

RETO

Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México



**Tecnológico
de Monterrey**

Aplicación de métodos multivariados en ciencia de datos (Gpo 102)

MA2003B.102

Zareth Abigail Rodríguez Reyes A01659378

José Juan González Rebollo A01753090

Leonardo Huitrón Díaz Barriga A01800389

Ximena Lamparero García A01751331

Profesor: Saúl Juárez

Septiembre 2025

Objetivo y justificación

El conjunto de datos fue elegido debido a que nos resultó muy interesante, porque sabemos que la ansiedad en los estudiantes es cada vez más común, y al nosotros ser parte de esta comunidad nos resulta relevante entender sus causas y algunos factores relacionados.

El impacto sería muy beneficioso debido a que, al conocer estos factores, se podría hacer una detección temprana y así prevenir la ansiedad. También se pueden crear estrategias para el cuidado de la salud mental basándose en este análisis.

Nuestro objetivo es conocer cuáles son las variables que más impacto tienen en la ansiedad de los estudiantes, con el fin de identificar los

factores más determinantes y comprender como influyen en la salud mental.

Resumen

Este trabajo aplica diferentes métodos estadísticos multivariados para analizar los factores asociados a la ansiedad en estudiantes. Utilizando el Student Stress Monitoring Dataset (1,100 registros y 21 variables), se implementaron técnicas como análisis factorial, análisis discriminante, regresión lineal múltiple y clustering. Los resultados muestran que variables como autoestima, calidad del sueño, apoyo social, carga de estudio y nivel de estrés tienen un impacto significativo en los niveles de ansiedad. A través de la clasificación y la segmentación de los estudiantes, fue posible identificar grupos con diferentes perfiles de salud mental y rendimiento académico. Los hallazgos resaltan la importancia de detectar tempranamente factores de riesgo y diseñar estrategias de prevención y apoyo para la salud mental estudiantil.

Índice

- Introducción
- Marco teórico
- Metodología
- Implementación
 - Importar base de datos
 - Preprocesamiento y estandarización
- Análisis factorial
 - Obtener las comunidades
 - Visualización
- Análisis discriminante
 - Alternativa para cuando las fronteras no son lineales
 - Visualizar en ejes discriminantes
- Regresión lineal múltiple
- Conglomerados
 - Preparación de datos
 - Determinar el número de clusters
 - Algoritmo de clustering
 - Visualización de clusters
 - Interpretación de resultados
- Análisis y Resultados
- Conclusión
- Referencias

Introducción

Los estudiantes están sometidos constantemente a la presión de exámenes, tareas y proyectos. El conjunto de datos que usamos contiene los niveles de ansiedad, estrés, autoestima, calidad de sueño, entre otros factores que tienen relación entre sí. La importancia de comprender y estudiar estos elementos ayuda a mejorar la vida estudiantil, de manera que los estudiantes demuestren todo su potencial sin ser afectados negativamente por diferentes aspectos escolares.

Marco Teórico

Análisis factorial

El análisis factorial es una técnica estadística de reducción de datos para explicar las correlaciones entre variables observadas. Se agrupan variables observadas en un número menor de factores que concentran la mayor parte de la varianza común (Universitat de València, s. f.).

Modelo general:

$$X_i = \lambda_{i1}F_1 + \lambda_{i2}F_2 + \cdots + \lambda_{im}F_m + \varepsilon_i$$

donde:

- X_i : variable observada
- λ_{ij} : cargas factoriales (peso de la variable en el factor)
- F_m : factores comunes
- ε_i : error específico o varianza no explicada

El número de factores se determina con criterios como el de **Kaiser** (eigenvalores mayores que 1) o el **Scree Plot**.

Para verificar la pertinencia del análisis se usan pruebas como:

- **Índice KMO (Kaiser-Meyer-Olkin)**

Análisis discriminante

El análisis discriminante busca clasificar individuos en grupos previamente definidos a partir de variables explicativas. Su propósito es encontrar una combinación lineal de variables independientes que maximice la separación entre categorías, al mismo tiempo que minimiza la varianza dentro de los grupos (IBM, s.f.).

En la salud mental, el análisis discriminante permite identificar los factores que mejor diferencian a estudiantes con distintos niveles de ansiedad o estrés, aportando un modelo útil para predecir a qué grupo pertenece un nuevo dato.

Regresión lineal múltiple

Permite modelar la relación entre una variable dependiente y variables independientes. El modelo se basa en la estimación de coeficientes mediante el método de mínimos cuadrados ordinarios OLS. Estos indican la magnitud, dirección e impacto en de cada variable predictora (Amat, 2023). En este caso es útil este método para cuantificar factores como rendimiento académico, calidad de sueño o apoyo social y ver qué tanto impactan en los niveles de ansiedad.

Modelo general:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

donde:

- Y : variable dependiente
- β_0 : intercepto
- β_j : coeficiente de cada predictor
- X_j : variables independientes
- ε : término de error

La estimación de los coeficientes se realiza mediante **Mínimos Cuadrados Ordinarios (OLS)**, minimizando:

$$\min \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Pruebas de hipótesis Las pruebas de hipótesis permiten tomar decisiones sobre parámetros poblacionales con base en información muestral (Ortega, s.f.).

Pasos generales:

1. Formular hipótesis nula H_0 y alternativa H_1 .
2. Seleccionar un nivel de significancia α .
3. Calcular el estadístico de prueba (t , F , χ^2 , etc.).
4. Obtener el valor p y compararlo con α .

Análisis de conglomerados

Para analizar patrones se emplea el análisis de conglomerados (clustering), técnica de aprendizaje no supervisado que agrupa individuos con características similares. Uno de los algoritmos más usados es K-Means, que minimiza la distancia entre cada punto y el centroide de su grupo:

$$J = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$

donde:

k = número de clusters,

C_i = conjunto de puntos en el cluster i ,

μ_i = centroide del cluster.

Este enfoque permite identificar perfiles de estudiantes: desde aquellos con altos niveles de riesgo (ansiedad y depresión elevados) hasta los más resilientes (autoestima alta y bajo estrés), lo que facilita diseñar estrategias de apoyo académico y psicológico.

Metodología

Explicar los pasos de manera general Explicar la estructura de la base de datos

- **Obtención y preparación de datos**

Se utilizó el conjunto de datos Student Stress Monitoring Dataset de Kaggle que contiene 1100 registros y 21 variables con indicadores psicológicos, académicos, sociales, etc. Algunos son el nivel de ansiedad, autoestima, historial mental, descripción, calidad de sueño, rendimiento, carga de estudio, nivel de estrés, etc.

- **Preprocesamiento y Estandarización**

Los datos venían completos, por lo que no fue necesario eliminar filas por valores nulos, ni nada por el estilo. En todo caso, el único preprocesamiento que se hizo fue antes del análisis discriminante en el cuál se modificó la variable de "anxiety_level" a una variable categórica para que pudiera usarse en el modelo.

Para la estandarización se usó la siguiente fórmula en todo el conjunto de datos:

$$z = \frac{x - \mu}{\sigma}$$

Esto se hizo para buscar que los datos queden en una escala con media cero y desviación estándar 1 y también como preparación para los métodos estadísticos que se aplicaron después.

- **Análisis factorial**

Se instaló factor_analyzer, posteriormente se realizaron dos pruebas; Barlett's Sphericity Test y la Kaiser-Meyer-Olkin Test para saber si el conjunto de datos era apropiado para hacerle un análisis factorial.

Primero se realizó un análisis factorial con el mismo número de factores latentes que el número de variables, posteriormente, con un gráfico de scree plot, se eligieron un total de dos factores latentes para el entrenamiento del modelo.

Por último se obtuvieron las comunalidades y se hizo una visualización con un mapa de calor.

- **Análisis discriminante**

Como se indicó anteriormente, se convirtió la variable "anxiety_level" en una variable categórica tomando en cuenta la media de la variable original, siendo 1 si era mayor a la media y 0 si era igual o menor.

Posteriormente, la muestra se dividió en datos de entrenamiento y datos de prueba para aplicarlos a ambos el modelo de LDA. Se hizo una búsqueda de hiperparámetros y se evaluó el modelo con un reporte de clasificación general por encima de 0.76 tanto para precision, recall y f1-score.

- **Regresión lineal múltiple**

Para el modelo de Regresión Multiple se dividió el conjunto de datos usando todas las variables. "anxiety_level" fue nuestra variable predictora y todas las demás variables fueron nuestra X.

También se evaluaron las correlaciones para elegir X, sin embargo, al intentar colocar sólo las variables con mayor correlación observábamos que el R^2 disminuía bastante.

Posteriormente se hizo la regresión con el modelo OLS sin usar ni training ni testing y realmente el modelo no mejoró mucho cuando dividimos el conjunto en datos de entrenamiento y en datos de prueba. Esto podemos saberlo ya que el R^2 fue tan sólo 2 milesimas mayor que el primero.

- **Análisis de conglomerados**

Para analizar patrones se emplean las graficas para seleccionar la cantidad de clusters, y se aplica K-Means técnica de aprendizaje no supervisado que agrupa individuos con características similares. Uno de los algoritmos más usados es , que minimiza la distancia entre cada punto y el centroide de su grupo. Y posterior a calcular los clusters se interpretan las agrupaciones hechas.

Descripción de las variables

Las variables del conjunto de datos son:

'anxiety_level': Nos dice el nivel de ansiedad que tiene el estudiante en un rango de 0 a 21.

'selfEsteem': Nos indica el nivel de autoestima del estudiante en un rango de 0 a 30.

'mental_health_history': Nos señala si el estudiante tiene historial de salud mental en un rango de 0 a 1.

'depression': Nos dice el nivel de depresión que presenta el estudiante en un rango de 0 a 27.

'headache': Nos muestra la presencia de dolor de cabeza en un rango de 0 a 5.

'blood_pressure': Nos indica el nivel de presión arterial del estudiante en un rango de 1 a 3.

'sleep_quality': Nos dice la calidad del sueño que tiene el estudiante en un rango de 0 a 5.

'breathing_problem': Nos señala la presencia de problemas respiratorios en un rango de 0 a 5.

'noise_level': Nos indica el nivel de ruido en el entorno del estudiante en un rango de 0 a 5.

'living_conditions': Nos muestra las condiciones de vida del estudiante en un rango de 0 a 5.

'safety': Nos dice el nivel de seguridad percibida por el estudiante en un rango de 0 a 5.

'basic_needs': Nos señala si el estudiante tiene cubiertas sus necesidades básicas en un rango de 0 a 5.

'academic_performance': Nos indica el rendimiento académico del estudiante en un rango de 0 a 5.

'study_load': Nos dice la carga de estudio que tiene el estudiante en un rango de 0 a 5.

'teacher_student_relationship': Nos señala la calidad de la relación maestro-estudiante en un rango de 0 a 5.

'future_career_concerns': Nos indica las preocupaciones del estudiante sobre su carrera futura en un rango de 0 a 5.

'social_support': Nos dice el nivel de apoyo social con el que cuenta el estudiante en un rango de 0 a 3.

'peer_pressure': Nos señala el nivel de presión de grupo que experimenta el estudiante en un rango de 0 a 5.

'extracurricular_activities': Nos indica la participación del estudiante en actividades extracurriculares en un rango de 0 a 5.

'bullying': Nos muestra si el estudiante sufre acoso escolar en un rango de 0 a 5.

'stress_level': Nos dice el nivel de estrés que presenta el estudiante en un rango de 0 a 2.

Implementación

Importar base de datos

Se importa la base de datos desde Kaggle y se asigna a la variable data.

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("mdsultanolislamovi/student-stress-monitoring-datasets")

print("Path to dataset files:", path)
```

Using Colab cache for faster access to the 'student-stress-monitoring-datasets' dataset.

Path to dataset files: /kaggle/input/student-stress-monitoring-datasets

```
import os
print(os.listdir(path))

['StressLevelDataset.csv', 'Stress_Dataset.csv']
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd
import os

# Construct the full path to the CSV file
csv_file_path = os.path.join(path, "StressLevelDataset.csv")

# Read the CSV file into a DataFrame
data = pd.read_csv(csv_file_path)
data
```

| | anxiety_level | selfEsteem | mentalHealthHistory | depression | headache | bloodPressure |
|------|---------------|------------|---------------------|------------|----------|---------------|
| 0 | 14 | 20 | | 0 | 11 | 2 |
| 1 | 15 | 8 | | 1 | 15 | 5 |
| 2 | 12 | 18 | | 1 | 14 | 2 |
| 3 | 16 | 12 | | 1 | 15 | 4 |
| 4 | 16 | 28 | | 0 | 7 | 2 |
| ... | ... | ... | | ... | ... | ... |
| 1095 | 11 | 17 | | 0 | 14 | 3 |
| 1096 | 9 | 12 | | 0 | 8 | 0 |
| 1097 | 4 | 26 | | 0 | 3 | 1 |
| 1098 | 21 | 0 | | 1 | 19 | 5 |
| 1099 | 18 | 6 | | 1 | 15 | 3 |

1100 rows × 21 columns

Se explora la base de datos y sus métricas principales.

