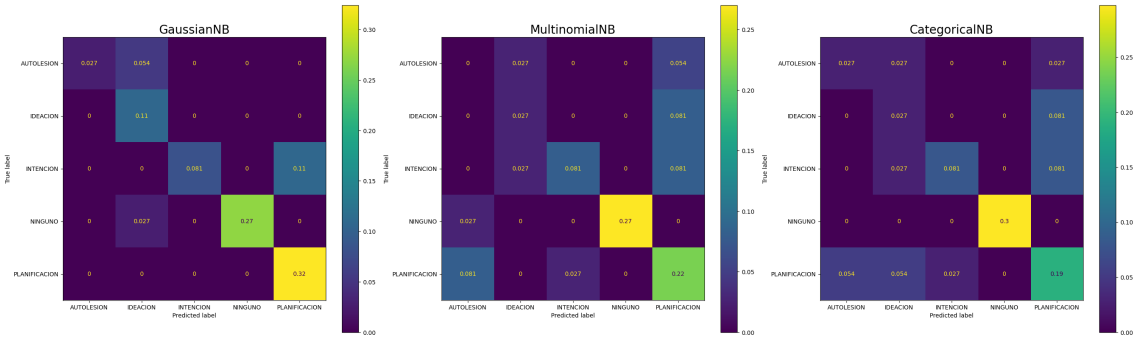


Figura 3.24: Matrices de confusión. Autoinforme. Iteración 15

El prompt inicial utilizado es el siguiente:

Eres un español experto en psicología en salud mental entre jóvenes adolescentes.

Se le han realizado cuestionarios a los padres sobre el comportamiento de sus hijos. Genera un dataset con respecto a los valores devueltos por los padres sobre sus hijos y el diagnóstico hecho por ti en el campo desenlace sobre el posible desenlace del hijo en base a los factores de riesgo. Los campos son los siguientes:

- *id: identificador (entero).*
- *curso: Nivel educativo actual del hijo (cadena: "Ninguno", "Primaria", "`ESO`", "Bachillerato", "`Universidad`", "`Otro`").*
- *situacion_laboral_madre : Situacion laboral de la madre (cadena: "Trabaja", "No trabaja", "Pensionado").*
- *situacion_laboral_padre : Situacion laboral del padre (cadena: "Trabaja", "No trabaja", "Pensionado").*
- *nivel_profesional_madre : Nivel educativo de la madre (cadena: "Ninguno", "Bachillerato", "Formacion Profesional", "`Universidad`", "`Otro`").*
- *nivel_profesional_padre : Nivel educativo del padre (cadena: "Ninguno", "Bachillerato", "Formacion Profesional", "`Universidad`", "`Otro`").*
- *problemas_interiorizados : Problemas interiorizados del hijo (cadena: "No", "Si").*
- *problemas_exteriorizados : Problemas exteriorizados del hijo (cadena: "No", "Si").*
- *problemas_contexto : Problemas de contexto del hijo (cadena: "No", "Si").*
- *problemas_recursos_psicologicos : Problemas de recursos psicológicos del hijo (cadena: "No", "Si").*
- *madre_adolescente : Madre adolescente (cadena: "No", "Si").*
- *padre_adolescente : Padre adolescente (cadena: "No", "Si").*
- *padres_divorciados : Padres divorciados (cadena: "No", "Si").*
- *familia_monoparental : Familia monoparental (cadena: "No", "Si").*

- *tratamiento_psicologico_padre_madre : Padre y/o madre en tratamiento psicologico (cadena: "No", "Si").*
- *adiccion_padre_madre : Padre y/o madre con adiccion (cadena: "No", "Si").*
- *relaciones_conflictivas_hijo_padre_madre : Relaciones conflictivas entre el hijo y el padre o la madre (cadena: "No", "Si").*
- *familia_reconstruida : Familia reconstruida (cadena: "No", "Si").*
- *aceptacion_rechazo_parental : aceptación o rechazo parental hacia al hijo (cadena: "Aceptacion", Rechazo").*
- *control_parental : existencia de control parental de los padres al hijo (cadena: "No", "Si").*
- *situacion_economica_precaria : la situación económica de la familia es precaria (cadena: "No", "Si").*
- *ingreso_familiar_mensual : ingreso familiar mensual (cadena: "x<=999", "1000-1499", "1500-1999", "x>=2000")*
- *desenlace : Desenlace de menor a mayor gravedad (cadena: "Ninguno", "`Autolesion`", "`Ideacion`", "Planificacion", "`Intencion`").*

Ten en cuenta que los datos deben ser coherentes, variados y realistas. Por ejemplo, las familias que tienen una situación familiar precaria, no suelen tener un ingreso mensual mayor de 1500€. Además, ten en cuenta que todos los campos son factores que afectan directamente a la columna desenlace, que es el resultado final.

Consta de 100 filas utilizando la estructura de la Figura B.3. Lo parámetros establecidos fueron los siguientes.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/familia/v3/dataset_v3_1.csv* - 100 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

En este caso, escogí este balance y este valor del *average* porque eran los parámetros seleccionados de la Iteración 13, que es el entrenamiento elegido para el modelo de los jóvenes. Esto lo hice con el objetivo de mantener los mismos parámetros en todos los modelos, en lugar de tener unos parámetros para un modelo y otros distintos para el resto.

Este primer entrenamiento ha devuelto unos resultados medianamente correctos.

Por un lado, el modelo gausiano acierta con las categorías de IDEACION, INTENCION y NINGUNO, pero falla en la mayoría de las predicciones con AUTOLESION y PLANIFICACION.

Por otro lado, el modelo multinomial acierta con IDEACION, NINGUNO y PLANIFICACION, pero falla en AUTOLESION e INTENCION.

Finalmente, el modelo categórico, que es el ha obtenido mejor rendimiento en este caso, acierta con todas las categorías, excepto con AUTOLESION.

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.767	0.800	0.833
Precision	0.892	0.812	0.909
Recall	0.767	0.800	0.833
F1	0.735	0.763	0.811
Error medio	0.0399	0.1319	0.1029

Tabla 3.17: Resultados de las métricas de la iteración 16

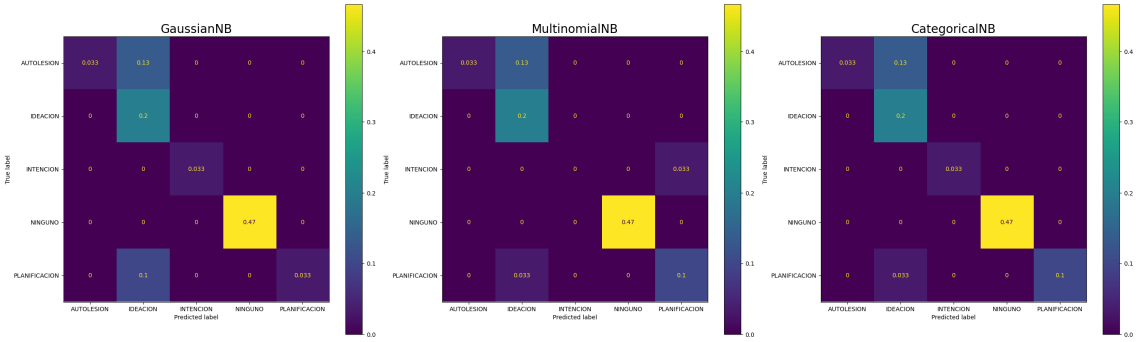


Figura 3.25: Matrices de confusión. Familia. Iteración 16

En líneas generales, la categoría que fallan todas las clasificaciones es la de AUTOLESION, por lo que hay que intentar corregir este fallo, añadiendo más filas de AUTOLESION, además de las categorías INTENCION y PLANIFICACION.

3.3.3.8. Iteración 17

Se ha ampliado el dataset de la anterior iteración a través del modelo GPT-4o, dando un total de 150 filas, añadiendo filas de AUTOLESION, INTENCION y PLANIFICACION. Los parámetros son los siguientes:

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/familia/v3/dataset_v3_2.csv* - 150 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Se obtuvieron las siguientes puntuaciones:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.822	0.711	0.756
Precision	0.886	0.688	0.768
Recall	0.822	0.711	0.756
F1	0.805	0.691	0.742
Error medio	0.1445	0.1218	0.1103

Tabla 3.18: Resultados de las métricas de la iteración 17

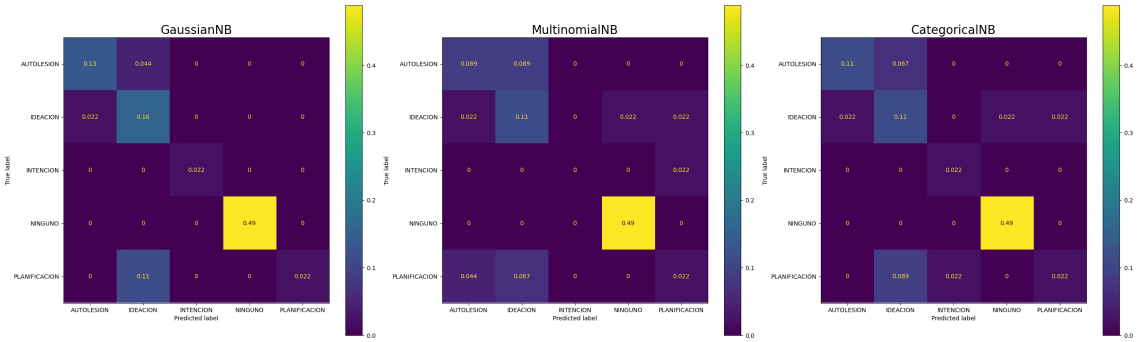


Figura 3.26: Matriz de confusión. Familia. Iteración 17

El rendimiento ha mejorado en el caso de los clasificadores gausiano y categórico, pero disminuye en el multinomial. Sin embargo, el punto más destacado de este entrenamiento es el aumento del error medio en todos ellos, siendo un ratio demasiado elevado como para detener el entrenamiento en este punto. Además, siguen habiendo fallos en las predicciones en distintas categorías en cada uno de los clasificadores.

3.3.3.9. Iteración 18

El aumento del error medio y los fallos en las predicciones en la Iteración 17 hicieron que se tuviese que generar un nuevo dataset. Este dataset consta de 107 filas.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/familia/v3/dataset_v3_3.csv* - 107 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Después de realizar el entrenamiento para los tres clasificadores, se obtuvieron los siguientes resultados:

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.848	0.727	0.788
Precision	0.864	0.668	0.792
Recall	0.848	0.727	0.788
F1	0.852	0.678	0.773
Error medio	0.0682	0.0997	0.0818

Tabla 3.19: Resultados de las métricas de la iteración 18

Se ha conseguido reducir el error medio en todos los clasificadores y la matriz de confusión muestra mejores predicciones para todas las clases en el modelo gausiano, aunque el rendimiento del multinomial y categórico han bajado. Por este motivo, como se ha conseguido que al menos uno de los modelos ha logrado una predicción correcta de los datos en general, se tomó la decisión de finalizar el entrenamiento de los modelos de la familia.

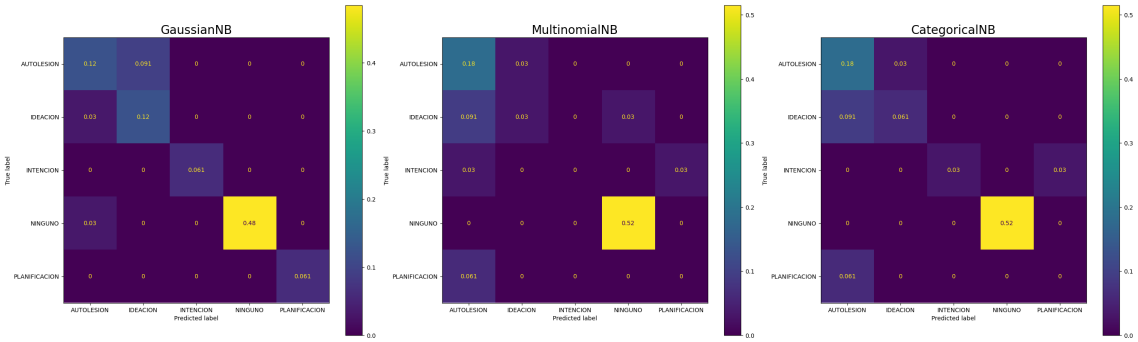


Figura 3.27: Matriz de confusión. Familia. Iteración 18

3.3.3.10. Iteración 19

A partir de esta iteración, el entrenamiento se enfocó al modelo de los profesionales. De nuevo, los datasets fueron generados a través de GPT-4o.

El prompt inicial fue el siguiente:

Eres un experto español en psicología española en jóvenes adolescentes y salud mental juvenil. Has realizado un diagnóstico de jóvenes adolescentes. Genera un dataset con 200 ejemplos de diagnósticos de profesionales a los jóvenes adolescentes de entre 12 y 21 años.

Las columnas son las siguientes:

- *ID.*
- *Curso: (Ninguno, Primaria, ESO, Bachillerato, Universidad, Otro).*
- *Situación laboral padre: (No trabaja, Trabaja, Pensionado).*
- *Situación laboral madre: (No Trabaja, Trabaja, Pensionado).*
- *Nivel profesional padre: (Ninguno, Bachillerato, Formación Profesional, Universidad, Otro).*
- *Nivel profesional madre: (Ninguno, Bachillerato, Formación Profesional, Universidad, Otro).*
- *Situación económica familiar precaria: (Sí,No).*
- *Ingreso familiar mensual: (x<=999,1000-1499, 1500-1999, x>=2000).*
- *Tratamiento psiquiátrico previo: (Sí, No).*
- *Problemas interiorizados: (Sí, No).*
- *Problemas exteriorizados: (Sí, No).*
- *Problemas de contexto: (Sí, No).*
- *Problemas de recursos psicológicos: (Sí, No).*
- *Madre adolescente: (Sí, No).*
- *Padre adolescente: (Sí, No).*

- *Padres divorciados: (Sí, No).*
- *Familia monoparental: (Sí, No).*
- *Tratamiento psicológico de padre o madre: (Sí, No).*
- *Adicción de padre o madre: (Sí, No).*
- *Relaciones conflictivas del hijo con padre o madre: (Sí, No).*
- *Familia reconstruida: (Sí, No).*
- *Supervisión parental insuficiente: (Sí, No).*
- *Maltrato al adolescente: (Sí, No).*
- *Maltrato a la pareja: (Sí, No).*
- *Duelo por un ser querido: (Sí, No).*
- *Desenlace: (Ninguno, Autolesión, Ideación, Planificación, Intención).*

Aclaraciones:

- *Asegúrate de que los datos sean coherentes y realistas.*
- *Todas las columnas son factores que afectan directamente al resultado "Desenlace".*

Al haber menos columnas que en el resto, es más fácil y rápido para el LLM generar las filas necesarias para realizar el entrenamiento. Se comenzó con un primer dataset de 200 filas y con los siguientes parámetros:

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/profesional/v3/dataset_v3_1.csv* - 200 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

De esta iteración se obtuvieron los siguientes resultados:

	GaussianNB	**MultinomialNB**	**CategoricalNB**
Accuracy	0.283	0.317	0.217
Precision	0.278	0.321	0.227
Recall	0.283	0.317	0.217
F1	0.278	0.314	0.218
Error medio	0.0556	0.0578	0.0748

Tabla 3.20: Resultados de las métricas de la iteración 19

En este primer entrenamiento del modelo de los profesionales, se puede observar una gran disparidad en la matriz de confusión, además de unas métricas considerablemente bajas. Esto puede deberse a la falta de patrones claros dentro del dataset de los profesionales, por lo que las predicciones fallan con demasiada frecuencia.

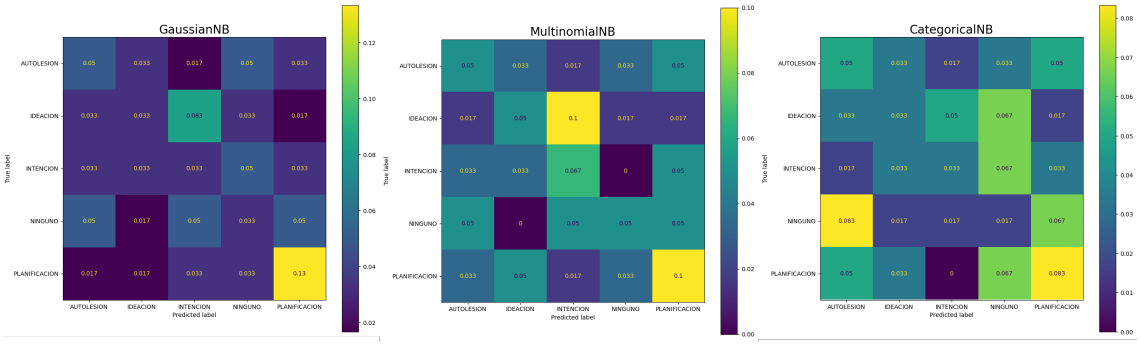


Figura 3.28: Matrices de confusión. Profesionales. Iteración 19

3.3.3.11. Iteración 20

En esta iteración, se iba a crear un dataset nuevo, que iba a estar constituido de 220 filas, este dataset era el que se encontraba en el directorio *scripts/datasets/profesional/v3/dataset_v3_2.csv*.

Sin embargo, durante la preparación de la actual iteración, me percaté de un fallo que había en el contenido de este dataset y también en el de la iteración 19. En ellas, existen cuatro columnas que no corresponden al modelo de los profesionales. Estos son los siguientes:

- Problemas interiorizados.
- Problemas exteriorizados.
- Problemas de recursos.
- Problemas de recursos psicológicos.

Estos campos no corresponden a este modelo porque son variables que se miden en base al cuestionario SENA de la página de SIVARIA. Sin embargo, se comprobó que no estaba presente en el cuestionario de los profesionales, por lo que no habría manera alguna de obtener esos valores si se conectasen a la aplicación. Es por ello que tuve que rehacer el dataset, eliminando estas cuatro columnas y volver a entrenar el modelo.

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset**: *scripts/datasets/profesional/v3/dataset_v3_3.csv* - 220 filas.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Después del entrenamiento, se obtuvieron los resultados representados en la Tabla 3.21 y las matrices de la Figura 3.29.

Se puede observar que el rendimiento ha disminuido. Esto se debe a la eliminación de las cuatro filas, que representaban factores importantes para diferenciar entre los distintos desenlaces. No obstante, consideré que se podría aumentar el rendimiento si se intentaba ajustar los hiperparámetros del modelo.

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.439	0.530	0.621
Precision	0.611	0.396	0.609
Recall	0.439	0.530	0.621
F1	0.447	0.412	0.605
Error medio	0.01398	0.0265	0.0334

Tabla 3.21: Resultados de las métricas de la iteración 20

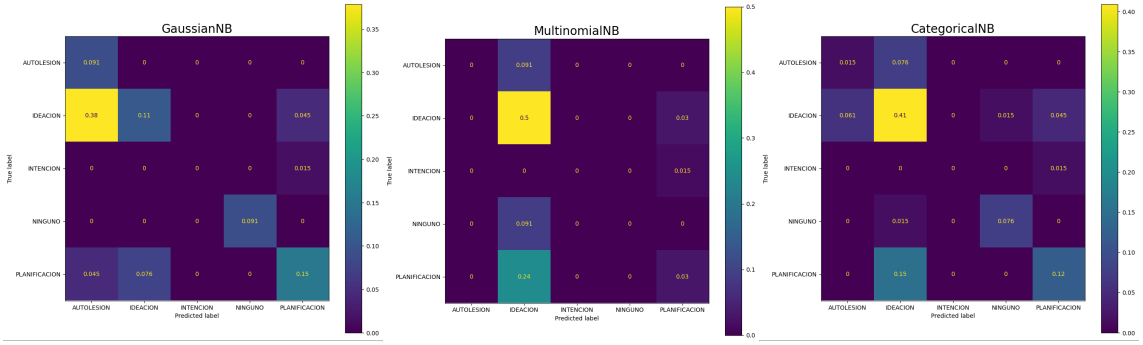


Figura 3.29: Matrices de confusión. Profesionales. Iteración 20

3.3.4. Cuarta versión: ajuste de hiperparámetros

A pesar de la considerable mejora en los resultados de la tercera versión, no se había realizado ningún ajuste de hiperparámetros y poder encontrar el mejor modelo con la mejor combinación de hiperparámetros posible. Para ello, se aplicó el algoritmo *GridSearchCV* de la librería *Scikit-learn*. El *GridSearch* es un algoritmo que recibe el tipo de modelo que queremos combinar y una lista de valores de cada uno de los hiperparámetros. Los hiperparámetros que se especifiquen determinarán el número de combinaciones que debe hacer el algoritmo posteriormente. A continuación, internamente realiza una validación cruzada mediante la división de los datos en *K-folds* o K subconjuntos.

En este caso, se ha establecido que sean cinco folds, ya que es el valor por defecto. Por lo tanto, el algoritmo dividirá los datos en cinco subconjuntos, de los cuáles k - 1 se usarán para entrenar al clasificador, mientras que el uno restante será para el proceso de evaluación. El algoritmo combina los cinco folds dentro de cada clasificador, por lo tanto, por cada clasificador habrá 5 combinaciones posibles. De esta manera, se puede obtener el clasificador que mejor rendimiento ofrezca.

El script del entrenamiento corresponde con el cuaderno de Jupyter:

scripts/forth_version_model_training_scikit_learn.ipynb

3.3.4.1. Iteración 21

En este caso, se decidió entrenar a los 3 modelos, usando los datasets que se seleccionaron en su momento que dieron mejores resultados en sus entrenamientos.

- Autoinforme: *scripts/datasets/autoinforme/v3/dataset_v3_3.csv* (Iteración 13)
- Familia: *scripts/datasets/familia/v3/dataset_v3_3.csv* (Iteración 18)

- Profesional: *scripts/datasets/profesional/v3/dataset_v3_3.csv* (Iteración 20)

Los parámetros seleccionados fueron los siguientes:

- **Clasificador**: GaussianNB, MultinomialNB, CategoricalNB.
- **Balance del dataset**: 70 % - 30 %.
- ***average***: weighted.

Y los hiperparámetros a los que se han ajustado fueron los siguientes:

- **GaussianNB**:
  - **var_smoothing** (suavizado de la variable): es un parámetro de estabilidad para suavizar la curva, y, por lo tanto, tener en cuenta más muestras de distribución. En este caso, *numpy.logspace(-9, 9, n=100)*. Devuelve un array de 100 números separados uniformente en una escala logarítmica de base 10. Por lo que los valores estarán comprendidos entre 10^{-9} (poco suavizado) y 10^{9} (mucho suavizado).

```
param_grid = {
  'var_smoothing': np.logspace(-9, 9, num=100) # devuelve un array de 100 elementos del 1.e-09 a 1.e+09
}

```

Listing 3.1: Posibles valores de los hiperparámetros del modelo gausiano

Teniendo en cuenta el tamaño del array, se puede saber que habrá 100 modelos que se van a probar, uno por cada valor del hiperparámetro.

