Universidad de San Carlos de Guatemala
Fab Lab - Dirección General de Investigación
Asesor Institución: MSc. Ing. Julio César Alvarez Guillén

MANUAL TÉCNICO

Análisis Inteligente de Resultados Psicológicos mediante Machine Learning

Autor	Rol	Correo	Carné
Moises David Maldonado de León	*Desarrollador principal*	*moises.david.maldonado337@gmail.com*	*202010833*

Guatemala, 30 de octubre de 2025

Versión 1.0

Resumen

La salud mental en el entorno universitario constituye un factor determinante para el rendimiento académico y el bienestar general del estudiantado. En la Universidad de San Carlos de Guatemala, se ha recopilado una gran cantidad de información psicológica mediante instrumentos estandarizados como *PHQ, CD-RISC y MHC-SF*, los cuales evalúan síntomas emocionales, resiliencia y bienestar mental. Sin embargo, estos datos permanecían dispersos y sin un análisis sistemático que permitiera transformarlos en conocimiento útil para la toma de decisiones institucionales.

El proyecto propone una solución basada en técnicas de *Machine Learning* para el análisis inteligente de dichos resultados psicológicos. A través de un flujo estructurado de procesamiento de datos (limpieza, exploración, modelado estadístico y entrenamiento de algoritmos) se busca extraer patrones relevantes y generar modelos predictivos capaces de identificar perfiles psicológicos y detectar factores de riesgo emocional de forma temprana.

La implementación se desarrolló en el entorno Python utilizando librerías especializadas como *Pandas*, *NumPy*, *Matplotlib* y *Scikit-learn*, dentro de Jupyter Notebook. El proceso incluyó la preparación de un conjunto de 7,819 registros y la aplicación de modelos supervisados y no supervisados para la clasificación del bienestar mental (*MHC-SF*) en tres categorías: *Languishing*, *Moderado* y *Flourishing*. Además, se documentó la estructura del proyecto, incluyendo carpetas de datos, modelos, notebooks, código fuente y la aplicación local.

Los resultados obtenidos permiten no solo comprender las tendencias generales del bienestar psicológico en la población estudiantil, sino también sentar las bases para desarrollar herramientas de apoyo a la gestión institucional y a la promoción del bienestar emocional en el ámbito universitario.

Introducción

La salud mental se ha convertido en un tema de creciente interés dentro del ámbito universitario, donde factores académicos, personales y sociales influyen directamente en el bienestar emocional de los estudiantes. En instituciones como la Universidad de San Carlos de Guatemala, la atención psicológica es un tema de vital importancia que permite no solo conocer el estado de los estudiantes, si no también explorar y analizar tendencias y patrones que ayuden a tomar acciones y obtener una visión global del estado emocional de la comunidad estudiantil. Esta situación ha generado la necesidad de incorporar herramientas tecnológicas que permitan analizar grandes volúmenes de información psicológica y extraer de ellos conocimiento útil para la prevención y el acompañamiento institucional.

El presente proyecto técnico propone el desarrollo de un sistema de análisis inteligente de datos psicológicos mediante la aplicación de técnicas de Machine Learning (ML). A partir de instrumentos estandarizados tales como los cuestionarios PHQ**,** CD-RISC **y** MHC-SF. Las respuestas de miles de estudiante se procesan y analizan con el objetivo de predecir el nivel de bienestar mental **y** detectar patrones asociados a factores de riesgo o resiliencia.

El enfoque metodológico del proyecto contempla un flujo integral de procesamiento de datos que abarca desde la limpieza y transformación del conjunto de datos, hasta la implementación, evaluación y comparación de modelos supervisados y no supervisados. Se emplearon herramientas del ecosistema Python, como *Pandas*, *NumPy*, *Matplotlib* y *Scikit-learn*, desarrolladas en el entorno Jupyter Notebook. Asimismo, se diseñó una estructura modular del proyecto que facilita la trazabilidad del código, la reutilización de modelos y la implementación de una aplicación local para la predicción del bienestar mental.

Desde una perspectiva institucional, este proyecto no solo busca demostrar la aplicabilidad de los algoritmos de aprendizaje automático en el ámbito de la psicología, sino también aportar una base técnica que permita el desarrollo de sistemas predictivos de salud mental orientados a la detección temprana, la prevención y la toma de decisiones estratégicas dentro de la universidad o para la entidad encargada que desee hacer uso de la misma.

OBJETIVOS

Generales

Analizar y modelar los resultados de instrumentos psicológicos aplicados a estudiantes universitarios mediante técnicas de Machine Learning, con el propósito de identificar factores de riesgo y bienestar psicológico que sirvan de apoyo para el diseño de estrategias institucionales de intervención y acompañamiento en salud mental.

Específicos

1. Analizar la estructura y calidad del conjunto de datos recopilado, identificando los tipos de variables, la presencia de valores nulos, atípicos o inconsistentes, y la distribución general de los datos, en un periodo no mayor a dos semanas desde el inicio del proyecto.

2. Aplicar técnicas de limpieza y preprocesamiento de datos, incluyendo imputación de valores faltantes, transformación de variables categóricas, escalamiento de datos numéricos y detección de outliers, utilizando herramientas del ecosistema *Python* como *Pandas*, *NumPy* y *Scikit-learn*, dentro del entorno *Jupyter Notebook*.

3. Realizar un análisis exploratorio de datos (EDA) con el fin de identificar patrones, correlaciones y relaciones significativas entre los distintos instrumentos psicológicos (PHQ, CD-RISC, MHC) y las variables sociodemográficas relevantes del estudio.

4. Seleccionar y justificar los algoritmos de aprendizaje automático (supervisados y no supervisados) más adecuados para la estructura y propósito del conjunto de datos, contemplando técnicas de clasificación, regresión, agrupamiento y, cuando sea pertinente, el uso de redes neuronales.

5. Entrenar, evaluar y validar los modelos de *Machine Learning* mediante una partición estratégica del conjunto de datos, empleando métricas como precisión, sensibilidad, F1-score y matrices de confusión para determinar su rendimiento predictivo y utilidad práctica en el análisis automatizado de resultados psicológicos.

6. Implementar un flujo completo de procesamiento de datos y modelado, documentando cada etapa del proceso para garantizar la trazabilidad, replicabilidad y transparencia de los resultados.

7. Proveer una base analítica sólida que respalde la toma de decisiones institucionales en materia de bienestar psicológico estudiantil, a través del uso responsable y ético de los modelos generados.

8. Desplegar una versión de la aplicación en el servidor de la institución Fab Lab (NAS Synology DS223) para disponibilidad tanto para consumo inmediato como uso en aplicación local.

Alcances

El presente proyecto abarca el desarrollo e implementación de un sistema de análisis basado en Machine Learning orientado al estudio del bienestar mental en estudiantes universitarios. El alcance incluye la integración, procesamiento y modelado de información proveniente de los instrumentos psicológicos PHQ, CD-RISC y MHC-SF, junto con variables demográficas complementarias.

El sistema está diseñado para analizar y clasificar los niveles de bienestar mental de acuerdo con tres categorías principales: Languishing, Moderado y Flourishing. Además, permite identificar patrones y correlaciones relevantes entre variables psicológicas y sociodemográficas, proporcionando una base empírica para el diseño de estrategias institucionales de prevención y acompañamiento.

Desde el punto de vista técnico, el proyecto contempla la implementación completa del flujo de procesamiento de datos, desde la limpieza y transformación hasta la evaluación de modelos predictivos. Los resultados se integran en una aplicación local y desplegable, la cual se ejecuta en un entorno NAS Synology DS223 alojado en el laboratorio institucional Fab Lab, permitiendo el consumo inmediato de los resultados y su uso en entornos locales o académicos si se desea descargar los archivos de instalación.

Los principales usuarios del sistema son profesionales de psicología, investigadores universitarios y personal técnico de apoyo institucional, quienes pueden interpretar los resultados o reutilizar los modelos entrenados para nuevos análisis. Incluso para los estudiantes mismos que quieran un análisis temprano y posibles resultados sobre su estado (siempre y cuando se verifique la información con un profesional capacitado y certificado).

Limites

El alcance del proyecto presenta algunas restricciones técnicas y metodológicas derivadas de la naturaleza de los datos y del entorno de ejecución:

- El análisis se basa exclusivamente en los datos recopilados por la Universidad de San Carlos de Guatemala, sin incluir información externa o longitudinal.
- El tamaño del conjunto de datos (7,819 registros) puede influir en la generalización de los modelos, especialmente en clases con menor representación.
- Las predicciones generadas reflejan patrones estadísticos, no diagnósticos clínicos definitivos, y deben interpretarse como herramientas de apoyo, no como evaluaciones psicológicas definitivas.
- El rendimiento del sistema está condicionado por los recursos de hardware del servidor *NAS Synology DS223*, los cuales limitan el entrenamiento de modelos de alta complejidad como redes neuronales profundas o modelos que requieran mayor potencia de hardware.
- El acceso a los datos originales se restringe por motivos de confidencialidad y privacidad, conforme a las políticas institucionales de manejo de información sensible.

Marco Teórico

1. Ciencia de Datos

1.1 Ciencia de Datos (Data Science)

La ciencia de datos es un campo interdisciplinario que combina métodos estadísticos, matemáticos y computacionales para extraer conocimiento e información útil a partir de grandes volúmenes de datos. Involucra etapas como la recopilación, limpieza, análisis, modelado y visualización de datos, con el fin de apoyar la toma de decisiones basada en evidencia.

1.2 Machine Learning (Aprendizaje Automático)

El *Machine Learning* es una rama de la inteligencia artificial que permite a los sistemas aprender automáticamente a partir de los datos, sin ser programados explícitamente para realizar tareas específicas. Su objetivo es identificar patrones y hacer predicciones o clasificaciones basadas en la experiencia previa contenida en los datos.

1.3 Tipos de Aprendizaje Automático

- **Aprendizaje supervisado:** se entrena el modelo con datos etiquetados (por ejemplo, clasificación o regresión).
- **Aprendizaje no supervisado:** el modelo busca estructuras o agrupamientos ocultos en datos sin etiquetas, como en el caso de *K-Means* o *PCA*.
- **Aprendizaje por refuerzo:** el modelo aprende mediante la interacción con un entorno, recibiendo recompensas o penalizaciones según sus acciones (usado en robótica o juegos).

2. Conjunto de Datos y Preprocesamiento

2.1 Dataset (Conjunto de datos)

Es el conjunto estructurado de información que contiene las variables relevantes para el análisis o entrenamiento del modelo. Su calidad, tamaño y representatividad influyen directamente en el desempeño del modelo de *Machine Learning*.

2.2 Preprocesamiento de datos

Etapa donde se preparan los datos para ser utilizados por los algoritmos, eliminando errores, inconsistencias o ruido. Incluye limpieza, codificación, escalamiento y balanceo.

2.3 Normalización / Escalamiento

La normalización ajusta las características numéricas a un rango específico (por ejemplo, [0,1] con *MinMaxScaler*), mientras que la estandarización (*StandardScaler*) transforma los datos para que tengan media 0 y desviación estándar 1. Estas técnicas evitan que variables con escalas diferentes dominen el aprendizaje.

2.4 Codificación de variables categóricas

Transforma variables no numéricas en valores que los modelos puedan procesar:

- *Label Encoding*: asigna un número entero a cada categoría.
- *One-Hot Encoding*: crea variables binarias para representar cada categoría.

2.5 Manejo de valores faltantes y *outliers*

Los valores faltantes se pueden eliminar o imputar mediante estrategias estadísticas (media, mediana, moda). Los *outliers* (valores atípicos) se detectan mediante métodos como desviación estándar o IQR, y se manejan según su impacto en el modelo.

2.6 SMOTE (Synthetic Minority Oversampling Technique)

Es una técnica de sobremuestreo que genera instancias sintéticas de la clase minoritaria para equilibrar conjuntos de datos desbalanceados. Mejora el rendimiento de clasificadores supervisados al evitar sesgos hacia la clase mayoritaria.

3. Análisis Exploratorio de Datos (EDA)

3.1 Definición e importancia

El *Exploratory Data Analysis* (EDA) permite comprender la estructura, relaciones y patrones de los datos antes del modelado. Ayuda a identificar correlaciones, anomalías y tendencias que orientan la selección de algoritmos y el preprocesamiento adecuado.

3.2 Técnicas de visualización y correlación

Incluyen histogramas, diagramas de dispersión, *boxplots* y mapas de calor (*heatmaps*) para visualizar distribuciones y relaciones entre variables. El coeficiente de correlación de Pearson o Spearman se usa para medir la asociación lineal entre variables numéricas.

3.3 Identificación de patrones o anomalías

El EDA facilita la detección de comportamientos inusuales o valores extremos, así como la identificación de variables predictoras relevantes.

4. Modelado y Algoritmos

4.1 Algoritmos supervisados

- **Gaussian Naive Bayes (GNB):** clasificador probabilístico basado en el teorema de Bayes, que asume independencia entre las variables predictoras. Es eficiente para datos con distribución normal.
- **K-Nearest Neighbors (KNN):** clasifica una muestra según las clases predominantes de sus *k* vecinos más cercanos en el espacio de características.
- **Random Forest:** conjunto de árboles de decisión entrenados sobre subconjuntos aleatorios del conjunto de datos; combina sus resultados para mejorar precisión y reducir sobreajuste.
- **Redes Neuronales Artificiales (MLP – Multilayer Perceptron):** modelo inspirado en el cerebro humano, compuesto por capas de neuronas artificiales conectadas entre sí; utiliza funciones de activación y aprendizaje mediante retropropagación.