```
data.describe()
```

| | anxiety_level | selfEsteem | mentalHealthHistory | depression | headache | bloodPressure |
|-------|---------------|-------------|---------------------|-------------|-------------|---------------|
| count | 1100.000000 | 1100.000000 | 1100.000000 | 1100.000000 | 1100.000000 | 1100.000000 |
| mean | 11.063636 | 17.777273 | | 0.492727 | 12.555455 | 2.508182 |
| std | 6.117558 | 8.944599 | | 0.500175 | 7.727008 | 1.409356 |
| min | 0.000000 | 0.000000 | | 0.000000 | 0.000000 | 0.000000 |
| 25% | 6.000000 | 11.000000 | | 0.000000 | 6.000000 | 1.000000 |
| 50% | 11.000000 | 19.000000 | | 0.000000 | 12.000000 | 3.000000 |
| 75% | 16.000000 | 26.000000 | | 1.000000 | 19.000000 | 3.000000 |
| max | 21.000000 | 30.000000 | | 1.000000 | 27.000000 | 5.000000 |

8 rows × 21 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1100 entries, 0 to 1099
Data columns (total 21 columns):
 #   Column           Non-Null Count Dtype  
 ---  -- 
 0   anxiety_level    1100 non-null   int64  
 1   selfEsteem        1100 non-null   int64  
 2   mentalHealthHistory 1100 non-null   int64  
 3   depression        1100 non-null   int64  
 4   headache          1100 non-null   int64  
 5   bloodPressure     1100 non-null   int64  
 6   sleepQuality      1100 non-null   int64  
 7   breathingProblem  1100 non-null   int64  
 8   noiseLevel        1100 non-null   int64  
 9   livingConditions  1100 non-null   int64  
 10  safety            1100 non-null   int64  
 11  basicNeeds        1100 non-null   int64  
 12  academicPerformance 1100 non-null   int64  
 13  studyLoad         1100 non-null   int64  
 14  teacherStudentRelationship 1100 non-null   int64  
 15  futureCareerConcerns 1100 non-null   int64  
 16  socialSupport     1100 non-null   int64  
 17  peerPressure      1100 non-null   int64  
 18  extracurricularActivities 1100 non-null   int64  
 19  bullying          1100 non-null   int64  
 20  stressLevel       1100 non-null   int64  
dtypes: int64(21)
memory usage: 180.6 KB
```

```
data['anxiety_level'].mean()
```

```
np.float64(11.063636363636364)
```

```
data['depression'].mean()
```

```
np.float64(12.555454545454545)
```

Preprocesamiento y estandarización

Se **estandariza** la base de datos a través de `StandardScaler` de Scikit Learn. La estandarización es necesaria para que las variables numéricas tengan la misma escala y se puedan comparar.

Lo que hace StandardScaler es:

- Restar la media de cada variable (centrado en 0).
- Dividir entre la desviación estándar (ajustando la dispersión a 1).

De esta manera, cada variable se convierte en una distribución con media 0 y desviación estándar 1.

```
#Estandarización de data
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = scaler.fit_transform(data)
df = pd.DataFrame(df_scaled, columns = data.columns)
```

```
df
```

| | anxiety_level | selfEsteem | mentalHealthHistory | depression | headache | blood |
|------|---------------|------------|---------------------|------------|-----------|-----------|
| 0 | 0.480208 | 0.248612 | | -0.985559 | -0.201393 | -0.360741 |
| 1 | 0.643746 | -1.093590 | | 1.014653 | 0.316508 | 1.768859 |
| 2 | 0.153131 | 0.024912 | | 1.014653 | 0.187033 | -0.360741 |
| 3 | 0.807284 | -0.646189 | | 1.014653 | 0.316508 | 1.058992 |
| 4 | 0.807284 | 1.143414 | | -0.985559 | -0.719293 | -0.360741 |
| ... | ... | ... | | ... | ... | ... |
| 1095 | -0.010407 | -0.086938 | | -0.985559 | 0.187033 | 0.349125 |
| 1096 | -0.337484 | -0.646189 | | -0.985559 | -0.589818 | -1.780475 |
| 1097 | -1.155175 | 0.919713 | | -0.985559 | -1.237193 | -1.070608 |
| 1098 | 1.624976 | -1.988391 | | 1.014653 | 0.834408 | 1.768859 |
| 1099 | 1.134361 | -1.317290 | | 1.014653 | 0.316508 | 0.349125 |

1100 rows × 21 columns

Análisis Factorial con FactorAnalyzer

```
!pip install factor_analyzer --quiet
████████████████████████████████████████████████████████████████ 0.0/42.8 KB ? eta -:--:-
████████████████████████████████████████████████████████████████ 42.8/42.8 KB 1.6 MB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Building wheel for factor_analyzer (pyproject.toml) ... done
```

- Bartlett's Sphericity Test: Con esta prueba vamos a observar si las variables están significativamente correlacionadas. Un p-value que sea menor a 0.05 nos indica que las variables están suficientemente relacionadas para aplicar un análisis factorial.
- Kaiser-Meyer-Olkin (KMO) Test: Esta prueba mide la adecuación de las variables y la muestra para un análisis factorial. Los valores mayores a 0.6 se consideran aceptables, y mientras más cercanos estén a 1 significa que hay una buena adecuación muestral.

```
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity, calculate_kmo
barlett, barlett_p_value = calculate_bartlett_sphericity(df)
kmo_all, kmo_avg = calculate_kmo(df)
```

```
print(f"Barlett Sphericity Test:")
print(f"Test Statistic: {barlett:.4f}")
print(f"p-value: {barlett_p_value:.4f}")
print(f"\nKaiser-Meyer-Olkin (KMO) Test:")
print(f"Overall KMO statistic: {kmo_avg:.4f}")
```

Barlett Sphericity Test:
Test Statistic: 19237.2252
p-value: 0.0000

Kaiser-Meyer-Olkin (KMO) Test:
Overall KMO statistic: 0.9712

Como el p-value de la prueba de Bartlett fue menor a 0.05, y el estadístico de KMO fue mayor a 0.6, significa que los datos son adecuados para realizar un análisis factorial.

```
fa = FactorAnalyzer(n_factors = df.shape[1], rotation = None, method = 'principal',
fa.fit(df)
eigenvalues, vectors = fa.get_eigenvalues()

print("\nEigenvalues:")
print(eigenvalues)
print("\nFactor Loadings (Unrotated):\n")
fa_loa = pd.DataFrame(fa.loadings_, index = df.columns)
display(fa_loa)
```

Eigenvalues:
[12.70294466 1.19861812 0.69394774 0.59529366 0.55921525 0.5262102
 0.47423875 0.45801512 0.4063027 0.38595835 0.36433672 0.34838292
 0.3287309 0.31322604 0.31227913 0.28248605 0.27322623 0.26646354
 0.23296411 0.17489164 0.10226818]

Factor Loadings (Unrotated):

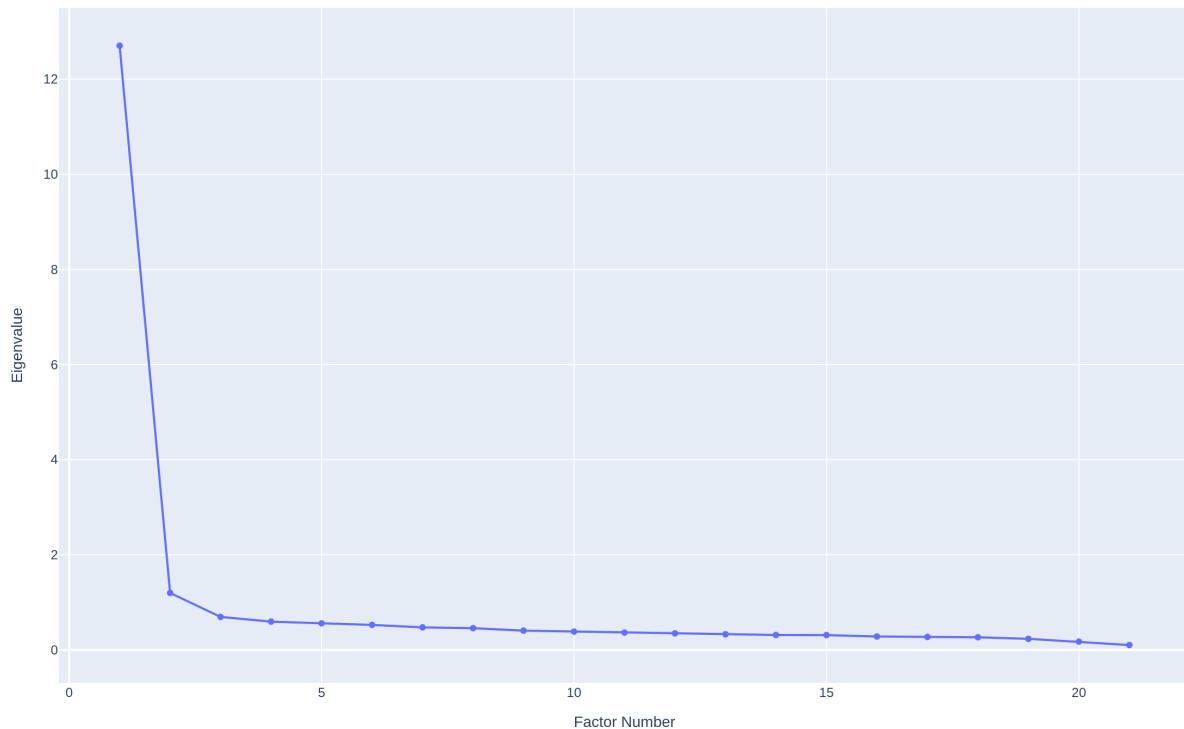
| | 0 | 1 | 2 | 3 | 4 |
|-------------------------------------|-----------|-----------|-----------|-----------|-----------|
| anxiety_level | 0.841936 | -0.117255 | 0.044923 | 0.010691 | -0.003797 |
| self_esteem | -0.836260 | -0.166344 | 0.032087 | -0.037019 | -0.007189 |
| mental_health_history | 0.753948 | -0.124007 | 0.056088 | -0.233917 | -0.182433 |
| depression | 0.836865 | 0.050960 | 0.058918 | -0.038787 | -0.016083 |
| headache | 0.792363 | -0.011939 | -0.023030 | -0.200462 | -0.126740 |
| blood_pressure | 0.496893 | 0.816636 | 0.108016 | 0.114269 | 0.013035 |
| sleep_quality | -0.830279 | 0.144787 | 0.023117 | 0.046734 | -0.049896 |
| breathing_problem | 0.655116 | -0.323514 | 0.088424 | 0.481076 | 0.264720 |
| noise_level | 0.722494 | -0.016348 | 0.230413 | -0.022209 | 0.211857 |
| living_conditions | -0.683458 | 0.099032 | 0.171880 | -0.302926 | 0.566039 |
| safety | -0.791837 | 0.089442 | 0.288754 | -0.093607 | -0.049026 |
| basic_needs | -0.783683 | 0.113665 | 0.203475 | 0.195876 | -0.155195 |
| academic_performance | -0.793299 | 0.146104 | 0.218384 | 0.120637 | -0.094879 |
| study_load | 0.705639 | 0.006698 | 0.338616 | -0.217764 | -0.120927 |
| teacher_student_relationship | -0.813431 | -0.006622 | 0.288672 | 0.043474 | -0.048128 |
| future_career_concerns | 0.852881 | 0.013418 | 0.042605 | 0.030677 | -0.041285 |
| social_support | -0.749100 | -0.529390 | 0.260828 | -0.026674 | -0.095140 |
| peer_pressure | 0.780942 | -0.024505 | 0.272556 | -0.038972 | -0.028831 |
| extracurricular_activities | 0.779552 | 0.037407 | 0.246444 | 0.199695 | -0.043469 |
| bullying | 0.838185 | -0.094335 | 0.069609 | 0.046503 | 0.053969 |
| stress_level | 0.897116 | -0.052925 | 0.030341 | -0.047572 | 0.052015 |

21 rows × 21 columns

```
import plotly.graph_objects as go

#Scree plot
fig = go.Figure(data = go.Scatter(x = list(range(1, len(eigenvalues)+1)), y = eigenvalues))
fig.update_layout(
    title = 'Scree Plot',
    xaxis_title = 'Factor Number',
    yaxis_title = 'Eigenvalue',
    hovermode = 'closest'
)
fig.show()
```

Scree Plot



Descripción del gráfico: El gráfico de sedimentación muestra los valores propios (eigenvalues) de cada factor. Permite identificar cuántos factores retener según el criterio de Kaiser (valores > 1). En este caso, se observan dos factores principales.