- **MultinomialNB y CategoricalNB**:
  - **alpha**: hiperparámetro similar al *var_smoothing* del modelo gausiano. Es un parámetro de suavizado cuyo valor por defecto es 1.0, y que se encarga de añadir internamente a los recuentos de datos para eliminar posibles errores de divisiones por 0. Esto es debido a que el modelo calcula las frecuencias de las características en base al recuento de datos, y es probable que si algunas características no están presentes en el dataset, el denominador de la frecuencia arrojaría un error de división por 0.
  - **force_alpha**: sirve para establecer un valor alpha fijo. Si su valor es True, el valor de alpha no se modifica. Sin embargo, si es False y alpha tiene un valor inferior a $1 \cdot 10^{-10}$, entonces alpha tendrá este valor establecido.
  - **fit_prior**: si el valor es True, entonces las probabilidades *a priori* se calcularán en base a los datos recibidos del dataset. Pero si es False, estas probabilidades tendrán un valor igual fijo.

```
param_grid_mul_cat = {
  'alpha': np.logspace(-9,9,num=100), # devuelve un array de valores que van desde 1.e-09 y 1.e+09
  'force_alpha': [True, False],
  'fit_prior': [True, False]
}

```

Listing 3.2: Posibles valores de los hiperparámetros del modelo multinomial

En este caso, el número total de combinaciones sería $|alpha| \cdot |force\_alpha| \cdot |fit\_prior| = 100 \cdot 2 \cdot 2 = 400$

Se ha configurado el *GridSearchCV* para que entrene al modelo utilizando 5 folds, utilizando la métrica de *accuracy* para la evaluación del modelo con el fold restante. Mediante este método, a cada candidato se le aplican 5 posibles combinaciones de datos. Esto da como resultado un total de $n\_candidatos \cdot n\_folds = 100 \cdot 5 = 500$ ajustes en el caso del modelo GaussianNB, y $400 \cdot 5 = 2000$ posibles ajustes en el modelo MultinomialNB y CategoricalNB.

Finalmente, los rendimientos obtenidos fueron las siguientes:

	Autoinforme	**Familia**	**Profesionales**
Accuracy	0.854	0.848	0.712
Precision	0.834	0.864	0.741
Recall	0.854	0.848	0.712
F1	0.838	0.852	0.711
Error medio	0.0573	0.0682	0.0231
Hiperparámetros	var_smoothing= $1 \cdot 10^{-9}$	var_smoothing= $1 \cdot 10^{-9}$	var_smoothing= 0,02848

Tabla 3.22: Métricas del modelo de Gauss. Iteración 21

	Autoinforme	**Familia**	**Profesionales**
Accuracy	0.798	0.758	0.530
Precision	0.720	0.753	0.396
Recall	0.798	0.758	0.530
F1	0.751	0.750	0.412
Error medio	0.0783	0.0743	0.0265
Hiperparámetros	alpha=10 force_alpha=True fit_prior=True	alpha=$1 \cdot 10^{-9}$ force_alpha=True fit_prior=False	alpha=1.0 force_alpha=True fit_prior=True

Tabla 3.23: Métricas del modelo multinomial. Iteración 21

	Autoinforme	**Familia**	**Profesionales**
Accuracy	0.798	0.758	0.636
Precision	0.766	0.753	0.624
Recall	0.798	0.758	0.636
F1	0.753	0.749	0.623
Error medio	0.0.4497	0.07760	0.0265
Hiperparámetros	alpha=10 force_alpha=True fit_prior=False	alpha=0.8111 force_alpha=True fit_prior=False	alpha=0.1 force_alpha=True fit_prior=True

Tabla 3.24: Métricas del modelo categórico. Iteración 21

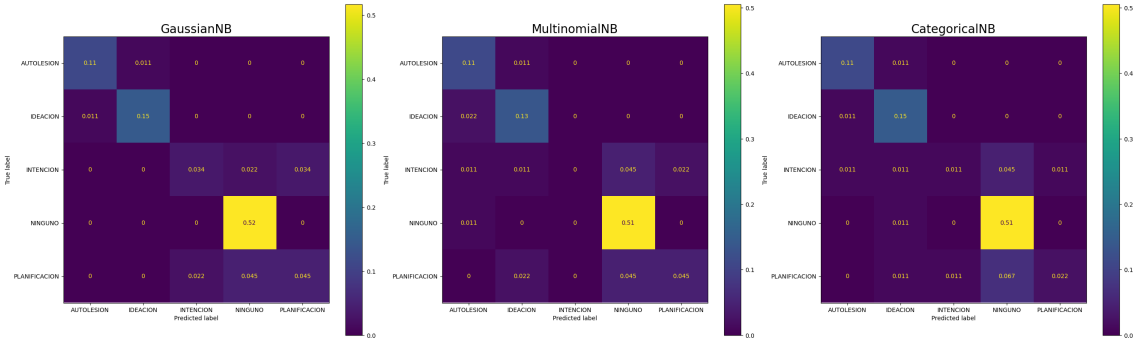


Figura 3.30: Matrices de confusión del Autoinforme. Iteración 21

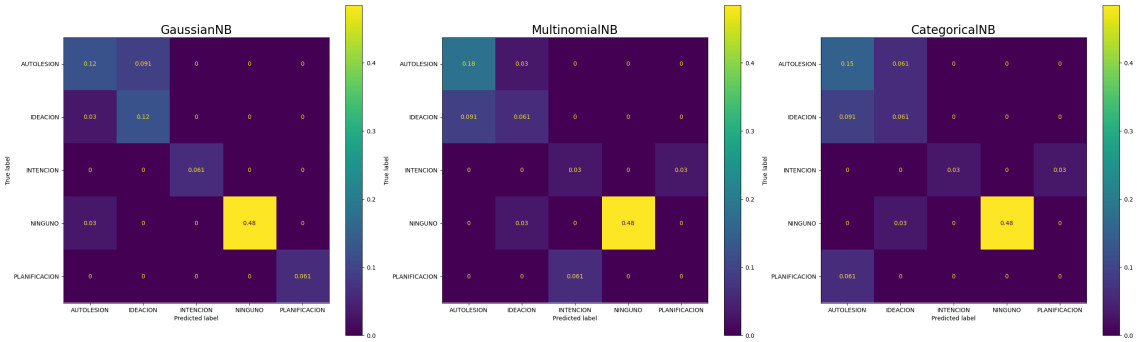


Figura 3.31: Matrices de confusión de las familias. Iteración 21

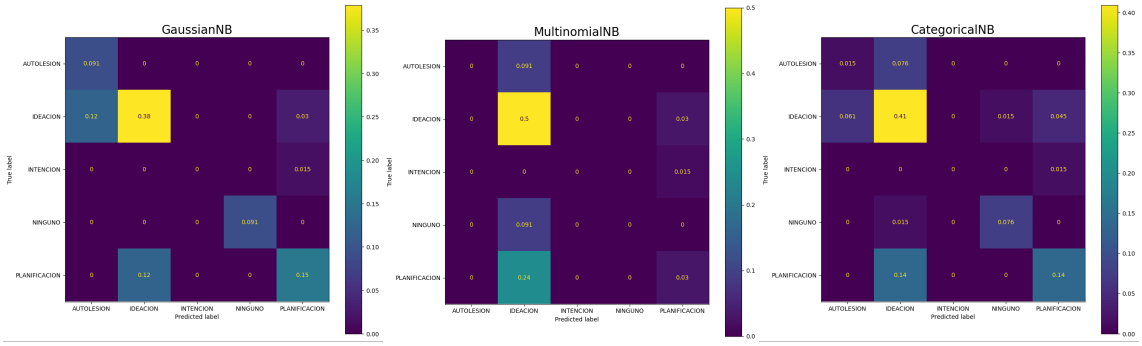


Figura 3.32: Matrices de confusión del modelo de los profesionales. Iteración 21

Haciendo tres tablas comparativas (Tabla 3.25, 3.26 y 3.27), se puede observar que en el modelo de las familias, ha habido en general una bajada de rendimiento en los clasificadores multinomial y categórico. que ha habido una ligera mejora en el rendimiento del modelo de las familias y, especialmente, de los profesionales, tanto en el modelo gausiano como el multinomial. Por lo tanto, el entrenamiento de la iteración 21 será la seleccionada para implementarlo en la aplicación y usarlo para los procesos de predicción.

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.854	0.854	0.848	0.848	0.439	0.712
Precision	0.834	0.834	0.864	0.864	0.611	0.741
Recall	0.854	0.854	0.848	0.848	0.439	0.712
F1	0.838	0.838	0.852	0.852	0.447	0.711

Tabla 3.25: Tabla comparativa de las métricas del modelo de Gauss

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.820	0.798	0.727	0.758	0.530	0.530
Precision	0.803	0.720	0.668	0.753	0.396	0.396
Recall	0.820	0.798	0.727	0.758	0.530	0.530
F1	0.796	0.751	0.678	0.750	0.412	0.412

Tabla 3.26: Tabla comparativa de las métricas del modelo multinomial

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.831	0.798	0.788	0.758	0.621	0.636
Precision	0.816	0.766	0.792	0.753	0.609	0.624
Recall	0.831	0.798	0.788	0.758	0.621	0.636
F1	0.815	0.753	0.773	0.749	0.605	0.623

Tabla 3.27: Tabla comparativa de las métricas del modelo categórico

3.3.5. Conclusiones del entrenamiento

Finalmente, después de realizar el entrenamiento, se ha realizado una tabla comparativa de las puntuaciones obtenidas en los mejores entrenamientos de cada uno de los 3 modelos de acuerdo a la siguiente Tabla 3.28.

En general, ha habido una mejora considerable en el rendimiento con respecto a la primera versión. Sin embargo, entre los tres clasificadores, el clasificador gausiano es el que ha obtenido mejores resultados que el multinomial y el categórico, en todos los modelos (Autoinforme, Familia y Profesionales). Es por esta razón por la que decidí utilizar el primer clasificador para realizar las predicciones de la aplicación. Posteriormente, tuve que decidir qué métrica usar para el cálculo de la fiabilidad del Sistema Experto. Por otra parte, entre las 4 métricas gausianas, se tuvo en cuenta finalmente la métrica de la precisión, debido que es la métrica más favorable para 2 de los 3 modelos: el de las familias y el de los

	Autoinforme			**Familia**			**Profesionales**		
Clasificador	C1	C2	C3	C1	C2	C3	C1	C2	C3
Accuracy	0.854	0.798	0.798	0.848	0.758	0.758	0.712	0.530	0.621
Precision	0.834	0.720	0.766	0.864	0.753	0.753	0.741	0.396	0.617
Recall	0.854	0.798	0.798	0.848	0.758	0.758	0.712	0.530	0.621
F1	0.838	0.751	0.753	0.852	0.750	0.749	0.711	0.412	0.610
Error medio	0.057	0.078	0.045	0.068	0.074	0.077	0.023	0.044	0.044
	C1=Gauss, C2=Multinomial, C3=Categórico								

Tabla 3.28: Tabla comparativa de las métricas. Versión final.

profesionales. Esta elección, aunque es perjudicial para el modelo del Autoinforme, porque es la métrica con el valor más bajo, en líneas generales, es la más alta para los otros 2, y por lo tanto, es más beneficioso para la fiabilidad del Sistema Experto.

Tomando en consideración los resultados obtenidos y el análisis posterior, se puede concluir que se ha obtenido una fiabilidad del 0.834 (83.4 %) en las predicciones en el modelo del autoinforme, 0.864 (86.4 %) en el modelo de las familias, y un 0.741 (74.1 %) en el modelo de los profesionales.

Realizamos una media aritmética de las tres mediciones de fiabilidad F:

$$F_{SistemaExperto} = \frac{F_{autoinforme}+F_{familias}+F_{profesionales}}{NumeroModelos} = \frac{0{,}834+0{,}864+0{,}741}{3} \approx 0{,}813$$

Por ende, podemos afirmar que hemos logrado crear un Sistema Experto con una fiabilidad general del 0.813 (81.3 %), lo que supone un aumento del 64 % con respecto a la primera versión (17.3 %). Esta fiabilidad final se ha considerado lo suficientemente alta como para detener el entrenamiento en este punto y comenzar con el desarrollo de la aplicación multiplataforma.

Capítulo 4

Aplicación

En este apartado se va a explicar el trabajo realizado para la elaboración de la aplicación.

Después de entrenar los modelos, se procedió al desarrollo de una aplicación móvil y web que pudiese conectarse al Sistema Experto y pudiese realizar predicciones en tiempo real. Para poder desarrollar dicha aplicación, era necesario realizar un análisis en profundidad sobre las tecnologías y herramientas necesarias. Finalmente, se llegó a la conclusión de que se iban a crear dos proyectos independientes que iban a ejercer como servidores locales. Ambos proyectos se encuentran dentro de la carpeta *app* del repositorio de Github. Estos son el backend y el frontend.

4.1. Backend

El primer proyecto es un servidor local API REST que funcionará como backend de la aplicación para recibir las peticiones de la aplicación, y devolver la respuesta correspondiente.

4.1.1. API REST

Una API es una interfaz de programación de aplicaciones que permite la comunicación entre sistemas de información a través del protocolo HTTP (*HyperText Transfer Protocol*). Se basa en un modelo cliente-servidor, en el que el cliente, que puede ser una aplicación o un sistema de software, se comunica con el servidor API mediante la utilización de peticiones HTTP para obtener datos como respuesta o realizar operaciones sobre estos datos en diferentes formatos, como XML o JSON.

Por otro lado, la transferencia de estado representacional o *REST* es una arquitectura de software que establece una serie de condiciones sobre las que debe trabajar una API. Uno de los principios fundamentales para que un API se considere REST es que debe ser una interfaz *stateless*, es decir, que debe ser capaz de procesar todas las peticiones del cliente sin tener en cuenta el estado de las solicitudes anteriores. De esta manera, se puede garantizar que el servidor pueda cumplir con las solicitudes en todo momento. Además, tiene que tener una interfaz uniforme y devolver un recurso con el mismo formato para todos los clientes.

Dentro de la API, se distinguen principalmente 4 tipos de peticiones HTTP:

- POST: crea un dato nuevo en base de datos o realiza una operación.
- GET: devuelve datos.
- PUT: modifica un dato en concreto.
- DELETE: borra un dato.

4.1.2. Django y Django Rest Framework

Para poder implementar una API REST, tomé la decisión de utilizar Django (Django (2024)), que es un framework de código abierto de Python. El motivo principal de su selección fue porque así será más fácil conectarlo con el Sistema Experto, que también fue desarrollado y entrenado en Python. Habían otras alternativas, como NodeJS, que es un servidor backend de Javascript, pero consideré que sería más problemática su implementación y comunicación con el Sistema Experto.

Django sigue una arquitectura MVT (*Model View Template*), que es una variante del conocido MVC (*Model View Controller*), por lo que se pueden separar de la siguiente forma:

- **Model**: la interfaz para los datos en la aplicación. Un Model de Django es la representación de una tabla de la base de datos conectada.
- **View**: es un *endpoint* o URL que recibirá las peticiones HTTP, se conecta a los modelos (Model) de Django para realizar las operaciones correspondientes a la base de datos, y posteriormente devolverá una respuesta HTTP al navegador.
- **Template**: es el código HTML que devuelve Django para que se renderice en el frontend. En este caso, no es importante ya que va a funcionar como una API REST, así que su funcionamiento se va a centrar en la gestión de peticiones del cliente.

Sin embargo, Django no está configurado de forma predterminada para que funcione como un API REST, por lo que fue necesario implementar un módulo adicional llamado Django Rest Framework (DRF), que ofrece una interfaz para acceder al API, y una serie de herramientas para desarrollar los endpoints, peticiones y respuestas.

4.1.3. Base de datos

Se utilizó una base de datos (BBDD) relacional, por mi familiaridad con este tipo de bases de datos, aunque se podría haber optado por una NoSQL, como por ejemplo MongoDB, orientado a documentos; o Neo4j, orientado a grafos.

Por defecto, Django crea una base de datos local de SQLite. Esta forma se caracteriza por el almacenamiento de la información en disco, lo que facilita el acceso, inserción, modificación y eliminación de filas.

Sin embargo, opté por utilizar Oracle como servidor de base de datos SQL, principalmente por la familiaridad de haberlo usado anteriormente en el máster, además de que permite implementar de manera sencilla políticas de enmascaramiento para la anonimización de los datos. Además, con esta implementación se puede mantener la base de datos y el backend en dos servidores separados e independientes.

Se ha utilizado la versión más reciente, Oracle 21c, a través de la herramienta Oracle Database Express, que permite crear un servidor de base de datos en la máquina local.

Una vez creado, para que Django se conecte a la base de datos de Oracle, se ha creado un usuario nuevo llamado SIVARIA_BACKEND, utilizando el puerto 1521 y el servicio xepdb1, que es una PDB (Pluggable Database) que existe por defecto dentro del servidor. Finalmente, se ha usado la herramienta Oracle SQL Developer, para conectarse a la base de datos y desarrollar scripts SQL y PL/SQL sobre las tablas, y así poder implementar el enmascaramiento o realizar pruebas de los datos devueltos.

De todas formas, se ha modificado la configuración del servidor para mantener como respaldo el SQLite de Django, en caso de que haya algún problema con la conexión o las operaciones en Oracle.

Dentro de esta BBDD, Django crea una serie de tablas por defecto, para implementar las funcionalidades de seguridad, permisos, grupos de usuarios, autenticación, autorización, entre otros. Sin embargo, también se han creado tablas nuevas a través de los modelos que he desarrollado para la aplicación, y que se generan mediante la ejecución de las migraciones creadas a lo largo del desarrollo del proyecto. Las tablas generadas finalmente son los siguientes:

- sivaria_appuser (Model: AppUser): tabla de usuarios de la aplicación. Deriva de la tabla de usuarios que crea por defecto Django, por lo que los campos *DATE_JOINED*, *LAST_LOGIN*, *IS_STAFF*, *IS_SUPERUSER* y *IS_ACTIVE* pertenecen a la tabla de Django.

- sivaria_userhasparent (Model: UserHasParent): tabla con la asignación de los padres y profesional correspondiente al joven.

- sivaria_rol (Model: Rol): contiene los roles de la plataforma, que son *joven* (J), *padre* (P), *madre* (M) y *profesional* (PR).

- sivaria_pushnotificationtype (Model: PushNotificationType): tabla con los templates de las notificaciones push que se enviarán al dispositivo móvil. El contenido de la notificación variará en base al rol del usuario y del nivel de riesgo predicho.

- sivaria_emailtemplate (Model: EmailTemplate): tabla con los templates de los correos electrónicos que se enviarán a los usuarios. El contenido del email que se envíe dependerá de su rol y del nivel de riesgo predicho.

- sivaria_injuryform (Model: InjuryForm): contiene las respuestas del usuario al cuestionario sobre la eficiencia del joven a hacerse daño.

- sivaria_rrssform (Model: RrssForm): contiene las respuestas al cuestionario de redes sociales (RRSS).

- sivaria_atiform (Model: AtiForm): contiene las respuestas al cuestionario ATI.

- sivaria_ateform (Model: AteForm): contiene las respuestas al cuestionario ATE.

- sivaria_inqform (Model: InqForm): contiene las respuestas al cuestionario INQ.

- sivaria_multicagecad4form (Model: MulticageCad4Form): contiene las respuestas al cuestionario MULTICAGE-CAD-4.

- sivaria_ebipqecipqform (Model: EbipqEcipqForm): contiene las respuestas al cuestionario EBIPQ y ECIPQ.

- sivaria_erform (Model: ErForm): contiene las respuestas al cuestionario ER.
- sivaria_edform (Model: EdForm): contiene las respuestas al cuestionario ED.
- sivaria_senaform (Model: SenaForm): contiene las respuestas del formulario SENA del cuestionario de los jóvenes en SIVARIA.
- sivaria_cerqsform (Model: CerqsForm): contiene las respuestas al cuestionario CERQS.
- sivaria_familysubform (Model: FamilySubForm): contiene las respuestas a las preguntas relacionadas con las familias, tanto en el cuestionario de jóvenes, como en el de los padres y profesionales.
- sivaria_parqform (Model: ParqForm): contiene las respuestas al formulario PARQ del cuestionario de las familias.
- sivaria_socialdataform (Model: SocialDataForm): contiene las respuestas a las preguntas relacionadas con las variables de identificación del usuario.
- sivaria_senafamilyform (Model: SenaFamilyForm): contiene las respuestas al cuestionario SENA para el cuestionario de las familias.
- sivaria_youngform (Model: YoungForm): contiene las respuestas a todos los formularios del cuestionario de los jóvenes.
- sivaria_familyform (Model: FamilyForm): contiene las respuestas a todos los formularios del cuestionario de las familias.
- sivaria_professionalform (Model: ProfessionalForm): contiene las respuestas a todos los formularios del cuestionario de los profesionales.

Todas las estructuras de estas tablas se encuentran en las Figuras B.5, B.6, B.7, B.8 y B.9.

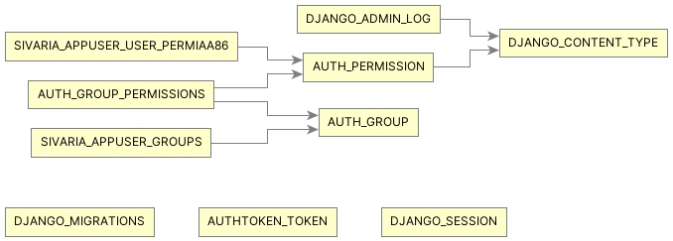


Figura 4.1: Diagrama de las relaciones de las entidades creadas por Django

Por otro lado, si se crean y se ejecutan las migraciones por primera vez en la base de datos SQLite, se genera un archivo nuevo llamado *db.sqlite3*, que es la base de datos SQLite en local.

Todos los modelos creados se encuentran disponibles dentro del panel de administración de Django, cuya URL es https://127.0.0.1:8000/admin, y a la que sólo tienen acceso los superusuarios de la aplicación. Por lo que, para ingresar en el panel, es necesario crearse previamente una cuenta de superusuario.