4.2 Algoritmos no supervisados

- **K-Means:** algoritmo de agrupamiento (*clustering*) que divide el conjunto de datos en *k* grupos, minimizando la distancia intracluster.
- **PCA (Análisis de Componentes Principales):** técnica de reducción de dimensionalidad que transforma las variables originales en nuevas componentes lineales, reteniendo la mayor varianza posible.

4.3 Redes Neuronales y Keras

Keras es una biblioteca de alto nivel para construir y entrenar redes neuronales de forma eficiente sobre frameworks como TensorFlow. Permite definir capas, funciones de activación (ReLU, Sigmoid, Softmax) y optimizadores (Adam, SGD), facilitando el modelado de arquitecturas complejas.

5. Evaluación de Modelos

5.1 Métricas de desempeño

- **Accuracy:** proporción de predicciones correctas.
- **Precision y Recall:** precisión mide cuántas predicciones positivas fueron correctas; *recall* mide cuántos casos positivos fueron identificados correctamente.
- **F1-score:** media armónica entre *precision* y *recall*.
- **Matriz de confusión:** resume los aciertos y errores del modelo.
- **Validación cruzada:** divide el dataset en varios subconjuntos para evaluar el modelo de forma más robusta y evitar sobreajuste.

5.2 Sobreajuste y generalización

El sobreajuste ocurre cuando el modelo se ajusta excesivamente a los datos de entrenamiento, perdiendo capacidad de generalizar. Se mitiga mediante regularización, aumento de datos o técnicas como *dropout* en redes neuronales.

6. Despliegue de Modelos

6.1 Persistencia de modelos (Joblib, Pickle)

Una vez entrenado, el modelo se puede guardar (serializar) para su reutilización sin tener que volver a entrenarlo. *Joblib* es ideal para modelos de gran tamaño, mientras que *Pickle* ofrece mayor compatibilidad general en Python.

6.2 Aplicación práctica / Consumo del modelo

El modelo persistido puede integrarse en aplicaciones web o de escritorio, APIs o sistemas automatizados para realizar predicciones en tiempo real.

6.3 Entornos de ejecución (NAS, local, nube)

El despliegue puede realizarse en entornos locales, servidores NAS (como Synology) o plataformas en la nube (AWS, Google Cloud, Azure), según los requerimientos de disponibilidad, escalabilidad y seguridad. El entorno de interacción con el usuario final es mediante laa librería streamlit.

Metodología

Adquisición de datos

Los datos fueron obtenidos internamente mediante la aplicación de encuestas durante un periodo no especificado, recopilando información demográfica y resultados de instrumentos psicológicos estandarizados y validados: PHQ, CD-RISC, MHC y GAD.

Las encuestas fueron elaboradas mediante formularios de *Google Forms* y respondidas por estudiantes de la Universidad de San Carlos de Guatemala, pertenecientes a distintas carreras y centros asociados al Campus Central.

El proceso de levantamiento de datos tuvo como finalidad reunir indicadores de salud mental y bienestar psicológico para su posterior análisis mediante técnicas de ciencia de datos y aprendizaje automático.

Fuente de datos

El conjunto de datos fue proporcionado por la Escuela de Ciencias Psicológicas del Centro Universitario Metropolitano (CUM) a través del coordinador del programa de investigación de psicología en salud y clínica.

El archivo fue entregado en formato CSV, conteniendo los resultados anonimizados de encuestas aplicadas en línea. Los datos fueron importados al entorno de análisis mediante la librería Pandas en Python dentro de un entorno Jupyter Notebook.

Así mismo se estableció comunicación con el encargado institucional del Fab Lab, MSc. Ing. Julio César Alvarez Guillén, para la coordinación del almacenamiento y procesamiento en el servidor del laboratorio.

Formato y tamaño del dataset

El dataset original contiene 8,207 registros y 92 columnas, lo que refleja una estructura con una cantidad considerable de variables. Del total, 57 columnas son de tipo float64, correspondientes principalmente a puntajes numéricos derivados de los instrumentos psicológicos; 25 columnas son de tipo int64, utilizadas para variables categóricas codificadas numéricamente; y 10 columnas son de tipo object, correspondientes a respuestas abiertas o descripciones textuales. El uso total de memoria es de aproximadamente 5.8 MB, lo cual se considera razonable y permite trabajar de manera eficiente en un entorno local de análisis sin limitaciones de hardware.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8207 entries, 0 to 8206
Data columns (total 92 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   ResponseID                                              8207 non-null   int64
 1   ¿Hacomprendidolosaspectosgeneralesdelproyectoydeseaparticiparen  8207 non-null   int64
 2   Trabajo                                                 8206 non-null   float64
 3   Sexo                                                    8206 non-null   float64
 4   Religion                                                8206 non-null   float64
 5   OtraEscríbalaReligion                                   55 non-null     object
 6   Etnia                                                   8206 non-null   float64
 7   OtroEscríbalaEtnia                                      15 non-null     object
 8   EstCivil                                                8206 non-null   float64
 9   edad                                                    8206 non-null   float64
 10  Terapia                                                 8206 non-null   float64
 11  TrataPsi                                                8206 non-null   float64
 12  ¿Tienehijos                                             8000 non-null   float64
 13  HijoEscribalacantidadennúmerosNúmerodehijosehijasquedependenec  588 non-null    float64
 14  HijaEscribalacantidadennúmerosNúmerodehijosehijasquedependenec  586 non-null    float64
 15  UnAca                                                   8207 non-null   int64
 16  OtraEscríbalaUnAca                                      234 non-null    object
 17  CEntroU                                                 8207 non-null   int64
 18  CEntroU1                                                0 non-null      float64
 19  OtroEscríbalo¿Enquécentrouniversitarioestáinscrito      7 non-null      object
 20  Grado                                                   8207 non-null   int64
 21  Semestre                                                7942 non-null   float64
 22  Jornada                                                 8206 non-null   float64
 23  Carrera                                                 7942 non-null   object
 24  Maestríaenlaquerealizasuresidencia                      36 non-null     float64
 25  Gradoderesidenciaquecursa                               34 non-null     float64
```

Imagen 1. Primera parte de variables originales

Elaboración propia

```
 26  @01PHQ                                                  8090 non-null   float64
 27  @02PHQ                                                  8088 non-null   float64
 28  @03PHQ                                                  8087 non-null   float64
 29  @04PHQ                                                  8087 non-null   float64
 30  @05PHQ                                                  8085 non-null   float64
 31  @06PHQ                                                  8082 non-null   float64
 32  @07PHQ                                                  8083 non-null   float64
 33  @08PHQ                                                  8083 non-null   float64
 34  @09PHQ                                                  8083 non-null   float64
 35  @01CDrisc                                               8207 non-null   int64
 36  @02CDrisc                                               8207 non-null   int64
 37  @03CDrisc                                               8207 non-null   int64
 38  @04CDrisc                                               8207 non-null   int64
 39  @05CDrisc                                               8207 non-null   int64
 40  @06CDrisc                                               8207 non-null   int64
 41  @07CDrisc                                               8207 non-null   int64
 42  @08CDrisc                                               8206 non-null   float64
 43  @09CDrisc                                               8204 non-null   float64
 44  @10CDrisc                                               8202 non-null   float64
 45  mhc1                                                    8206 non-null   float64
 46  mhc2                                                    8206 non-null   float64
 47  mhc3                                                    8206 non-null   float64
 48  mhc4                                                    8206 non-null   float64
 49  mhc5                                                    8206 non-null   float64
 50  mhc6                                                    8206 non-null   float64
 51  mhc7                                                    8206 non-null   float64
 52  mhc8                                                    8206 non-null   float64
 53  mhc9                                                    8206 non-null   float64
 54  mhc10                                                   8206 non-null   float64
 55  mhc11                                                   8206 non-null   float64
 56  mhc12                                                   8206 non-null   float64
 57  mhc13                                                   8206 non-null   float64
 58  mhc14                                                   8206 non-null   float64
```

Imagen 2. Segunda parte de variables originales

Elaboración propia

```
59  @1GAD                                                  8206 non-null   float64
60  @2GAD                                                  8201 non-null   float64
61  @3GAD                                                  8205 non-null   float64
62  @4GAD                                                  8204 non-null   float64
63  @5GAD                                                  8205 non-null   float64
64  @6GAD                                                  8205 non-null   float64
65  @7GAD                                                  8204 non-null   float64
66  ConsumoSustancias                                      7350 non-null   float64
67  Tabaco¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltimos   786 non-null    object
68  Marihuana¿Quétipodesustanciaspsicoactivashaconsumidoenlosülti   261 non-null    object
69  Cocaína¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltim    53 non-null     object
70  Alcohol¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltimo   1721 non-null   object
71  Otras¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltimos1   163 non-null    object
72  SUMPHQ                                                 8081 non-null   float64
73  AGrupPHQ                                               8081 non-null   float64
74  SUMCDrisc                                              8207 non-null   int64
75  NivCDRisc                                              8207 non-null   int64
76  Total_MHC                                              8206 non-null   float64
77  SumaGAD                                                8207 non-null   int64
78  NivelesGAD                                             8207 non-null   int64
79  mhc_total                                              8206 non-null   float64
80  mhc_ewb                                                8206 non-null   float64
81  mhc_swb                                                8206 non-null   float64
82  mhc_pwb                                                8206 non-null   float64
83  hiaff                                                  8207 non-null   int64
84  loaff                                                  8207 non-null   int64
85  hifunc                                                 8207 non-null   int64
86  lofunc                                                 8207 non-null   int64
87  hiaffect                                               8207 non-null   int64
88  loaffect                                               8207 non-null   int64
89  hifunct                                                8207 non-null   int64
90  lofunct                                                8207 non-null   int64
91  mhc_dx                                                 8207 non-null   int64
dtypes: float64(57), int64(25), object(10)
```

Imagen 3. Tercera parte de variables originales

Elaboración propia

Preprocesamiento

Esta sección describe todas las operaciones realizadas para preparar el dataset antes de entrenar los modelos.

- **Limpieza de datos**

```
In [6]: #Busqueda de datos nulos
        df.isnull().sum()[df.isnull().sum() > 0]
        #Si el resultado es mayor a 0 es porque tenemos nulos