Basándose en el criterio de Kaiser (eigenvalues>1), hay 2 factores con eigenvalues mayores que 1. Ahora entrenamos FactorAnalyzer de nuevo con 3 factores y 'varimax' rotation y se imprimen las cargas factoriales.

```
fa_n = FactorAnalyzer(n_factors = 2, rotation = 'varimax', method = 'principal', svd_method = 'auto')
fa_n.fit(df)
#la rotación 'varimax' es ortogonal y minimiza el número de variables que tienen cada factor

print("\nFactor Loadings (Varimax Rotation):\n")
fac_loa = pd.DataFrame(fa_n.loadings_, index = df.columns)
display(fac_loa.loc[:, 0:1])
```

Factor Loadings (Varimax Rotation):

| | 0 | 1 |
|-------------------------------------|-----------|-----------|
| anxiety_level | 0.173826 | 0.138681 |
| self_esteem | -0.146451 | -0.317702 |
| mental_health_history | 0.125892 | 0.116722 |
| depression | 0.166276 | 0.232792 |
| headache | 0.134426 | 0.174248 |
| blood_pressure | 0.031579 | 0.955451 |
| sleep_quality | -0.713471 | -0.115776 |
| breathing_problem | 0.105484 | 0.035615 |
| noise_level | 0.112989 | 0.162687 |
| living_conditions | -0.098067 | -0.116607 |
| safety | -0.147446 | -0.134594 |
| basic_needs | -0.121675 | -0.124339 |
| academic_performance | -0.155365 | -0.105126 |
| study_load | 0.103343 | 0.158062 |
| teacher_student_relationship | -0.158103 | -0.190512 |
| future_career_concerns | 0.146023 | 0.224286 |
| social_support | -0.118249 | -0.692347 |
| peer_pressure | 0.144700 | 0.183612 |
| extracurricular_activities | 0.127609 | 0.212792 |
| bullying | 0.166203 | 0.172710 |
| stress_level | 0.194484 | 0.186219 |

Factor 1, denominado Bienestar personal y social, agrupa principalmente variables como calidad del sueño y apoyo social, las cuales reflejan condiciones de soporte emocional y hábitos de vida que impactan en la salud mental.

Factor 2, denominado Condición fisiológica, está dominado por la variable presión arterial, que representa aspectos más ligados a la respuesta física del organismo frente al estrés.

Obtener las comunidades

Comunalidad = suma de cargas cuadradas para cada variable en todos los factores.

```
from sklearn.decomposition import FactorAnalysis
fa_sklearn = FactorAnalysis(n_components = 2, rotation = 'varimax')
```

```
fa_sklearn.fit(df)

sklearn_communalities = (fa_sklearn.components_.T**2).sum(axis=1)

print("Communalities (from fa_sklearn - Varimax Rotation):")
display(sklearn_communalities)
```

```
Communalities (from fa_sklearn - Varimax Rotation):  
array([0.70345915, 0.70089602, 0.5499376 , 0.68523512, 0.60809088,  
       0.98690298, 0.69533789, 0.43714513, 0.4967947 , 0.44252478,  
       0.62339003, 0.6092427 , 0.63786016, 0.47225303, 0.6446642 ,  
       0.71355084, 0.75068138, 0.5867806 , 0.58733268, 0.69012195,  
       0.80563197])
```

Visualización

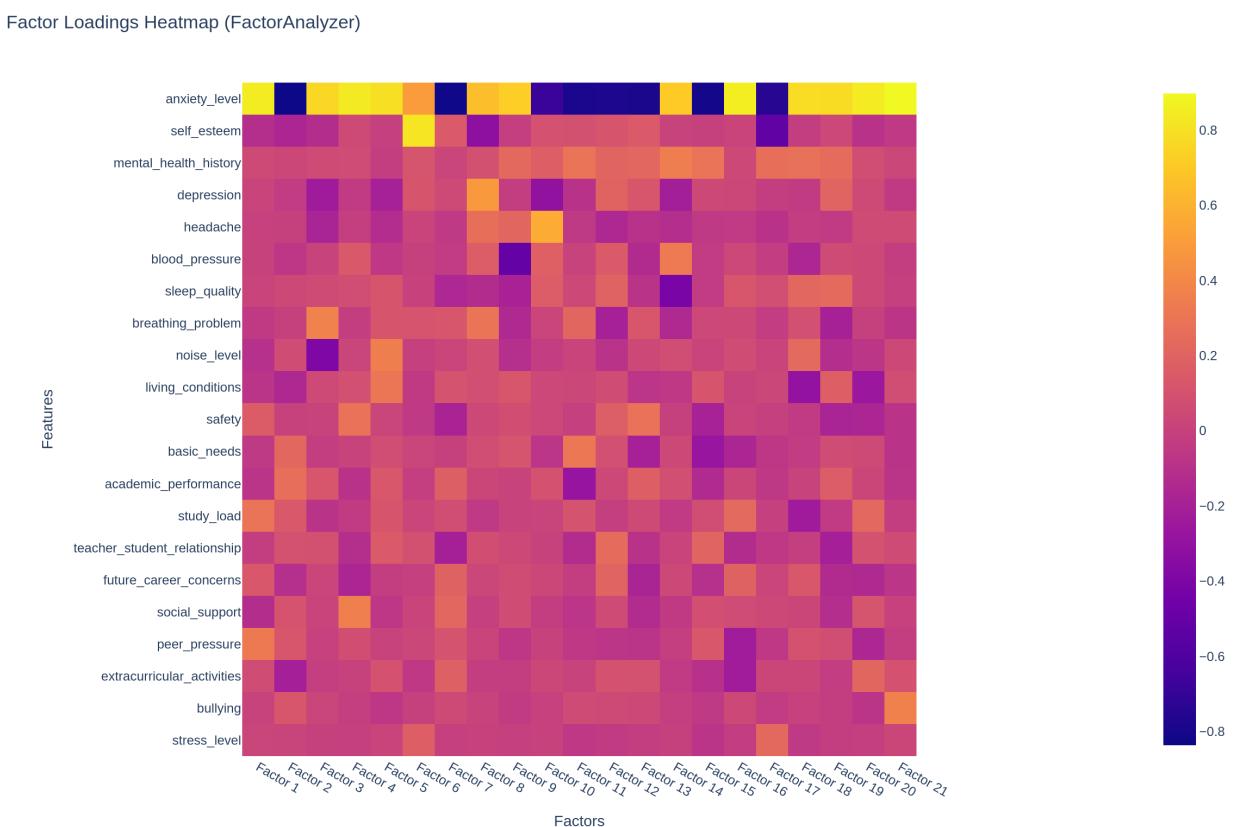
Presentamos mapas de calor de las cargas factoriales usando plotly para visualizar las relaciones entre los parámetros y los factores

```
import plotly.express as px

fa_loadings_df = pd.DataFrame(fa.loadings_, index = df.columns).T

fig = px.imshow(fa_loadings_df,
                 x=[f'Factor {i+1}' for i in range(fa_loadings_df.shape[1])],
                 # y=fa_loadings_df.index,
                 y = df.columns,
                 title='Factor Loadings Heatmap (FactorAnalyzer)',
                 labels={'x': 'Factors', 'y': 'Features'})

fig.show()
```



Descripción del gráfico: Mapa de calor de las cargas factoriales de las variables en cada factor. Permite visualizar qué variables tienen mayor peso en cada dimensión latente, facilitando la interpretación de los factores. Se puede ver que hay una buena separación, sin embargo, en el centro se puede ver como se combinan algunos de los datos que son de diferente cluster.

Análisis Discriminante

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, f1_score, classification_report, confusion_matrix
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDisc
```

```
# ===== 1. Datos =====
X = data.drop('anxiety_level', axis=1)

# Hacer 'anxiety_level' categorica: 1 si superior a la media, 0 si no
mean_anxiety = data['anxiety_level'].mean()
y = (data['anxiety_level'] > mean_anxiety).astype(int)
```

```
# ===== 2. Split estratificado =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Estandarización de data
# Se estandariza después de dividirlo en train y test para que el test no esté sesgado y solo estandariza el train con fit transform

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train) # 'fit_transform' calcula media y desviación estándar
X_train = pd.DataFrame(X_train_s, columns=X_train.columns)

X_test_s = scaler.transform(X_test) # 'transform' usa el mismo scaler que se generó
X_test = pd.DataFrame(X_test_s, columns=X_test.columns)
```

```
# ===== 3. Pipeline + búsqueda de hiperparámetros (LDA) =====
pipe_lda = Pipeline([
    ('scaler', StandardScaler()),
    ('lda', LinearDiscriminantAnalysis())
])
```

```
# Solvers: 'svd' (no admite shrinkage), 'lsqr' y 'eigen' (sí admiten shrinkage)
param_grid = {
    'lda_solver': ['svd', 'lsqr', 'eigen'],
    'lda_shrinkage': [None, 'auto']
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Convierte el 'anxiety_level' a variable categórica basada en la media
```

```

mean_anxiety = y_train.mean()
y_categorical = (y_train > mean_anxiety).astype(int) # 1 si superior a la media, 0
# 0 si inferior o igual a la media

gcv = GridSearchCV(pipe_lda, param_grid, cv=cv, scoring='f1_macro', n_jobs=-1)
gcv.fit(X_train, y_categorical) # Use the categorical target variable

print("Mejor config LDA:", gcv.best_params_)
print("F1_macro CV (mejor):", gcv.best_score_)
# Aquí se imprime el mejor mejor estimador lda_solver

```

Mejor config LDA: {'lda__shrinkage': 'auto', 'lda__solver': 'lsqr'}
F1_macro CV (mejor): 0.7615384360240119

```

# ===== 4. Evaluación en test =====
# Evalúa las métricas de desempeño para el mejor estimador
best_lda = gcv.best_estimator_
y_pred = best_lda.predict(X_test)

print("\nAccuracy test:", accuracy_score(y_test, y_pred))
print("F1_macro test:", f1_score(y_test, y_pred, average='macro'))
print("\nReporte de clasificación:\n", classification_report(y_test, y_pred))
print("Matriz de confusión:\n", confusion_matrix(y_test, y_pred))

```

n\Accuracy test: 0.7772727272727272
F1_macro test: 0.7771576227390181

Reporte de clasificación:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.79 | 0.78 | 0.78 | 113 |
| 1 | 0.77 | 0.78 | 0.77 | 107 |
| accuracy | | | 0.78 | 220 |
| macro avg | 0.78 | 0.78 | 0.78 | 220 |
| weighted avg | 0.78 | 0.78 | 0.78 | 220 |

Matriz de confusión:

[[88 25]
[24 83]]

Clase 0 (baja ansiedad):

- Precisión: 0.79 cuando predice "baja ansiedad", acierta 79 % de las veces.
- Recall: 0.78 detecta correctamente 78 % de los estudiantes con baja ansiedad.
- F1: 0.78.

Clase 1 (alta ansiedad):

- Precisión: 0.77 cuando predice "alta ansiedad", acierta 77 %.
- Recall: 0.78 detecta correctamente 78 % de los casos de alta ansiedad.
- F1: 0.77.

Matriz de confusión

88 → estudiantes de ansiedad baja correctamente clasificados.

83 → estudiantes de ansiedad alta correctamente clasificados.

25 → casos de ansiedad baja mal clasificados como alta.

24 → casos de ansiedad alta mal clasificados como baja.