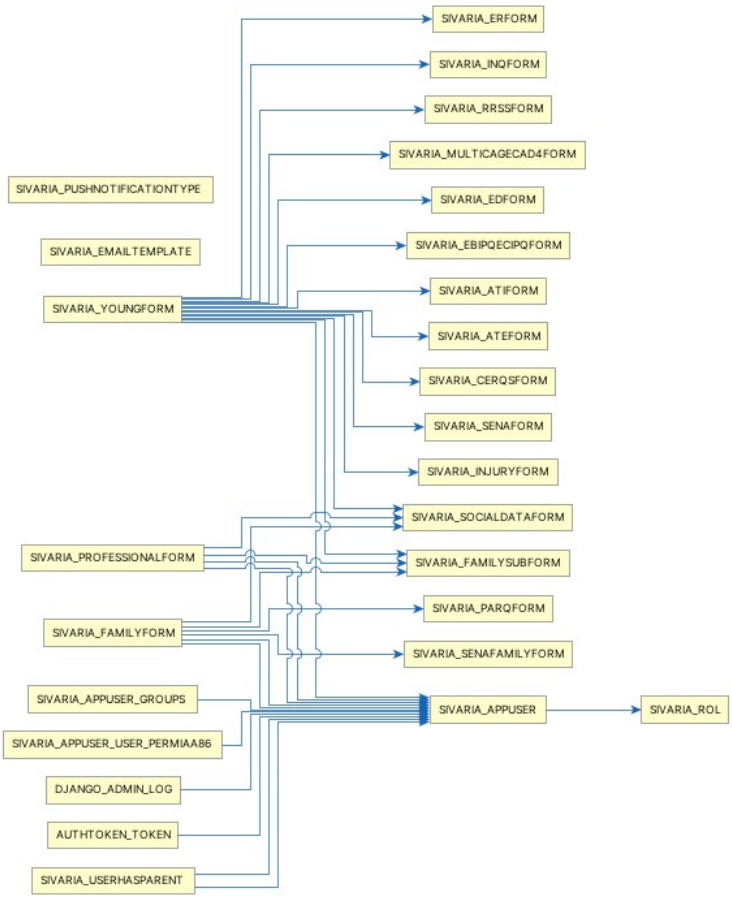


Figura 4.2: Diagrama de las relaciones de las entidades

4.2. Frontend

El segundo proyecto es un servidor frontend desde donde se desarrollará la interfaz de la aplicación. Para ello, había que tener en cuenta que había que seleccionar herramientas y lenguajes que permitiesen crear una plataforma disponible en Android, Web y iOS.

4.2.1. React Native

Teniendo en cuenta la condición de multiplataforma, se decidió utilizar React Native (React (2024)). Es un framework de Javascript de código abierto utilizado para el desarrollo de aplicaciones móviles nativas. La gran ventaja y la principal razón por la que seleccione esta herramienta es que permite crear plataformas compatibles con los sistemas operativos iOS y Android, por lo que no es necesario tener que crear dos proyectos para cada uno.

4.2.2. Expo framework

Posteriormente, se investigó un framework específico de React Native para poder tener un entorno predefinido en donde pueda disponer de herramientas ya creadas que me puedan facilitar el desarrollo. En este contexto, se seleccionó Expo (Expo (2024)), que es un framework específico para aplicaciones de React Native. Proporciona una serie de herramientas adicionales para desarrollar, construir y desplegar aplicaciones en Web, además

de los otros dos sistemas operativos nombrados anteriormente. Esta posibilidad de poder hacer compatible la app en escritorio es lo que me hizo decantarme por él. Utiliza en todos los casos componentes basados en código Javascript o Typescript.

Otra de las grandes ventajas es que dispone de multitud de funcionalidades nativas del móvil a Javascript, como el uso de las cámaras, notificaciones, síntesis de voz, entre otros. Además, dispone de una serie de plataformas para poder desarrollar, desplegar y publicar nuestro proyecto mientras al mismo tiempo estamos trabajando en ellos, similar a Github, aunque esta funcionalidad no es utilizada. Finalmente, este framework se encuentra actualmente activo y recibe constantes actualizaciones, por lo que se evita el riesgo de desarrollar código desfasado y con posibles vulnerabilidades.

Para probar la aplicación, se tuvo que emplear un dispositivo físico. En mi caso, las pruebas se hicieron con un móvil con un sistema operativo Android 14, y para hacerlo es necesario descargar una aplicación en Google Play llamado Expo Go, que permite ejecutar el proyecto a través de un código QR que se genera automáticamente. Aunque también se puede ejecutar el código en un simulador de un móvil Android creado a través de Android Studio. No obstante, esta opción presenta ciertas limitaciones con respecto a uno real. Por ejemplo, la gestión de las notificaciones push no se puede realizar desde el simulador, o el uso de la cámara.

4.3. Conexión entre el frontend y el backend

Una vez explicados ambos proyectos, a continuación se procederá a explicar la conexión entre el frontend y el backend. Para poder realizar peticiones del frontend hasta el servidor, se ha requerido el uso de una librería externa llamada *axios*.

Axios (Axios (2024)), es una librería de Javascript basada en promesas. Es isomórfico, es decir, que se puede emplear tanto en el lado del cliente, a través del navegador con peticiones XMLHttpRequests, como en el lado del servidor, a través del módulo nativo *http* de Node.js.

En este caso, se hace desde el frontend, donde se crea una instancia de axios con una configuración por defecto, añadiendo la URL base del servidor para poder realizar las peticiones, y por lo tanto, establecer conexiones HTTP. Está URL va a variar dependiendo de si la conexión se hace desde el navegador web o desde el dispositivo móvil. Esto es debido a que el backend se ejecuta en local, y por lo tanto, no se puede acceder de forma natural por dispositivos externos a la máquina local.

Si la conexión se realiza desde la web, la URL es el que genera el servidor cuando se inicia, por lo que es más sencilla.

Sin embargo, si se hace desde un dispositivo móvil, se requiere de un servicio extra para crear un puente entre el móvil y el servidor local.

Para resolver este problema, decidí utilizar Ngrok, que es un servicio que permite hostear un servidor local en un subdominio suyo. De esta manera, cualquier dispositivo externo puede acceder al servidor introduciendo la URL del dominio. El proceso de acceso se realiza mediante *Tunnelling*, en donde se crea un túnel para recibir peticiones del exterior. Esta petición se redirige a la URL local donde se aloja el servidor.

[1] Fuente: `https://ngrok.com/blog-post/deploying-ngrok-in-production`

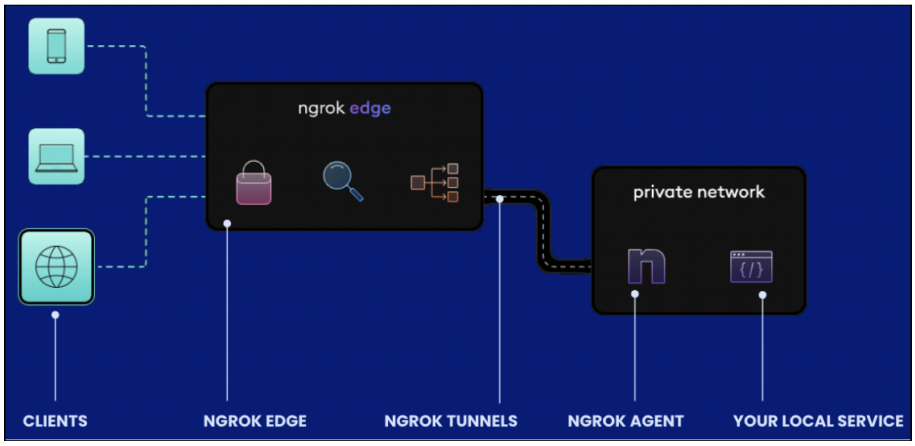


Figura 4.3: Arquitectura de Ngrok[1].

4.4. Casos de uso

En esta sección se van a mostrar los casos de uso diseñados para la aplicación, así como la relación que hay entre ellos.

Para empezar, especifiqué cuáles eran los actores presentes en la aplicación, que son los siguientes:

- Usuario de la aplicación, que puede ser: joven, familiar o profesional.
- Sistema Experto.
- Sistema email
- Sistema de notificaciones push de Expo

Después, realicé el siguiente diagrama de casos de uso, para visualizar y representar la relación entre los actores y los casos de uso.

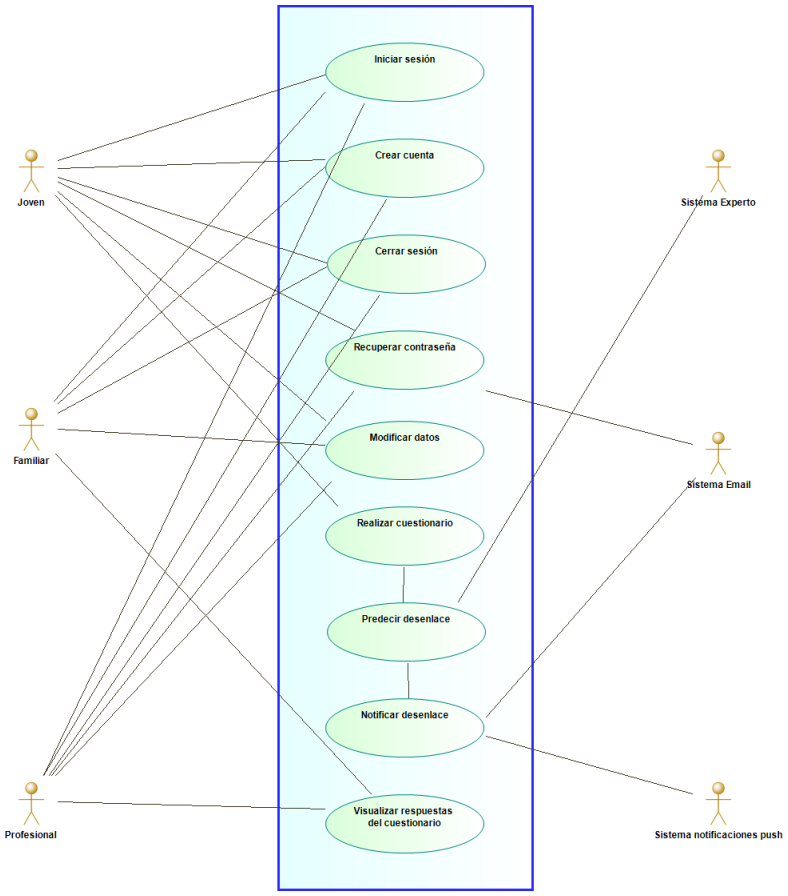


Figura 4.4: Diagrama de casos de uso.

A continuación, desarrollé los correspondientes casos de uso.

4.4.1. Iniciar sesión

El usuario introduce el email y la contraseña. Posteriormente, cuando el usuario dé al botón de *Iniciar sesión*, se envía una petición al backend. Cabe resaltar que en este punto, el usuario todavía no se ha autenticado, por lo que el endpoint de login es de los pocos que es de acceso libre.

Dentro del backend, se validan el email y la contraseña y se comprueba si existe un usuario con esas credenciales. Si no existe, devuelve un mensaje de error. Sino, se genera un nuevo token de autorización y se devuelve en la respuesta, junto con los datos del usuario. Finalmente, estos datos los recibe el cliente, los guarda en el almacenamiento local del móvil o navegador, y es redirigido a la página *Home* de la aplicación para los usuarios registrados.

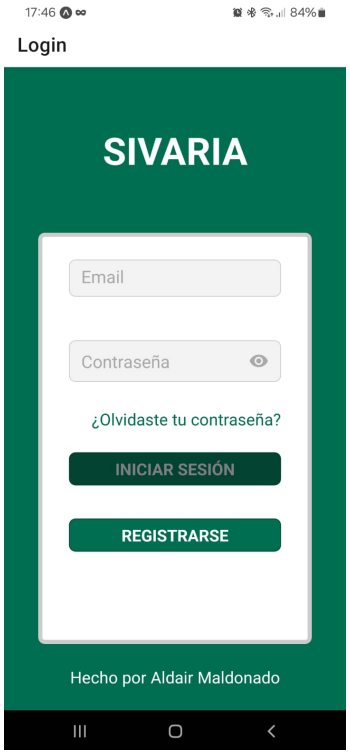


Figura 4.5: Pantalla de inicio de sesión

UC-01	Iniciar sesión	
Descripción	El usuario inicia sesión introduciendo el correo electrónico y la contraseña como credenciales.	
Actores	Usuario (joven, familiar, profesional)	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce el email y la contraseña.
	2	La aplicación valida el email y la contraseña (AS-01).
	3	La aplicación revisa si un usuario existe en la plataforma que coincida con las credenciales introducidas (AS-02).
	4	El usuario es validado.
	5	La aplicación muestra al usuario un mensaje de éxito.
Postcondición	Las credenciales son validadas y el usuario accede a la plataforma.	
Secuencias Alternativas	AS-01	El email y/o la contraseña no son válidos.
	1	Se muestra un mensaje de error al usuario indicando que el email y/o la contraseña no son válidos.
	AS-02	El usuario no existe en la plataforma.
	1	La aplicación muestra un mensaje de error indicando que el usuario no ha sido encontrado en la plataforma.

Tabla 4.1: Caso de uso 1: Iniciar sesión

4.4.2. Registrarse

Para registrarse, el usuario debe dar al botón de *Registrarse*. Después, el usuario es redirigido a la pantalla de registro, donde tiene que introducir sus datos: nombre, apellidos, email, contraseña, número de teléfono y fecha de nacimiento. Dependiendo de la fecha que se introduzca, la aplicación mostrará más campos adicionales o no.

- Si el usuario es menor de 21 años, se motrarán 3 campos nuevos para introducir el email del padre o figura parental 1, madre o figura parental 2, y el profesional asignado.
- Si el usuario es mayor de 21 años, aparecerá un dropdown donde puede ser identificado como padre, madre o profesional. Si el usuario es un padre o madre, entonces se aparecerá un campo nuevo donde debe especificar el correo de su hijo o hija. Si selecciona la opción de *profesional*, no aparecerá nuevos campos, así que no se le pedirá más información.

Finalmente, después de una serie de validaciones en el frontend, los datos son enviados al backend, donde se obtienen los datos del JSON, se genera el hash de la contraseña, se realizan las mismas validaciones que en el frontend y se crea un nuevo registro en base de datos.

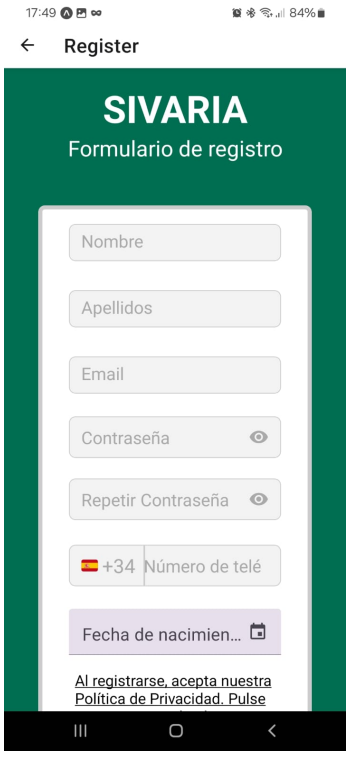


Figura 4.6: Pantalla de registro

UC-02	Crear cuenta	
Descripción	El usuario se registra en la plataforma introduciendo una serie de datos iniciales.	
Actores	Usuario (joven, familiar, profesional).	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce su nombre, apellidos, correo electrónico, contraseña, número de teléfono y fecha de nacimiento (AS-01, AS-02).
	2	El usuario envía los datos introducidos.
	3	La aplicación valida los datos (AS-05).
	4	El usuario es validado
	5	La aplicación crea una cuenta nueva al usuario (AS-06).
	6	La aplicación muestra un mensaje indicando que la cuenta se ha creado exitosamente.
Postcondición	Se crea la cuenta del usuario en la plataforma.	
Secuencias Alternativas	AS-01	El usuario ha puesto una fecha de nacimiento menor de 21 años.
	1	El usuario debe introducir el email del padre o figura parental 1, o madre o figura parental 2, y email del profesional asignado.
	2	Se vuelve al paso 2 de la secuencia principal.
	AS-02	El usuario ha puesto una fecha de nacimiento mayor de 21 años
	1	El usuario debe especificar si es padre, madre o profesional (AS-03, AS-04).
	AS-03	El usuario indica que es padre o madre.
	1	El usuario introduce el email del hijo.
	2	Se vuelve al paso 2 de la secuencia principal.
	AS-04	El usuario especifica que es un profesional
	1	Se vuelve al paso 2 de la secuencia principal.
	AS-05	Los datos introducidos no son válidos.
	1	Se muestra mensaje de error al usuario de que los datos no son válidos.
	AS-06	Ya existe un usuario con el mismo correo electrónico.
	1	La aplicación muestra un mensaje de error indicando que el usuario ya existe en la plataforma.

Tabla 4.2: Caso de uso 2: Crear cuenta

4.4.3. Recuperar contraseña

Si el usuario no recuerda la contraseña de su cuenta, puede pinchar en el enlace de recuperación de contraseña y se le redirige a la pantalla de *Recuperación de Contraseña*. En ella, hay un campo para introducir el correo electrónico que utilizó para crearse la cuenta. Posteriormente, se da al botón de *Enviar* y se enviará automáticamente un correo al usuario usando el sistema de emails de Django. Si no lo recibe es probable que esté en *Correo no deseado*, aunque puede probar a enviar el email múltiples veces. En dicho email, se le proporciona un enlace **web** con el correo del usuario y el token de autorización. El enlace se debe dar desde el navegador, ya que utiliza la URL de la aplicación. Si se da al enlace, se abrirá la página de *Nueva contraseña*. En ella, introduce la nueva contraseña, la envía al servidor, éste la valida y actualiza la contraseña del usuario.

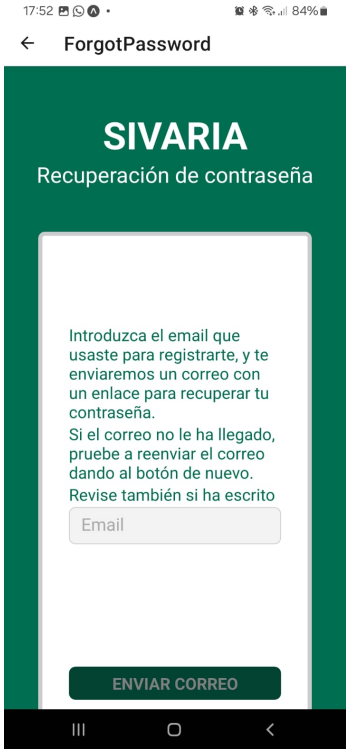


Figura 4.7: Pantalla de recuperación de contraseña

UC-03	Recuperar contraseña	
Descripción	El usuario recupera la contraseña de su cuenta.	
Actores	Usuario (joven, familiar, profesional), sistema de emails	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce un correo electrónico.
	2	El sistema de email envía un correo electrónico al usuario.
	3	El usuario accede al enlace.
	4	El usuario establece una nueva contraseña.
	5	La aplicación valida la nueva contraseña (AS-01).
	6	La aplicación actualiza la contraseña.
	7	Se muestra un mensaje de éxito indicando que se ha modificado la contraseña del usuario.
Postcondición	El usuario crea una nueva contraseña para su cuenta.	
Secuencias Alternativas	AS-01	La contraseña no es válida
	1	La aplicación muestra un mensaje de error indicando que la contraseña no es válida.

Tabla 4.3: Caso de uso 3: Recuperar contraseña

4.4.4. Hacer cuestionarios

Una vez el usuario consigue iniciar sesión, es redirigido a la pantalla de *Home*, donde habrá un botón visible para realizar el cuestionario correspondiente. Una vez se pulse, se le redirige a la pantalla del cuestionario, con una serie de formularios a rellenar, tratando de replicar las preguntas del sitio web de SIVARIA. Cabe resaltar que, dependiendo del rol del usuario (joven, padre o madre, y profesional), las preguntas del cuestionario serán distintas.

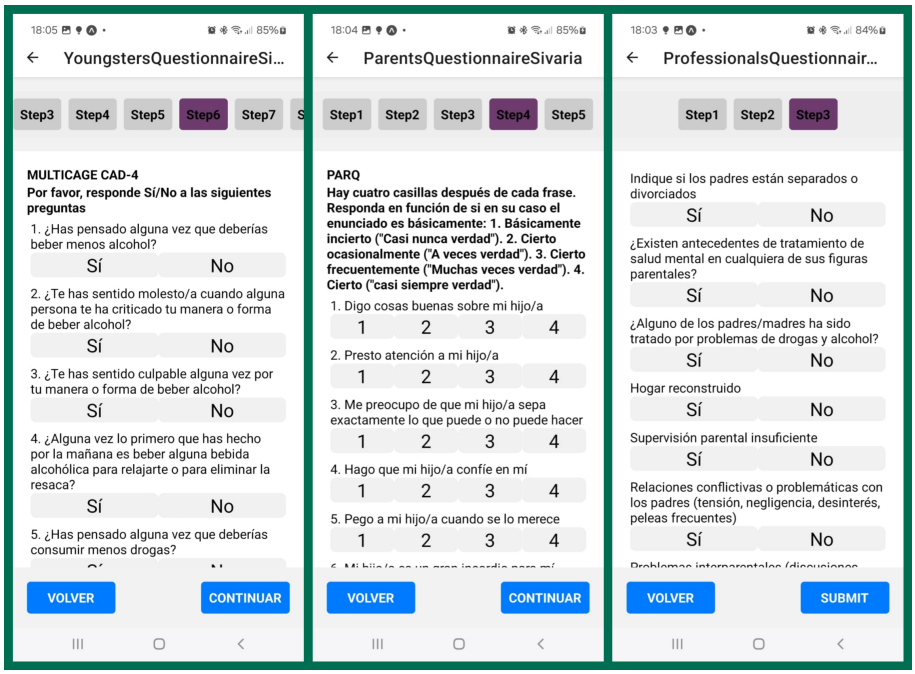


Figura 4.8: Pantallas de cuestionarios de jóvenes (I), familiares (II) y profesionales (III)

Para poder enviar el cuestionario, todos los campos deben estar completados, sino devolverá un error indicando los pasos donde faltan datos. Posteriomente, cuando el usuario envíe el cuestionario al servidor, se hará una petición POST al endpoint de predicción, que realizará las siguientes tareas:

1. Guarda las respuestas del cuestionario en base de datos. Cuando estos cuestionarios son almacenados, se genera un código para cada tipo de sub-cuestionario, y para el cuestionario principal, que también se guardan en base de datos. Este código consta de una secuencia de caracteres alfanuméricos formados por el ID del usuario que ha rellenado el cuestionario en base de datos, su rol (*J*, joven; *P*, padre; *M*, madre; *PR*, profesional), la fecha (AAAA/MM/DD) y hora (HH:MM:SS) de realización del cuestionario.

   <ROL><ID><FECHA><HORA>

   Por ejemplo, si un profesional cuya ID es 12 y rellena el formulario el día 16/08/2024 a las 16:04:34, entonces el código sería PR 15 20240816 160434. O si fuera de un padre en lugar de un profesional, sería P1520240816160434.

   Además, el código de cada sub-cuestionario sigue la misma estructura, pero añadiendo el nombre del sub-cuestionario dentro del propio código.

<ROL><CODIGO_CUESTIONARIO><ID><FECHA><HORA>

Siguiendo con el ejemplo anterior, el sub-cuestionario SENA del cuestionario de los padres, su código sería PSENAFAMILY1520240816160434. Mediante esta técnica, podemos obtener un código único.