Out[6]: Trabajo                   1
        Sexo                      1
        Religion                  1
        OtraEscríbalaReligion    8152
        Etnia                     1
                                 ...
        Total_MHC                 1
        mhc_total                 1
        mhc_ewb                   1
        mhc_swb                   1
        mhc_pwb                   1
        Length: 67, dtype: int64
```

Imagen 4. Fragmento del Notebook para validar datos

Elaboración propia

Como se observa en la imagen 4, el dataset original presenta valores nulos y deben ser tratados de manera adecuada para mantener una integridad en la información. En el cuadernillo con el nombre "*1.Exploracion_datos.ipynb"* ubicado en la carpeta "*/notebooks*" se detalla el proceso de análisis para eliminar y conservar columnas según la relevancia de la información; no obstante, a continuación se muestran las variables descartadas reduciendo de 91 columnas a 77.

- ✓ **ResponseID:** Es un identificador para cada respuesta y por lo tanto se considera de información no relevante.
- ✓ **¿Hacomprendidolosaspectosgeneralesdelproyectoydeseaparticiparen:** es una pregunta sobre consentimiento, al observar el archivo se observa que solo 1 persona dijo que no; sin embargo, contestó las preguntas. Por tanto, la columna no aporta información valiosa para el análisis y se elimina.
- ✓ **OtraEsríbalaReligion:** Se eliminó por baja cobertura (solo 55 respuestas, <1%). Además, la columna Religión ya captura esta categoría con el valor 8 = "Otra". Los valores ingresados son variados, no estructurados y difíciles de procesar de forma útil.
- ✓ **OtroEscríbalaEtnia:** Esta variable fue descartada por tener solo 15 respuestas no nulas (≈0.18%). Los valores en su mayoría repetían opciones existentes o eran irrelevantes. La variable Etnia ya contempla un valor 5 = "Otro", lo que hace redundante esta columna.
- ✓ **HijoEscribalacantidad…/HijaEscribalacantidad…:** Las variables que pedían número de hijos o hijas (Hijo..., Hija...) fueron descartadas debido a inconsistencias entre respuestas (ej. "Sí" pero cantidad 0). Alta cantidad de valores nulos. Presencia de outliers como 99. Bajo número de registros útiles frente al total.
- ✓ **OtraEscríbalaUnAca:** En la mayoría de casos corresponde a carreras específicas o malinterpretaciones que ya están abarcadas por las categorías existentes en "UnAca". Bajo volumen y posible ruido: Solo 234 respuestas no nulas (menos del 3%), y entre estas, muchas no aportan información realmente diferenciadora.
- ✓ **CEntroU1:** Tiene 0 datos no nulos, es decir, completamente vacío. Su etiqueta confunde (dice "Unidad Académica", pero la variable se llama como si fuera centro universitario). Información ya cubierta por UnAca.
- ✓ **OtroEscríbalo¿Enquécentrouniversitarioestáinscrito**: Solo tiene 7 respuestas (< 0.1% del total). Las respuestas ya están incluidas en las categorías de CEntroU. No aporta diferenciación ni mejora la clasificación.
- ✓ **Maestríaenlaquerealizasuresidencia**: Tiene un número muy reducido de respuestas (solo 36 casos), lo cual representa una proporción mínima del total de registros (8207). Además, no se cuenta con una codificación clara de los valores, lo que dificulta su interpretación.

- ✓ **Gradoderesidenciaquecursa**: Similarmente, contiene únicamente 34 respuestas y no se dispone de una descripción o etiquetas que permitan comprender con claridad a qué se refieren los valores registrados.
- ✓ **Cocaína¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltim:** El número de respuestas es demasiado bajo (53 de 8207) para análisis significativo o entrenamiento de modelos. Además, su utilidad ya está indirectamente contenida en ConsumoSustancias.
- ✓ **Otras¿Quétipodesustanciaspsicoactivashaconsumidoenlosúltimos1:** No aporta valor. No especifica qué sustancia fue consumida ni se puede codificar de forma útil. No tiene profundidad.
- ✓ **Total_MHC:** Es redundante con mhc_total, es decir, es columna repetida.

- **Transformación de datos**

  En esta sección se realiza el tratamiento de los datos para garantizar su calidad y adecuación antes del análisis. Esto incluye la identificación y manejo de valores nulos, la transformación de variables categóricas a formatos adecuados como variables binarias, y la conversión o recodificación de datos según sea necesario para facilitar su interpretación y análisis posterior.

  Este proceso es fundamental para evitar sesgos, errores o pérdidas de información relevantes durante el modelado o interpretación estadística. En el cuadernillo con el nombre "*2.Limpieza_y_Transformacion.ipynb"* ubicado en la carpeta "*/notebooks*" se detalla este proceso. De forma técnica se describen algunos criterios utilizados.

  Antes de suprimir o imputar (reemplazar) valores que faltan, se pregunta lo siguiente:

  - ¿Cuáles fueron los mecanismos que causaron los valores que faltan?
  - ¿Estos valores que faltan, faltan al azar?
  - ¿Hay filas o columnas faltantes de las que no esté al tanto?

  Durante la limpieza de datos para análisis o modelado, también se debe prestar atención a los datos que no están presentes. Esto permite identificar posibles problemas en la recolección, sesgos en los resultados o desviaciones estadísticas relevantes.

En general, hay dos caminos para tratar los valores faltantes:

1. *Eliminar los registros con datos ausentes.*
2. *Imputar los valores con alguna estrategia (media, mediana, moda, interpolación, etc.).*

La elección depende del contexto. Eliminar filas con datos faltantes puede llevar a la pérdida significativa de información, lo cual afectaría la calidad del modelo. Por otro lado, imputar sin una justificación sólida puede introducir sesgos importantes, especialmente si se realiza de forma arbitraria. En casos donde una fila tiene múltiples valores ausentes, puede ser más razonable descartarla por completo, ya que requeriría hacer demasiadas suposiciones para su reconstrucción.

Observaciones extras

Para el caso de la imputación de datos, se utiliza la técnica de imputación estadística, esta implica reemplazar los valores faltantes con una medida de tendencia central como la media, la mediana o la moda. Cada una debe ser justificable para reducir las probabilidades de sesgar la información.

Técnica	Cuando usarla	Recomendación
Media	Cuando los datos están simétricamente distribuidos (sin sesgo).	Útil en datos continuos sin valores atípicos fuertes.
Mediana	Cuando los datos son sesgados o tienen outliers (valores extremos).	Preferible en distribuciones asimétricas, como ingresos, edad, etc.
Moda	Cuando los datos son categóricos (por ejemplo, género, país, etc.).	Se usa para datos no numéricos o con baja cardinalidad (pocos valores distintos).

Tabla 1. Técnicas de imputación de datos.

Elaboración propia.

Tipo de Datos Faltantes en el Dataset

En el análisis preliminar del dataset proporcionado (aproximadamente 8000 respuestas), se ha identificado que los valores faltantes no parecen estar distribuidos aleatoriamente. Por el contrario, los patrones observados y el contexto de recolección sugieren que los datos ausentes podrían deberse a factores relacionados con la propia naturaleza de las preguntas o con las características del encuestado. Algunas observaciones relevantes incluyen:

- En ciertos casos, los participantes omitieron preguntas sensibles o de carácter personal, posiblemente por incomodidad o desconfianza.
- Otros valores faltantes podrían deberse a fatiga en la respuesta del cuestionario, especialmente si estas preguntas se encontraban en secciones finales del formulario.
- También se identifica la posibilidad de que algunos participantes hayan respondido de forma deshonesta o poco comprometida, generando no solo omisiones sino también respuestas ruidosas o inconsistentes.

Dado este contexto, los datos faltantes se clasifican mayoritariamente como: Missing Not At Random (MNAR). La ausencia de datos está relacionada con el valor que falta o con factores no observables directamente. Debido a que la muestra ya ha sido recolectada y no es posible volver a obtener una nueva, el análisis debe realizarse con la información disponible. Por ello, se adoptará el siguiente enfoque:

- Se analizará cada columna con valores faltantes para determinar el tipo de variable, la proporción de faltantes y su posible relación con otras variables.
- Se evaluará la viabilidad de realizar imputaciones estadísticas simples (como media, mediana o moda) en variables numéricas o categóricas, cuando el contexto lo permita.
- Para casos más complejos o con mayor sesgo, se considerará el uso de técnicas avanzadas como MICE (Multiple Imputation by Chained Equations), que permiten imputar valores teniendo en cuenta múltiples variables relacionadas.
- En columnas donde la proporción de datos faltantes sea muy alta o no exista una justificación válida para la imputación, se optará por eliminar la variable del análisis o tratarla por separado, según corresponda.

Consideraciones Finales

Es importante reconocer que toda imputación introduce cierto grado de incertidumbre o sesgo, especialmente cuando se trabaja con datos MNAR. Por lo tanto, cualquier modelo construido a partir de este dataset deberá ser interpretado con cautela, documentando adecuadamente las decisiones tomadas durante la limpieza y transformación de los datos.

Este enfoque busca equilibrar el rigor metodológico con la practicidad del análisis, permitiendo obtener modelos útiles sin perder de vista las limitaciones del proceso de recolección de datos. Si la distribución cambia mucho, la imputación podría estar introduciendo sesgo artificial. Se realizaron gráficas para una comparación visual, como herramienta exploratoria muy valiosa cuando no se dispone de los verdaderos valores.

- **Validación de coherencia de datos**

  En esta fase se realiza una revisión exhaustiva de los datos para asegurar su consistencia lógica y validez estadística. Se verifican posibles incongruencias entre variables relacionadas, se revisan valores fuera de rango esperado y se identifican valores atípicos (outliers) que podrían distorsionar los análisis posteriores. Además, se confirma que las transformaciones anteriores (como la imputación de valores faltantes) hayan mantenido la integridad del conjunto de datos. Esta etapa es esencial para garantizar la calidad del dataset antes de su análisis estadístico o modelado. En el cuadernillo con el nombre "*3.Validacion_coherencia_datos.ipynb"* ubicado en la carpeta "*/notebooks*" se detalla este proceso, en el cual se hacen gráficas para validar valores atípicos. A continuación, algunos hallazgos y procesos relevantes.

	Carrera
0	Arqui
1	medicina veterinaria
2	Administración de Empresas
3	Administracion de Empresas
4	Auditoría
5	Profesorado en enseñanza media lengua y litera...
6	Contaduría publica
7	Física Matemáticas
8	Contadoria pública y auditoria
9	Licenciatura en las ciencias economico contables

Imagen 5. Validación de coherencia en "Carrera"

Elaboración propia

Se observa que las carreras no están escritas en un estándar adecuado, además de que deben codificarse para poder usarlas en el entrenamiento del modelo que se quiera. Debido

a que este campo fue digitado por el encuestado se tiene una gran cantidad de datos ambiguos, otros donde solo se tiene la unidad académica en lugar de la carrera o bien, datos equivocados. Por tal razón se toma la decisión de eliminar la columna de Carrera.

Identificación de valores atípicos

En el cuadernillo antes mencionado se exploran los datos y se concluye que no hay valores atípicos artificiales, no obstante, si en un futuro se manifiestan entonces para el tratamiento de Valores Atípicos se puede optar por alguno de los siguientes enfoques:

A. ***Eliminación del Valor Atípico:*** Cuando se identifica que el valor no refleja la realidad y proviene de un error de captura o ingreso (error artificial). Esto permite limpieza directa del ruido en los datos, pero puede perder información, especialmente si hay pocos datos.