```
# ===== 5. Interpretación rápida =====
lda_model = best_lda.named_steps['lda']

# Varianza explicada por las componentes discriminantes (si solver =/ 'svd', usar .
if hasattr(lda_model, "explained_variance_ratio_"):
    print("\nVarianza explicada por ejes LD:", lda_model.explained_variance_ratio_)

# Medias por clase en el espacio original:
print("\nMedias por clase (original):\n", lda_model.means_)

# Coeficientes (direcciones discriminantes en problemas binarios; en multiclas es
if hasattr(lda_model, "coef_"):
    print("\nCoeficientes:\n", lda_model.coef_)

print()
# Variables predictoras
features = X.columns.tolist()

# Coeficientes del modelo LDA
coeficientes = lda_model.coef_[0]

# Imprimir como lista
for nombre, valor in zip(features, coeficientes):
    print(f"{nombre}: {valor}")
```

Medias por clase (original):

```
[[ 0.62043368 -0.55350711 -0.61479909 -0.53877028 -0.33695744  0.61948223  
-0.47850406 -0.53529137  0.48863923  0.57577083  0.55167579  0.55108452  
-0.50851299  0.58772423 -0.62125426  0.52476863 -0.56672477 -0.5726765  
-0.60910801 -0.63823422]  
[-0.65821185  0.58721012  0.65223417  0.57157597  0.35747476 -0.65720246  
0.50764014  0.56788522 -0.51839244 -0.61082947 -0.58526729 -0.58464002  
0.53947631 -0.62351072  0.65908239 -0.55672175  0.6012326   0.60754673  
0.64619655  0.67709625]]
```

Coeficientes:

```
[[ -0.57974115  0.2721281   0.380774   -0.0909789  -0.06975679 -0.47645552  
0.21070865  0.35012345 -0.02395052 -0.18840485 -0.15306441  0.00058417  
0.1443725   -0.28385184  0.34477326 -0.10392479  0.17066496  0.3532553  
0.2448081   0.07051087]]
```

self_esteem: -0.5797411484452131
mental_health_history: 0.2721281018019095
depression: 0.38077400129080863
headache: -0.09097889709855372
blood_pressure: -0.06975679068339102
sleep_quality: -0.4764555164120783
breathing_problem: 0.2107086462131734
noise_level: 0.3501234516289575
living_conditions: -0.02395051916501509
safety: -0.18840484765439158
basic_needs: -0.15306441465992823
academic_performance: 0.0005841662596912625
study_load: 0.14437249838950633
teacher_student_relationship: -0.2838518429600942
future_career_concerns: 0.3447732590609164
social_support: -0.10392479397655521
peer_pressure: 0.1706649600294058
extracurricular_activities: 0.35325530203606137
bullying: 0.2448080984484341
stress_level: 0.07051087313558659

Los resultados muestran las medias de las variables por clase y los coeficientes del modelo LDA.

Las medias reflejan que cada grupo (clase 0 y clase 1) presenta tendencias opuestas: mientras una clase tiene valores positivos en ciertas variables, la otra los tiene negativos.

Por otro lado, los coeficientes señalan el peso e importancia de cada variable en la separación de las clases. Por ejemplo, la autoestima (-0.57) y la calidad del sueño (-0.47) influyen fuertemente de manera negativa, mientras que factores como depresión (0.38), actividades extracurriculares (0.35) y nivel de ruido (0.35) contribuyen positivamente a distinguir entre los grupos.

En conjunto, esto sugiere que hay variables que influyen más que otras y algunas de ellas son esenciales para el análisis.

A continuación se muestran las variables más significativas del modelo con magnitud mayor a 0.35.

```
# Filtrar variables con coeficientes de magnitud > 0.35
print("\nVariables más significativas con |coeficiente| > 0.35:\n")
for nombre, valor in zip(features, coeficientes):
    if abs(valor) > 0.35:
        print(f"{nombre}: {valor:.4f}")
```

Variables más significativas con $|coeficiente| > 0.35$:

```
selfEsteem: -0.5797
depression: 0.3808
sleep_quality: -0.4765
noise_level: 0.3501
extracurricular_activities: 0.3533
```

Alternativa para cuando las fronteras no son lineales

```
import numpy as np
from sklearn.model_selection import StratifiedKFold

pipe_qda = Pipeline([
    ("scaler", StandardScaler()),
    ("qda", QuadraticDiscriminantAnalysis(reg_param=0.0)) # 'reg_param>0' añade re
])

# Puedes validar 'reg_param' para estabilizar covarianzas si hay pocas muestras o m
param_grid_qda = {"qda__reg_param": np.linspace(0, 0.5, 6)}
gcv_qda = GridSearchCV(pipe_qda, param_grid_qda, scoring="f1_macro", cv=cv, n_jobs=n_cv)
gcv_qda.fit(X_train, y_train)

print("Mejor QDA:", gcv_qda.best_params_, " | F1_macro CV:", gcv_qda.best_score_)
y_pred_qda = gcv_qda.best_estimator_.predict(X_test)
print("QDA Accuracy test:", accuracy_score(y_test, y_pred_qda))
```

Mejor QDA: {'qda__reg_param': np.float64(0.5)} | F1_macro CV: 0.7713247678688265
QDA Accuracy test: 0.7590909090909090

Visualizar en ejes discriminantes

```
# Proyecta a los primeros ejes LD (útil para ver separabilidad)
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Ajusta un nuevo modelo LDA con un solver que soporte la transformación ('svd' o 'l
lda_visualize = LinearDiscriminantAnalysis(n_components=1, solver='svd') # Usando 'svd'
lda_visualize.fit(X_train, y_train) # Ajustar usando los datos de entrenamiento

X_train_ld = lda_visualize.transform(StandardScaler().fit(X_train).transform(X_train))
plt.figure()

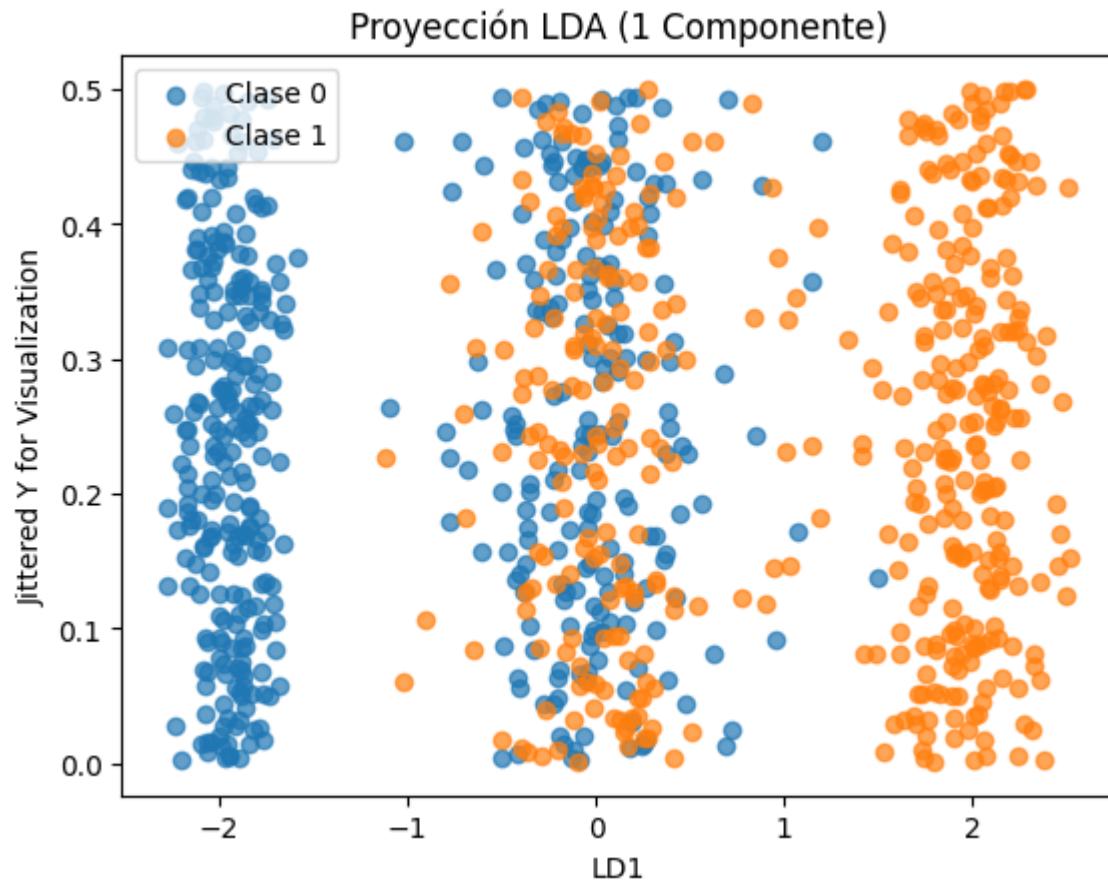
# Para un solo componente, el diagrama de dispersión será una línea 1D, así que se
# Para visualizar mejor la separación, podemos usar un strip plot o simplemente dis
# Aquí dispersaremos en el eje x con una pequeña cantidad de jitter para una mejor

y_jitter = np.random.rand(len(X_train_ld)) * 0.5 # Añade un poco de ruido aleatorio
for k in np.unique(y_train.to_numpy()):
    # Filtra los datos para la clase k y aplica j
    class_indices = y_train.to_numpy() == k
```

```

plt.scatter(X_train_ld[class_indices, 0], y_jitter[class_indices], label=f"Clas
plt.xlabel("LD1"); plt.ylabel("Jittered Y for Visualization"); plt.legend(); plt.ti
plt.show()

```



Descripción del gráfico: Proyección de los estudiantes sobre el primer eje discriminante (LD1). Muestra la separación entre los grupos de baja y alta ansiedad, con cierto solapamiento en los casos cercanos al promedio.

Regresión lineal multiple

```

import statsmodels.api as sm
import plotly.figure_factory as ff
import numpy as np

```

```

#Objetivo y parámetro
X = df.drop('anxiety_level', axis=1)
y = df['anxiety_level']