2. Mapea los datos a un formato JSON que sea válido para enviar al Sistema Experto.
3. Se envía el nuevo JSON al Sistema Experto, ejecutando internamente el comando de predicción. Dicha predicción devolverá un desenlace probable del caso.
4. Procede a enviar un correo y notificación push a los padres y profesionales involucrados en el caso. Dependiendo del tipo de desenlace predicho, el contenido del correo y push será distinto.
5. Finalmente devuelve una respuesta al frontend, indicando que la operación se ha realizado correctamente.

UC-04	Realizar cuestionario	
Descripción	El usuario realiza el cuestionario y obtiene un predicción.	
Actores	Usuario (joven, familiar, profesional), Sistema Experto, Sistema de Emails, Sistema de notificaciones push	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	El usuario responde al cuestionario.
	2	El usuario envía el cuestionario.
	3	La aplicación revisa que todas las preguntas no están vacías (AS-01).
	4	La aplicación guarda las respuestas.
	5	La aplicación conecta con el Sistema Experto.
	6	La aplicación envía las respuestas del cuestionario.
	7	El Sistema Experto realiza una predicción (UC-07).
	8	La aplicación notifica del desenlace al usuario (UC-08).
	9	La aplicación muestra un mensaje indicando que el mensaje se ha enviado correctamente.
Postcondición	El usuario ha completado el cuestionario y se le ha notificado el resultado de la predicción.	
Secuencias Alternativas	AS-01	Al menos una pregunta del cuestionario está vacía.
	1	La aplicación muestra un mensaje de error indicando las preguntas que están vacías.

Tabla 4.4: Caso de uso 4: Hacer cuestionario

UC-05	Predecir desenlace	
Descripción	El Sistema Experto realiza una predicción	
Actores	Sistema Experto	
Precondición	El usuario debe de estar autenticado y el usuario debe haber respondido a un cuestionario.	
Secuencia Principal	Paso	Acción
	1	El Sistema Experto recibe las respuestas del cuestionario
	2	El sistema se conecta con el modelo dependiendo del rol del usuario.
	3	El Sistema Experto envía las respuestas al modelo.
	4	El modelo realiza la predicción (AS-01)
	5	El Sistema Experto devuelve la predicción
Postcondición	El Sistema Experto devuelve una predicción.	
Secuencias Alternativas	AS-01	Error durante la predicción del Sistema Experto
	1	La aplicación muestra un mensaje de error indicando el fallo en la predicción del Sistema Experto.

Tabla 4.5: Caso de uso 5: Predecir desenlace

UC-06	Notificar desenlace	
Descripción	La aplicación notifica a los usuarios involucrados a través del sistema de emails y el sistema de notificaciones push	
Actores	Sistema de Emails, Sistema de notificaciones push	
Precondición	El usuario debe de estar autenticado y debe haber una predicción realizada previamente.	
Secuencia Principal	Paso	Acción
	1	La aplicación selecciona el contenido del email y push en base a la predicción recibida.
	2	La aplicación busca a los familiares y profesionales relacionados con el que ha rellenado el cuestionario.
	3	La aplicación se conecta al Sistema de Emails (AS-01).
	4	La aplicación se conecta al Sistema de notificaciones push (AS-02).
	5	La aplicación envía un correo electrónico a los usuarios relacionados con el que ha realizado el cuestionario (familiares y profesionales) (AS-03).
	6	La aplicación envía una notificación push a los usuarios relacionados con el que ha realizado el cuestionario (familiares y profesionales) (AS-04).
Postcondición	El Sistema Experto devuelve una predicción.	
Secuencias Alternativas	AS-01	Error de conexión con el sistema de emails.
	1	Muestra mensaje de error indicando el error de conexión con el sistema.
	AS-02	Error de conexión con el sistema de notificaciones push.
	1	Muestra mensaje de error indicando el error de conexión con el sistema.
	AS-03	Error en el envío del correo electrónico.
	1	Muestra un mensaje error indicando el fallo en el envío del correo.
	AS-04	Error en el envío del push.
	1	Muestra un mensaje error indicando el fallo en el envío de la notificación.

Tabla 4.6: Caso de uso 6: Notificar desenlace

4.4.5. Ver respuestas de cuestionarios

Dentro de la aplicación, hay un apartado exclusivo para los padres y profesionales donde pueden ver de los cuestionarios realizados por los profesionales, padres y joven involucrados, y así realizar un seguimiento de los casos de cada joven.

En el caso de los padres, podrán observar los cuestionarios realizados por su pareja, sus hijos o hijas, y el profesional responsable. Mientras que el profesional puede visualizar los cuestionarios de los familiares y pacientes que tiene asignados. De esta manera, se garantiza que las personas deseadas puedan ver toda la actividad de las personas involucradas en el seguimiento y monitorización del paciente.

En la tabla aparecerá el código del cuestionario realizado, el nombre de la persona que lo ha realizado, qué rol tiene, la fecha y hora en la que lo hizo y el desenlace predicho. Esta tabla se puede recargar deslizando hacia arriba en el móvil, o dándole al botón de recarga de la página del navegador, en caso de la aplicación se esté ejecutando desde la web. Por otra parte, si el usuario pincha sobre una de las filas, aparecerá un pop up donde se mostrará todas las respuestas de ese cuestionario en concreto.

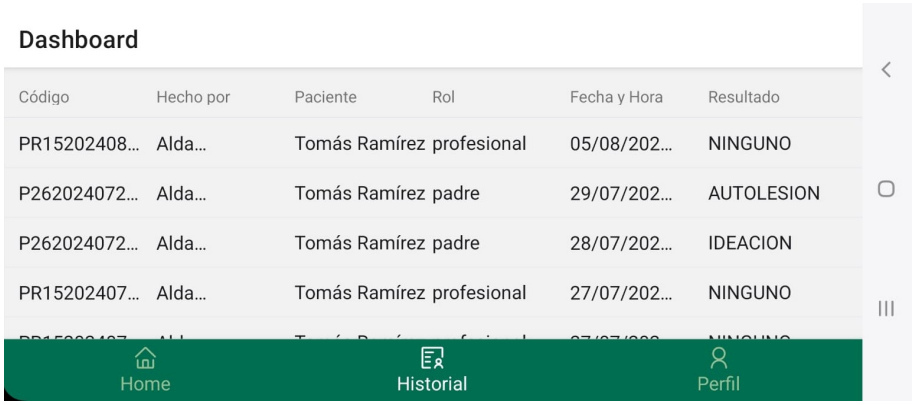


Figura 4.9: Pantalla de registro de cuestionarios

UC-07	Visualizar respuestas del cuestionario	
Descripción	El usuario puede visualizar las respuestas de los cuestionarios	
Actores	Familiar, profesional	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	La aplicación identifica las personas relacionadas con el familiar o profesional que solicita visualizar las respuestas.
	2	La aplicación selecciona los cuestionarios respondidos por las personas relacionadas y el propio usuario.
	3	La aplicación muestra estos resultados.
	4	El usuario selecciona uno de los cuestionarios.
	5	La aplicación muestra las respuestas de ese cuestionario.
Postcondición	El familiar y profesional obtienen todos los cuestionarios de las personas relacionadas con ellos, y obtienen las respuestas de todos los cuestionarios.	

Tabla 4.7: Caso de uso 7: Ver respuestas del cuestionario

4.4.6. Acciones de perfil

Dentro del apartado del perfil, se le mostrará al usuario su información personal, como los nombres y apellidos, email, número de teléfono y el rol que tiene. En el caso de que sea un joven, aparte de la información previa, también se le va a mostrar los emails de sus padres, así como el email de su profesional asignado. Con los padres, aparecerá el nombre y apellidos de sus hijos o hijas registrados en la plataforma. Finalmente, a los profesionales se les muestra el nombre de los pacientes que están a su cargo.

Aparte, en la pantalla de perfil cuenta con funcionalidades adicionales, como son el cierre sesión, la modificación de los datos de su usario, e incluso la posibilidad de poder eliminar su cuenta.

UC-08	Cerrar sesión	
Descripción	El usuario cierra sesión	
Actores	Usuario (joven, familiar, profesional).	
Precondición	El usuario debe de ser autenticado previamente.	
Secuencia Principal	Paso	Acción
	1	El usuario selecciona la opción de cerrar sesión
	2	La aplicación revisa los datos del usuario (AS-01).
	3	La aplicación elimina los datos almacenados en el dispositivo.
	4	La aplicación redirige al usuario a la pantalla de inicio de sesión
	5	La aplicación muestra un mensaje indicando que se ha cerrado la sesión exitosamente.
Postcondición	Se cierra la sesión del usuario de la plataforma	
Secuencias Alternativas	AS-01	Ha habido un error revisando los datos del usuario.
	1	La aplicación muestra un mensaje de error indicando que ha habido un fallo en el cierre de sesión.

Tabla 4.8: Caso de uso 8: Cerrar sesión

UC-09	Modificar datos	
Descripción	El usuario actualiza sus datos	
Actores	Usuario (joven, familiar, profesional)	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	El usuario introduce los datos nuevos (contraseña y número de teléfono).
	2	El usuario envía los datos nuevos.
	3	La aplicación valida la contraseña y número de teléfono (AS-01, AS-02).
	4	La aplicación actualiza los datos del usuario.
	5	La aplicación muestra un mensaje de éxito indicando que se han modificado correctamente los datos del usuario.
Postcondición	La contraseña y/o el número de teléfono se han actualizado.	
Secuencias Alternativas	AS-01	La contraseña no es válida
	1	La aplicación muestra un mensaje de error indicando que la contraseña no es válida.
	AS-02	El número de teléfono no es válido
	1	La aplicación muestra un mensaje de error indicando que el número de teléfono no es válida.

Tabla 4.9: Caso de uso 9: Modificar datos

4.5. Envío de notificaciones push

Dentro de la aplicación, se informan de las respuestas de los cuestionarios a los padres y profesionales a través de un correo electrónico y una notificación push que les llega al dispositivo móvil. Con respecto a esto último, internamente tienen una lógica diferente dependiendo de si se debe enviar a un dispositivo Android o iOS.

- Si el receptor es Android, se debe enviar la notificación a través de FCM (*Firebase Cloud Messaging*), que es una API nativa de Google específica para la recepción y envío de notificaciones push. Una vez la petición llega a Firebase, lo reenvia al destinatario.
- Si por el contrario, el sistema operativo es iOS, el servicio al que se envía es APNS (*Apple Push Notification Service*), que realiza las mismas funciones que el FCM.

Esto implica que el desarrollador tiene que crear dos códigos distintos para ambas plataformas, duplicando funcionalidades y código para ambos casos, lo cuál es poco eficiente. Para solucionar este problema, Expo ha implementado un servicio suyo de API que agrega una capa de lógica por encima, permitiendo al programador enviar notificaciones a través del API de Expo, y desde ahí se envía a cualquiera de los dos sistemas operativos sin necesidad de repetir trabajo.

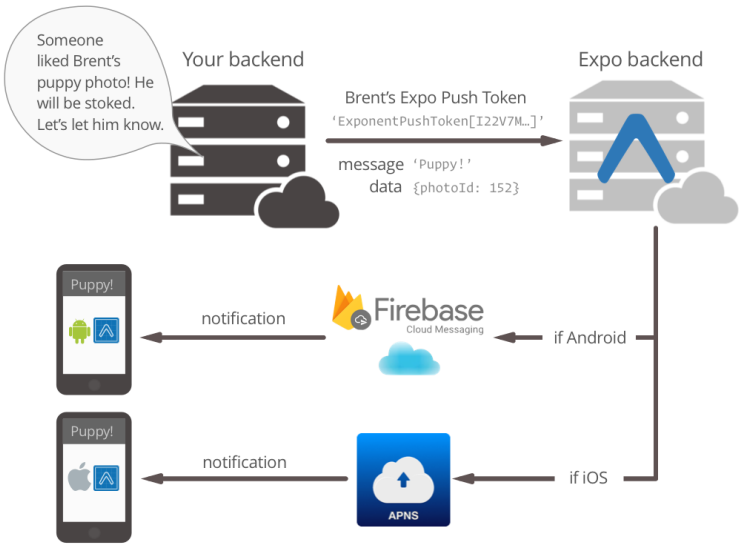


Figura 4.10: Arquitectura del envío de notificaciones de Expo[2].

Para ello, el usuario, cuando entre por primera vez a la aplicación, se le solicitará si puede recibir notificaciones. Si lo acepta, automáticamente se genera un token de Expo denominado *Expo token*. Este token se almacena en base de datos junto con la información del usuario. Cuando se envíe la notificación, el backend accede a la base de datos para obtener el token correspondiente, y posteriormente llama a un endpoint del API de Expo, añadiendo el token del dispositivo, junto el mensaje y datos adicionales, en caso de que los haya. Posteriormente, Expo evalúa el token y determina a qué servidor debe conectarse para enviar el push.

[2]Fuente: `https://docs.expo.dev/push-notifications/sending-notifications/`

4.6. Conexión con el Sistema Experto

La conexión con el Sistema Experto se hará desde el backend a través de una línea de comandos interno explicado en el Apéndice A.

Se desarrolló un prototipo de Sistema Experto en la carpeta *scripts*. Dicha versión del prototipo se replicó dentro del proyecto de backend, en la carpeta *app/backend/sivaria/expert_ system*. Por lo tanto, cuando se necesita ejecutar alguna función del Sistema Experto, el backend inicia un servicio llamado *ExpertService*, que ejecutará el comando especificado por nosotros a través del código. Por ejemplo, si queremos recibir una predicción en base a un JSON recibido de la aplicación de SIVARIA, habría que ejecutar el siguiente comando por código.

```
python controller.py -p --json <base64\_encoded\_json>
```

Para utilizar los modelos, se copiaban los ficheros SAV, que contienen los modelos guardados y que se generaban en los cuadernos de Jupyter. A continuación, se colocaban en una carpeta *configFiles* dentro del backend de la aplicación. Así, el Sistema Experto tendría acceso a él para realizar la predicción.

El prototipo consta de los siguientes componentes.

- **Controller**: recibe el comando introducido por el usuario y ejecuta la correspondiente opción y devuelve el resultado realizado por el Sistema Experto.
- **Expert System**: Sistema Experto de la aplicación. Se encarga internamente de entrenar un modelo seleccionado por el usuario, testear, crear la matriz de confusión y realizar predicciones dado un CSV o un JSON.
- **Checker**: realiza revisiones en el código sobre el formato de los valores recibidos. Si el formato no se cumple, se lanza una excepción.
- **File Manager**: se encarga de administrar los archivos de guardado de las versiones de los modelos. Su trabajo es crear, mostrar y eliminar los archivos de guardado de un modelo. De esta manera, podemos garantizar el control de versiones para todos los modelos y la persistencia de los entrenamientos realizados.
- **Configurator**: se encarga de administrar el archivo de configuración del Sistema Experto. Es capaz de añadir y sustituir valores de la configuración, como el tipo de modelo o puntuación seleccionado.
- **Decoder**: se encarga de recibir el dataframe del CSV, y lo prepara para el entrenamiento del modelo, sustituyendo las variables continuas del CSV por intervalos de valores, y de esta manera, discretizándolos.

Para poder desarrollar todos los componentes, se ha seguido la programación orientada a objetos en Python (*Object Oriented Programming*). Gracias a este paradigma, se pueden dividir los objetos en clases u objetos, lo que permite crear scripts de Python específicos para cada objeto.

Por otro lado, es de importancia mencionar que las peticiones del usuario no se realizan de manera directa al Sistema Experto, sino que al principio van dirigidos al Controlador, quien es el que recibe las órdenes del usuario y llama al Sistema Experto para realizar las correspondientes acciones.

Entonces, la estructura sería similar a lo que se puede ver en la Figura 4.11.

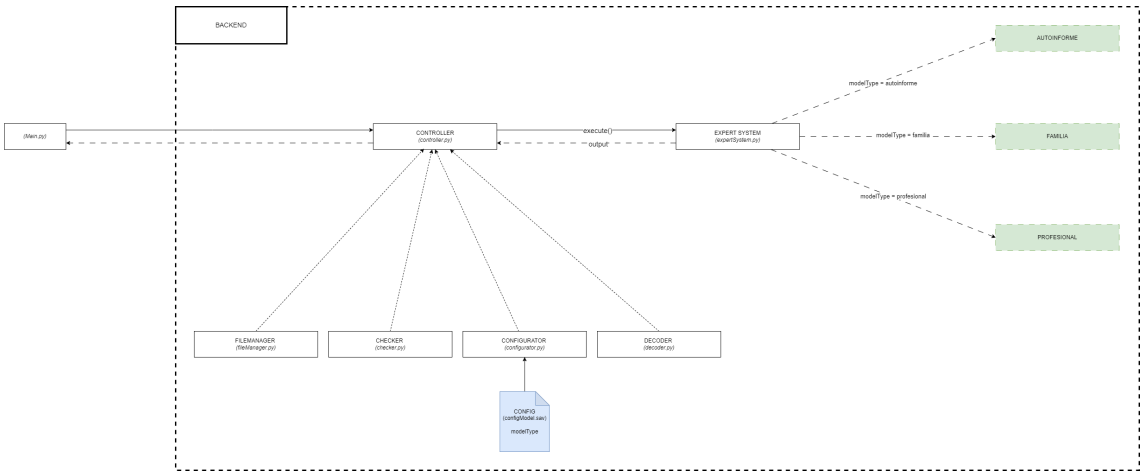


Figura 4.11: Estructura del Controlador con el Sistema Experto

4.6.1. Funcionalidades del código

En esta sección se va mostrar las funcionalidades que hay en el Sistema Experto.

- **Establecimiento del tipo de modelo**: esta funcionalidad permite mostrar y establecer el tipo de modelo que se va a utilizar para el entrenamiento del modelo. Los valores permitidos son: *autoinforme*, *familia* y *profesional*.

  Posteriormente, el valor establecido se guarda en un archivo de configuración en la carpeta *configFiles* del repositorio. Si esto no se especifica, luego el Sistema Experto no sabe qué modelo escoger para realizar el entrenamiento o la predicción.

  En la Figura B.10 se puede ver un diagrama de secuencia del comando en cuestión.

- **Listado de archivos de guardado**: devuelve una lista de los archivos de guardado SAV que hay dentro de la carpeta *configFiles/<tipo del modelo>*. Se puede especificar el modelo para ver los archivos de guardado de ese modelo, o no se introduce ningún parámetro extra, y como resultado, se devuelve los archivos de guardado de todos los modelos.

  En la Figura B.11 se puede observar el diagrama de secuencia del comando que muestra la lista de archivos de guardado de los modelos.

- **Información del archivo de guardado**: se recibe el nombre de un archivo de guardado y devuelve la información básica de ese archivo, que consta de:

  - El nombre del archivo.
  - La fecha y hora del último acceso al archivo.
  - La fecha y hora de la última modificación.
  - La fecha hora de la última vez que ha habido un cambio en los metadatos en Unix y la hora de creación en Windows.
  - El tamaño del archivo
  - La unidad del tamaño del archivo (KiB, MiB, GiB, TiB, etc.).

  En la Figura B.12 se puede observar el diagrama de secuencia del comando.

- **Eliminación de un archivo de guardado**: elimina un archivo de guardado dado su nombre. En la Figura B.13 se puede observar el diagrama de secuencia del comando del borrado del archivo de guardado.

- **Entrenamiento del modelo**: dado un dataset de entrenamiento por parte del usuario (se debe especificar el path del archivo), divide el dataset en base al balance seleccionado en la Sección 3.3.1.7.

  Posteriormente, se entrena al modelo con la parte correspondiente del dataset, y los datos restantes se usarán para crear la matriz de confusión y testear el modelo, creando las métricas *accuracy*, *precision*, *recall* y *f1*.

  Finalmente, las métricas modelo y el modelo entrenado se guarda en un archivo SAV dentro de la carpeta *configFiles/<tipo de modelo>*, y se devuelve un diccionario de Python, que contiene el número de verdaderos/falsos positivos y negativos, tanto totales como de cada clase de *Desenlace*. Es necesario que antes de realizar el entrenamiento se haya especificado el tipo de modelo. Para visualizarlo mejor, en la Figura B.14 se puede observar el diagrama de secuencia donde se muestra los pasos internos del entrenamiento.

- **Predicción**: dado un dataset enviado a través de los parámetros del comando, se realiza una predicción en base a los datos proporcionados. Se carga el modelo de un fichero SAV dentro de la carpeta *configFiles*, dependiendo del tipo de modelo seleccionado en el archivo de configuración. Si hay un modelo especificado, entonces se conecta al Sistema Experto, carga la versión del modelo más reciente, realiza una predicción con los datos del dataset, y devuelve un dataframe con los resultados de esa predicción. En la Figura B.15 se puede observar el diagrama de secuencia del comando.

4.7. Seguridad de la plataforma

Se han implementado medidas de seguridad para la comunicación entre el cliente y servidor, ya que se encuentran en dominios distintos, y pueden generar ciertos problemas de seguridad.

En primer lugar, se han optado medidas en relación a la conexión y comunicación entre el dispositivo móvil y el servidor, que se realiza mediante *Tunnelling* con Ngrok. Este proceso genera un agujero en el firewall, ya que implica publicar un servidor local, lo cuál puede generar un problema de vulnerabilidad. Ante este problema, el comando *ngrok* aloja los servidores en dominios HTTPS, garantizando que las comunicaciones entre ambos componentes vayan cifradas. Por otro lado, la URL se regenera automáticamente cada vez que se ejecuta el comando *ngrok*. Por último, para añadir más seguridad, la URL no se comparte públicamente, sino que se guarda en un fichero .env, que se usa para la configuración de variables de entorno. Este archivo no se sube al repositorio, sino que el desarrollador debe generarlo de forma local.

Con respecto al servidor backend, por lo general un servidor está expuesto a ataques de inyecciones SQL, en donde el atacante inserta en un formulario consultas o parte de una consulta SQL para que se ejecute en el lado del servidor, y devuelva datos comprometidos de la base de datos. Para solucionar esta vulnerabilidad, Django se encarga internamente de limpiar y escapar caracteres especiales. Esto está implementado a través del conjunto de consultas de Django, que están creados usando la parametrización de la query. Este

proceso consiste en definir el código SQL de manera separada de los parámetros recibidos del exterior, ya que pueden ser inseguros.

Por otro lado, el servidor de Django, por defecto, está configurado para recibir peticiones solamente de su propio dominio. Esto se hace para garantizar la protección frente a ataques de *Cross-Site Scripting* (XSS), ataques CSRF (*Cross-site Request Forgery*) o ataques de envenenamiento de caché o envenenamiento de los links de los correos electrónicos. No obstante, como el frontend se aloja en un servidor distinto, se tuvo que cambiar dicha configuración para añadir una constante nueva llamada *ALLOWED_HOSTS*, que es similar a una *whitelist*. Este valor es una lista donde se colocan los dominios que pueden conectarse al servidor. En ella, se añade la URL del frontend, tanto de nuestra máquina local, como la del dispositivo móvil, que es la que genera Ngrok. Cualquier otra petición hecha desde otro servidor, tanto externo como interno, será rechazada.