B. ***Transformación del Valor Atípico*:** Cuando los outliers son reales, pero se desea reducir su impacto. Esto Conserva todos los datos, pero puede hacer menos interpretable el modelo si no se transforma de nuevo al final.
C. ***Imputación de un Valor Nuevo*:** Cuando el valor atípico se considera incorrecto y se prefiere sustituirlo por una estimación. Esto conserva la estructura del dataset sin perder filas, pero introducir un valor artificial que podría sesgar el análisis.

Identificación visual de outliers

Se procede a graficar cada columna y su análisis estadístico para determinar valores atípicos y tratarlos, así como también interpretar los valores obtenidos. Se utiliza un Boxplot, o también conocido como gráfica de cajas y bigotes. La "Caja" representa aproximadamente el 50% de los datos concentrados alrededor de la media. La línea que parte la caja es la mediana (valor central de datos) y sirve para ver si tenemos simetría en la "caja". Las líneas perpendiculares son los bigotes y representan el otro 50% de los datos; Se distribuyen como el 25% más pequeño y el 25% más grande según corresponda al bigote inferior o superior. Los puntos son los valores que se salen de la media y son los que se van a analizar como atípicos.

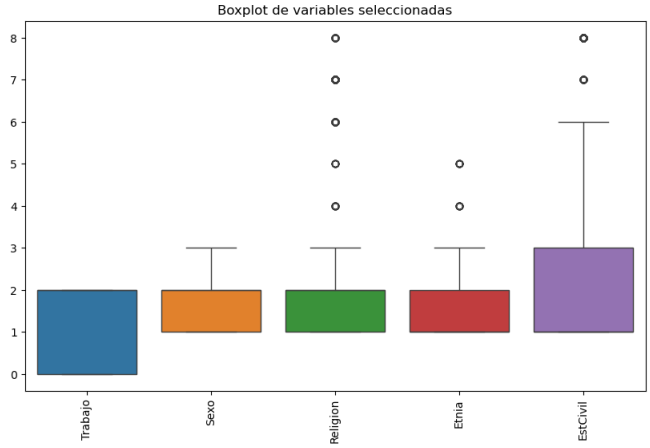

Imagen 6. Gráfica de ejemplo

Elaboración propia

- **Mapa de calor**

  El mapa de calor es una forma visual para encontrar correlación entre variables mediante el color. En el cuadernillo con el nombre "*4mapaDeCalor.ipynb"* ubicado en la carpeta "*/notebooks*" se detalla este proceso. En este se definió un color verde para la baja correlación; entre más oscuro el verde menos correlación lineal se tiene. Entre más intenso sea el color rojo más correlación lineal se tiene.

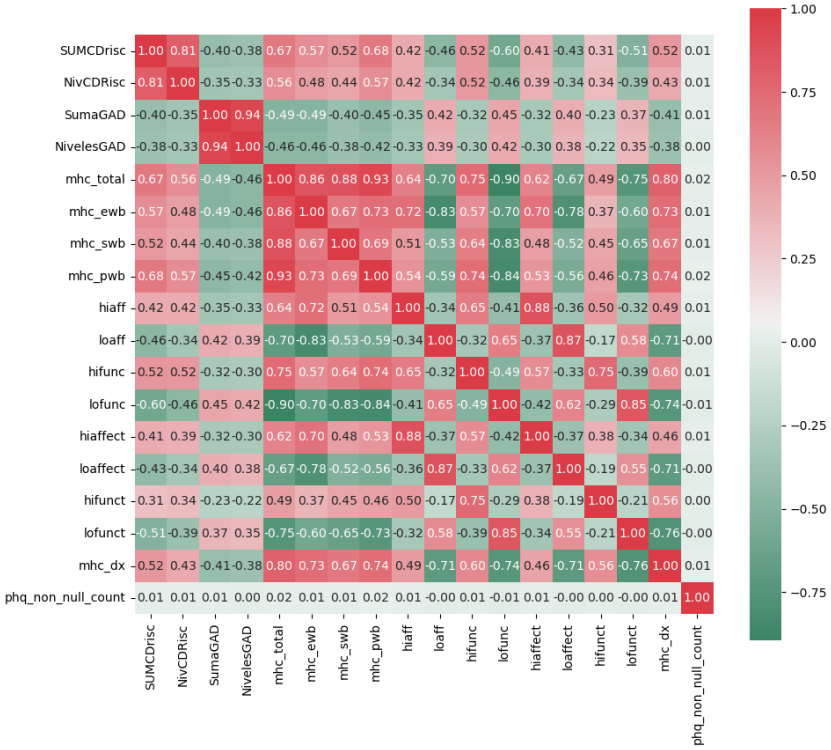

Imagen 7. Mapa de calor – parte I

Elaboración propia

Implementación de Modelos

1. Modelo de Selección de Características con Random Forest

El algoritmo Random Forest (Bosque Aleatorio) se emplea para estimar la relevancia de las variables predictoras (features) respecto a una variable objetivo (target). Su principio se basa en la construcción de múltiples árboles de decisión independientes, cada uno entrenado con una muestra aleatoria de los datos y un subconjunto aleatorio de variables. El promedio de las predicciones de estos árboles permite obtener un modelo robusto y menos propenso al sobreajuste**.**

En este proyecto, el modelo fue utilizado para identificar las variables de mayor impacto en el bienestar psicológico de los estudiantes, permitiendo filtrar características irrelevantes antes de entrenar modelos clasificadores más complejos.

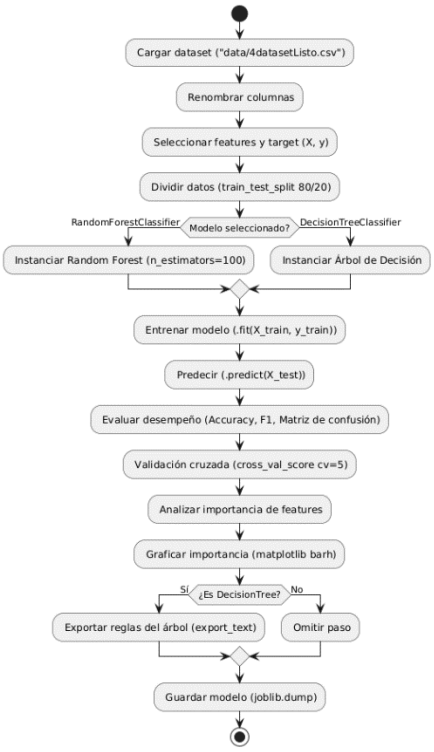

Imagen 8. Diagrama de flujo para Random Forest

Elaboración propia

Pasos implementados:

1. División del dataset en subconjuntos de entrenamiento y prueba (80/20).
2. Entrenamiento del modelo `RandomForestClassifier` con 100 árboles (`n_estimators=100`).
3. Evaluación de métricas de desempeño.
4. Cálculo de importancia de variables (`feature_importances_`).
5. Ajuste del hiperparámetro `class_weight='balanced'` para mitigar el desbalance de clases.

Hiperparámetros

Parámetro	Valor	Descripción
n_estimators	100	Número de árboles generados en el bosque.
max_depth	None	Permite crecimiento completo de los árboles.
class_weight	'balanced'	Ajusta pesos según frecuencia de clases.
random_state	42	Reproducibilidad de resultados.
n_jobs	-1	Usa todos los núcleos disponibles.

Tabla 2. Hiperparámetros de Random Forest.

Elaboración propia

Resultados

Métrica	Valor	Interpretación
Accuracy	0.74	Proporción total de aciertos.
Balanced Accuracy	0.51	Corrige por desbalance de clases.
Macro F1	0.53	Promedio del F1-score por clase, indica rendimiento moderado.

Tabla 3. Métricas de evaluación Random Forest.

Elaboración propia

Clase	Precision	Recall	F1-score	Observaciones
0 (Desanimado)	0.69	0.56	0.60	Buen desempeño, pero pierde sensibilidad.
1 (Moderado)	0.76	0.89	0.82	Excelente rendimiento; clase predominante.
2 (Florecido)	0.80	0.08	0.15	Alto error; el modelo casi no detecta esta clase.

Tabla 4. Desempeño por clase de Random Forest.

Elaboración propia

Variable	Importancia
SUMCDrisc	0.2086
SUMPHQ	0.1794
SumaGAD	0.1159
edad	0.0826
Semestre	0.0573
UnAca	0.0548
Religion	0.0392
Jornada	0.0374
CEntroU	0.0365
Trabajo	0.0284
EstCivil	0.0266

Tabla 5. Features más relevantes seleccionadas.

Elaboración propia

Ajuste y optimización

Se probó la inclusión de class_weight='balanced' para mitigar el desbalance de clases. Si bien la sensibilidad de las clases minoritarias aumentó ligeramente, las métricas globales no mostraron mejoras sustanciales.

Métrica	Valor anterior	Valor ajustado
Accuracy	0.733	0.727
Balanced Accuracy	0.505	0.492
Macro F1	0.531	0.511

Tabla 6. Métricas optimizadas.

Elaboración propia

Conclusiones del modelo

- El Random Forest resultó útil para estimar importancias de variables y comprender la estructura predictiva de los datos.
- La clase moderado presenta el mejor rendimiento, mientras que la clase *florecido* requiere técnicas adicionales (p. ej., SMOTE o modelos neuronales).
- Se conserva este modelo como referencia base para selección de características en posteriores etapas del pipeline.

2. Análisis de Componentes Principales (PCA)

Es una técnica estadística utilizada para la reducción de dimensionalidad. Su objetivo es transformar un conjunto de variables originales correlacionadas en un nuevo conjunto de **c**omponentes principales no correlacionados, que explican la mayor parte de la variabilidad presente en los datos. En este proyecto, el PCA se aplicó como un método exploratorio para identificar patrones latentes y agrupaciones naturales en los datos, y como paso complementario al modelo de Random Forest, con el fin de evaluar la redundancia entre variables y la estructura interna del dataset.

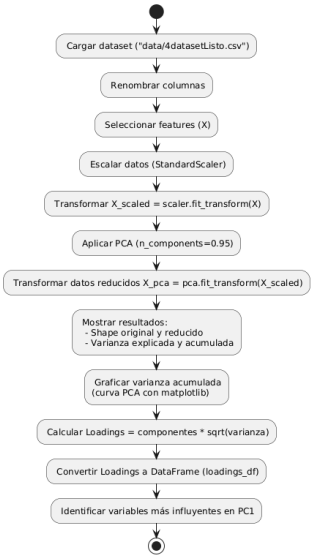

Imagen 9. Diagrama de flujo para PCA.

Elaboración propia

Característica	Valor
Variables originales	21
Varianza retenida	95%
Componentes resultantes	18
Reducción efectiva	3 variables (≈14%)

Tabla 7. Métricas PCA.

Elaboración propia

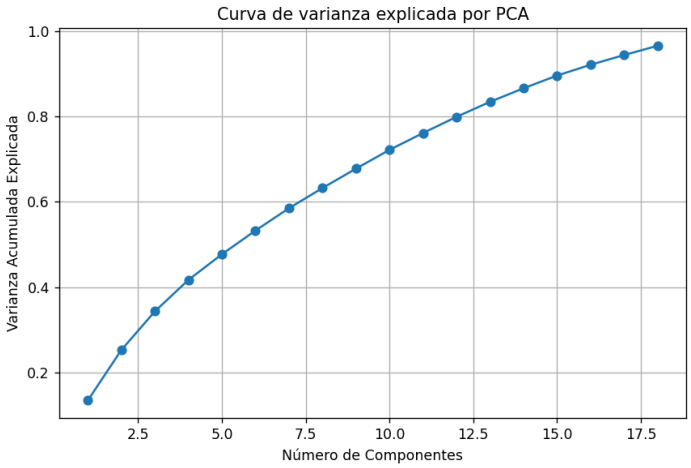

Imagen 10. Curva de varianza PCA.

Elaboración propia

Aunque la reducción no fue drástica, el resultado indica que las variables contienen información no redundante, lo cual valida la riqueza del dataset y la relevancia de mantener múltiples dimensiones en el análisis predictivo.

N° de componentes	Varianza explicada (%)	Varianza acumulada (%)
1	13.4	13.4
2	11.8	25.2
3	9.0	34.2
5	48.0	48.0
10	72.0	72.0
18	96.6	96.6

Tabla 8. Varianza de componentes.

Elaboración propia

Los primeros 10 componentes explican más del 70% de la varianza, y los 18 componentes retienen casi toda la información original, garantizando una representación fiel del espacio de datos.

Componente	Varianza (%)	Principales variables asociadas	Interpretación
PC1	13.4	Consumo de sustancias (Alcohol, Tabaco, Marihuana, ConsumoSustancias), SUMPHQ, SumaGAD	Eje de **consumo y salud mental**
PC2	11.8	Edad, trabajo, tener hijos (en oposición a escalas psicológicas)	Eje **sociodemográfico**
PC3	9.0	Sexo, ansiedad (SumaGAD), nivel académico	Eje **académico/psicológico**
PC4–PC5	—	Centro universitario, grado, terapia, jornada	Factores secundarios o contextuales

Tabla 9. Análisis de Loadings.

Elaboración propia

Resultados y análisis

1. No hay alta redundancia entre variables.
   Se necesitaron 18 componentes para conservar el 95% de la información, lo que confirma que las variables originales aportan información complementaria.
2. Los componentes principales revelan dimensiones interpretativas claras**:**
   - ✓ ***PC1 (Consumo y salud mental*)**: indica relación entre uso de sustancias y síntomas psicológicos.
   - ✓ ***PC2 (Factores sociodemográficos):*** diferencia por edad, trabajo y responsabilidades familiares.
3. Desde el punto de vista predictivo, PCA no mejoró la clasificación de `mhc_dx`, ya que no reduce significativamente la dimensionalidad y los modelos basados en árboles (como Random Forest) ya manejan la redundancia internamente.
4. Desde el punto de vista exploratorio, PCA sí aporta una visión estructural y conceptual del fenómeno estudiado, facilitando la interpretación de ejes psicológicos y sociodemográficos en el bienestar mental.

Conclusiones del modelo

- El PCA permite visualizar las dimensiones latentes del bienestar psicológico universitario.
- Los resultados evidencian al menos dos grandes ejes interpretativos:
  1. *Consumo + síntomas psicológicos* (salud mental y hábitos).
  2. *Factores sociodemográficos* (edad, trabajo, familia).
- Aunque no optimiza la predicción, proporciona una base teórica útil para comprender las relaciones entre variables.
- Para la fase de modelado predictivo, se conservaron las 11 variables seleccionadas por Random Forest**,** ya que mantienen mayor interpretabilidad directa**.**

Recomendaciones

- Aplicar PCA únicamente con fines de **análisis exploratorio** o visualización.
- En tareas de predicción, priorizar **selección de variables basada en importancia** (Random Forest).
- Considerar PCA en versiones futuras para **reducción previa a redes neuronales** (si se manejan muchas variables correlacionadas).

3. Modelo Gaussian Naive Bayes (GNB)

Este modelo utiliza el clasificador Gaussian Naive Bayes (GNB) para predecir el nivel de bienestar mental categórico (mhc_dx) a partir de variables psicológicas, demográficas y contextuales. Se realiza preprocesamiento de los datos, división estratificada de entrenamiento y prueba, escalado, entrenamiento y evaluación con métricas estándar. Debido al desbalance de clases (la clase floreciente es minoritaria), el análisis enfatiza tanto el rendimiento global como la capacidad del modelo para detectar categorías minoritarias.

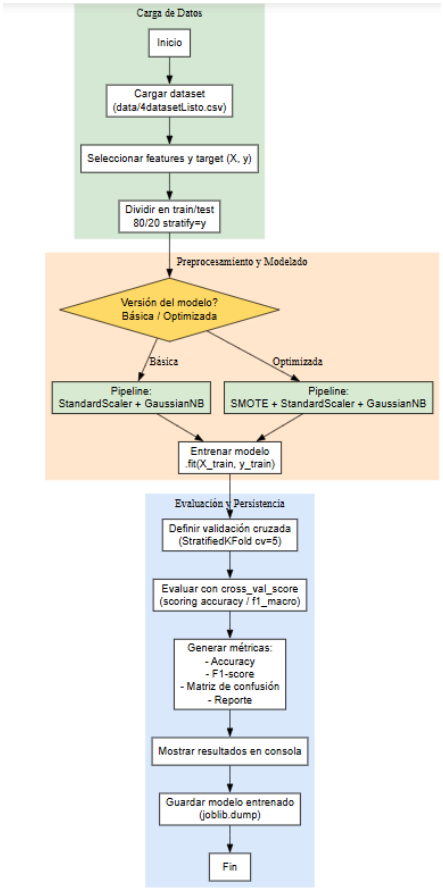

Imagen 11. Diagrama de flujo GNB.

Elaboración propia

Librerías y funcionalidad clave

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

- **GaussianNB**: implementación del modelo probabilístico con distribución gaussiana.
- **train_test_split(stratify=y)**: garantiza representatividad proporcional de clases en train/test.
- **StandardScaler**: estandariza las variables numéricas para mejorar estabilidad y comparación entre modelos.

División de datos y escalado

Para evitar sesgo y fuga de información, se implementó la siguiente estrategia:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- **Estratificación (stratify=y)**: conserva proporción de clases minoritarias.
- **Escalado estándar**: aunque GNB no lo requiere estrictamente, se aplica para consistencia intermodelo.
- **Semilla (random_state=42)**: asegura reproducibilidad del experimento.

Resultados del modelo original (sin balanceo)

Métrica	Valor
Accuracy	0.699
Macro F1	0.581
Balanced Accuracy	0.579
CV Accuracy (5 folds)	0.711 ± 0.008

Tabla 10. Métricas principales.

Elaboración propia

Reporte de clasificación:

Clase	Precision	Recall	F1-score	Support
0	0.604	0.611	0.608	450
1	0.770	0.771	0.771	1018
2	0.378	0.354	0.366	96

Tabla 11. Clasificación de clases.

Elaboración propia

Matriz de confusión:

Real \ Pred	0	1	2
0	275	173	2
1	179	785	54
2	1	61	34

Tabla 12. Matriz de confusión.

Elaboración propia

Interpretación

- La clase **1 (moderado)** es la mejor predicha.
- La clase **2 (floreciente)** tiene baja detección (recall=0.35).
- El modelo presenta **sesgo hacia la clase mayoritaria**.

Ajuste del modelo con SMOTE (balanceo de clases)

Para abordar el desbalance de clases, se aplicó *SMOTE (Synthetic Minority Over-sampling Technique)*, generando ejemplos sintéticos de clases minoritarias mediante interpolación.

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(k_neighbors=5, sampling_strategy={0:2500, 2:1200},
random_state=42)
X_res, y_res = sm.fit_resample(X_train_scaled, y_train)
```