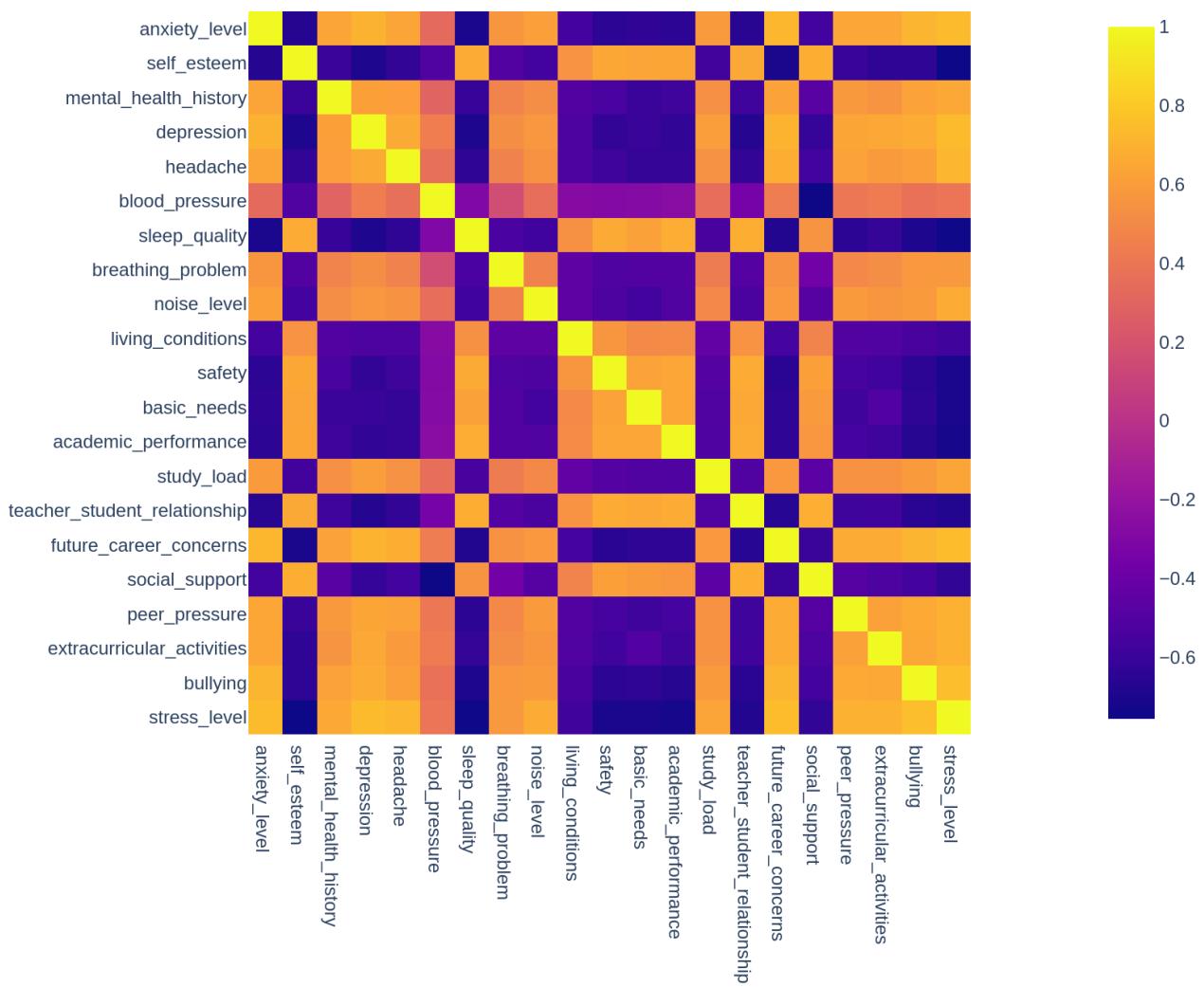
#Constante
#X = sm.add_constant(X)
X.sample(5)

```

| | self_esteem | mental_health_history | depression | headache | blood_pressure | sleep |
|-----|-------------|-----------------------|------------|-----------|----------------|-----------|
| 796 | 1.367114 | | -0.985559 | -0.719293 | -1.070608 | -0.218218 |
| 108 | -0.981739 | | 1.014653 | 1.611258 | 0.349125 | 0.981981 |
| 753 | 1.031564 | | -0.985559 | -1.366668 | -1.070608 | -0.218218 |
| 238 | 0.360463 | | -0.985559 | -0.071918 | -0.360741 | -1.418416 |
| 946 | -1.988391 | | -0.985559 | -0.978243 | 0.349125 | 0.981981 |

```
import plotly.express as px

figcorr = px.imshow(df.corr(), width=800, height=700)
figcorr.show()
```



Descripción del gráfico: Mapa de calor que representa las correlaciones entre todas las variables del conjunto de datos. Se observan relaciones fuertes entre ansiedad, estrés y depresión.

```
import plotly.express as px

# Calculate the correlation matrix
correlation_matrix = df.corr()
```

```

# Select the correlations of 'stress_level', 'anxiety_level', and 'depression' with
selected_correlations = correlation_matrix[['stress_level', 'anxiety_level']]

# Create a heatmap of the selected correlations
fig_selected_corr = px.imshow(selected_correlations, width=1920, height=1080,
                               text_auto=True,
                               aspect="auto",
                               title="Correlation of Stress and Anxiety with Other
color_continuous_scale="Blues") # Changed color scale

# Update layout for black background
fig_selected_corr.update_layout(
    plot_bgcolor='black', # Set plot background color
    paper_bgcolor='black', # Set paper background color
    font_color='white' # Set font color to white for better visibility on black
)

fig_selected_corr.show()

```



Descripción del gráfico: Gráfico que resalta la correlación de stress_level y anxiety_level con el resto de variables. Sirve para identificar factores más relacionados con el bienestar emocional.

```

#Objetivo y parámetro
X = df.drop('anxiety_level', axis=1)
y = df['anxiety_level']

#Constante
#X = sm.add_constant(X)
X.sample(5)

```

| | self_esteem | mental_health_history | depression | headache | blood_pressure | sleep |
|-----|-------------|-----------------------|------------|-----------|----------------|-------|
| 622 | 0.696013 | 1.014653 | -0.071918 | -0.360741 | -1.418416 | - |
| 885 | -0.534339 | 1.014653 | 1.611258 | 0.349125 | 0.981981 | - |
| 674 | -0.086938 | -0.985559 | 0.057558 | 0.349125 | -1.418416 | - |
| 97 | -1.205440 | 1.014653 | 1.352308 | 1.058992 | 0.981981 | - |
| 168 | 0.248612 | 1.014653 | 0.057558 | 0.349125 | -1.418416 | - |

Acercamiento estadístico puro sin usar training y testing:

```

import statsmodels.api as sm

# Re-fit the OLS model to get the equation
# X is the DataFrame of independent variables (excluding 'anxiety_level')
# y is the target variable ('anxiety_level')

# Add a constant term to the independent variables for the intercept
X_with_const = sm.add_constant(X)

# Fit the OLS model
ols_model = sm.OLS(y, X_with_const).fit()

# Get the coefficients and intercept from the OLS model results
intercept = ols_model.params['const']
coefficients = ols_model.params.drop('const') # Drop the constant's coefficient

# Get the names of the independent variables
independent_variables = X.columns

# Construct the regression equation string
equation = f'anxiety_level = {intercept:.4f}'
for i, coef in enumerate(coefficients):
    equation += f' + {coef:.4f} * {independent_variables[i]}'

# Display the regression equation
print("OLS Regression Equation:")
print(equation)

# Display the OLS summary again for completeness
display(ols_model.summary())

```

OLS Regression Equation:

anxiety_level = -0.0000 + -0.0481 * self_esteem + 0.0715 * mental_health_history + 0.0863 * depression + -0.0015 * headache + -0.1026 * blood_pressure + -0.1135 * sleep_quality + 0.0500 * breathing_problem + 0.0865 * noise_level + -0.0527 * living_conditions + -0.0259 * safety + -0.0123 * basic_needs + -0.0192 * academic_performance + 0.0578 * study_load + -0.0230 * teacher_student_relationship + 0.1292 * future_career_concerns + -0.0905 * social_support + 0.0516 * peer_pressure + 0.0511 * extracurricular_activities + 0.1060 * bullying + 0.0334 * stress_level

| OLS Regression Results | | | | | | | |
|----------------------------|---------------------------|-------------------|---------------------|-----------|-------|--------|--------|
| Dep. Variable: | anxiety_level | | R-squared: | 0.690 | | | |
| Model: | OLS | | Adj. R-squared: | 0.685 | | | |
| Method: | Least Squares | | F-statistic: | 120.2 | | | |
| Date: | Mon, 27 Oct 2025 | | Prob (F-statistic): | 9.88e-258 | | | |
| Time: | 04:57:26 | | Log-Likelihood: | -916.18 | | | |
| No. Observations: | 1100 | | AIC: | 1874. | | | |
| Df Residuals: | 1079 | | BIC: | 1979. | | | |
| Df Model: | 20 | | | | | | |
| Covariance Type: | nonrobust | | | | | | |
| | | coef | std err | t | P> t | [0.025 | 0.975] |
| | const | -1.683e-16 | 0.017 | -9.93e-15 | 1.000 | -0.033 | 0.033 |
| | selfEsteem | -0.0481 | 0.031 | -1.560 | 0.119 | -0.109 | 0.012 |
| | mentalHealthHistory | 0.0715 | 0.025 | 2.844 | 0.005 | 0.022 | 0.121 |
| | depression | 0.0863 | 0.030 | 2.901 | 0.004 | 0.028 | 0.145 |
| | headache | -0.0015 | 0.027 | -0.053 | 0.957 | -0.055 | 0.052 |
| | bloodPressure | -0.1026 | 0.032 | -3.164 | 0.002 | -0.166 | -0.039 |
| | sleepQuality | -0.1135 | 0.030 | -3.808 | 0.000 | -0.172 | -0.055 |
| | breathingProblem | 0.0500 | 0.023 | 2.215 | 0.027 | 0.006 | 0.094 |
| | noiseLevel | 0.0865 | 0.024 | 3.595 | 0.000 | 0.039 | 0.134 |
| | livingConditions | -0.0527 | 0.023 | -2.326 | 0.020 | -0.097 | -0.008 |
| | safety | -0.0259 | 0.028 | -0.910 | 0.363 | -0.082 | 0.030 |
| | basicNeeds | -0.0123 | 0.028 | -0.442 | 0.659 | -0.067 | 0.042 |
| | academicPerformance | -0.0192 | 0.028 | -0.681 | 0.496 | -0.074 | 0.036 |
| | studyLoad | 0.0578 | 0.023 | 2.465 | 0.014 | 0.012 | 0.104 |
| teacherStudentRelationship | | -0.0230 | 0.030 | -0.757 | 0.449 | -0.083 | 0.037 |
| | futureCareerConcerns | 0.1292 | 0.031 | 4.155 | 0.000 | 0.068 | 0.190 |
| | socialSupport | -0.0905 | 0.041 | -2.230 | 0.026 | -0.170 | -0.011 |
| | peerPressure | 0.0516 | 0.027 | 1.882 | 0.060 | -0.002 | 0.105 |
| | extracurricularActivities | 0.0511 | 0.027 | 1.895 | 0.058 | -0.002 | 0.104 |
| | bullying | 0.1060 | 0.030 | 3.493 | 0.000 | 0.046 | 0.165 |
| | stressLevel | 0.0334 | 0.038 | 0.890 | 0.374 | -0.040 | 0.107 |
| Omnibus: | 51.877 | Durbin-Watson: | 2.068 | | | | |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 146.235 | | | | |

| | | | |
|------------------|--------|------------------|----------|
| Skew: | -0.166 | Prob(JB): | 1.76e-32 |
| Kurtosis: | 4.755 | Cond. No. | 10.8 |

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Con los datos sin entrenamiento ni prueba obtuvimos los siguientes resultados:

- R²: 0.690
 - El modelo explica aproximadamente el 69 % de la variabilidad de la variable dependiente.
 - Adj. R²: 0.685
 - Las variables incluidas son relevantes y no hay sobreajuste importante.
 - F-statistic: 120.2
 - El modelo es altamente significativo
 - Prob (F-statistic): 9.88e-258

Coeficientes que son estadísticamente significativos y tienen mayor importancia para la predicción de 'anxiety level':

- blood_pressure : -0.1026
 - sleep_quality : -0.1135
 - future_carreer_concerns : 0.1292
 - social_support : -0.905
 - depression : 0.0863
 - mental health history : 0.0715

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.api as sm
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Redefine X and y from the original data before splitting
X = data.drop('anxiety_level', axis=1)
y = data['anxiety_level']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Scale the data *after* splitting
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert scaled arrays back to DataFrames with original indices
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train.columns, index=X_train.index)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_test.columns, index=X_test.index)

# Add a constant term to the independent variables for the intercept for both training sets
X_train_with_const = sm.add_constant(X_train_scaled)
X_test_with_const = sm.add_constant(X_test_scaled)

# Train the OLS model on the training data
ols_model_split = sm.OLS(y_train, X_train_with_const).fit()

# Display the summary of the OLS model on the training data
display(ols_model_split.summary())

# Evaluate the model on the test set
# Make predictions on the test set
y_pred_ols = ols_model_split.predict(X_test_with_const)

# Evaluate the model using R-squared on the test set
r2_ols_test = r2_score(y_test, y_pred_ols)

print(f"OLS R-squared on the test set: {r2_ols_test:.4f}")

# Evaluate the model using R-squared on the training set
y_train_pred_ols = ols_model_split.predict(X_train_with_const)
r2_ols_train = r2_score(y_train, y_train_pred_ols)

print(f"OLS R-squared on the training set: {r2_ols_train:.4f}")
```