Con respecto al acceso a la aplicación, se sigue un método de autenticación y autorización por token. Por lo tanto, cada vez que el usuario inicia sesión en la plataforma, el servidor genera un nuevo token de autorización al cliente para poder ejecutar posteriormente los endpoints del backend, ya que la gran mayoría de ellos requieren de acceso autorizado. Por último, las contraseñas de los usuarios no se almacenan en texto en claro, puesto que esto representaría una vulnerabilidad en caso de que un atacante pueda acceder a los datos de los usuarios. Para evitar esto, antes de crear un nuevo registro, se genera un hash de la contraseña utilizando el algoritmo criptográfico SHA-256, que genera una secuencia de números y caracteres de 256 bits, y añadiendo una *salt* aleatoria. Este hash es el se guarda finalmente en base de datos.

Como medida adicional a lo anterior, durante la generación de un hash o de un token, Django los firma mediante el uso de una clave secreta. Esta clave secreta se crea automáticamente cuando se inicia un proyecto nuevo, y se almacena en la variable de configuración *SECRET_KEY*. Pero esta clave secreta no se puede compartir en el repositorio de Github, porque se estaría exponiendo públicamente. Es por ello que se ha cambiado la configuración del servidor para que dicha clave la genere cada uno de forma local, y se debe indicar el valor de la clave en las variables de entorno. De este modo, evitamos que la clave secreta se comparta a través del repositorio, garantizando su seguridad.

Finalmente, con respecto a la base de datos, Oracle proporciona un paquete llamado DBMS_REDACT, que pertenece a Oracle Data Redaction, y que permite implementar políticas de enmascaramiento o redacción a las tablas de la base de datos. Mediante esta técnica, se garantiza la anonimización de los datos, en donde la información confidencial de un usuario se oculta para los usuarios o aplicaciones con pocos privilegios. Existen diferentes tipos de enmascaramiento:

- DBMS_REDACT.FULL : se enmascaran todos los datos confidenciales.
- DBMS_REDACT.PARTIAL : se enmascaran sólo una parte de los datos.
- DBMS_REDACT.REGEXP : los datos son enmascarados según reglas definidas en expresiones regulares.
- DBMS_REDACT.RANDOM : se enmascaran los datos confidenciales por datos aleatorios.

Para implementar estas polítcias, en primer lugar, se definió cuáles eran los datos más sensibles para poder redactarlos. Posteriormente, se establecieron las políticas desde el usuario SYSTEM, que tiene el rol de administrador, de tal forma que se apliquen para

todos los usuarios que no sea el propio SYSTEM o SIVARIA_BACKEND. Las columnas de las tablas que han sido seleccionadas para ser redactadas se encuentran en la Tabla B.1, en el B.

Todas las políticas se implementaron desde Oracle SQL Developer mediante procedimientos PL/SQL que se encuentran en el fichero *sivaria_connection_xepdb1_policies.sql* en la carpeta raíz de *app/backend*.

4.8. Conclusión

La aplicación permite que los usuarios finales puedan acceder a una interfaz desde donde pueden realizar los cuestionarios, hacer peticiones al servidor desde el navegador o desde el dispositivo móvil para realizar una predicción por el Sistema Experto; y finalmente recibir una notificación push y email a los familiares y profesionales. Todo esto gracias a los casos de uso y los diagramas donde se muestra la relación entre todas ellas.

La división del proyecto en dos servidores frontend y backend permite separar la interacción con el usuario y el acceso y manipulación de los datos.

Por otra parte, cuenta con una base de datos Oracle que supera a la opción predeterminada de Django, que era SQSLite, y ofrece opciones más avanzadas para el almacenamiento y enmascaramiento de los datos, consiguiendo que la información permanezca oculta para usuarios no autorizados. Sin embargo, a gran escala, creo que sería necesario una migración a una base de datos NoSQL.

Por último, las medidas de seguridad buscan garantizar la protección de los datos de los usuarios, ya sea aplicando medidas de seguridad propias, como las validaciones manuales o la limpieza de caracteres, o utilizando las opciones que ya vienen en los frameworks, como la configuración de Django para permitir qué dominios pueden enviar peticiones.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En conclusión, este proyecto tiene como objetivo desarrollar el Sistema Experto y la aplicación que ayuden a familias y expertos a monitorizar el riesgo de conductas autolesivas suicidas y no suicidas en adolescentes.

Durante este desarrollo, he aprendido el potencial de ChatGPT, que es una gran herramienta para generar todo tipo de datasets basados en un prompt inicial, con respecto a otros LLMs. Antes, se intentó crearlos a través de otros métodos, como un script o con otro LLM como Gemini, pero ninguno ofreció resultados más completos y realistas que ChatGPT. Esto es debido a que Gemini era incapaz de aplicar diversidad a los datos, y por lo tanto, todos acababan en el mismo desenlace. Por otro lado, ChatGPT ofrece diferentes modelos GPT, como son GPT-3.5 y GPT-4 la versión básica, y GPT-4o el que proporciona respuestas más elaboradas y rápidas. De esta forma, pude entender la gran influencia de los datasets en el entrenamiento de un modelo bayesiano, y la necesidad de que estos sean coherentes y realistas en la medida de lo posible. El gran inconveniente es que ChatGPT limita el número de mensajes que se pueden enviar a GPT-4o en su versión gratuita, lo que ralentizaba la generación de los datos. Una vez superado el límite, sólo se podía utilizar el GPT-4 o GPT-3.5, dependiendo de la demanda. Y si se superaba ese límite también, ChatGPT no permitía enviar más mensajes y había que esperar un tiempo relativamente largo para volver a acceder.

En segundo lugar, pude comprender los distintos tipos de modelos bayesianos que pueden haber, dependiendo de la distribución que haya de los datos con los que trabajemos. Si estos son numéricos, habría que optar por un modelo gausiano o multinomial, y si la distribución es categórica, entonces por una alternativa categórica.

En tercer lugar, con respecto a la aplicación, una enseñanza muy útil fue el aprendizaje de nuevas herramientas y tecnologías para desarrollar la aplicación. He comprendido la importancia de React Native para la creación de aplicaciones para dispositivos móviles por su compatibilidad con iOS y Android, y Python para la creación y entrenamiento de los modelos. Asimismo, este trabajo me ha ayudado a aprender sobre el framework de Expo, con las facilidades que otorga para el envío de notificaciones, y su compatibilidad con Web, además de iOS y Android; y Django, con el que he profundizado mis conocimientos acerca de su funcionamiento como API REST.

En cuarto lugar, una conclusión interesante fue la comparación entre una base de datos

relacional Oracle y SQLite. Oracle es una opción mucho más completa y avanzada que SQLite, ya que ofrece control de las cuentas de usuarios, espacio de tablas para cada tipo de usuario, gestión de roles, y la posibilidad de aplicar políticas de enmascaramiento de datos. Mientras que SQLite es una solución más a corto plazo e ideal para manejar pequeñas cantidades de datos, sus datos se guardan en disco y no proporcionan ninguna de las ventajas de Oracle.

Finalmente, el incremento de este tipo de proyectos que implementan la técnica de las redes bayesianas ayudarán a contribuir al campo de la salud mental, ayudando a los profesionales realizar diagnósticos con mayor precisión, y permitiendo a los jóvenes y familiares pueden recibir una predicción del modelo, que se hará con gran exactitud ya que estará basado en datos de otros pacientes anteriores, que se habrán usado durante el entrenamiento y el testeo.

5.2. Trabajo a futuro

Con respecto al trabajo a futuro, se ha planteado la posibilidad de llevar el trabajo del Sistema Experto a la nube. La velocidad y el rendmiento de los entrenamientos de los modelos han sido de los principales inconvenientes a la hora de escoger una librería de Python para desarrollar el Sistema Experto, ya que la gran mayoría necesitaban demasiado tiempo para realizar los procesos de creación, entrenamiento, testeo y predicción del modelo en una máquina local. En algunos casos, ha provocado fallos en la ejecución. Con esta solución, se podría distribuir la carga de trabajo entre varios servidores que tengan mayor capacidad que una máquina local, así como el almacenamiento, agilizando el trabajo y obteniendo respuestas más rápidas de los modelos.

Por otro lado, se podrían aplicar ciertas mejoras al Sistema Experto. A pesar de que se han podido obtener unos resultados buenos con los entrenamientos de los modelos, se ha tenido que recurrir a datasets generados a través de un LLM, en lugar de emplear los datasets de los psicólogos que colaboraban en este proyecto. Se espera que en el futuro, se puedan obtener los datasets originales de los psicólogos, lo que permitiría diseñar un modelo o modelos más precisos, y acorde a sus intereses y necesidades. También evitarían posibles incoherencias en los datos, lo que mejoraría el entrenamiento. Además, con esta mejora se podría añadir más variables y valores al modelo, calibrandólos más y teniendo en cuenta nuevos factores que afecten a la salud mental de los jóvenes.

En el tema de la aplicación, una de las principales mejoras es la migración de la base de datos. Actualmente, la base de datos que se está utilizando es una relacional, que es Oracle, aunque también se está usando SQLite como respaldo. Oracle es un servidor que nos permite poder separar el servidor de la base de datos, además de añadir más funcionalidades para la gestión de usuarios y almacenamiento de las tablas. Por otro lado, SQLite se caracteriza por almacenar datos en disco, y es de utilidad cuando vamos a manejar pocos datos. Sin embargo, a largo plazo, puede ser contraproducente el seguir usándolo como forma de almacenamiento, aunque sea de respaldo. Por otra parte, el uso de una base de datos relacional puede no ser el mejor enfoque para guardar los datos de los cuestionarios y las predicciones. Es por eso se ha planteado en un futuro, migrar la base de datos a una orientada a grafos, que es una base de datos no relacional. Con esta modalidad, se pueden establecer relaciones más fácilmente entre los padres, jóvenes y profesionales.

También se tiene pensado implementar nuevas opciones a la aplicación para que puedan generar gráficos y estadísticas bajo demanda de los profesionales, además de la posibilidad de crear reportes de cada paciente mediante el uso de un LLM, ya sea el que se ha utilizado

para los datasets u otro distinto.

Finalmente, se debería probar la aplicación sobre usuarios finales, para poder validar tanto la eficiencia del Sistema Experto como de las funcionalidades de la propia aplicación. Para ello, se debería evaluar sobre un número determinado de personas jóvenes, padres y profesionales que puedan reflejar la mayor cantidad de variables posibles de los modelos.

Capítulo 6

Introduction

6.1. Motivation

In recent years, suicide has become a global health problem, being one of the leading causes of unnatural death among young people and adults.

This is why its prevention and mitigation has presented a challenge for health professionals, since this problem affects both the individuals themselves, as well as their family members and close friends.

For this reason, in recent years, efforts and economic resources have been focused on the creation of suicide prevention programs. In these plans, it is essential to make diagnoses as early as possible in order to detect mental symptoms in the person in time, and therefore, be able to offer treatment. The problem is that many times, people at risk of adopting these suicidal and self-injurious behaviors are not aware that they suffer from them or that they are at risk of suicide or self-injury, so they do not attend prevention plans. It leads to the aggravation of the problem, until it ends in a fatal outcome. It is important to be able to inform the person in time about his or her mental health condition automatically, quickly and effectively, so that he or she is aware of the risk and therefore be able to ask for help in advance.

Against this backdrop, the aim is to develop tools to prevent this type of behavior in young people. This is where artificial intelligence begins to play a fundamental role. The artificial intelligence is being used in the field of mental health, being of great use in suicide prevention and to support professionals in achieving an accurate assessment of suicide risk. In this way, potential patients with these self-injurious or non-self-injurious behaviors can be treated at very early stages in the development of these behaviors.

The main reason for the choice of this topic is because of the relevance that this subject has been acquiring over the years. In addition, it is a project that covered the section of web and mobile application development, which I am more familiar with.

6.2. Goals

The main objective of the project is to develop a platform that, through a series of questionnaires, can make predictions of possible suicidal and non-suicidal self-injurious behaviors in young adolescents between 12 and 21 years of age by querying an Expert

System. To achieve this main objective, the following project objectives were established throughout the development of the thesis.

- To design an Expert System for the assessment of the risk of suicidal and non-suicidal self-injurious behaviors in adolescent population (12-21 years old).
- To design and develop a web and mobile application from which the predictions of the Expert System can be made.

6.3. Work Plan

Based on the objectives set out in the previous section, the following work plan was established.

- Selection of the type of Expert System. A search of the different types of Expert Systems available will be carried out and the most suitable one will be selected for each type.
- Understanding the SIVARIA project. Access the SIVARIA project website and investigate and understand the questionnaires on the page, which will be useful to design the Expert System models.
- Selection of tools and libraries to implement the models for the selected Expert System.
- Design and implementation of the models using the selected tools and libraries.
- Train and validate the models created through several cycles or iterations. The results of the iterations will be interpreted and the parameters and input data will be improved until an acceptable performance of the Expert System models is achieved.
- Study of tools and technologies for the creation of the mobile application.
- Creation of the use cases for the development of the application.
- Development of the application is carried out based on the use cases specified by using the tools studied.

All the code made during the development of the project can be found in the following Github repository.

`https://github.com/NILGroup/TFM-2324-SIVARIA`

It has MIT license, and the repository is public. Inside the repository there are 4 main folders:

6.4. Report Structure

The content of the report consists of the following sections.

6.4.1. State of the Art

This section will present the current state of youth suicide and the plans and strategies created to prevent the risk of suicide. Some examples of physical plans applied to schools will be shown, since these are the areas where students may interact the most. Further on, more examples will be cited, but in this case, of web and mobile applications that seek to apply their respective prevention, intervention and training plans, by approaching young people with the use of mobile devices. Finally, a comparative analysis is made between the applications and the strengths and weaknesses of each one are evaluated. In this way, we can obtain an overall view of the functions to be performed by the application.

6.4.2. Job Description

This section will explain everything related to the design, creation and training of the models. The work done to design the initial models of the Expert System is shown. Also, the versions of the training performed are shown, together with the interpretation of the results. A series of analyses, and their corresponding conclusions, on each version of the trainings are also included.

6.4.3. Application

This section will show all the work related to the application theme. The two main servers of the application will be explained: the frontend and backend, and the frameworks used to create them, as well as how the connection between the two projects is made, depending on whether it is done from the local machine or the mobile device.

In addition, the structure and design of the database, the use case diagrams, and the operation of the application will be taught.

Finally, we will delve into the security measures implemented in the application, both by the client and the server, in addition to the database.

6.4.4. Conclusions and Future Work

At the end of the report, a series of conclusions will be made about the work done during the training phase of the Expert System models, as well as the application development phase. Subsequently, a series of improvements that can be added to the project in the future will be proposed, such as the use of real datasets, validation of the Expert System with other users, or the possible migration to other alternatives, among others.

6.4.5. Appendix A

The first appendix includes an instruction manual with the steps to perform in order to run a version of the application, in case you want to test it on the local machine. It explains the creation of the virtual environment, the download of the necessary frontend and backend project packages, and the creation and configuration of the database.

6.4.6. Appendix B

It includes a number of figures and tables in addition to the memory.

Capítulo 7

Conclusions and Future Work

7.1. Conclusions

In conclusion, the objectives of the project are to develop the Expert System and the application that will help families and experts to monitor the risk of suicidal and the application to help families and experts to monitor the risk of suicidal and non-suicidal self-injurious behaviors in adolescents.

During this development, I have learned the potential of ChatGPT, which is a great tool to generate all kinds of datasets based on an initial prompt, with respect to other LLMs. Previously, attempts were made to create them through other methods, such as a script or with another LLM such as Gemini, but none offered more complete and realistic results than ChatGPT. This is because Gemini was unable to apply diversity to the data, and therefore, they all ended up with the same outcome. ChatGPT, on the other hand, offers different GPT models, such as GPT-3.5 and GPT-4 as the basic version, and GPT-4 as the one that provides more elaborate and faster responses. In this way, I could understand the great influence of the datasets in the training of a Bayesian model, and the need for these to be consistent and realistic as much as possible. The big drawback is that ChatGPT limits the number of messages that can be sent to GPT-4o in its free version, which slowed down data generation. Once the limit was exceeded, only GPT-4 or GPT-3.5 could be used, depending on demand. And if you exceeded that limit too, ChatGPT did not allow you to send any more messages and you had to wait a relatively long time to get access again.

Secondly, I was able to understand the different types of Bayesian models that can exist, depending on the distribution of the data we are working with. If the data are numerical, we should opt for a Gaussian or multinomial model, and if the distribution is categorical, then a categorical alternative.

Thirdly, with respect to the application, a very useful lesson was learning new tools and technologies to develop the application. I have understood the importance of React Native for the creation of applications for mobile devices because of its compatibility with iOS and Android, and Python for the creation and training of the models. Likewise, this work has helped me learn about the Expo framework, with the facilities it grants for sending notifications, and its compatibility with Web, in addition to iOS and Android; and Django, with which I have deepened my knowledge about its operation as a REST API.

Fourthly, an interesting conclusion was the comparison between an Oracle relational

database and SQLite. Oracle is a much more complete and advanced option than SQLite, as it offers control of user accounts, tablespace for each type of user, role management, and the ability to apply data masking policies. While SQLite is a more short-term solution and ideal for handling small amounts of data, its data is stored on disk and does not provide any of the advantages of Oracle.

Finally, the increase of these type of projects that implement the Bayesian Networks technique will contribute in the mental health field, helping professionales in their diagnoses, and allowing young people and relatives receive a highly accurate prediction from the model, based on training data coming from previous patients that had similar pathologies.

7.2. Future work

With respect to future work, the possibility of taking the work of the Expert System to the cloud has been raised. The speed and performance of the training of the models have been the main drawbacks when choosing a Python library to develop the Expert System, since most of them required too much time to perform the processes of of creating, training, testing and predicting the model on a local machine. In some cases, it has led to execution failures. With this solution, the workload could be distributed among several servers that have more capacity than a local machine, as well as the storage, speeding up the work and obtaining faster responses from the models.

On the other hand, certain improvements could be applied to the Expert System. Although we have been able to obtain good training results, I have had to resort to datasets generated by an LLM, instead of using the datasets of the psychologists that were supposed to be collaborating in this project. It is hoped that in the future, it will be possible to obtain the original datasets from the psychologists, which would allow the design of a more accurate models, and according to their interests and needs. This solution would avoid possible inconsistencies in the data, improving model training. In addition, with this improvement, more variables and values could be introduced to the model, calibrating them further and taking into account new factors affecting the mental health of young people.

Regarding the application, one the main improvements is the migration of the database. Currently, the database that is used is a relational one, which is Oracle, but SQLite is also used as a backup database. Oracle is a database server that allows us to separate both database and backend servers, in addition to adding more functionalities for users management and table storage. On the other hand, SQLite is characterized by storing data on disk, and is useful to handle small data. Nevertheless, as a long-term solution, could be counterproductive to continue using it as form of storage, even if it is a backup. Additionally, the usage of a relational database could not be the best approach to store the questionnaire data and predictions. For this reason, in the future, it is planned to move to a graph-oriented database, which is non-relational. With this approach, relationships between parents, young people and professionals can be established more easily.

It also planned to implement new functionalities in the application in order to generate graphics and statistics on demand by professionals, in addition to the possibility of creating patient reports by using an LLM, either using GPT or trying a new one.

Finally, the application should be tested on end users, in order to validate both the efficiency of the Expert System and the functionalities of the application itself. of the application itself. For this purpose, it should be evaluated on a certain number of young people, parents and professionals who can reflect the greatest possible number of variables of the models. models.

Bibliografía

Agüero-Nate, A. I., Ruíz-Sánchez, O. R. y Fernández-Cruz, H. M. Estrategia de orientación educativa para la prevención de la depresión en adolescentes. *Revista Mexicana de Investigación e Intervención Educativa*, vol. 2(3), páginas 19–28, 2023.

Ankan, A. y Panda, A. pgmpy: Probabilistic graphical models using python. En *SciPy*, páginas 6–11. Citeseer, 2015.

Axios. *Axios Documentation*, 2024. Último acceso: 3 de agosto de 2024.

Burn, A.-M., Hall, P., Anderson, J. et al. A web-based training program for school staff to respond to self-harm: design and development of the supportive response to self-harm program. *JMIR formative research*, vol. 8(1), página e50024, 2024.

Bustamante, F. y Florenzano, R. Programas de prevención del suicidio adolescente en establecimientos escolares: una revisión de la literatura. *Revista chilena de neuro-psiquiatría*, vol. 51(2), páginas 126–136, 2013.

Del Barrio, V., Ramírez-Uclés, I., Romero, C. y Carrasco, M. Á. Adaptación del child-parq/control: versiones para el padre y la madre en población infantil y adolescente española. *Acción psicológica*, vol. 11(2), páginas 27–46, 2014.

Django. *Django Documentation*, 2024. Último acceso: 3 de agosto de 2024.

Expo. *Expo Documentation*, 2024. Último acceso: 3 de agosto de 2024.

Fernández-Pinto, I., Santamaría, P., Sánchez-Sánchez, F., Carrasco, M. y Del Barrio, V. Sistema de evaluación de niños y adolescentes. *SENA. Madrid: TEA Ediciones*, 2015.

Fonseka, T. M., Bhat, V. y Kennedy, S. H. The utility of artificial intelligence in suicide risk prediction and the management of suicidal behaviors. *Australian & New Zealand Journal of Psychiatry*, vol. 53(10), páginas 954–964, 2019.

Garnefski, N. y Kraaij, V. Cognitive emotion regulation questionnaire–development of a short 18-item version (cerq-short). *Personality and individual differences*, vol. 41(6), páginas 1045–1053, 2006.

Gilbert, P. y Allan, S. The role of defeat and entrapment (arrested flight) in depression: an exploration of an evolutionary view. *Psychological medicine*, vol. 28(3), páginas 585–598, 1998.

Grosan, C. y Abraham, A. *Rule-Based Expert Systems*, páginas 149–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21004-4.

Hill, R. M., Rey, Y., Marin, C. E., Sharp, C., Green, K. L. y Pettit, J. W. Evaluating the interpersonal needs questionnaire: Comparison of the reliability, factor structure, and predictive validity across five versions. *Suicide and Life-Threatening Behavior*, vol. 45(3), páginas 302–314, 2015.

King, K. A., Strunk, C. M. y Sorter, M. T. Preliminary effectiveness of surviving the teens® suicide prevention and depression awareness program on adolescents'suicidality and self-efficacy in performing help-seeking behaviors. *Journal of school health*, vol. 81(9), páginas 581–590, 2011.

Kolodner, J. L. An introduction to case-based reasoning. *Artificial intelligence review*, vol. 6(1), páginas 3–34, 1992.