Resultados con SMOTE:

Métrica	Sin SMOTE	Con SMOTE
Accuracy	0.699	0.646
Macro F1	0.581	0.570
Balanced Accuracy	0.579	**0.636**
Recall Clase 0	0.611	**0.660**
Recall Clase 1	0.771	0.643
Recall Clase 2	0.354	**0.604**

Tabla 13. Comparación entre métricas antes y después de optimizar.

Elaboración propia

Observaciones:

- Se logró mejor balance entre clases, aunque con ligera pérdida de accuracy global.
- La **clase 2** mejora sustancialmente su detección (+25 puntos en recall).
- Se selecciona k_neighbors=5 como óptimo por equilibrio entre precisión y recall (Revisar código y repositorio para ver las pruebas comparativas realizadas).

Interpretación y análisis comparativo

Aspecto	Modelo Original	Modelo con SMOTE
Sesgo	Hacia clase 1	Más equilibrado
Recall Clase 2	0.35	0.60
Balanced Accuracy	0.58	0.64
Consistencia (CV)	Alta (±0.008)	Estable (±0.011)

Tabla 14. Comparación de rendimiento entre modelos.

Elaboración propia

Conclusión

El balanceo mejora la equidad interclase, reduciendo el sesgo del modelo hacia la clase dominante. Aunque el accuracy baja ligeramente, se obtiene un desempeño más justo y robusto frente al desbalance de la muestra.

Limitaciones y recomendaciones

- La clase 2 sigue limitada en cantidad y variabilidad, lo que afecta su generalización.
- GaussianNB asume independencia total entre features, lo que no se cumple plenamente en variables psicológicas.
- El modelo es adecuado como baseline explicativo o filtro inicial, pero no como clasificador final.

Sugerencias futuras:

- Evaluar modelos que capturen correlaciones (KNN, Random Forest, MLP).
- Aplicar estrategias de recolección de datos o ajuste de pesos por clase.

El Gaussian Naive Bayes ofrece una primera aproximación sencilla y transparente para la clasificación del bienestar mental. Aunque sus suposiciones limitan el rendimiento con datos correlacionados, su bajo costo computacional **y** estabilidad lo convierten en un modelo base adecuado**.** El uso de SMOTE evidenció mejoras significativas en la detección de clases minoritarias, demostrando el valor de las estrategias de balanceo en contextos de datos psicológicos desiguales.

4. Clasificación con K-Nearest Neighbors (KNN)

El modelo K-Nearest Neighbors (KNN) se emplea para predecir el nivel de bienestar categórico (`mhc_dx`) usando variables psicológicas y demográficas. Se construye un pipeline que incluye preprocesamiento**,** balanceo de clases**,** escalado **y** clasificación, con el objetivo de evaluar la capacidad del modelo para identificar correctamente cada categoría de bienestar. KNN es un algoritmo de clasificación **basado en instancia** que asigna a un punto nuevo la clase más frecuente entre sus (k) vecinos más cercanos, según una métrica de distancia. El rendimiento depende de (k), la métrica de distancia y el escalado de las features. Es sensible a ruido y clases desbalanceadas.

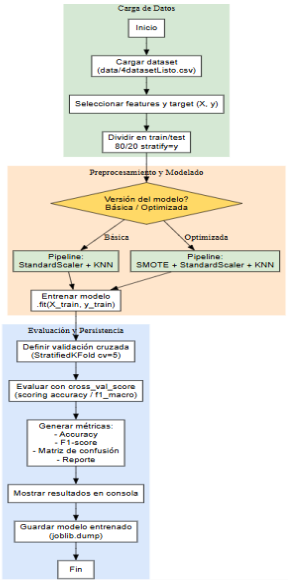

Imagen 12. Diagrama de flujo KNN.

Elaboración propia

Librerías y funcionalidad clave

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
```

- *KNeighborsClassifier*: algoritmo base de KNN.
- *Pipeline*: estructura para secuenciar balanceo, escalado y clasificación.
- *StandardScaler*: normaliza features, evitando dominancia por escala.
- *SMOTE*: balancea clases minoritarias generando ejemplos sintéticos.

Pipeline y parámetros clave

```
pipeline = Pipeline(steps=[
    ('smote', SMOTE(sampling_strategy={0:2500, 2:1200},
random_state=42)),
    ('scaler', StandardScaler()),
    ('knn', KNeighborsClassifier(
        n_neighbors=7,
        weights='distance',
        metric='minkowski',
        p=2,
        algorithm='auto'
    ))
])
```

- *SMOTE*: aumenta clases minoritarias para mitigar sesgo.
- *StandardScaler*: normaliza para que la distancia euclidiana sea coherente.
- *n_neighbors=7*: balance entre precisión y recall de todas las clases.
- *weights='distance'*: vecinos más cercanos tienen mayor peso.
- *metric='minkowski', p=2*: distancia euclidiana estándar.

Resultados del modelo original (sin balanceo)

*Accuracy***:** 0.691
Validación cruzada (5 folds): (0.634 \pm 0.049)

Clase	Precision	Recall	F1-score	Soporte
0	0.58	0.47	0.52	450
1	0.73	0.84	0.78	1018
2	0.52	0.11	0.19	96

Tabla 15. Métricas de rendimiento.

Elaboración propia

Matriz de confusión:

Real \ Pred	0	1	2
0	211	239	0
1	150	858	10
2	1	84	11

Tabla 16. Matriz de confusión KNN.

Elaboración propia

La clase mayoritaria (1) domina, mientras que la clase minoritaria 2 casi no se detecta.

Resultados con balanceo (SMOTE) y ajuste de k

Se evaluó el desempeño variando (k) vecinos:

k vecinos	Accuracy	F1-macro	Recall clase 0	Recall clase 1	Recall clase 2
3	0.660	0.519	0.56	0.74	0.25
4	0.685	0.500	0.48	0.83	0.15
5	0.657	0.540	0.56	0.74	0.33
6	0.669	0.542	0.56	0.75	0.33
7	**0.669**	**0.545**	**0.58**	0.74	0.34
8	0.677	0.551	0.58	0.75	0.31

Tabla 17. Resultado de diferentes pruebas del valor k.

Elaboración propia

Conclusión: (k=7) proporciona el mejor balance entre precisión global y recall para las clases minoritarias.

Matrices de confusión comparadas

Original:

Clase real ↓ / Predicha →	**Clase 0**	**Clase 1**	**Clase 2**
Clase 0	211	239	0
Clase 1	150	858	10
Clase 2	1	84	11

Tabla 18. Matriz de confusión original.

Elaboración propia

Ajustada (SMOTE + KNN k=7):

Clase real ↓ / Predicha →	**Clase 0**	**Clase 1**	**Clase 2**
Clase 0	260	186	4
Clase 1	193	754	71
Clase 2	3	60	33

Tabla 19. Matriz de confusión optimizada.

Elaboración propia

- **Clase 0:** recall mejora de 0.47 a 0.58
- **Clase 1:** recall baja ligeramente 0.84 a 0.74, sacrificando precisión para balancear otras clases.
- **Clase 2:** recall mejora significativamente 0.11 a 0.34, detectando más casos minoritarios.

Conclusiones generales

1. **KNN puro** favorece a la clase mayoritaria, penalizando la detección de minorías.
2. **SMOTE** mejora el recall de minorías y el F1-macro, equilibrando desempeño.
3. **k=7** es el mejor compromiso entre recall y precisión global.

5. MLP (Multi-layer Perceptron)

Este modelo entrena un clasificador **MLP (Multi-layer Perceptron)** para predecir el estado de bienestar categórico (`mhc_dx`) a partir de **escalas psicológicas y variables demográficas/contextuales**. El proceso incluye la división de datos, preprocesamiento (escalado), entrenamiento, predicción y evaluación con métricas estándar.

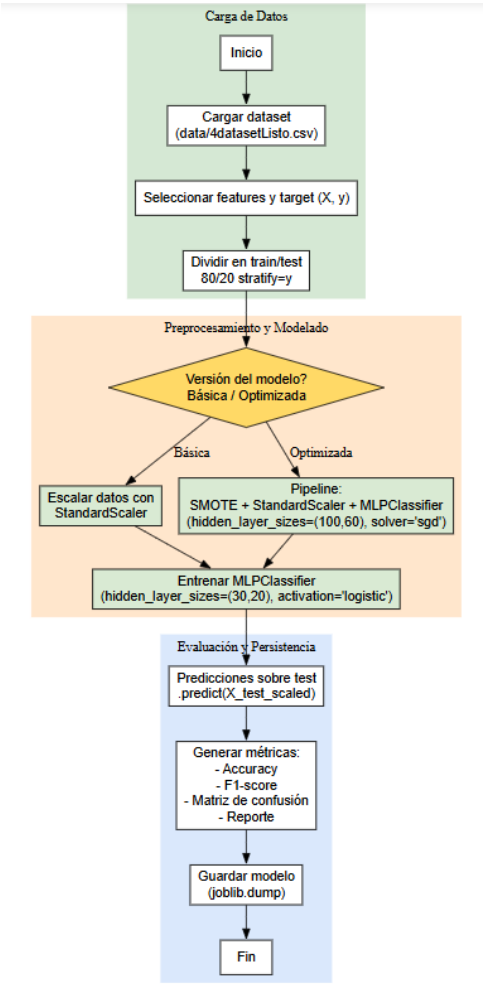

Imagen 13. Diagrama de flujo MLP.

Elaboración propia

Análisis del código

1. **Carga de datos:** El dataset se importa desde un archivo CSV previamente procesado.
2. **Selección de características y target:** Se seleccionan 10 variables predictoras (escalas psicológicas + datos demográficos) y la variable objetivo `mhc_dx`.
3. **División del dataset:** Entrenamiento (80%) y prueba (20%), con **estratificación** para mantener la proporción de clases.
4. **Preprocesamiento:** Normalización mediante `StandardScaler`, asegurando media 0 y desviación estándar 1.

5. **Entrenamiento:**
    - `MLPClassifier(hidden_layer_sizes=(30,20), activation='logistic', alpha=0.01, max_iter=3000, random_state=42)`
    - Arquitectura de dos capas ocultas.
    - Regularización para evitar sobreajuste.
6. **Evaluación:** Predicciones y métricas de desempeño sobre el conjunto de prueba.
7. **Persistencia:** El modelo se guarda para despliegue futuro.

Resultados del modelo original