| OLS Regression Results | | | | | | | |
|-----------------------------------|----------------------------------|---------|----------------------------|-----------|-------|--------|--------|
| Dep. Variable: | anxiety_level | | R-squared: | 0.686 | | | |
| Model: | OLS | | Adj. R-squared: | 0.679 | | | |
| Method: | Least Squares | | F-statistic: | 93.83 | | | |
| Date: | Mon, 27 Oct 2025 | | Prob (F-statistic): | 4.41e-200 | | | |
| Time: | 04:57:27 | | Log-Likelihood: | -2332.0 | | | |
| No. Observations: | 880 | | AIC: | 4706. | | | |
| Df Residuals: | 859 | | BIC: | 4806. | | | |
| Df Model: | 20 | | | | | | |
| Covariance Type: | nonrobust | | | | | | |
| | | coef | std err | t | P> t | [0.025 | 0.975] |
| | const | 11.1182 | 0.117 | 95.147 | 0.000 | 10.889 | 11.348 |
| | selfEsteem | -0.3289 | 0.216 | -1.522 | 0.128 | -0.753 | 0.095 |
| | mentalHealthHistory | 0.3419 | 0.174 | 1.967 | 0.049 | 0.001 | 0.683 |
| | depression | 0.6906 | 0.211 | 3.267 | 0.001 | 0.276 | 1.106 |
| | headache | -0.0639 | 0.190 | -0.337 | 0.736 | -0.436 | 0.309 |
| | bloodPressure | -0.5731 | 0.222 | -2.579 | 0.010 | -1.009 | -0.137 |
| | sleepQuality | -0.8137 | 0.206 | -3.942 | 0.000 | -1.219 | -0.409 |
| | breathingProblem | 0.4347 | 0.156 | 2.783 | 0.005 | 0.128 | 0.741 |
| | noiseLevel | 0.6198 | 0.169 | 3.662 | 0.000 | 0.288 | 0.952 |
| | livingConditions | -0.2752 | 0.156 | -1.766 | 0.078 | -0.581 | 0.031 |
| | safety | -0.1110 | 0.202 | -0.551 | 0.582 | -0.507 | 0.285 |
| | basicNeeds | -0.1121 | 0.192 | -0.584 | 0.560 | -0.489 | 0.265 |
| | academicPerformance | -0.0669 | 0.192 | -0.348 | 0.728 | -0.444 | 0.310 |
| | studyLoad | 0.4385 | 0.162 | 2.712 | 0.007 | 0.121 | 0.756 |
| teacherStudentRelationship | | -0.2334 | 0.212 | -1.099 | 0.272 | -0.650 | 0.184 |
| | futureCareerConcerns | 0.7450 | 0.214 | 3.479 | 0.001 | 0.325 | 1.165 |
| | socialSupport | -0.4613 | 0.280 | -1.649 | 0.100 | -1.010 | 0.088 |
| | peerPressure | 0.1390 | 0.192 | 0.725 | 0.468 | -0.237 | 0.515 |
| | extracurricularActivities | 0.3159 | 0.191 | 1.650 | 0.099 | -0.060 | 0.692 |
| | bullying | 0.8348 | 0.209 | 3.986 | 0.000 | 0.424 | 1.246 |
| | stressLevel | -0.1783 | 0.256 | -0.695 | 0.487 | -0.681 | 0.325 |
| Omnibus: | 36.994 | | Durbin-Watson: | 2.061 | | | |
| Prob(Omnibus): | 0.000 | | Jarque-Bera (JB): | 100.978 | | | |
| Skew: | -0.106 | | Prob(JB): | 1.18e-22 | | | |
| Kurtosis: | 4.646 | | Cond. No. | 10.8 | | | |

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

OLS R-squared on the test set: 0.6942

OLS R-squared on the training set: 0.6860

Para el modelo con datos de entrenamiento obtuvimos los siguientes valores:

- R^2 : 0.686

- La capacidad explicativa se mantiene casi igual. El modelo sigue generalizando bien.

- Adj. R^2 : 0.679

- F-statistic: 93.83

- El modelo sigue siendo altamente significativo

- Prob (F-statistic): 4.41e-200

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Evaluate the model on the training set
y_train_pred = ols_model_split.predict(X_train_with_const)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
mae_train = mean_absolute_error(y_train, y_train_pred)

print("Evaluation on Training Set:")
print(f"Mean Squared Error (MSE): {mse_train:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_train:.4f}")
print(f"Mean Absolute Error (MAE): {mae_train:.4f}")
print("-" * 30)

# Evaluate the model on the test set
y_test_pred = ols_model_split.predict(X_test_with_const)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
mae_test = mean_absolute_error(y_test, y_test_pred)

print("Evaluation on Test Set:")
print(f"Mean Squared Error (MSE): {mse_test:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_test:.4f}")
print(f"Mean Absolute Error (MAE): {mae_test:.4f})
```

```
Evaluation on Training Set:  
Mean Squared Error (MSE): 11.7293  
Root Mean Squared Error (RMSE): 3.4248  
Mean Absolute Error (MAE): 2.5755
```

```
-----  
Evaluation on Test Set:  
Mean Squared Error (MSE): 11.4595  
Root Mean Squared Error (RMSE): 3.3852  
Mean Absolute Error (MAE): 2.4371
```

Error Cuadrático Medio (MSE): 11.4595 Similar al MSE de entrenamiento. Es ligeramente menor que el MSE de entrenamiento (11.4595 vs 11.7293), lo que sugiere que el modelo generaliza bien los datos no y no hay sobreajuste.

Raíz Cuadrada del Error Cuadrático (RMSE): 3.3852 La raíz cuadrada del MSE de prueba nos indica que el error promedio en la predicción del nivel de ansiedad en los datos de prueba es de 3.3852 unidades. Este valor es muy cercano al RMSE de entrenamiento, lo que refuerza la idea de una buena generalización.

Error Medio Absoluto (MAE): 2.4371 El error promedio absoluto en las predicciones del conjunto de prueba también es ligeramente menor que el MAE de entrenamiento (2.4371 vs 2.5755), lo que nuevamente nos sugiere una buena capacidad del modelo para predecir.

Conglomerados

Preparación de datos

```
import numpy as np  
  
# Transformamos el formato de los datos para el agrupamiento  
df_cluster=df.copy()  
data_for_clustering = df_cluster.values  
data_for_clustering = data_for_clustering.astype(float)
```

```
data_for_clustering
```

```
array([[ 0.48020782,  0.24861241, -0.98555881, ...,  0.16424895,  
       -0.40337716,  0.00442758],  
       [ 0.64374608, -1.09358954,  1.0146528 , ...,  1.57576339,  
       1.5570715 ,  1.22201077],  
       [ 0.15313129,  0.02491208,  1.0146528 , ..., -0.54150826,  
       -0.40337716,  0.00442758],  
       ...,  
       [-1.15517484,  0.91971338, -0.98555881, ..., -0.54150826,  
       -1.05686005, -1.21315562],  
       [ 1.62497568, -1.98839084,  1.0146528 , ...,  0.87000617,  
       0.90358861,  1.22201077],  
       [ 1.13436088, -1.31728986,  1.0146528 , ..., -1.24726548,  
       0.90358861,  1.22201077]])
```

Determinar el número de clusters

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

inertia = []
silhouette_scores = []

for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
    kmeans.fit(data_for_clustering)
    inertia.append(kmeans.inertia_)
    if n_clusters > 1:
        silhouette_avg = silhouette_score(data_for_clustering, kmeans.labels_)
        silhouette_scores.append(silhouette_avg)

import matplotlib.pyplot as plt

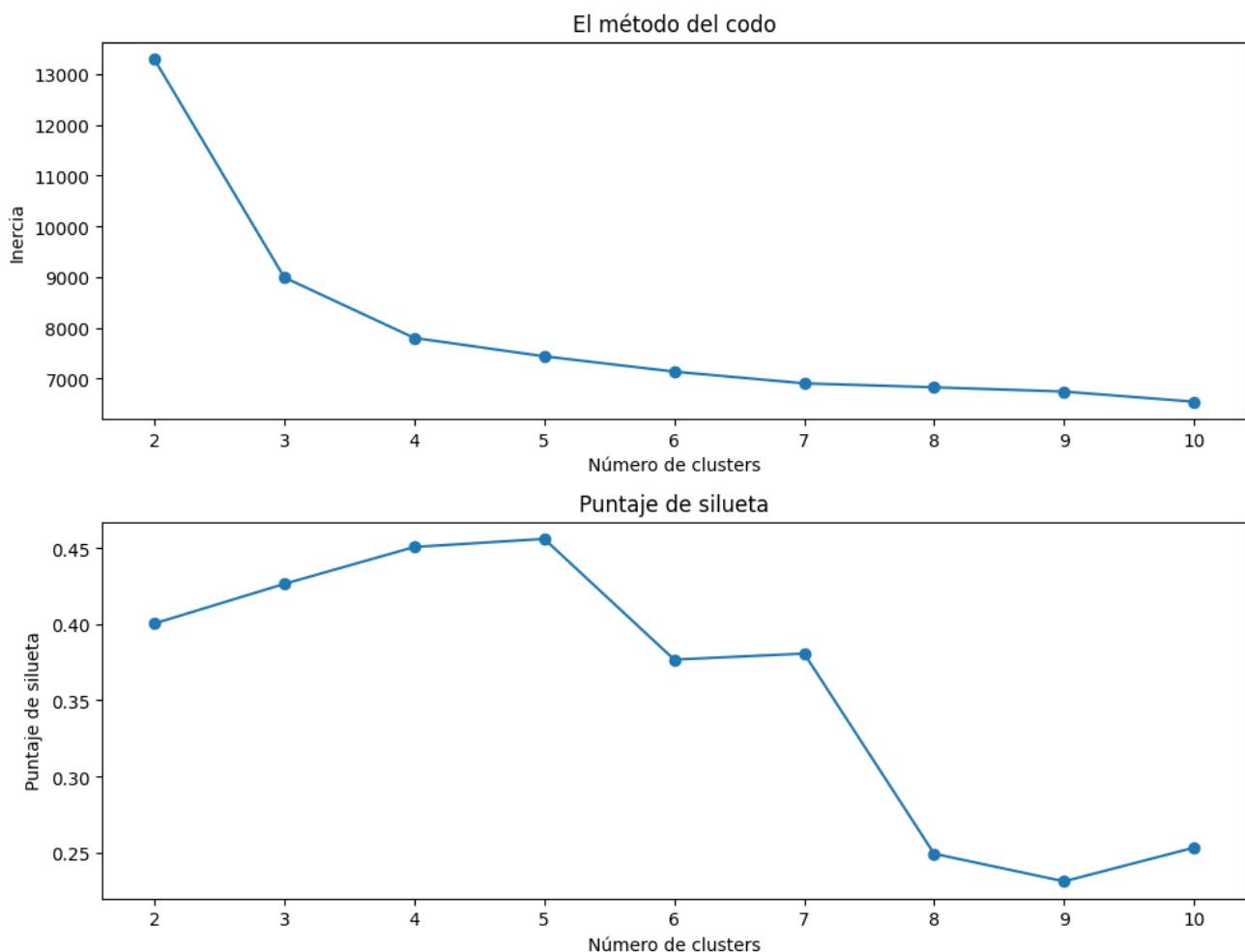
fig, axes = plt.subplots(2, 1, figsize=(10, 8))

# Elbow Method
axes[0].plot(range(2, 11), inertia, marker='o')
axes[0].set_xlabel('Número de clusters')
axes[0].set_ylabel('Inercia')
axes[0].set_title('El método del codo')

# Silhouette Score
axes[1].plot(range(2, 11), silhouette_scores, marker='o')
axes[1].set_xlabel('Número de clusters')
axes[1].set_ylabel('Puntaje de silueta')
axes[1].set_title('Puntaje de silueta')

fig.suptitle('El método del codo y puntaje de silueta para el óptimo número de clusters')
plt.tight_layout()
plt.show()
```

El método del codo y puntaje de silueta para el óptimo número de clusters



La grafica del Elbow Method muestra una especie de codo y donde se encuentra este codo es la cantidad de clusters que se recomiendan usar con los datos que se tienen, a su vez la gráfica de Silhouette Score le da una puntuación a cada cantidad de clusters, entre mayor puntuación de mejor manera se lleva a cabo la agrupación.