Kruzan, K. P., Whitlock, J., Bazarova, N. N., Bhandari, A. y Chapman, J. Use of a mobile peer support app among young people with nonsuicidal self-injury: Small-scale randomized controlled trial. *JMIR Form Res*, vol. 6(1), página e26526, 2022. ISSN 2561-326X.

Kurniawan, K., Pratama, A., Amalia, A., Robbani, A. N., Lathifah, A., Khoirunnisa, K. y Mulyana, A. M. Exploring effective interventions to reduce self-harm behavior in adolescents: A scoping review. *International Journal of Africa Nursing Sciences*, página 100762, 2024.

Monteagudo Rodenas, J. V. Prevención del suicidio en adolescentes. propuestas educativas desde una perspectiva bioética. 2022.

Ortega-Ruiz, R., Del Rey, R. y Casas, J. A. Evaluar el bullying y el cyberbullying validación española del ebip-q y del ecip-q. *Psicología educativa*, vol. 22(1), páginas 71–79, 2016.

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

Pérez, E. P., Monje, M. R., Alonso, F. G., Girón, M. F., López, M. P. y Romero, J. C. Validación de un instrumento para la detección de trastornos de control de impulsos y adicciones: el multicage cad-4. *Trastornos adictivos*, vol. 9(4), páginas 269–278, 2007.

React. *React Native Documentation*, 2024. Último acceso: 3 de agosto de 2024.

Salsman, N. y Linehan, M. M. Dialectical-behavioral therapy for borderline personality disorder. *Primary Psychiatry*, vol. 13(5), página 51, 2006.

Sánchez, M. I. S., de Pedro, M. M. y Izquierdo, M. G. Propiedades psicométricas de la versión española de la escala de resiliencia de 10 ítems de connor-davidson (cd-risc 10) en una muestra multiocupacional. *Revista latinoamericana de psicología*, vol. 48(3), páginas 159–166, 2016.

Sola, E. J., Alés, G. M., Mazuecos, E. R., Castro, P. S., de Dios Perrino, C., Vega, B. R. y Ortiz, M. F. B. Implementación de un programa de prevención del riesgo de suicidio en la comunidad autónoma de madrid. la experiencia arsuic. *Actas españolas de psiquiatría*, vol. 47(6), páginas 229–235, 2019.

Sucar, L. E. y Tonantzintla, M. Redes bayesianas. *Aprendizaje Automático: conceptos básicos y avanzados*, vol. 77, página 100, 2006.

Wikipedia. Sistema experto — wikipedia, la enciclopedia libre. 2024. [Internet; descargado 17-mayo-2024].

Apéndice A

Guía de instalación

Todo el desarrollo del TFM, tanto código como bocetos de los diagramas, se encuentran en el repositorio online de Github.

A.1. Contenido del repositorio

Al descargarse el repositorio, se pueden observar tres carpetas principales:

- **DiagramBayesSketches**: esta carpeta contiene los bocetos de los diagramas de Bayes de todos los modelos: autoinforme, familia y profesional. Además, aparte hay una subcarpeta con el boceto de cada grupo de variables, para poder ver con mayor claridad cómo está compuesto cada grupo.
- **Memoria**: en ella se encuentra la presente memoria.
- **scripts**: en esta carpeta se encuentran los cuadernos de Jupyter de los entrenamientos, junto con un prototipo del Sistema Experto. Ambos componentes realizan la misma función de entrenamiento y testeo.
- **app**: en esta carpeta se encuentran los dos proyectos de la aplicacíon, tanto el frontend como el backend.

A.1.1. Scripts

Constan de los siguientes archivos:

- **exceptions**: carpeta que contiene una serie de excepciones personalizadas creadas para el Sistema Experto.
- **configFiles**: carpeta donde se almacenará los archivos de guardado de los modelos del Autoinforme, Familia y Profesional que genera el prototipo del Sistema Experto, no de los cuadernos de Jupyter.
- **datasets**: carpeta donde se encuentran los datasets de prueba creados de forma manual, aleatoria y artificial para entrenar a los modelos, y así, probar el rendimiento del Sistema Experto.

- **csvV1Creator**: en esta carpeta se encuentran los scripts que se utilzaron en la primera versión para crear los datasets aleatorios.

- **jupyterNotebooks**: contiene los cuadernos de Jupyter con el código de los entrenamientos de las 4 versiones, además habrá una carpeta **configFilesJupyter**, donde se almacenarán los modelos guardados en archivos de guardado SAV, dependiendo del tipo de modelo.

- *expertSystem.py*: contiene la clase Sistema Experto, que utiliza el módulo scikit learn para realizar las funciones de predicción, entrenamiento y testeo. Se encarga de construir el modelo inicial a partir del tipo de modelo especificado por el usuario, que puede ser el modelo del autoinforme, el de la familia o el de los profesionales. Además, también posee una función para guardar el modelo entrenado en un archivo .sav y que lo puede colocar en las carpetas *scripts/configFiles/autoinforme*, *scripts/configFiles/familia* o *scripts/configFiles/profesional*, también dependiendo del tipo de modelo.

- *controller.py*: ejerce de Controlador y es capaz de utilizar métodos del Sistema Experto a través de una serie de comandos:

  - -h: muestra un mensaje de ayuda que explica los comandos disponibles del controller.py. Tiene un formato similar a las páginas *man* de Linux.
  - -mt: muestra y establece el tipo de modelo seleccionado por el usuario (autoinforme, familia, profesional).
  - -lst: devuelve una lista de archivos de guardado que hay según el tipo de modelo.
  - -fs: devuelve la información del archivo dado su nombre.
  - -rm: elimina un archivo de guardado dado su nombre.
  - -t: entrena al modelo. Es necesario que previamente se establezca el tipo de modelo (-mt). Además, debemos añadir en los parámetros el dataset que se va a usar para el entrenamiento. Devuelve una lista de verdaderos/falsos positivos y verdaderos/falsos negativos de cada tipo de desenlace.
  - -p: realiza la predicción de un dataset especificado a través del path por los parámetros del comando.

- *constants.py*: contiene las constantes que se usa en los dos scripts principales.

- *checker.py*: objeto estático Checker que contiene métodos para revisar el formato del dato de una variable.

- *decoder.py*: objeto estático Decoder con métodos que decodifica un dataframe recibido.

- *fileManager.py*: objeto FileManager con métodos para administrar la creación y eliminación de archivos de guardado de los modelos.

- *main.py*: script principal que crea el objeto Controlador y ejecuta el comando.

A.2. Instrucciones de ejecución

Una vez descargado el repositorio, se necesita crear un entorno virtual para ejecutar los scripts de Python. Se recomienda instalar Anaconda, ya que permite crear entornos virtuales que incluyen una serie de paquetes que son utilizados en el código desarrollado, por ejemplo, Numpy, que ya se encuentra instalado en el entorno de Anaconda.

A.2.1. Entornos de Python con Anaconda

Por defecto, Anaconda tiene un entorno general llamado *base (root)*. Sin embargo, también existe la opción de crear un entorno nuevo aparte del principal. Para ello, dentro de Anaconda hay que ir a la sección *Environments*, y una vez ahí, dar al botón *Create*. Aparecerá una ventana donde se especifica la versión de Python. En mi caso, se ha utilizado la versión 3.9.19.

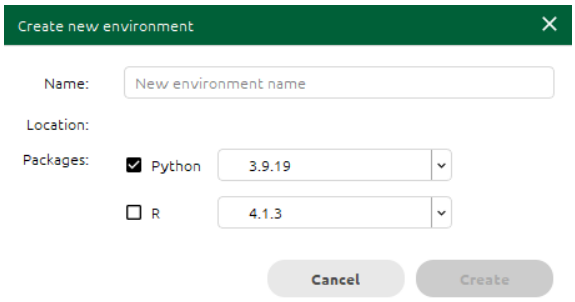


Figura A.1: Ventana de creación de nuevo entorno de Python en Anaconda

Finalmente, se termina el proceso dando botón *Create*.

A.2.2. Conexión del entorno virtual con Visual Studio Code

Cuando el se haya creado el entorno, el siguiente paso es conectar el nuevo entorno, o el *base* si no se ha creado ninguno, a Visual Studio Code.

Para ello, se debe abrir el panel de comandos (`CTRL+Shift+P`) y seleccionar la opción

```
>Python:Select Interpreter
```

Una vez ahí, seleccionamos el entorno. Finalmente, se abre un terminal que se ejecuta dentro del entorno virtual y desde dónde se podrán descargar los paquetes del proyecto y ejecutar los scripts.

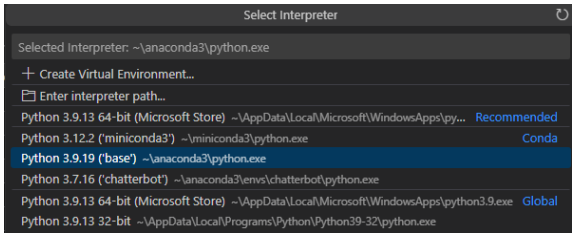


Figura A.2: Panel de selección de entorno de Python en Visual Studio Code

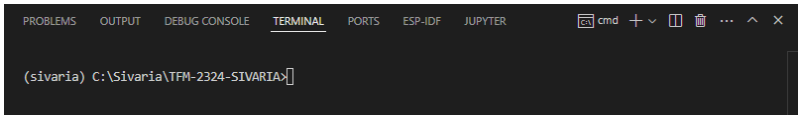


Figura A.3: Consola del nuevo entorno de Python

A.2.3. Instalación de los paquetes

A.2.3.1. Frontend

En esta sección, se explicarán los pasos necesarios para poder ejecutar el servidor frontend del repositorio en local. Todos los comandos que se mostrarán a continuación se deben ejecutar desde la consola del VSCode.

1. Instalar NodeJS, que habilita el uso del comando npm, lo que permite instalar las dependencias del proyecto.
2. Instalar Expo en el móvil sólo en el caso de que se quiera probar la aplicación en el móvil.
3. Instalar Ngrok, que permite crear un túnel en un dispositivo externo y el servidor backend local, también si se quiere probar la app en el dispositivo físico.
4. Ir al proyecto *app/sivaria/*
5. Instalar las dependencias de React Native y Expo mediante el siguiente comando:

   ```
   (sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\sivaria>npm install expo
   ```

6. Crear el archivo de variables de entorno: se debe crear un archivo .env en el directorio raiz de este proyecto. Este .env debe contener los siguientes valores.
   - API_SERVER_MOBILE : URL que genera Ngrok al ejecutarse en la consola. Este enlace se usa cuando se quiere probar la aplicación desde el dispositivo móvil.
   - API_SERVER_WEB : URL del servidor local para probar la aplicación desde un navegador web.

De esta manera, el frontend ya estaría listo para iniciarse en cualquier momento. Para arrancar el servidor, se ejecuta el siguiente comando dentro del directorio anteriormente especificado.

```
(sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\sivaria>npm start
```

Este comando inicia el servidor del proyecto, a la cuál se puede acceder de dos maneras: a través del navegador o a través de un dispositivo móvil.

Para ver la app desde un navegador, se debe pulsar la tecla *w* en la consola del servidor, y automáticamente se abre el nbavegador predeterminado y se realiza el build de la aplicación.

Para ver la app desde el móvil, es necesario utilizar e iniciar la aplicación de Expo. El servidor genera dos formas de ejecutar la aplicación: con un código QR o un enlace expo.

Es importante destacar que la máquina local donde se aloja el servidor API y la aplicación móvil deben estar conectados a la misma red WiFi.

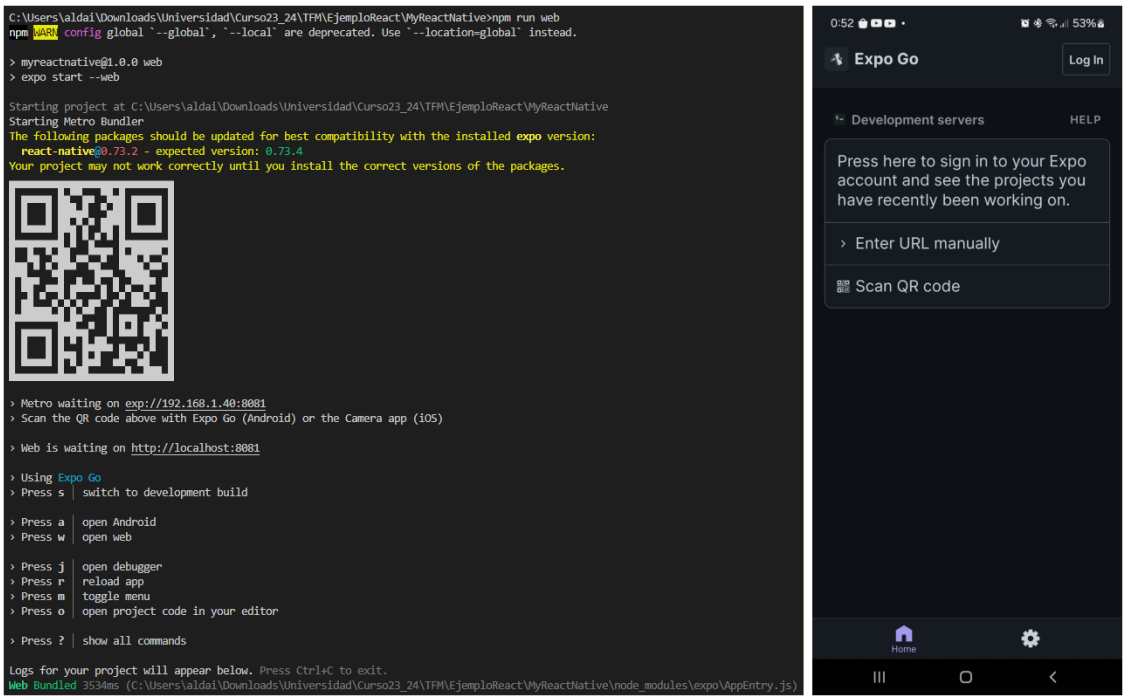


Figura A.4: Consola del servidor e interfaz de la aplicación de Expo Go

A.2.3.2. Scripts de Python y servidor backend

En esta sección, se explicarán los pasos necesarios para poder ejecutar el servidor backend del repositorio en local, además de los scripts de Python.

1. Navegar al directorio *app/backend*

2. Instalar los paquetes necesarios de Python.

   ```
   (sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\backend> pip install -r requirements.txt
   ```

   A partir de aquí se pueden ejecutar los scripts de Python. Sin embargo, para poder ejecutar el servidor backend, es necesario realizar más pasos.

3. Crear archivo de variables de entorno. Al igual que en el frontend, el backend tamibén tiene sus variables de entorno, a través del archivo .env. Este fichero se debe crear desde el directorio raíz del servidor de backend (*app/backend*). Sus variables deben ser las siguientes:

   - DEFAULT_FROM_EMAIL : dirección del correo electrónico desde donde se enviarán los corrreos de la aplicación.
   - EMAIL_HOST_PASSWORD : contraseña de la cuenta de correo electrónico.
   - FRONTEND_SIVARIA_URL : URL del frontend.
   - SECRET_KEY : clave secreta usada para firmar el hash de las contraseñas y los tokens. Se debe generar de forma local. Para eso, se debe abrir la consola de python e introducir las siguientes instrucciones:

```
(sivaria) C:\Sivaria\TFM-2324-SIVARIA>python
>>> from django.core.management.utils import get_random_secret_key
>>> print(get_random_secret_key())
```

Esto devolverá la clave secreta generada. Este valor se copia y pega en la variable SECRET_KEY.

- DEBUG: indica si la aplicación está en modo DEBUG o no. Debe estar a false cuando la aplicación se encuentre en modo de producción.
- ALLOWED_HOSTS : dominios que tienen permiso de conectarse al backend.
- ORACLE_PASSWORD : contraseña de la base de datos local de Oracle.

4. Crear una base de datos con la herramienta Oracle Database Express .
5. Conectarse a la nueva base de datos con el usuario SYSTEM utilizando la herramienta Oracle SQL Developer. Los datos deben ser los siguientes:
   - Usuario: system
   - Contraseña: <La que se estableció al momento de crear la base de datos Oracle>
   - Nombre del host: localhost
   - Puerto: 1521
   - Nombre del servicio: xepdb1
6. Crear el usuario SIVARIA_BACKEND con privilegios de modificación, inserción, eliminación y consulta de las tablas, además del privilegio de la creación y alteración de sesión, vistas y triggers.
7. Crear migraciones (opcional).

   ```
   (sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\backend>python manage.py makemigrations
   ```

8. Ejecutar las migraciones.

   ```
   (sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\backend>python manage.py migrate
   ```

9. Volver a conectarse a la BBDD con el usuario SYSTEM en Oracle SQL Developer y abrir el fichero SQL *sivaria_connection_xepdb1_policies.sql*, y ejecutarlo para aplicar las políticas de enmascaramiento a las tablas.
10. Crear un superusuario, para poder acceder al panel de administración de Django.

    ```
    (sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\backend>python manage.py createsuperuser
    ```

    En cuanto se haya creado el superusuario se puede probar su acceso introduciendo el email y la contraseña en el panel de administración, que se puede acceder a través de la siguiente URL:

    SERVER_URL/admin

    Donde *SERVER_URL* es la URL del servidor cuando se ejecuta.

    Por ejemplo, *http://127.0.0.1:8000/admin*.

De esta manera, ya se podría ejecutar el servidor backend con el siguiente comando.

```
(sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\backend>python manage.py runserver
```

Además, ya se podrían ejecutar los endpoints del API, a través del comando *curl*, desde el navegador accediendo a la URL exacta del endpoint, o usando la herramienta Postman, que es la usé para acceder y probar los endpoints.

Dentro del directorio *app/backend* se encuentra una colección JSON y las variables de entorno que se puede importar al Postman, y que contiene todos los endpoints del API. No obstante, para ejecutarlos, la mayoría de ellos requieren de un token de autorización, y para ello, se debe usar previamente el endpoint del login: *BASE_ URL/sivaria/v1/user/login*. Este devuelve en la respuesta el token de autorización dentro del campo *token* (Figura A.5).

De esta manera, cada vez que se quiera usar un endpoint con autorización, el token debe estar incluido en la cabecera de la petición, como se puede ver en la Figura A.6. Es necesario puntualizar que los token caducan después de un tiempo determinado, normalmente 1 día. Por lo tanto, es posible que cada día haya que usar el login para obtener el nuevo token.

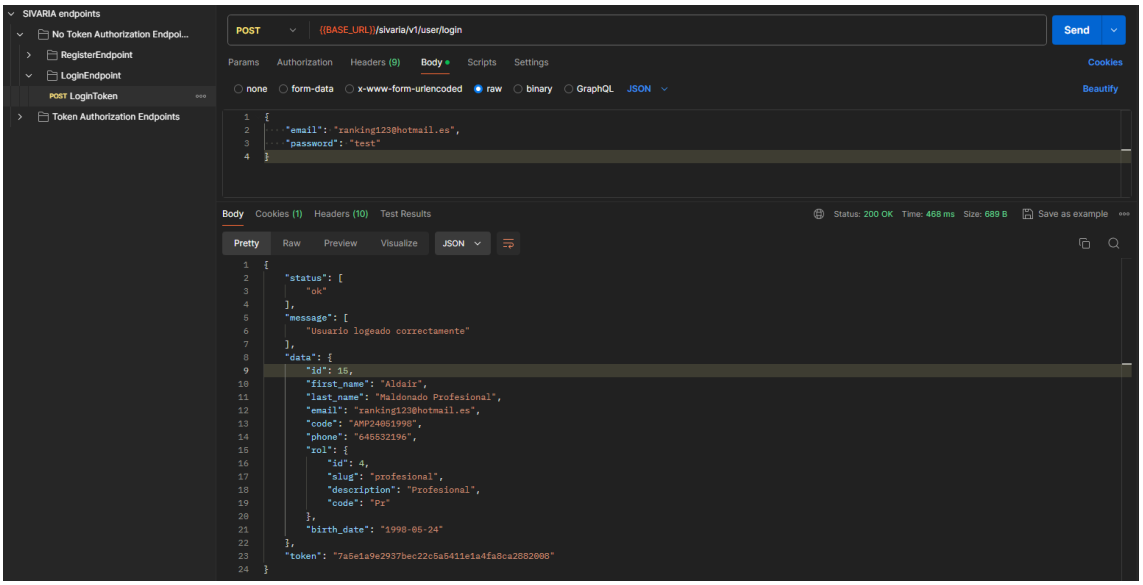


Figura A.5: Login endpoint en Postman

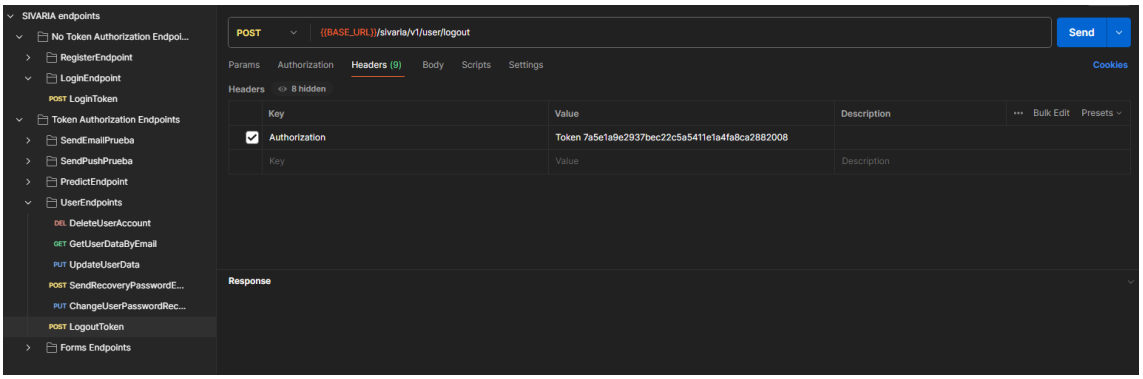


Figura A.6: Cabecera de la petición para los endpoints con autorización

Apéndice B

Figuras y tablas adicionales

B.1. Estructuras del dataset en cada entrenamiento

B.2. Tablas de las bases de datos

B.3. Diagramas de secuencia de las funcionalidades del Sistema Experto

B.4. Políticas de enmascaramiento a las Tablas SQL

Tabla B.1: Políticas de enmascaramiento en las tablas de la BBDD de Oracle

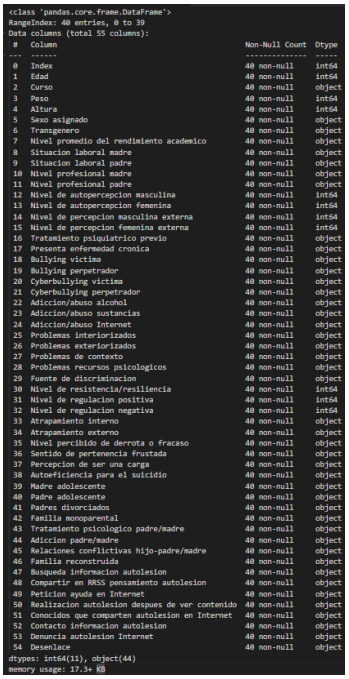
Tabla	**Columna**	**Política DBMS_REDACT aplicada**
AUTHTOKEN_TOKEN	USER_ID	FULL
	CREATED	FULL
	KEY	RANDOM
SIVARIA_APPUSER	EMAIL	REGEXP
	PHONE	FULL
	ROL_ID	FULL
	USERNAME	RANDOM
	FIRST_NAME	RANDOM
	LAST_NAME	RANDOM
	EXPO_TOKEN	RANDOM
	CODE	RANDOM
	BIRTH_DATE	FULL
	DATE_JOINED	FULL
SIVARIA_USERHASPARENT	EMAIL_PARENT_1	REGEXP
	EMAIL_PARENT_2	REGEXP
	RESPONSIBLE_ID	FULL
	CHILD_ID	FULL