```
Accuracy = 0.7404
```

El modelo acierta aproximadamente el **74%** de las predicciones sobre datos no vistos.

Reporte detallado por clase

Clase	Precision	Recall	F1-score	Soporte
0	0.66	0.58	0.62	450
1	0.77	0.87	0.81	1018
2	0.77	0.18	0.29	96

Tabla 20. Rendimiento MLP.

Elaboración propia

Interpretación:

- **Clase 1 (Moderado)** domina la predicción, con recall alto.
- **Clase 0 (Languishing)** aceptable, pero con errores cruzados con clase 1.
- **Clase 2 (Floreciente)** casi ignorada por el modelo (recall 0.18).

Matriz de confusión — Modelo original

Clase real ↓ / Predicha →	Clase 0	Clase 1	Clase 2
Clase 0	259	191	0
Clase 1	131	882	5
Clase 2	1	78	17

Tabla 21. Matriz de confusión MLP.

Elaboración propia

Análisis:

- Alta confusión entre clases 0 y 1.
- Clase 2 tiene baja tasa de acierto (17/96).

Optimización y pruebas con SMOTE

Para mejorar el balance de clases, se aplicó *SMOTE* y se exploraron distintas arquitecturas MLP. A continuación, se resume el desempeño comparativo de cada configuración.

Resumen de experimentos MLPClassifier + SMOTE

Escenario	Capas	Activación	Solver	Alpha	SMOTE	Accuracy	F1-macro	Recall C0	Recall C1	Recall C2	Observaciones
Original	(30,20)	logistic	adam	0.01	—	0.740	0.57	0.58	0.87	0.18	Clase 2 muy baja.
SMOTE + logistic	(30,20)	logistic	adam	0.01	{0:2500,2:1200}	0.676	0.54	0.59	0.75	0.30	Mejora recall C2.
Logistic	(50,30)	logistic	adam	0.01	{0:2500,2:1200}	0.692	0.57	0.57	0.77	0.41	Mejor balance.
Logistic	(60,30)	logistic	sgd	0.01	{0:2500,2:1200}	0.708	0.61	0.64	0.76	0.44	Óptimo general.
Logistic	(80,40)	logistic	sgd	0.01	{0:2500,2:1200}	0.708	0.61	0.64	0.76	0.46	Más estable, mejor recall C2.

Tabla 22. Tabla de escenarios trabajados.

Elaboración propia

Comparativa final — Modelo original vs óptimo (SMOTE + MLP)

Métrica	Original	Óptimo (SMOTE + MLP)
Accuracy	0.740	0.708
F1-macro	0.57	0.62
Recall Clase 0	0.58	0.64
Recall Clase 1	0.87	0.76
Recall Clase 2	0.18	0.47
Precision Clase 0	0.66	0.61
Precision Clase 1	0.77	0.79
Precision Clase 2	0.77	0.44

Tabla 23. Tabla comparativa de modelos.

Elaboración propia

Matrices de confusión comparadas

Clase real ↓ / Predicha →	Clase 0	Clase 1	Clase 2
Clase 0	259	191	0
Clase 1	131	882	5
Clase 2	1	78	17

Tabla 24. Matriz de confusión original.

Elaboración propia

Clase real ↓ / Predicha →	Clase 0	Clase 1	Clase 2
Clase 0	288	162	0
Clase 1	185	775	58
Clase 2	1	50	45

Tabla 25. Matriz de confusión optimizada.

Elaboración propia

Interpretación

- **Clase 2 (Floreciente)**: gran mejora (de 17 → 45 aciertos, recall 0.18 → 0.47).
- **Clase 0 (Languishing)**: mejora leve en recall (+6%).
- **Clase 1 (Moderado)**: ligera pérdida de recall, pero sigue siendo la más robusta.
- **F1-macro** aumenta de 0.57 a 0.62 → mejor equilibrio entre clases.

Conclusiones

1. SMOTE + MLP optimizado (capas 80,40) logra mayor equilibrio interclases, especialmente mejorando la detección de la clase minoritaria.
2. Aunque el accuracy global baja ligeramente, la mejora en recall y F1-macro justifica el cambio: el modelo deja de estar sesgado.
3. Es un modelo más justo y útil en contextos reales, donde las clases minoritarias (p. ej., "floreciente") son las más relevantes para la intervención psicológica.
4. Se recomienda su uso como versión de producción y continuar ajustes con nuevas muestras o técnicas de ensemble (como Bagging o Voting Classifier).

6. Análisis de Clustering con KMeans

Aplicar el algoritmo KMeans como técnica de aprendizaje no supervisado para agrupar individuos según variables clínicas, demográficas y sociales asociadas al bienestar mental (mhc_dx). El objetivo es identificar patrones latentes en los datos y explorar cómo se relacionan con las categorías de bienestar psicológico.

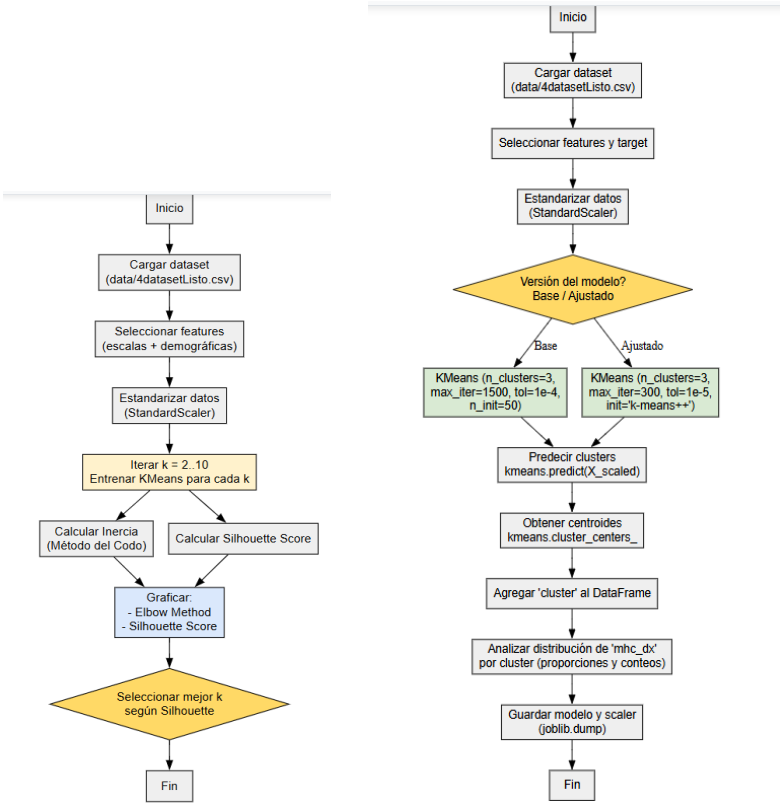

Imagen 14. Diagrama de flujo para número de cluster y KMeans .

Elaboración propia

Aspectos Clave del Proceso

- **Selección de variables (`features`):**
  Se emplearon 18 variables relevantes, incluyendo escalas psicológicas (`SUMPHQ`, `SumaGAD`, `mhc_total`) y variables demográficas (edad, sexo, estado civil, consumo de sustancias, entre otras).
- **Preprocesamiento:**
  Los datos fueron escalados con **`StandardScaler`** para igualar magnitudes y asegurar que las distancias euclidianas empleadas por KMeans tengan sentido comparativo entre variables.
- **Entrenamiento del modelo:**
  Se definieron 3 clusters (k=3) basados en el análisis de la curva del codo.
  El modelo se entrenó con datos escalados para optimizar la agrupación.
- **Predicción y análisis:**
  Se asignó a cada observación un cluster y se analizaron las proporciones y conteos de `mhc_dx` dentro de cada grupo.
- **Visualización:**
  Se construyeron gráficos de barras apiladas mostrando la proporción de las categorías `mhc_dx` en cada cluster.
- **Persistencia:**
  Tanto el modelo KMeans como el `StandardScaler` fueron guardados para su uso en futuras inferencias o validaciones.

Resultados del Modelo

Cluster	Desanimado (0)	Moderado (1)	Florecido (2)
0	0.114	0.760	0.125
1	0.192	0.759	0.049
2	0.582	0.410	0.008

Tabla 26. Distribución de mhc_dx por cluster (proporciones).

Elaboración propia

Cluster	Desanimado (0)	Moderado (1)	Florecido (2)	Total
0	293	1952	322	2567
1	543	2143	139	2825
2	1412	996	19	2427

Tabla 27. Conteo absoluto.

Elaboración propia

Cluster	Descripción	Características Principales
0 – Moderados/Florecidos	Grupo con funcionamiento positivo y resiliente.	Alta proporción de moderados (76%) y presencia significativa de florecidos (12%).
1 – Moderados/Intermedios	Riesgo medio, equilibrio entre moderados y desanimados.	Menos florecidos y más desanimados que el cluster 0.
2 – Críticos (Desanimados)	Grupo con mayor afectación clínica.	Más del 58% desanimados, casi sin florecidos. Debe recibir atención prioritaria.

Tabla 28. Interpretación de los clusters

Elaboración propia

Determinación del Número de Clusters

El análisis de la curva del codo (archivo `N_Clusters.py`) sugería **k=2**, pero este valor fusionaba los casos florecidos en un grupo general. Con **k=3**, se obtiene una segmentación más útil clínicamente, diferenciando:

1. Un grupo crítico (alta proporción de desanimados).
2. Un grupo intermedio (moderados con cierto riesgo).
3. Un grupo positivo (moderados y florecidos).

Configuración Original del Modelo

```
kmeans = KMeans(
    n_clusters=3,
    max_iter=1500,
    tol=1e-4,
    random_state=0,
    n_init=50
)
```

Resultados:

Cluster	0 (Desanimado)	1 (Moderado)	2 (Florecido)
0	0.1916	0.7584	0.0499
1	0.5812	0.4109	0.0078
2	0.1146	0.7606	0.1248

Tabla 29. Resultados del modelo original.

Elaboración propia

Versión Ajustada del Modelo