Algoritmo de clustering

```
# Calculamos los grupos
optimal_clusters = 4 # Se clasifican en 4 grupos
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42, n_init=10)
clusters = kmeans.fit_predict(data_for_clustering)
df_cluster['cluster'] = clusters
display(df_cluster)
```

| | anxiety_level | selfEsteem | mentalHealthHistory | depression | headache | blood |
|------|---------------|------------|---------------------|------------|-----------|-----------|
| 0 | 0.480208 | 0.248612 | | -0.985559 | -0.201393 | -0.360741 |
| 1 | 0.643746 | -1.093590 | | 1.014653 | 0.316508 | 1.768859 |
| 2 | 0.153131 | 0.024912 | | 1.014653 | 0.187033 | -0.360741 |
| 3 | 0.807284 | -0.646189 | | 1.014653 | 0.316508 | 1.058992 |
| 4 | 0.807284 | 1.143414 | | -0.985559 | -0.719293 | -0.360741 |
| ... | ... | ... | ... | ... | ... | ... |
| 1095 | -0.010407 | -0.086938 | | -0.985559 | 0.187033 | 0.349125 |
| 1096 | -0.337484 | -0.646189 | | -0.985559 | -0.589818 | -1.780475 |
| 1097 | -1.155175 | 0.919713 | | -0.985559 | -1.237193 | -1.070608 |
| 1098 | 1.624976 | -1.988391 | | 1.014653 | 0.834408 | 1.768859 |
| 1099 | 1.134361 | -1.317290 | | 1.014653 | 0.316508 | 0.349125 |

1100 rows × 22 columns

Tabla con todos los datos escalados y con una nueva columna que son los clusters al que se asignó cada estudiante.

```
df_cluster['cluster'].value_counts()
```

| | count |
|----------------|-------|
| cluster | |
| 3 | 307 |
| 2 | 301 |
| 0 | 300 |
| 1 | 192 |

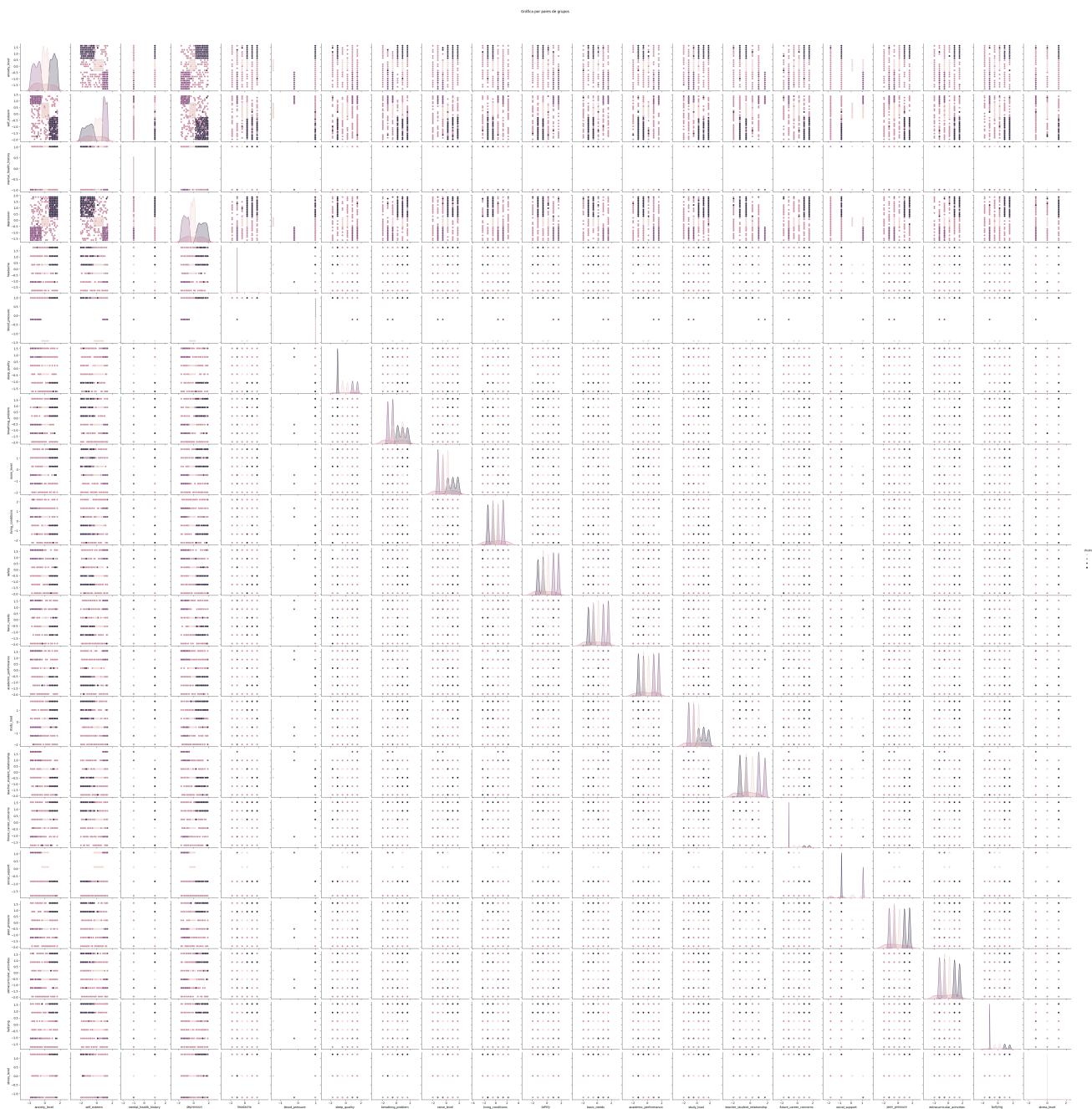
dtype: int64

La cantidad de estudiantes que hay en cada clusters.

Visualización de clusters

```
import seaborn as sns

# Creación de un gráfica por pares
sns.pairplot(df_cluster, hue='cluster', diag_kind='kde')
plt.suptitle('Gráfica por pares de grupos', y=1.02)
plt.show()
```



Una gráfica por pares que además de mostrar la como se grafican las variables entre si también le da un color a cada punto a partir de cómo fueron clasificados.

```
# Usamos los datos no transformados para tener los valores originales, y se agrega 1
df_cluster_original_scale = data.copy()
df_cluster_original_scale['cluster'] = df_cluster['cluster']

display(df_cluster_original_scale.head())
```

| | anxiety_level | selfEsteem | mentalHealthHistory | depression | headache | bloodPressure |
|---|---------------|------------|---------------------|------------|----------|---------------|
| 0 | 14 | 20 | | 0 | 11 | 2 |
| 1 | 15 | 8 | | 1 | 15 | 5 |
| 2 | 12 | 18 | | 1 | 14 | 2 |
| 3 | 16 | 12 | | 1 | 15 | 4 |
| 4 | 16 | 28 | | 0 | 7 | 2 |

5 rows × 22 columns

Tabla con todos los datos sin escalar y con la columna 'cluster' que los identifica a cual fue asignó cada estudiante, para interpretar mejor los datos.

Interpretación de resultados

```
display(df_cluster_original_scale.describe().transpose())
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|-------------------------------------|--------|-----------|----------|-----|------|------|------|------|
| anxiety_level | 1100.0 | 11.063636 | 6.117558 | 0.0 | 6.0 | 11.0 | 16.0 | 21.0 |
| self_esteem | 1100.0 | 17.777273 | 8.944599 | 0.0 | 11.0 | 19.0 | 26.0 | 30.0 |
| mental_health_history | 1100.0 | 0.492727 | 0.500175 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| depression | 1100.0 | 12.555455 | 7.727008 | 0.0 | 6.0 | 12.0 | 19.0 | 27.0 |
| headache | 1100.0 | 2.508182 | 1.409356 | 0.0 | 1.0 | 3.0 | 3.0 | 5.0 |
| blood_pressure | 1100.0 | 2.181818 | 0.833575 | 1.0 | 1.0 | 2.0 | 3.0 | 3.0 |
| sleep_quality | 1100.0 | 2.660000 | 1.548383 | 0.0 | 1.0 | 2.5 | 4.0 | 5.0 |
| breathing_problem | 1100.0 | 2.753636 | 1.400713 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 |
| noise_level | 1100.0 | 2.649091 | 1.328127 | 0.0 | 2.0 | 3.0 | 3.0 | 5.0 |
| living_conditions | 1100.0 | 2.518182 | 1.119208 | 0.0 | 2.0 | 2.0 | 3.0 | 5.0 |
| safety | 1100.0 | 2.737273 | 1.406171 | 0.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| basic_needs | 1100.0 | 2.772727 | 1.433761 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 |
| academic_performance | 1100.0 | 2.772727 | 1.414594 | 0.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| study_load | 1100.0 | 2.621818 | 1.315781 | 0.0 | 2.0 | 2.0 | 3.0 | 5.0 |
| teacher_student_relationship | 1100.0 | 2.648182 | 1.384579 | 0.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| future_career_concerns | 1100.0 | 2.649091 | 1.529375 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| social_support | 1100.0 | 1.881818 | 1.047826 | 0.0 | 1.0 | 2.0 | 3.0 | 3.0 |
| peer_pressure | 1100.0 | 2.734545 | 1.425265 | 0.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| extracurricular_activities | 1100.0 | 2.767273 | 1.417562 | 0.0 | 2.0 | 2.5 | 4.0 | 5.0 |
| bullying | 1100.0 | 2.617273 | 1.530958 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| stress_level | 1100.0 | 0.996364 | 0.821673 | 0.0 | 0.0 | 1.0 | 2.0 | 2.0 |
| cluster | 1100.0 | 1.559091 | 1.162486 | 0.0 | 0.0 | 2.0 | 3.0 | 3.0 |

```
# Interpretación basada en el resumen de los grupo con los datos en escala original
print("Interpretando los grupo basados en el resumen estadístico de los datos en es")

# Analizar Conglomerado 0
print("\nGrupo 0:")
display(df_cluster_original_scale[df_cluster_original_scale['cluster'] == 0].describe())
print("Estudiantes con niveles promedio de ansiedad, depresión y autoestima. Salud media")

# Analizar Conglomerado 1
print("\nGrupo 1:")
display(df_cluster_original_scale[df_cluster_original_scale['cluster'] == 1].describe())
print("Estudiantes con ansiedad y depresión moderadas, menor autoestima y apoyo social")

# Analizar Conglomerado 2
print("\nGrupo 2:")
display(df_cluster_original_scale[df_cluster_original_scale['cluster'] == 2].describe())
print("Estudiantes con la mejor salud mental: baja ansiedad y depresión, alta autoestima y apoyo social")

# Analizar Conglomerado 3
print("\nGrupo 3:")
```

```

display(df_cluster_original_scale[df_cluster_original_scale['cluster'] == 3].describe())
print("Estudiantes con mayor ansiedad y depresión, baja autoestima y apoyo social.")

# Crear un resumen general en forma de tabla
resumen_clusters = pd.DataFrame({
    "Cluster": ["0", "1", "2", "3"],
    "Descripción breve": [
        "Indicadores promedio de salud mental, vida y estudio.", 
        "Ansiedad y depresión moderadas, menor autoestima y apoyo social.", 
        "Mejor salud mental y social, condiciones positivas y alto rendimiento.", 
        "Peor salud mental y social, condiciones difíciles y bajo rendimiento."
    ]
})