Continuación de la Tabla B.1

Tabla	Columna	Política DBMS_REDACT aplicada
SIVARIA_YOUNGFORM	ATE_ID	FULL
	ATI_ID	FULL
	CERQS_ID	FULL
	ECIPQ_EBIPQ_ID	FULL
	ED_ID	FULL
	ER_ID	FULL
	FAMILY_ID	FULL
	INJURY_ID	FULL
	MCAD_ID	FULL
	PARTICIPANT_YOUNG_ID	FULL
	RRSS_ID	FULL
	SENA_ID	FULL
	CODE	FULL
	PREDICTION	FULL
SIVARIA_FAMILYFORM	TO_USER_FAMILY_FORM_ID	FULL
	PARTICIPANT_FAMILY_FORM_ID	FULL
	SENA_FAMILY_ID	FULL
	PARQ_ID	FULL
	FAMILY_ID	FULL
	SOCIAL_DATA_ID	FULL
	CODE	FULL
	PREDICTION	FULL
SIVARIA_PROFESSIONALFORM	TO_USER_PROFESSIONAL_FORM_ID	FULL
	PARTICIPANT_PROFESSIONAL_FORM_ID	FULL
	FAMILY_ID	FULL
	SOCIAL_DATA_ID	FULL
	CODE	FULL
	PREDICTION	FULL
SIVARIA_INJURYFORM	INJURY1	RANDOM
	CODE	RANDOM
SIVARIA_ATEFORM	CODE	RANDOM
SIVARIA_ATIFORM	CODE	RANDOM
SIVARIA_EBIPQ_ECIPQFORM	CODE	RANDOM
SIVARIA_EDFORM	CODE	RANDOM
SIVARIA_ERFORM	CODE	RANDOM
SIVARIA_INQFORM	CODE	RANDOM
SIVARIA_MULTICAGECAD4FORM	CODE	RANDOM

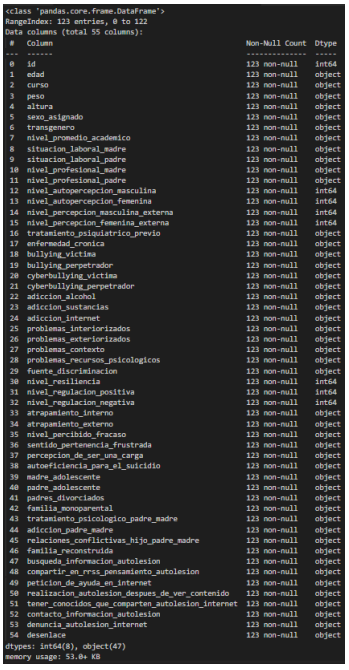
Continuación de la Tabla B.1

Tabla	Columna	Política DBMS_REDACT aplicada
SIVARIA_PARQFORM	CODE	RANDOM
SIVARIA_RRSSFORM	CODE	RANDOM
SIVARIA_SENAFAMILYFORM	CODE	RANDOM
SIVARIA_SENAFORM	CODE	RANDOM
SIVARIA_SOCIALDATAFORM	COURSE	RANDOM
	AGE	FULL
	GENDER	RANDOM
	TRANS	RANDOM
	JOB_SITUATION_FATHER	RANDOM
	JOB_SITUATION_MOTHER	RANDOM
	ACADEMIC_LEVEL_FATHER	RANDOM
	ACADEMIC_LEVEL_MOTHER	RANDOM
	ACADEMIC_PERFORMANCE	RANDOM
	PREVIOUS_PSYCHIATRIC_TREATMENT	RANDOM
	CHRONIC_DISEASE	RANDOM
	WEIGHT	FULL
	HEIGHT	FULL
	DISCRIMINATION_TYPE	RANDOM
	CODE	RANDOM
SIVARIA_FAMILYSUBFORM	ADICCION_PADRE_MADRE	RANDOM
	MADRE_ADOLESCENTE	FULL
	PADRE_ADOLESCENTE	FULL
	MALTRATO_AL_ADOLESCENTE	RANDOM
	MALTRATO_A_LA_PAREJA	RANDOM
	PADRES_DIVORCIADOS	RANDOM
	RELACIONES_CONFLICTIVAS_HIJO_PADRE_MADRE	RANDOM
	SITUACION_ECONOMICA_PRECARIA	RANDOM
	SUPERVISION_PARENTAL_INSUFICIENTE	RANDOM
	DUELO	RANDOM
	FAMILIA_MONOPARENTAL	RANDOM
	INGRESO_FAMILIAR_MENSUAL	FULL
	FAMILIA_RECONSTRUIDA	RANDOM
	CODE	RANDOM



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 55 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Index                                           40 non-null     int64
 1   Edad                                            40 non-null     int64
 2   Curso                                           40 non-null     object
 3   Peso                                            40 non-null     int64
 4   Altura                                          40 non-null     int64
 5   Sexo asignado                                   40 non-null     object
 6   Transgenero                                     40 non-null     object
 7   Nivel promedio del rendimiento academico        40 non-null     object
 8   Situacion laboral madre                         40 non-null     object
 9   Situacion laboral padre                         40 non-null     object
 10  Nivel profesional madre                         40 non-null     object
 11  Nivel profesional padre                         40 non-null     object
 12  Nivel de autopercepcion masculina               40 non-null     int64
 13  Nivel de autopercepcion femenina                40 non-null     int64
 14  Nivel de percepcion masculina externa           40 non-null     int64
 15  Nivel de percepcion femenina externa            40 non-null     int64
 16  Tratamiento psiquiatrico previo                 40 non-null     object
 17  Presenta enfermedad cronica                     40 non-null     object
 18  Bullying victima                                40 non-null     object
 19  Bullying perpetrador                            40 non-null     object
 20  Cyberbullying victima                           40 non-null     object
 21  Cyberbullying perpetrador                       40 non-null     object
 22  Adiccion/abuso alcohol                          40 non-null     object
 23  Adiccion/abuso sustancias                       40 non-null     object
 24  Adiccion/abuso Internet                         40 non-null     object
 25  Problemas interiorizados                        40 non-null     object
 26  Problemas exteriorizados                        40 non-null     object
 27  Problemas de contexto                           40 non-null     object
 28  Problemas recursos psicologicos                 40 non-null     object
 29  Fuente de discriminacion                        40 non-null     object
 30  Nivel de resistencia/resiliencia                40 non-null     int64
 31  Nivel de regulacion positiva                    40 non-null     int64
 32  Nivel de regulacion negativa                    40 non-null     int64
 33  Atrapamiento interno                            40 non-null     object
 34  Atrapamiento externo                            40 non-null     object
 35  Nivel percibido de derrota o fracaso            40 non-null     object
 36  Sentido de pertenencia frustada                 40 non-null     object
 37  Percepcion de ser una carga                     40 non-null     object
 38  Autoeficiencia para el suicidio                 40 non-null     object
 39  Madre adolescente                               40 non-null     object
 40  Padre adolescente                               40 non-null     object
 41  Padres divorciados                              40 non-null     object
 42  Familia monoparental                            40 non-null     object
 43  Tratamiento psicologico padre/madre             40 non-null     object
 44  Adiccion padre/madre                            40 non-null     object
 45  Relaciones conflictivas hijo-padre/madre        40 non-null     object
 46  Familia reconstruida                            40 non-null     object
 47  Busqueda informacion autolesion                 40 non-null     object
 48  Compartir en RRSS pensamiento autolesion        40 non-null     object
 49  Peticion ayuda en Internet                      40 non-null     object
 50  Realizacion autolesion despues de ver contenido 40 non-null     object
 51  Conocidos que comparten autolesion en Internet  40 non-null     object
 52  Contacto informacion autolesion                 40 non-null     object
 53  Denuncia autolesion Internet                    40 non-null     object
 54  Desenlace                                       40 non-null     object
dtypes: int64(11), object(44)
memory usage: 17.3+ KB
```

Figura B.1: Estructura del dataframe en la segunda versión de entrenamiento



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 55 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   id                                              123 non-null    int64
 1   edad                                            123 non-null    object
 2   curso                                           123 non-null    object
 3   peso                                            123 non-null    object
 4   altura                                          123 non-null    object
 5   sexo_asignado                                   123 non-null    object
 6   transgenero                                     123 non-null    object
 7   nivel_promedio_academico                        123 non-null    object
 8   situacion_laboral_madre                         123 non-null    object
 9   situacion_laboral_padre                         123 non-null    object
 10  nivel_profesional_madre                         123 non-null    object
 11  nivel_profesional_padre                         123 non-null    object
 12  nivel_autopercepcion_masculina                  123 non-null    int64
 13  nivel_autopercepcion_femenina                   123 non-null    int64
 14  nivel_percepcion_masculina_externa              123 non-null    int64
 15  nivel_percepcion_femenina_externa               123 non-null    int64
 16  tratamiento_psiquiatrico_previo                 123 non-null    object
 17  enfermedad_cronica                              123 non-null    object
 18  bullying_victima                                123 non-null    object
 19  bullying_perpetrador                            123 non-null    object
 20  cyberbullying_victima                           123 non-null    object
 21  cyberbullying_perpetrador                       123 non-null    object
 22  adiccion_alcohol                                123 non-null    object
 23  adiccion_sustancias                             123 non-null    object
 24  adiccion_internet                               123 non-null    object
 25  problemas_interiorizados                        123 non-null    object
 26  problemas_exteriorizados                        123 non-null    object
 27  problemas_contexto                              123 non-null    object
 28  problemas_recursos_psicologicos                 123 non-null    object
 29  fuente_discriminacion                           123 non-null    object
 30  nivel_resiliencia                               123 non-null    int64
 31  nivel_regulacion_positiva                       123 non-null    int64
 32  nivel_regulacion_negativa                       123 non-null    int64
 33  atrapamiento_interno                            123 non-null    object
 34  atrapamiento_externo                            123 non-null    object
 35  nivel_percibido_fracaso                         123 non-null    object
 36  sentido_pertenencia_frustrada                   123 non-null    object
 37  percepcion_de_ser_una_carga                     123 non-null    object
 38  autoeficiencia_para_el_suicidio                 123 non-null    object
 39  madre_adolescente                               123 non-null    object
 40  padre_adolescente                               123 non-null    object
 41  padres_divorciados                              123 non-null    object
 42  familia_monoparental                            123 non-null    object
 43  tratamiento_psicologico_padre_madre             123 non-null    object
 44  adiccion_padre_madre                            123 non-null    object
 45  relaciones_conflictivas_hijo_padre_madre        123 non-null    object
 46  familia_reconstruida                            123 non-null    object
 47  busqueda_informacion_autolesion                 123 non-null    object
 48  compartir_en_rrss_pensamiento_autolesion        123 non-null    object
 49  peticion_de_ayuda_en_internet                   123 non-null    object
 50  realizacion_autolesion_despues_de_ver_contenido 123 non-null    object
 51  tener_conocidos_que_comparten_autolesion_internet 123 non-null  object
 52  contacto_informacion_autolesion                 123 non-null    object
 53  denuncia_autolesion_internet                    123 non-null    object
 54  desenlace                                       123 non-null    object
dtypes: int64(8), object(47)
memory usage: 53.0+ KB
```

Figura B.2: Estructura del dataframe del modelo del autoinforme en la tercera versión

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107 entries, 0 to 106
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   id                                     107 non-null    int64
 1   curso                                  107 non-null    object
 2   situacion_laboral_madre                107 non-null    object
 3   situacion_laboral_padre                107 non-null    object
 4   nivel_profesional_madre                107 non-null    object
 5   nivel_profesional_padre                107 non-null    object
 6   problemas_interiorizados               107 non-null    object
 7   problemas_exteriorizados               107 non-null    object
 8   problemas_contexto                     107 non-null    object
 9   problemas_recursos_psicologicos        107 non-null    object
 10  madre_adolescente                      107 non-null    object
 11  padre_adolescente                      107 non-null    object
 12  padres_divorciados                     107 non-null    object
 13  familia_monoparental                   107 non-null    object
 14  tratamiento_psicologico_padre_madre    107 non-null    object
 15  adiccion_padre_madre                   107 non-null    object
 16  relaciones_conflictivas_hijo_padre_madre  107 non-null  object
 17  familia_reconstruida                   107 non-null    object
 18  aceptacion_rechazo_parental            107 non-null    object
 19  control_parental                       107 non-null    object
 20  situacion_economica_precaria           107 non-null    object
 21  ingreso_familiar_mensual               107 non-null    object
 22  desenlace                              107 non-null    object
dtypes: int64(1), object(22)
memory usage: 19.4+ KB
```

Figura B.3: Estructura del dataframe del modelo de las familias en la tercera versión

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220 entries, 0 to 219
Data columns (total 22 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   id                                     220 non-null    int64
 1   curso                                  220 non-null    object
 2   situacion_laboral_madre                220 non-null    object
 3   situacion_laboral_padre                220 non-null    object
 4   nivel_profesional_madre                220 non-null    object
 5   nivel_profesional_padre                220 non-null    object
 6   situacion_economica_precaria           220 non-null    object
 7   ingreso_familiar_mensual               220 non-null    object
 8   tratamiento_psiquiatrico_previo        220 non-null    object
 9   madre_adolescente                      220 non-null    object
 10  padre_adolescente                      220 non-null    object
 11  padres_divorciados                     220 non-null    object
 12  familia_monoparental                   220 non-null    object
 13  tratamiento_psicologico_padre_madre    220 non-null    object
 14  adiccion_padre_madre                   220 non-null    object
 15  relaciones_conflictivas_hijo_padre_madre  220 non-null  object
 16  familia_reconstruida                   220 non-null    object
 17  supervision_parental_insuficiente      220 non-null    object
 18  maltrato_al_adolescente                220 non-null    object
 19  maltrato_a_la_pareja                   220 non-null    object
 20  duelo                                  220 non-null    object
 21  desenlace                              220 non-null    object
dtypes: int64(1), object(21)
memory usage: 37.9+ KB
```

Figura B.4: Estructura del dataframe del modelo de los profesionales en la tercera versión

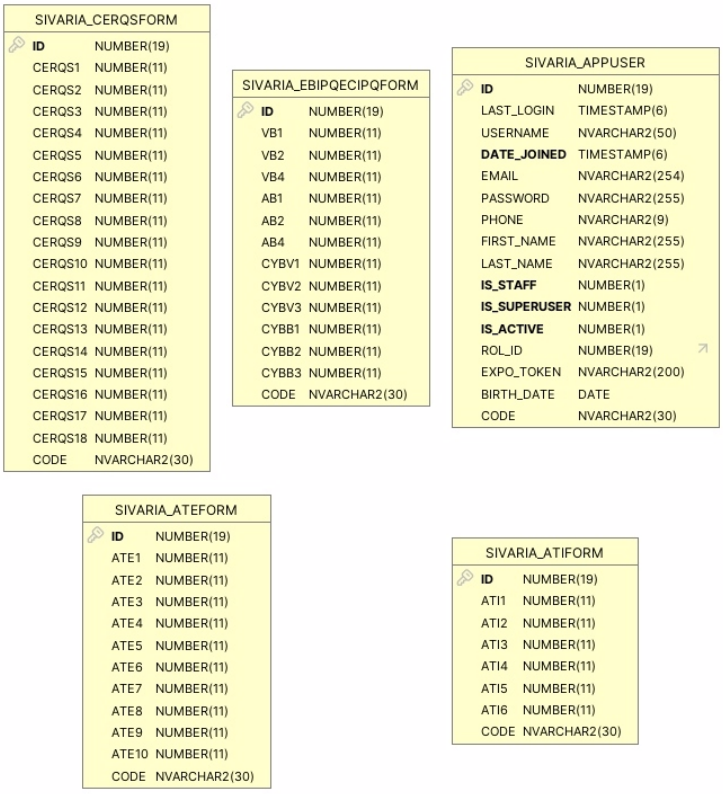


Figura B.5: Tablas de la base de datos SIVARIA (I)

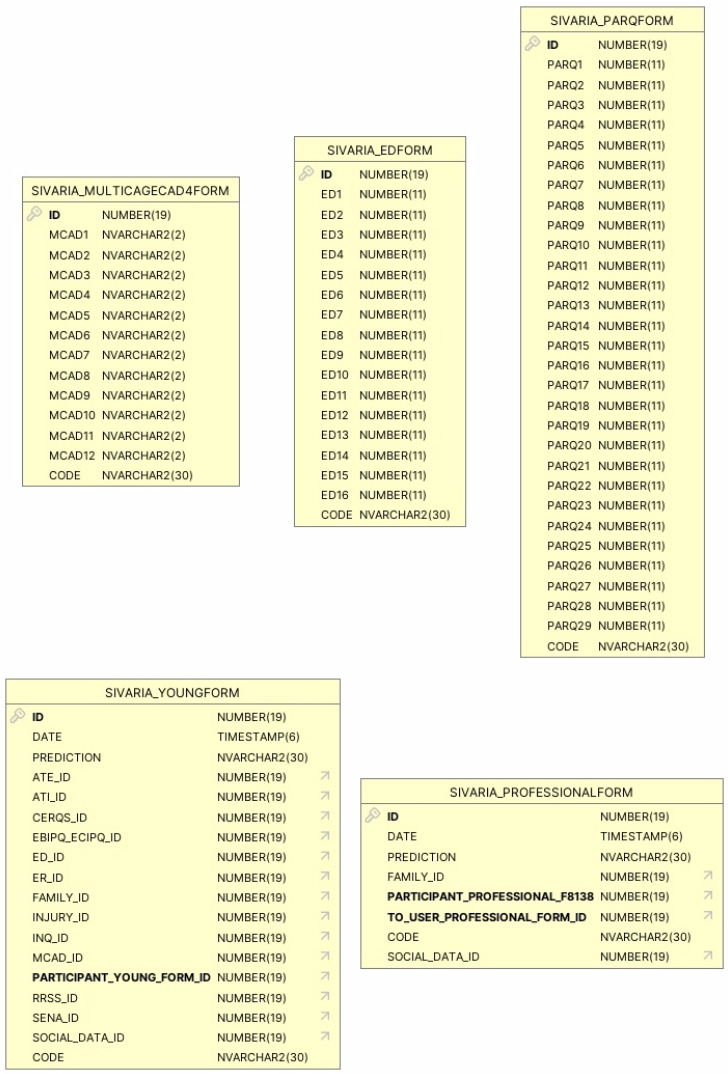


Figura B.6: Tablas de la base de datos SIVARIA (II)

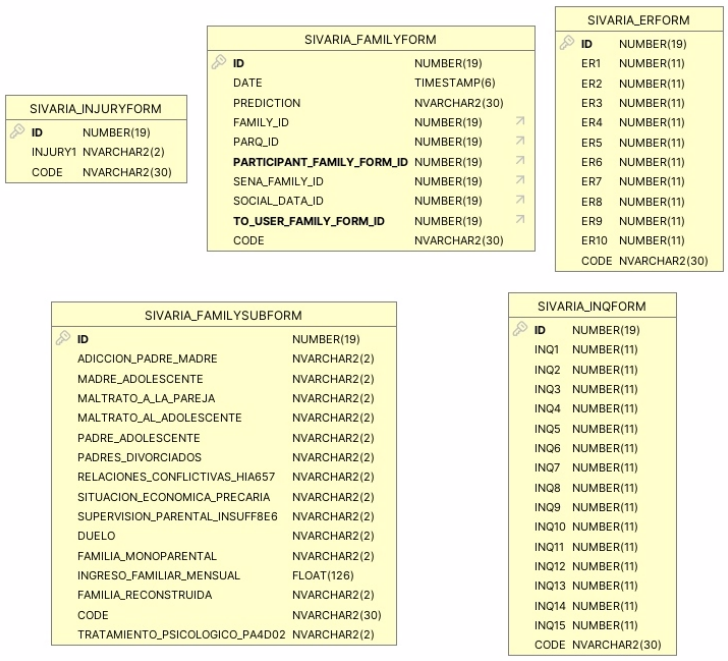


Figura B.7: Tablas de la base de datos SIVARIA (III)

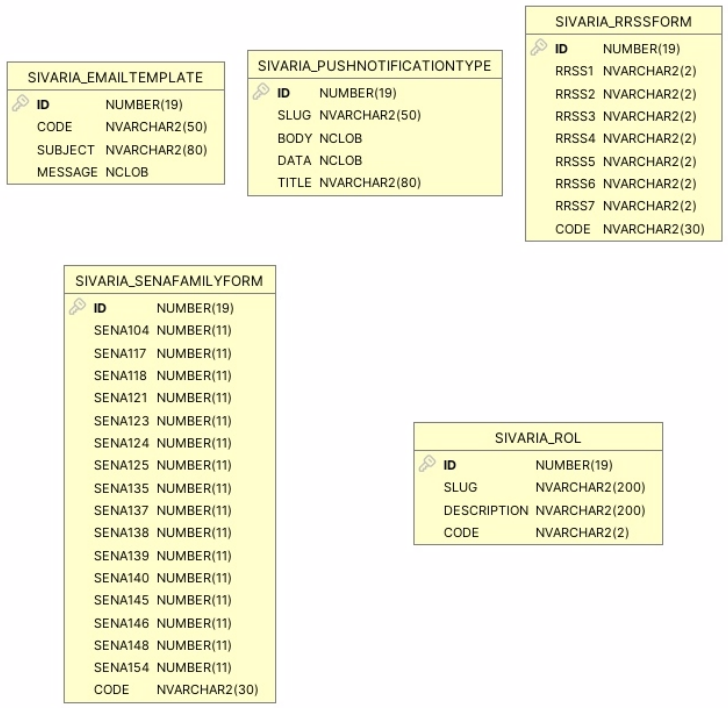


Figura B.8: Tablas de la base de datos SIVARIA (IV)