```
kmeans = KMeans(
    n_clusters=3,
    max_iter=300,
    tol=1e-5,
    random_state=0,
    init="k-means++"
)
```

Resultados:

Cluster	0 (Desanimado)	1 (Moderado)	2 (Florecido)
0	0.5803	0.4124	0.0074
1	0.1871	0.7610	0.0519
2	0.1189	0.7579	0.1232

Tabla 30. Resultados del modelo optimizado.

Elaboración propia

Cambios y Justificación

Parámetro	Antes	Ahora	Justificación
`max_iter`	1500	300	Reducción de iteraciones: el modelo ya convergía antes.
`tol`	1e-4	1e-5	Mayor precisión en la convergencia.
`init`	Aleatorio	`k-means++`	Mejora la estabilidad de los centroides iniciales.

Tabla 31. Ajuste de parámetros.

Elaboración propia

Impacto de los Ajustes

- **Cluster 0** pasó de representar mayormente la clase moderada (1) a reflejar principalmente la clase desanimada (0), mejorando la segmentación del grupo crítico.
- **Cluster 1** mantuvo su rol como grupo intermedio, con leve aumento en florecidos (de 4.9% a 5.2%).
- **Cluster 2** conservó su papel como el más asociado con florecidos, con resultados estables.

Conclusiones

1. **Mayor estabilidad:** `k-means++` reduce la variabilidad de los centroides entre ejecuciones.
2. **Redefinición útil del Cluster 0:** ahora se asocia más claramente con estudiantes en riesgo.
3. **Balance clínico más coherente:** cada cluster representa una dimensión distinta del bienestar psicológico.
4. **Eficiencia:** el modelo ajustado converge más rápido y mantiene calidad en la segmentación.
5. **Aplicabilidad:** esta agrupación puede servir como **base para estrategias de intervención diferenciadas**, identificando grupos vulnerables y promoviendo acciones preventivas específicas.

7. Redes Neuronales (Keras)

Este modelo se desarrolló como parte experimental del proyecto, con el objetivo de explorar el uso de redes neuronales (MLP) para la predicción del bienestar mental (`mhc_dx`). No forma parte del sistema principal implementado, pero permite comparar el rendimiento de un modelo basado en aprendizaje profundo frente a los enfoques clásicos utilizados (Random Forest, KNN, PCA + KMeans).

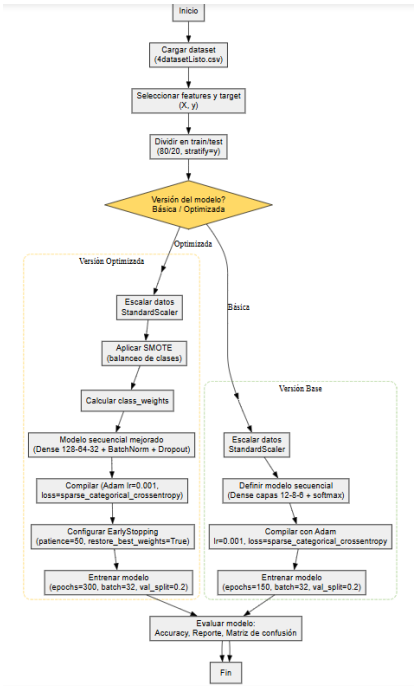

Imagen 15. Diagrama de flujo para modelo implementado en Keras.

Elaboración propia

Configuración del Modelo

- **División de datos:** 80% entrenamiento / 20% prueba, estratificada por clase.
- **Escalado:** `StandardScaler` aplicado a todas las features.
- **Batch size:** 32
- **Épocas:** 150
- **Validación:** 20% del set de entrenamiento reservado para validación interna.

Capa	Neuronas	Activación
Input	12	ReLU
Oculta 1	12	ReLU
Oculta 2	8	ReLU
Oculta 3	6	ReLU
Output	3	Softmax

Tabla 32. Configuración de capas.

Elaboración propia

- **Función de pérdida:** `sparse_categorical_crossentropy`
- **Optimizador:** Adam (`learning_rate=0.001`)
- **Métrica:** Accuracy

3. Resultados del Modelo Original

- **Accuracy en test:** 0.733 (~73%)

Clase	Precision	Recall	F1-score	Support
0 (Desanimado)	0.65	0.55	0.60	450
1 (Moderado)	0.76	0.87	0.81	1018
2 (Florecido)	0.71	0.16	0.26	96

Tabla 33. Rendimiento del modelo.

Elaboración propia

Promedio	Precision	Recall	F1-score
Macro avg	0.71	0.53	0.55
Weighted avg	0.73	0.73	0.71

Tabla 34. Métricas de rendimiento.

Elaboración propia

	Pred. 0	Pred. 1	Pred. 2
Real 0	249	201	0
Real 1	130	882	6
Real 2	2	79	15

Tabla 35. Matriz de confusión.

Elaboración propia

Análisis e Interpretación

1. **Clase Moderado (1):**
   - Alto recall (0.87). El modelo identifica bien esta clase, la más numerosa.
2. **Clase Desanimado (0):**
   - Recall moderado (0.55), confunde varios casos con “Moderado”.
3. **Clase Florecido (2):**
   - Recall bajo (0.16). Dificultad para detectar la clase minoritaria.
   - Causa probable: desequilibrio de clases**.**

Conclusión

Aunque la red neuronal obtiene un accuracy aceptable (~73%), no logra mejorar el rendimiento de los modelos tradicionales en las clases minoritarias. Esto evidencia la necesidad de balancear los datos o usar pesos de clase**.**

Modelo Ajustado (Balanceado con SMOTE + Class Weights)

Para mejorar la detección de clases minoritarias se implementó un modelo ajustado con las siguientes modificaciones:

Aspecto	Ajuste realizado
Balanceo	SMOTE aplicado al conjunto de entrenamiento
Pesos de clase	`compute_class_weight` para compensar el desbalance
Arquitectura	Red más profunda con Batch Normalization y Dropout
Regularización	Dropout 0.3–0.4
Entrenamiento	EarlyStopping con `patience=50`, 300 épocas
Optimización	Adam (learning_rate=0.001)

Tabla 36. Configuración del modelo.

Elaboración propia

Resultados Comparativos

Modelo	Accuracy
Original	0.733
Ajustado	0.641

Tabla 36. Precisión global.

Elaboración propia

Clase	Original	Ajustado
0 (Desanimado)	0.55	0.77
1 (Moderado)	0.87	0.59
2 (Florecido)	0.16	0.54

Tabla 37. Recall por clase.

Elaboración propia

Clase	Original	Ajustado
0 (Desanimado)	0.60	0.63
1 (Moderado)	0.81	0.68
2 (Florecido)	0.26	0.40

Tabla 38. F1-score por clase.

Elaboración propia

	Pred. 0	Pred. 1	Pred. 2
Real 0	249	201	0
Real 1	130	882	6
Real 2	2	79	15

Tabla 39. Matriz de confusión original.

Elaboración propia

	Pred. 0	Pred. 1	Pred. 2
Real 0	348	101	1
Real 1	306	602	110
Real 2	3	41	52

Tabla 40. Matriz de confusión optimizada.

Elaboración propia

Conclusiones Generales

- El modelo ajustado logra mayor equilibrio entre clases, mejorando sustancialmente el recall de las clases minoritarias (Desanimado y Florecido).
- Aunque disminuye el accuracy general, esto representa una mejora en equidad de clasificación**.**
- El experimento demuestra que las redes neuronales son una alternativa viable para problemas complejos de bienestar mental, pero requieren:
    - Más datos balanceados.
    - Optimización de arquitectura y regularización.
    - Mayor poder computacional (GPU, entorno Colab o NAS con soporte TensorFlow).

Entorno de Desarrollo

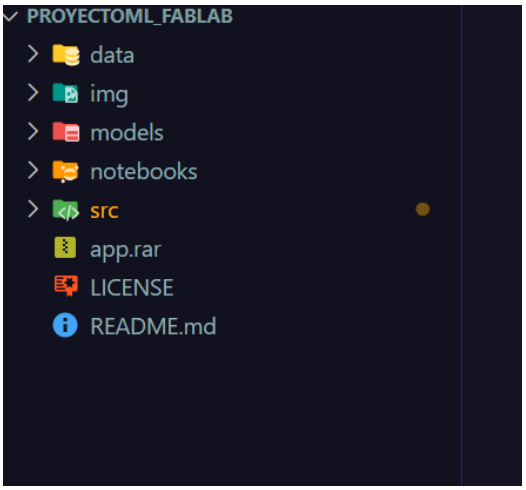

Imagen 16. Estructura del proyecto, rama "main"

Elaboración propia

Carpetas

- **Data:** Esta carpeta contiene los archivos en formato csv con la información utilizada, así como el libro de códigos que indican las variables, descripción y escalas utilizadas. Llevan un orden según el proceso en el que fueron generados, desde el dataset original hasta la parte de limpieza y transformación.

- **Img:** Esta carpeta contiene imágenes sobre resultados de implementaciones y otros procesos, son referenciadas en los archivos README.txt que se pueden encontrar en diferentes partes del código que detallan aún más aspectos y hallazgos relevantes.

- **Models:** Esta carpeta contiene los modelos implementados, cada uno en una subcarpeta que contienen los archivos ".joblib" para cargar y utilizar. También hay un modelo ".pkl"; esto solo es por temas de formato, ambos son válidos. El archivo "scaler.pkl" guarda el objeto StandardScaler usado para normalizar los datos de entrada. Es indispensable para mantener la coherencia entre el entrenamiento y las nuevas predicciones.

- **Notebooks:** En esta carpeta se encuentran los cuadernos de Jupyter trabajados para el preprocesamiento, cada uno tiene un orden y nombre descriptivo, así como el debido detalle de cada operación realizada.

- **Src:** Carpeta que contiene los archivos ".py" del entrenamiento y generación de modelos. Contiene subcarpetas enumeradas por orden de ejecución que contienen documentación y el script. También contienen una carpeta para imágenes de referencia, así como la subcarpeta "app" que contiene la implementación de la aplicación.
- **Dosc:** Contiene la documentación del proyecto a mayor detalle, Manual de Usuario y Manual Técnico (el presente documento).

Librerías y versiones

Librería	Versión	Descripción / Uso en el Proyecto
pandas	2.3.2	Manipulación y análisis de datos tabulares. Se utilizó para la carga del dataset, limpieza, transformación y generación de subconjuntos de entrenamiento y prueba.
numpy	1.26.4	Operaciones matemáticas y manejo eficiente de arreglos numéricos. Base para cálculos vectorizados utilizados en preprocesamiento y modelado.
scikit-learn	1.7.2	Biblioteca principal de *machine learning*. Se usó para la creación, entrenamiento y evaluación de modelos supervisados y no supervisados (Random Forest, KNN, KMeans, PCA, etc.). También incluye utilidades de escalado, partición de datos y métricas.
scipy	1.15.3	Soporte matemático y científico complementario. Se emplea en funciones estadísticas y cálculos avanzados usados indirectamente por `scikit-learn`.
joblib	1.2.0	Serialización eficiente de objetos Python, especialmente modelos y transformadores con grandes matrices NumPy. Se usó para guardar y cargar los modelos y escaladores entrenados.
streamlit	1.49.1	Framework para construir aplicaciones web interactivas de análisis y visualización de resultados. Se utilizó para el despliegue del modelo y la exploración de predicciones desde una interfaz gráfica.
imbalanced-learn	0.14.0	Técnicas de balanceo de datos (como SMOTE, RandomUnderSampler) para datasets desbalanceados.

Tabla 41. Librerías utilizadas.

Elaboración propia

Scripts

Se tienen 2 ramas, “main” como rama principal del proyecto, en esta se encuentra dentro de la carpeta “src” la subcarpeta “app”. Esta está pensada para ejecutar en entornos locales como aplicación de escritorio, a continuación, se detalla.

Nombre	Descripción
Requirements.txt	Archivo que contienen las librerías antes mencionadas, se deben instalar.
install_requirements.bat	El proyecto local de momento solo tiene cobertura para Windows, este archivo ejecuta comandos para instalar las dependencias de “Requirements.txt”
star_app.bat	Este inicia la aplicación, debe dar permisos necesarios y dejar abierta la ventana de la consola (CMD).
App.py	Archivo principal empleando streamlit para generar la interfaz e integrar la funcionalidad en un solo archivo monolítico. Funciona tanto de manera local como usando la plataforma oficial “streamlit.io”.

Tabla 42. Scripts más importantes.

Elaboración propia

La segunda rama es llamada “app”. Esta es para el despliegue directamente en la plataforma streamlit.io; Los archivos son una modificación de los archivos de Python de “src”, esto debido que Streamlit Cloud no guarda archivos binarios (.joblib, .pkl) fácilmente entre sesiones, ya que cada ejecución se da en un contenedor temporal. Por eso, lo correcto fue modificar el código para que genere los modelos en tiempo de ejecución (aunque tome más tiempo). Por tanto, solo se quedan los archivos de app.py y los “.py” son por cada modelo de la rama “main” ajustados para retornar el modelo.

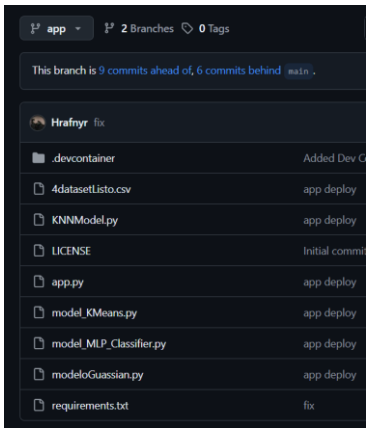

Imagen 17. Estructura del proyecto, rama “app”

Elaboración propia

Despliegue

Aplicación Local

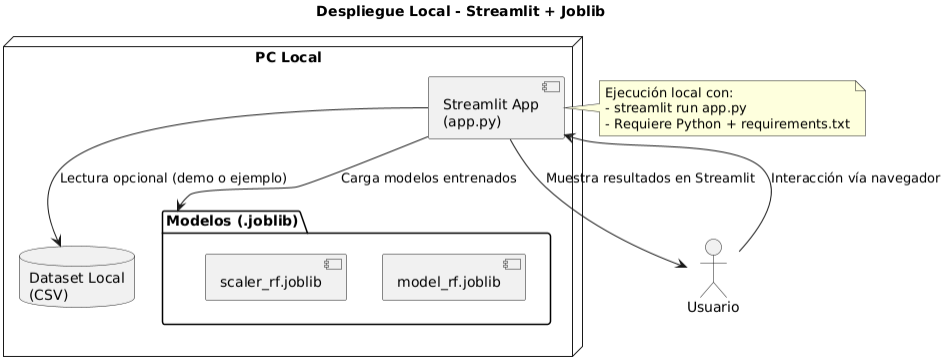

Imagen 18. Diagrama de despliegue local

Elaboración propia

Se descarga la carpeta “app” de la rama “main”, o en su defecto, el archivo “.rar”. Leer manual de usuario para configuración de entorno e inicio de aplicación. La aplicación se mostrará en una nueva ventana del navegador predeterminado ejecutándose de manera local, la ubicación no importa media vez se tengan los archivos en la carpeta integrada.

Stremlit Cloud

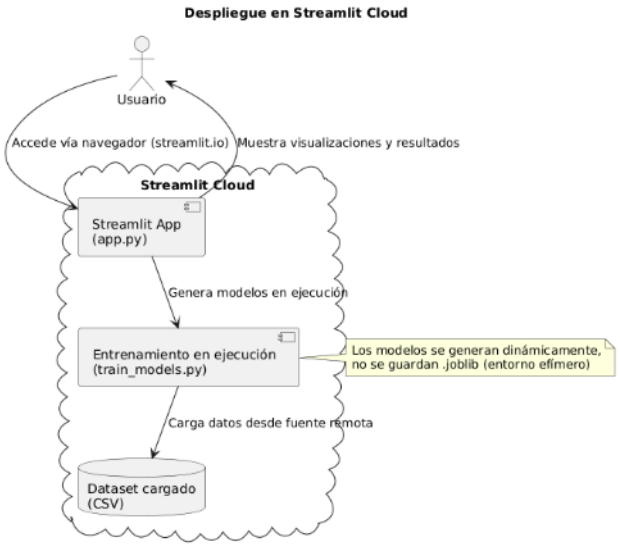

Imagen 19. Diagrama de despliegue en Streamlit

Elaboración propia

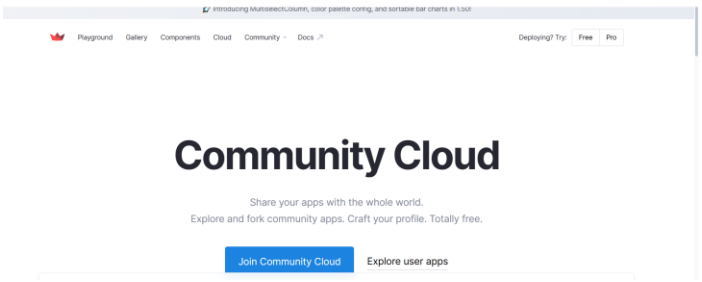

Imagen 20. Página de inicio oficial de streamlit

Elaboración propia

Streamlit Cloud es una plataforma en la nube gratuita ofrecida por Streamlit que permite desplegar y compartir aplicaciones interactivas de Python directamente desde un repositorio de GitHub. Funciona integrándose con el código fuente, ejecutando automáticamente el archivo principal (app.py) y gestionando las dependencias a partir de requirements.txt. No requiere infraestructura adicional ni configuración manual del servidor.

Ventajas principales:

- **Gratuito y sin mantenimiento:** permite publicar apps personales o académicas sin costo.
- **Despliegue automático:** solo es necesario subir el proyecto a GitHub; la app se ejecuta directamente en la nube.
- **Acceso web instantáneo:** genera una URL pública para compartir el proyecto con otros usuarios.
- **Entorno controlado:** aísla las dependencias, facilitando la reproducción de resultados sin conflictos locales.

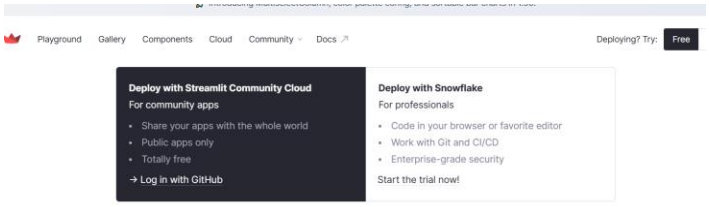

Imagen 21. Inicio de sesión con perfil de GitHub

Elaboración propia

Imagen 22. Crear un nuevo proyecto

Elaboración propia

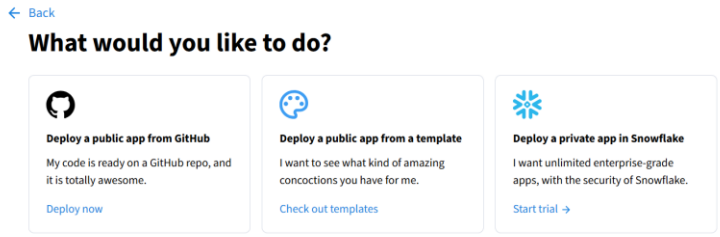

Imagen 23. Diferentes formas de despliegue en Streamlit

Elaboración propia

Para mayor facilidad se despliega el proyecto como una aplicación pública desde GitHub, luego de eso se despliega un menú donde se configura el repositorio, la rama y la ruta del archivo principal; en opciones avanzadas se puede elegir la versión de Python a utilizar (3.10).

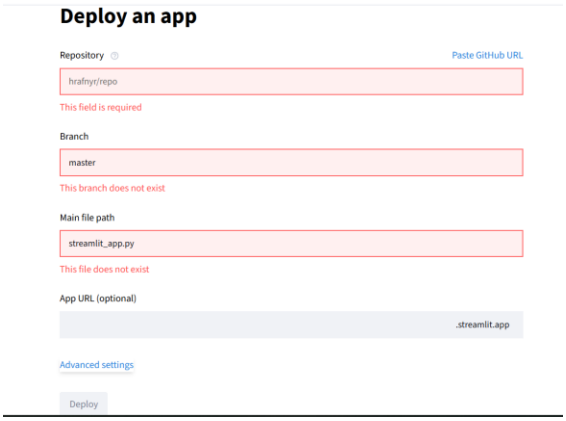

Imagen 24. Menú de configuración de despliegue

Elaboración propia

Como se mencionó anteriormente, se tienen 2 ramas, para el despliegue de en Streamlit Cloud se hizo la rama “app”, a continuación, se muestra y detalla la estructura del código.

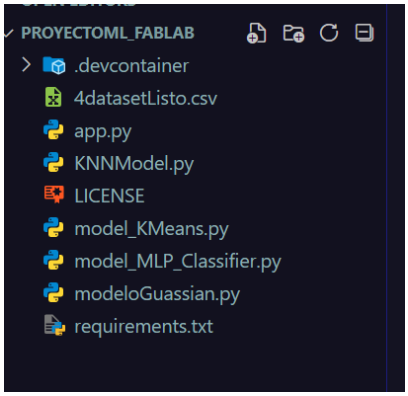

Imagen 25. Estructura del proyecto en rama “app”

Elaboración propia

Como se observa, en esta rama todo está al mismo nivel, los archivos “.py” son los mismos de la otra rama, pero ajustados para genera el modelo. También se adjunta el archivo csv.

NAS Synology DS223

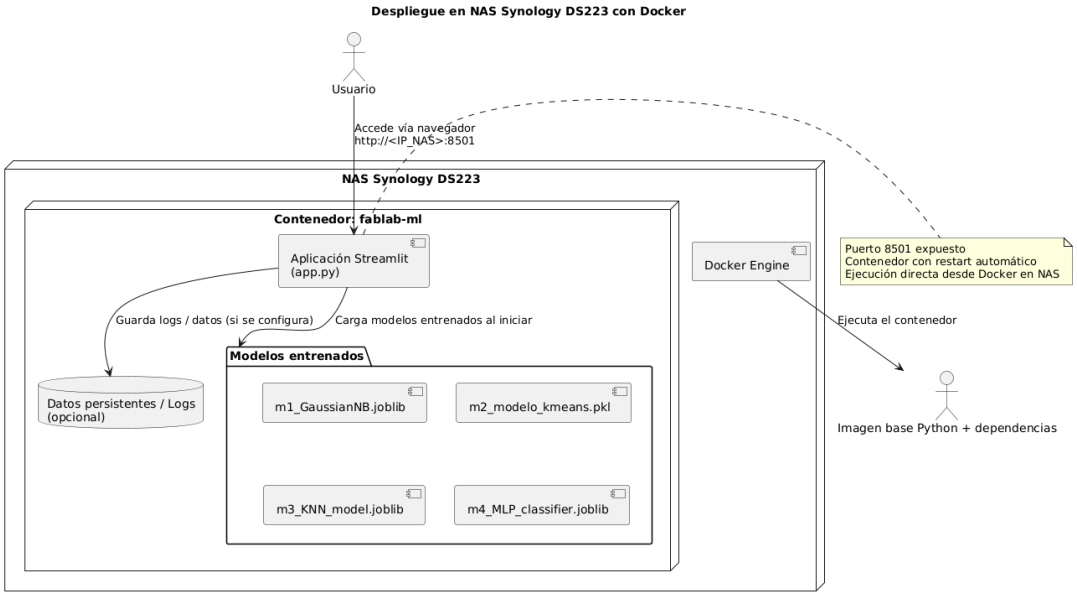

Imagen 25. Diagrama de despliegue – Servidor Interno

Elaboración propia

El Synology DiskStation DS223 es un NAS (Network Attached Storage) de gama de entrada con dos bahías diseñado para uso doméstico o de pequeña oficina. Permite almacenar, compartir y gestionar datos de forma centralizada. En este proyecto, se usa como plataforma para alojar una aplicación de machine learning mediante Docker, ejecutando una aplicación Streamlit que accede a modelos entrenados y sirve predicciones a los usuarios de la red local.

Característica	Detalle
CPU / Arquitectura	Realtek RTD1619B, 4 núcleos a 1.7 GHz, arquitectura 64 bits (Synology)
Memoria RAM	2 GB DDR4 (no expandible) (Synology)
Bahías de disco	2 (soporte para HDD/SSD de 3.5" o 2.5") (Synology)
Interfaces externas	1 puerto LAN Gigabit (RJ-45), 3 puertos USB 3.2 Gen 1 (Synology)
Sistemas de archivos soportados	Btrfs, ext4 internamente (NAS Compares)
Dimensiones / peso	165 × 108 × 232.7 mm, ~1.28 kg (Synology)
Consumo de energía	Aproximadamente 17.343 W en uso, 4.08 W en hibernación (Synology)
Ruido / ventilación	Ventilador de 92 mm, nivel de ruido ~14.6 dB(A) (Synology)
Sistema operativo	DSM (DiskStation Manager), sistema propio de Synology

Tabla 43. Tabla de especificaciones técnicas DS223.

Elaboración propia

Configuración de archivos

Debe ingresar con la IP y puerto del NAS e iniciar sesión ingresando las credenciales de usuario y contraseña. Una vez dentro entrar en *File Station* e ir a la carpeta dedicada.

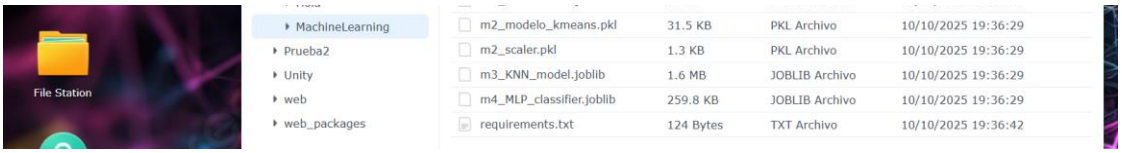

Imagen 65. Archivos en la ruta /homes/MachineLearning

Elaboración propia

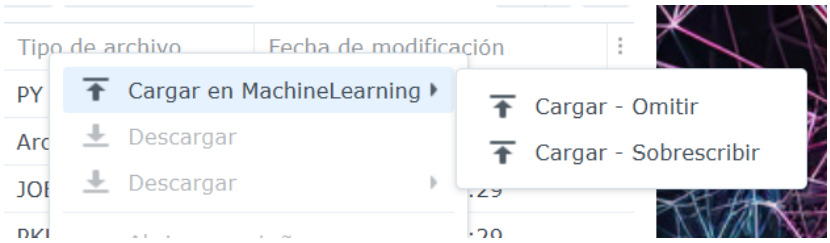

Imagen 66. Carga de archivos con clic derecho en la carpeta

Elaboración propia

Cuando se cargan archivos se le permite seleccionarlos desde la computadora facilitando el proceso. La carga es de todos los elementos en la carpeta "**app".** Note que debe incluir el archivo de construcción de imagen, Dockerfile.

Un Dockerfile es un archivo de texto que contiene las instrucciones necesarias para construir una imagen de Docker. Cada línea define una acción (como instalar dependencias o copiar archivos) que se ejecuta en una capa del sistema. El resultado es una imagen reproducible y portátil, lista para ejecutar una aplicación en cualquier entorno compatible con Docker.

En este caso, el Dockerfile define la imagen de la aplicación Streamlit de machine learning para que pueda desplegarse directamente en el NAS Synology DS223, que usa una arquitectura ARM64.

```dockerfile
# Imagen base ligera compatible con ARM (NAS DS223)
FROM python:3.11-slim

# Establecer directorio de trabajo
WORKDIR /app

# Copiar archivo de requisitos e instalarlos
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copiar el resto de tu proyecto
COPY . .

# Exponer el puerto de Streamlit
EXPOSE 8501

# Comando por defecto para iniciar la app
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]
```

Imagen 67. Dockerfile de la aplicación.

Elaboración propia

Línea	Instrucción	Descripción
`FROM python:3.11-slim`	Imagen base	Usa una imagen oficial de Python 3.11 ligera, ideal para dispositivos ARM (como el DS223). Incluye solo lo esencial para ejecutar Python.
`WORKDIR /app`	Directorio de trabajo	Crea y establece el directorio `/app` como el lugar donde se ejecutarán los comandos posteriores.
`COPY requirements.txt .`	Copiar dependencias	Copia el archivo `requirements.txt` al contenedor.
`RUN pip install --no-cache-dir -r requirements.txt`	Instalar dependencias	Instala las librerías necesarias de Python (Streamlit, pandas, scikit-learn, etc.) dentro del contenedor.
`COPY . .`	Copiar aplicación	Copia todos los archivos del proyecto (app.py, modelos, etc.) dentro del contenedor.
`EXPOSE 8501`	Exponer puerto	Indica que la aplicación usará el puerto **8501**, el que utiliza Streamlit por defecto.
`CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]`	Comando de inicio	Define el comando que se ejecuta cuando se inicia el contenedor: ejecuta la app de Streamlit y la hace accesible desde cualquier IP de la red.

Tabla 44. Explicación de la configuración del Dockerfile.

Elaboración propia

Despliegue

El despliegue de la aplicación se realizó directamente en el NAS Synology DS223, aprovechando su compatibilidad con Docker para ejecutar contenedores ligeros. Para ello, se utilizó una conexión SSH establecida mediante la herramienta Termius, que permitió acceder al sistema del NAS y ejecutar los comandos de construcción y ejecución del contenedor.

Termius es un cliente SSH multiplataforma que permite conectarse de forma segura a servidores remotos (como el NAS) desde Windows, macOS, Linux o dispositivos móviles. Proporciona una interfaz moderna y funcional para administrar conexiones, ejecutar comandos y transferir archivos sin necesidad de usar una terminal nativa. A continuación, listado del orden de comandos (Enter para ejecutarlos):

1. Entrar al directorio dedicado *MachineLearning:*

```
cd /var/services/homes/MachineLearning/
```

2. Construir la imagen Docker*:*

```
sudo docker build -t fablab-ml:latest .
```

3. Ejecutar contenedor:

```
sudo docker run -d -p 8501:8501 --name fablab-ml fablab-ml:latest
```

4. Configurar inicio de app por si el NAS se reinicia:

```
sudo docker update --restart unless-stopped fablab-ml
```

URL: http://192.168.1.164:8501/

De este modo se puede aplicar mantenimiento de la aplicación, cargar los nuevos archivos o modificados, reconstruir la imagen de Docker, así como detener y volver a levantar el servicio del contenedor.

Referencias

Docker, Inc. (s.f.). Docker: Accelerated container application development. Recuperado el 15 de octubre de 2025, de https://www.docker.com/

El Naqa, I., & Murphy, M. J. (2015). What is machine learning?. In Machine learning in radiation oncology: theory and applications (pp. 3-11). Cham: Springer International Publishing.

Grus, J. (2016). *Data science do zero* (Vol. 1). Rio d Janeiro: Alta books.

McKinney, W. (2022). Python para análisis de datos.

Mirjalili, V., & Raschka, S. (2020). *Python machine learning*. Marcombo.

Pandey, D., Niwaria, K., & Chourasia, B. (2019). Machine learning algorithms: a review. *Mach. Learn*, *6*(2).

Singh, A., Thakur, N., & Sharma, A. (2016, March). A review of supervised machine learning algorithms. In 2016 3rd international conference on computing for sustainable global development (INDIACom) (pp. 1310-1315).

Streamlit. (s.f.). *Streamlit: A faster way to build and share data apps*. Recuperado el 15 de octubre de 2025, de https://streamlit.io/

Synology Inc. (2025). DiskStation DS223 | Synology Inc. Recuperado el 15 de octubre de 2025, de https://www.synology.com/en-br/products/DS223

Termius. (s.f.). Termius: Modern SSH client. Recuperado el 15 de octubre de 2025, de https://termius.com/index.html