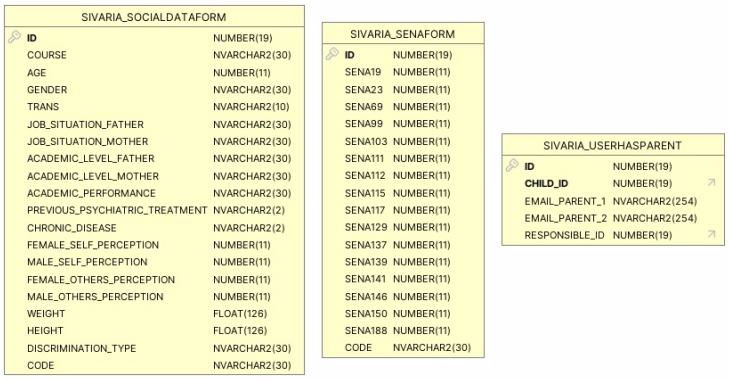


Figura B.9: Tablas de la base de datos SIVARIA (V)

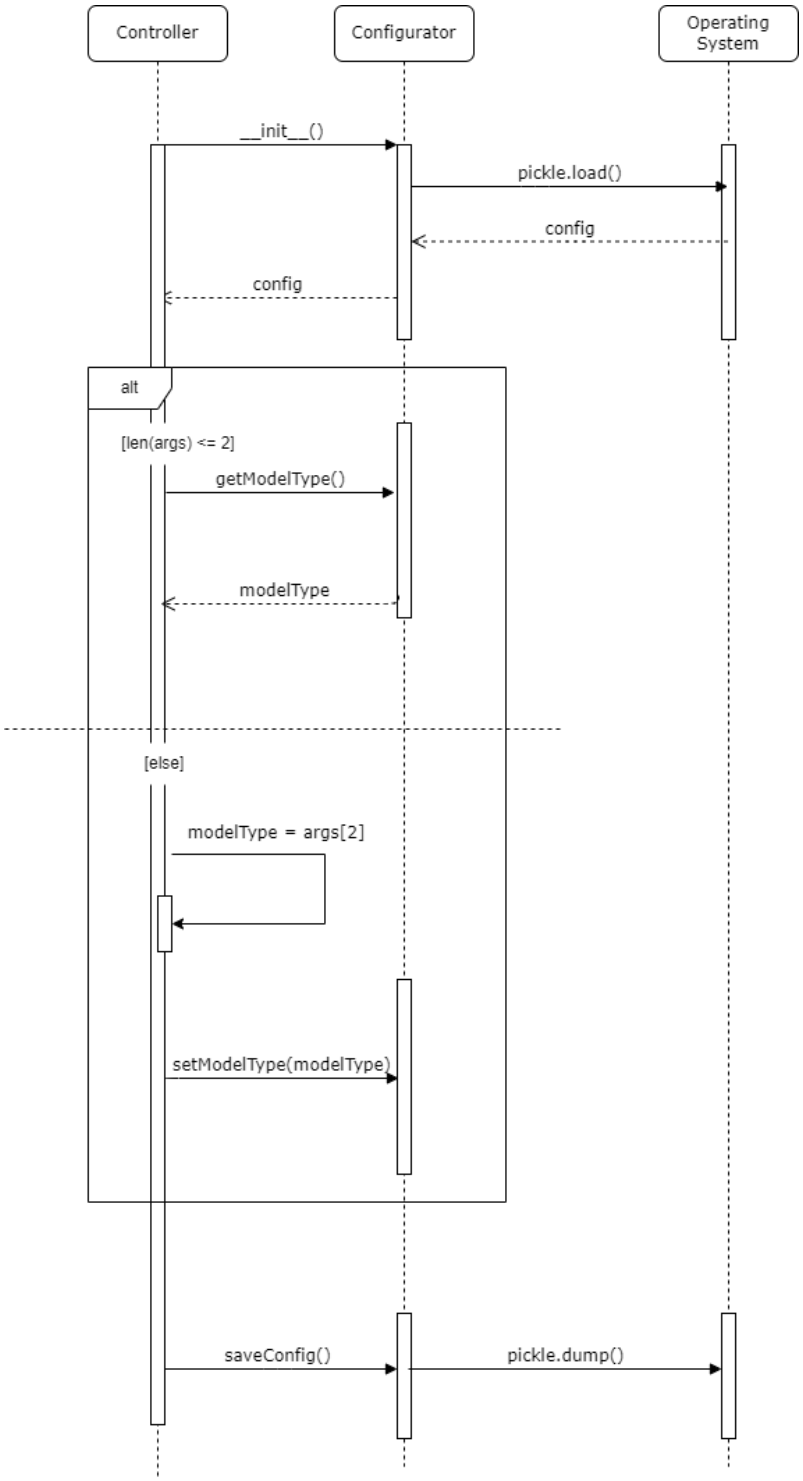


Figura B.10: Diagrama de secuencia del comando para establecer el tipo de modelo

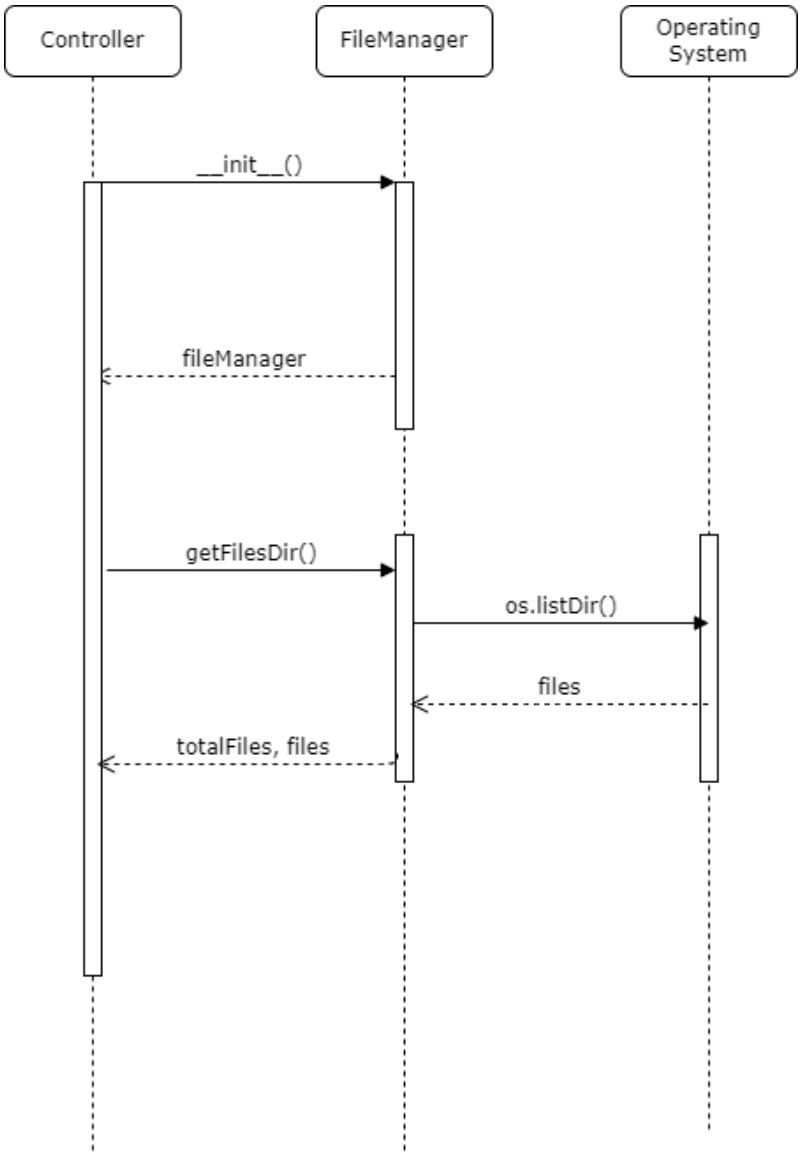


Figura B.11: Diagrama de secuencia del comando para listar los archivos de guardado

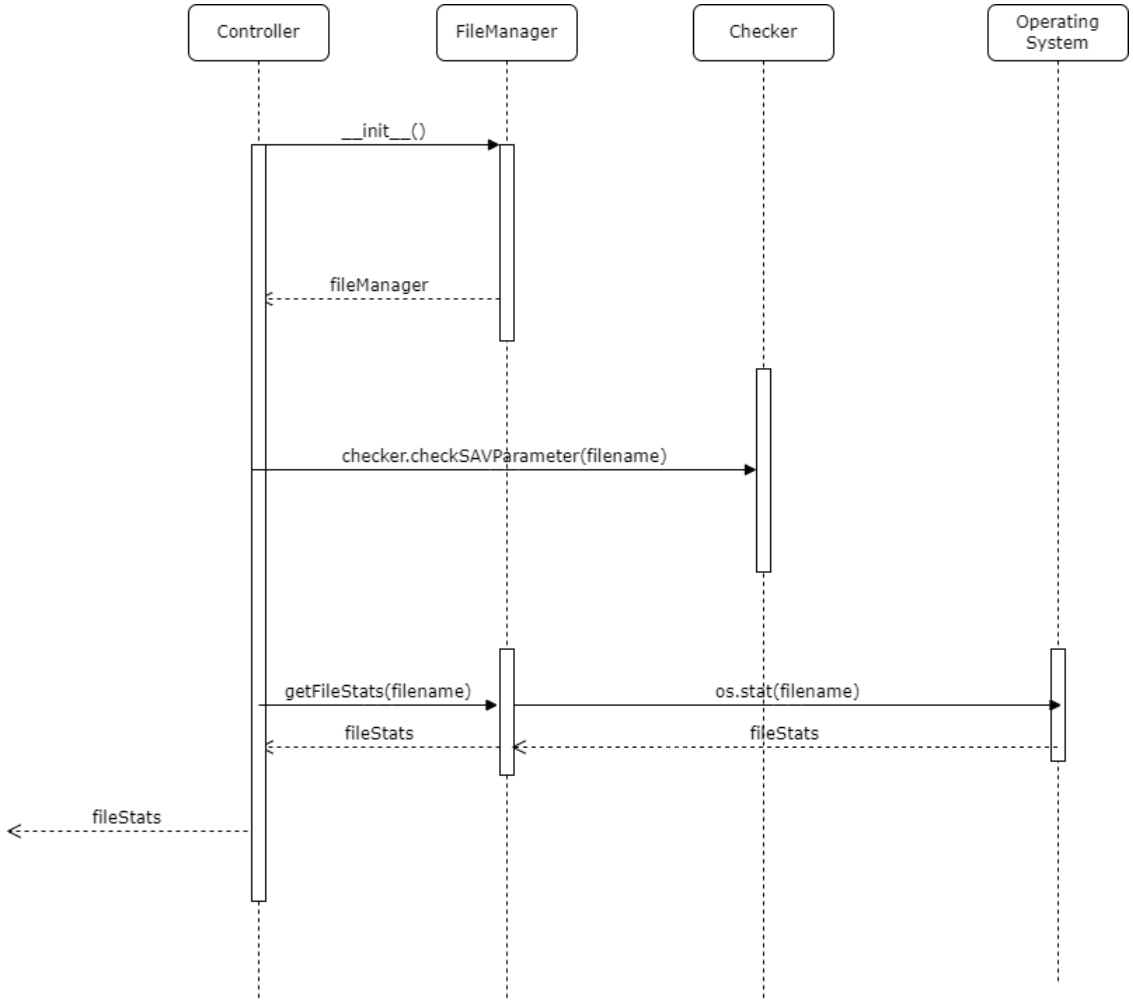


Figura B.12: Diagrama de secuencia del comando para mostrar la información del archivo de guardado

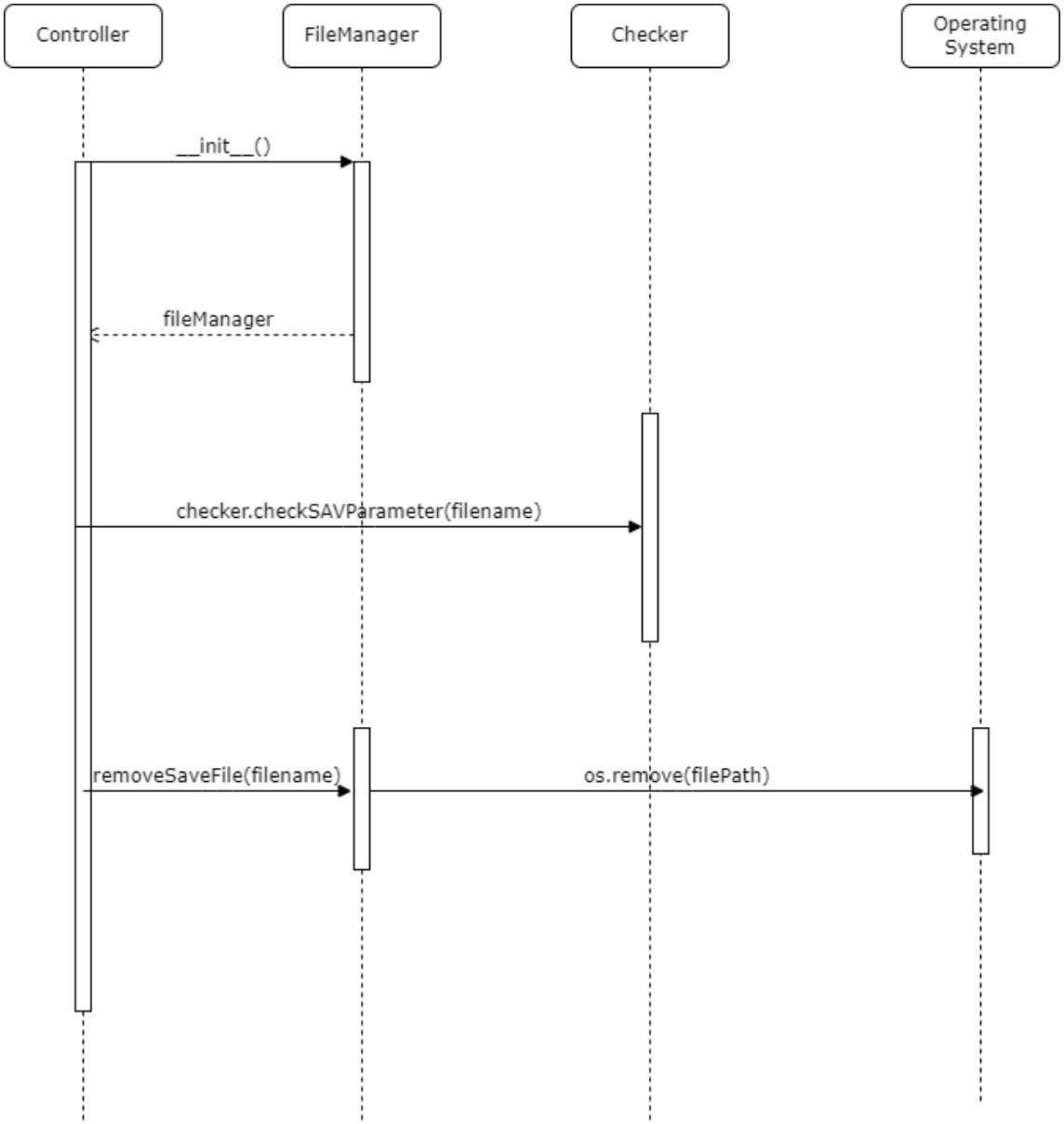


Figura B.13: Diagrama de secuencia del comando para eliminar un archivo de guardado

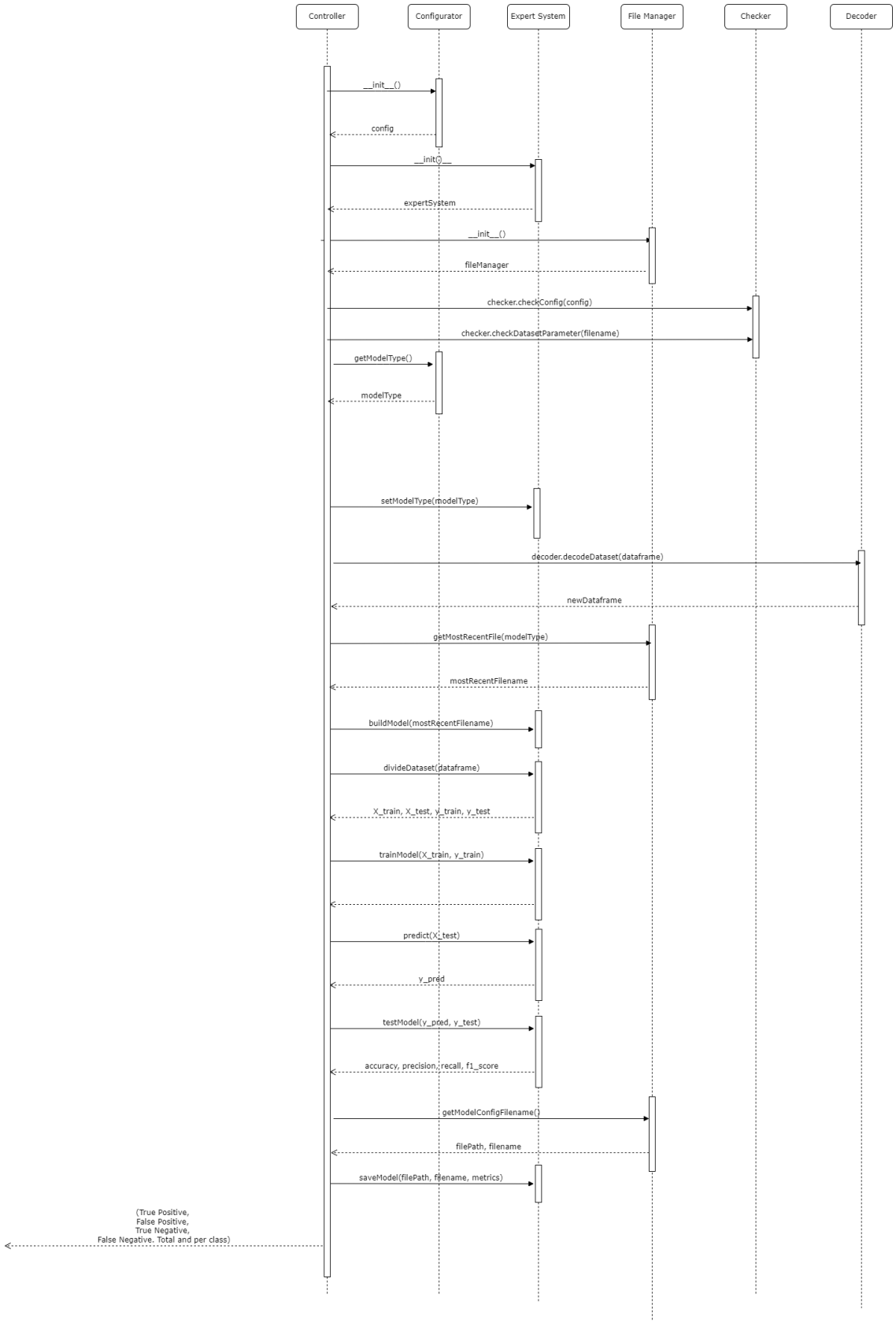


Figura B.14: Diagrama de secuencia del comando para entrenar al modelo

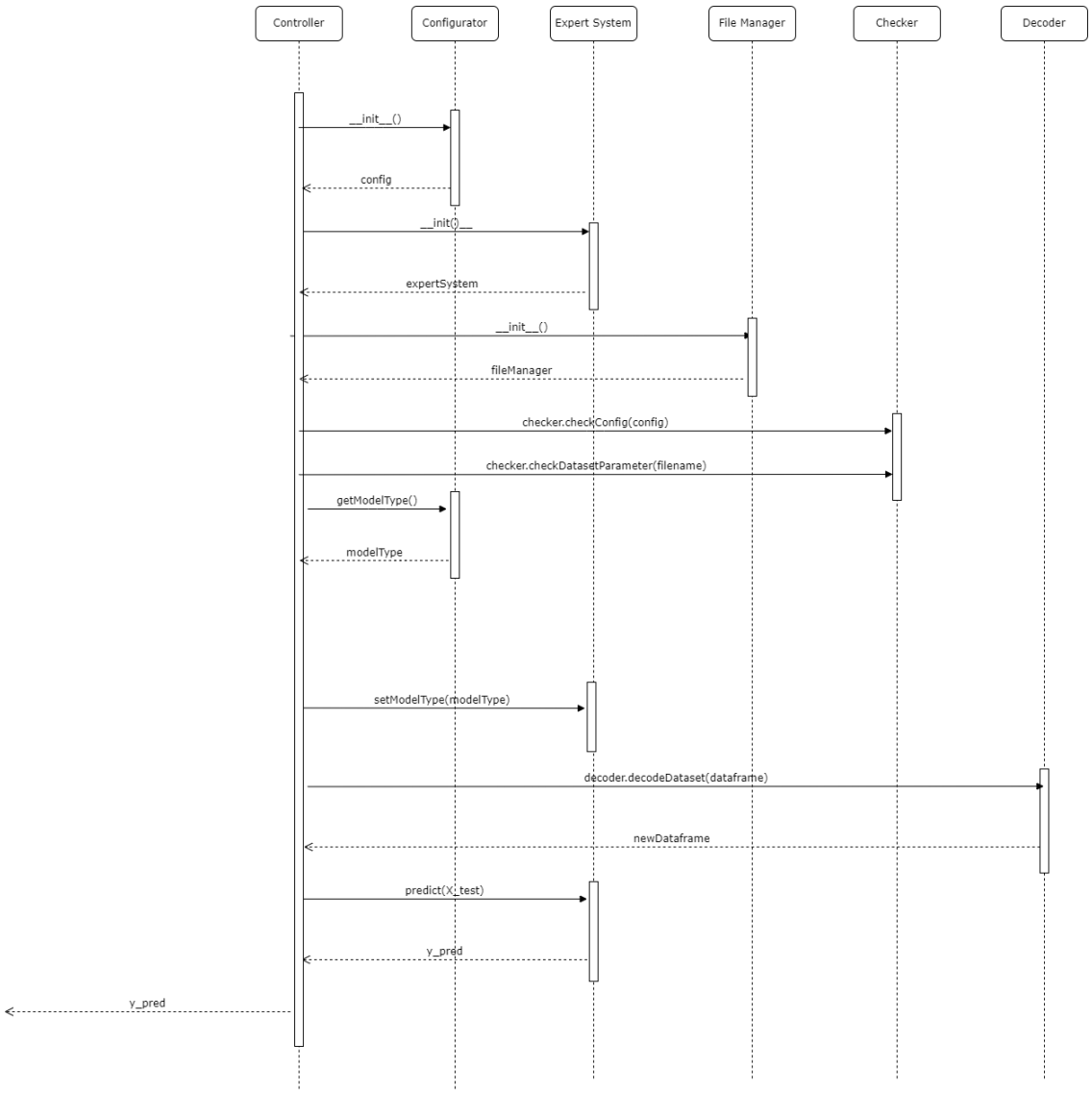


Figura B.15: Diagrama de secuencia del comando para realizar una predicción