print("\nResumen General en Tabla:")
display(resumen_clusters)

```

Interpretando los grupos basados en el resumen estadístico de los datos en escala original:

\Grupo 0:

| | | count | mean | std | min | 25% | 50% | 75% | max |
|-------------------------------------|------------------------------|-------|-----------|----------|------|------|------|------|------|
| | anxiety_level | 300.0 | 11.410000 | 1.708302 | 9.0 | 10.0 | 11.0 | 13.0 | 14.0 |
| | selfEsteem | 300.0 | 19.980000 | 3.152681 | 15.0 | 17.0 | 20.0 | 23.0 | 25.0 |
| | mental_health_history | 300.0 | 0.513333 | 0.500657 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| | depression | 300.0 | 11.640000 | 1.673240 | 9.0 | 10.0 | 12.0 | 13.0 | 14.0 |
| | headache | 300.0 | 2.483333 | 0.500557 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| | blood_pressure | 300.0 | 1.000000 | 0.000000 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | sleep_quality | 300.0 | 2.456667 | 0.498951 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| | breathing_problem | 300.0 | 3.026667 | 1.001315 | 2.0 | 2.0 | 4.0 | 4.0 | 4.0 |
| | noise_level | 300.0 | 2.523333 | 0.500290 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
| | living_conditions | 300.0 | 2.483333 | 0.500557 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| | safety | 300.0 | 2.476667 | 0.500290 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| | basic_needs | 300.0 | 2.513333 | 0.500657 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
| | academic_performance | 300.0 | 2.473333 | 0.500123 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| | study_load | 300.0 | 2.486667 | 0.500657 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| teacher_student_relationship | | 300.0 | 2.483333 | 0.500557 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| future_career_concerns | | 300.0 | 2.460000 | 0.499230 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| social_support | | 300.0 | 2.526667 | 0.500123 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
| peer_pressure | | 300.0 | 2.466667 | 0.499721 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| extracurricular_activities | | 300.0 | 2.483333 | 0.500557 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 |
| bullying | | 300.0 | 2.560000 | 0.497216 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
| stress_level | | 300.0 | 1.000000 | 0.000000 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| cluster | | 300.0 | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Estudiantes con niveles promedio de ansiedad, depresión y autoestima. Salud física y condiciones de vida regulares. Rendimiento y apoyo social moderados. Estrés estable en 1.

\Grupo 1:

| | count | mean | std | min | 25% | 50% | 75% | max |
|-------------------------------------|-------|-----------|----------|-----|-----|------|------|------|
| anxiety_level | 192.0 | 10.244792 | 6.325174 | 0.0 | 5.0 | 10.0 | 16.0 | 21.0 |
| selfEsteem | 192.0 | 15.083333 | 9.049216 | 0.0 | 7.0 | 15.0 | 23.0 | 30.0 |
| mental_health_history | 192.0 | 0.421875 | 0.495150 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| depression | 192.0 | 13.052083 | 8.169500 | 0.0 | 6.0 | 12.0 | 20.0 | 27.0 |
| headache | 192.0 | 2.630208 | 1.746717 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| blood_pressure | 192.0 | 3.000000 | 0.000000 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| sleep_quality | 192.0 | 2.750000 | 1.781228 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| breathing_problem | 192.0 | 2.380208 | 1.800589 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| noise_level | 192.0 | 2.536458 | 1.754194 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| living_conditions | 192.0 | 2.578125 | 1.660802 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| safety | 192.0 | 2.291667 | 1.617423 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| basic_needs | 192.0 | 2.427083 | 1.756035 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| academic_performance | 192.0 | 2.583333 | 1.647887 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| study_load | 192.0 | 2.473958 | 1.769239 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| teacher_student_relationship | 192.0 | 1.921875 | 1.361067 | 0.0 | 1.0 | 2.0 | 3.0 | 4.0 |
| future_career_concerns | 192.0 | 2.604167 | 1.659364 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| social_support | 192.0 | 0.552083 | 0.498580 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| peer_pressure | 192.0 | 2.234375 | 1.669644 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| extracurricular_activities | 192.0 | 2.531250 | 1.724193 | 0.0 | 1.0 | 3.0 | 4.0 | 5.0 |
| bullying | 192.0 | 2.307292 | 1.743905 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| stress_level | 192.0 | 0.979167 | 0.849751 | 0.0 | 0.0 | 1.0 | 2.0 | 2.0 |
| cluster | 192.0 | 1.000000 | 0.000000 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

Estudiantes con ansiedad y depresión moderadas, menor autoestima y apoyo social. Más dolores de cabeza y presión arterial alta. Circunstancias de vida variadas. Estrés cercano a 1 con más variación.

\Grupo 2:

| | count | mean | std | min | 25% | 50% | 75% | max |
|-------------------------------------|-------|-----------|----------|------|------|------|------|------|
| anxiety_level | 301.0 | 4.182724 | 2.586471 | 0.0 | 2.0 | 4.0 | 7.0 | 8.0 |
| self_esteem | 301.0 | 27.438538 | 1.730234 | 25.0 | 26.0 | 27.0 | 29.0 | 30.0 |
| mental_health_history | 301.0 | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| depression | 301.0 | 4.172757 | 2.936561 | 0.0 | 2.0 | 4.0 | 6.0 | 27.0 |
| headache | 301.0 | 1.000000 | 0.000000 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| blood_pressure | 301.0 | 2.003322 | 0.057639 | 2.0 | 2.0 | 2.0 | 2.0 | 3.0 |
| sleep_quality | 301.0 | 4.488372 | 0.500697 | 4.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| breathing_problem | 301.0 | 1.514950 | 0.500609 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 |
| noise_level | 301.0 | 1.468439 | 0.499834 | 1.0 | 1.0 | 1.0 | 2.0 | 2.0 |
| living_conditions | 301.0 | 3.531561 | 0.506459 | 3.0 | 3.0 | 4.0 | 4.0 | 5.0 |
| safety | 301.0 | 4.491694 | 0.507376 | 3.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| basic_needs | 301.0 | 4.514950 | 0.507224 | 3.0 | 4.0 | 5.0 | 5.0 | 5.0 |
| academic_performance | 301.0 | 4.504983 | 0.500808 | 4.0 | 4.0 | 5.0 | 5.0 | 5.0 |
| study_load | 301.0 | 1.485050 | 0.507224 | 0.0 | 1.0 | 1.0 | 2.0 | 2.0 |
| teacher_student_relationship | 301.0 | 4.465116 | 0.499612 | 4.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| future_career_concerns | 301.0 | 1.003322 | 0.057639 | 1.0 | 1.0 | 1.0 | 1.0 | 2.0 |
| social_support | 301.0 | 2.993355 | 0.115278 | 1.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| peer_pressure | 301.0 | 1.524917 | 0.500210 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 |
| extracurricular_activities | 301.0 | 1.501661 | 0.539287 | 1.0 | 1.0 | 1.0 | 2.0 | 5.0 |
| bullying | 301.0 | 1.006645 | 0.115278 | 1.0 | 1.0 | 1.0 | 1.0 | 3.0 |
| stress_level | 301.0 | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| cluster | 301.0 | 2.000000 | 0.000000 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |

Estudiantes con la mejor salud mental: baja ansiedad y depresión, alta autoestima y apoyo social. Buenas condiciones de vida y rendimiento académico alto. Estrés en 0.

\Grupo 3:

| | | count | mean | std | min | 25% | 50% | 75% | max |
|-------------------------------------|------------------------------|-------|-----------|----------|------|------|------|------|------|
| | anxiety_level | 307.0 | 17.983713 | 2.207325 | 5.0 | 16.0 | 18.0 | 20.0 | 21.0 |
| | self_esteem | 307.0 | 7.837134 | 4.829734 | 0.0 | 4.0 | 8.0 | 12.0 | 29.0 |
| | mental_health_history | 307.0 | 1.000000 | 0.000000 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | depression | 307.0 | 21.358306 | 3.707697 | 10.0 | 18.0 | 22.0 | 25.0 | 27.0 |
| | headache | 307.0 | 3.934853 | 0.833720 | 1.0 | 3.0 | 4.0 | 5.0 | 5.0 |
| | blood_pressure | 307.0 | 3.000000 | 0.000000 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| | sleep_quality | 307.0 | 1.009772 | 0.273985 | 0.0 | 1.0 | 1.0 | 1.0 | 5.0 |
| | breathing_problem | 307.0 | 3.934853 | 0.853093 | 1.0 | 3.0 | 4.0 | 5.0 | 5.0 |
| | noise_level | 307.0 | 4.000000 | 0.836269 | 2.0 | 3.0 | 4.0 | 5.0 | 5.0 |
| | living_conditions | 307.0 | 1.521173 | 0.519591 | 0.0 | 1.0 | 2.0 | 2.0 | 3.0 |
| | safety | 307.0 | 1.550489 | 0.582892 | 0.0 | 1.0 | 2.0 | 2.0 | 5.0 |
| | basic_needs | 307.0 | 1.534202 | 0.543503 | 0.0 | 1.0 | 2.0 | 2.0 | 4.0 |
| | academic_performance | 307.0 | 1.485342 | 0.526066 | 0.0 | 1.0 | 1.0 | 2.0 | 3.0 |
| | study_load | 307.0 | 3.960912 | 0.815557 | 2.0 | 3.0 | 4.0 | 5.0 | 5.0 |
| teacher_student_relationship | | 307.0 | 1.482085 | 0.519714 | 0.0 | 1.0 | 1.0 | 2.0 | 3.0 |
| future_career_concerns | | 307.0 | 4.475570 | 0.622473 | 0.0 | 4.0 | 5.0 | 5.0 | 5.0 |
| social_support | | 307.0 | 0.993485 | 0.080581 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| peer_pressure | | 307.0 | 4.495114 | 0.538525 | 1.0 | 4.0 | 5.0 | 5.0 | 5.0 |
| extracurricular_activities | | 307.0 | 4.433225 | 0.597875 | 1.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| bullying | | 307.0 | 4.446254 | 0.541914 | 1.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| stress_level | | 307.0 | 1.980456 | 0.160501 | 0.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| cluster | | 307.0 | 3.000000 | 0.000000 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |

Estudiantes con mayor ansiedad y depresión, baja autoestima y apoyo social. Dolores de cabeza frecuentes, presión alta y malas condiciones de vida. Rendimiento bajo. Estrés en 2.

Resumen General en Tabla:

| Cluster | Descripción breve |
|---------|---|
| 0 | 0 Indicadores promedio de salud mental, vida y e... |
| 1 | 1 Ansiedad y depresión moderadas, menor autoesti... |
| 2 | 2 Mejor salud mental y social, condiciones posit... |
| 3 | 3 Peor salud mental y social, condiciones difícili... |

Tabla por cada grupo para mostrar la cantidad, media, varianza, el dato mínimo, máximo y cuartiles, además de una tabla final con una descripción breve de cada grupo.

Análisis y Resultados

Análisis Discriminante

Se transformó la variable continua anxiety_level en una variable categórica binaria. Los estudiantes se clasificaron en dos grupos: aquellos con un nivel de ansiedad por debajo o igual a la media y aquellos con un nivel de ansiedad superior.

El conjunto de datos se dividió en entrenamiento (80 %) y prueba (20 %) con muestreo estratificado, y todas las variables predictoras fueron estandarizadas con StandardScaler para garantizar comparabilidad.

Posteriormente, se entrenaron dos variantes: Linear Discriminant Analysis (LDA) y Quadratic Discriminant Analysis (QDA) para fronteras no lineales.

Resultados de LDA: Se implementó un pipeline con búsqueda de hiperparámetros, evaluando distintos solvers (svd, lsqr, eigen) y configuraciones de shrinkage. La mejor configuración fue lsqr con shrinkage=auto, alcanzando un desempeño sólido en validación cruzada ($F1\text{-macro} \approx 0.76$). En el conjunto de prueba, los resultados fueron:

- Exactitud (Accuracy): 0.777
- $F1\text{-macro}$: 0.777

Precisión y recall fueron cercanos al 78 %. La matriz de confusión indicó una buena capacidad de discriminación, aunque con errores. Los coeficientes obtenidos sugieren que las variables más influyentes en la separación fueron calidad del sueño, apoyo social, rendimiento académico y nivel de estrés.

Resultados de QDA: Además, se ajustó un modelo Quadratic Discriminant Analysis (QDA), optimizando el parámetro de regularización. El mejor modelo correspondió a $\text{reg_param} = 0.5$, alcanzando un $F1\text{-macro}$ en validación cruzada de ≈ 0.77 y una exactitud en el conjunto de prueba de ≈ 0.75 . Se observó mayor sensibilidad al ruido en los datos y menor estabilidad en la clasificación.

Visualización: La proyección sobre el primer eje discriminante (LD1) mostró una separación clara entre los dos grupos de estudiantes, aunque en algunos casos las predicciones de clasificación se empalman. Esto evidencia que, aunque el modelo no es perfecto, sí logra identificar patrones relevantes que permiten distinguir entre estudiantes con baja y alta ansiedad.

Análisis factorial

Se verificaron si los datos eran adecuados para realizar un análisis factorial, para esto se aplicó la prueba de Bartlett y el índice KMO. Después, para decidir cuántos factores eran importantes, calculamos los eigenvalues, usamos el criterio de Kaiser y el gráfico de sedimentación. Luego, aplicamos una rotación varimax para facilitar ver que variables se agrupaban en cada factor. Finalmente, calculamos las communalidades las cuales indican qué parte de la varianza de cada variable es explicada por los factores seleccionados y utilizamos un mapa de calor para visualizar como cada variable se asocia con los factores.

Resultados:

-Prueba de Bartlett:

Test Statistic: 19237.2252

p-value: 0.0000

Lo que significa que las variables estaban suficientemente relacionadas para realizar el análisis

-KMO = 0.9712

Lo que significa que la muestra y las variables están bien para realizar este análisis.

- Se obtuvieron dos factores principales

-Factor 1: Bienestar social y académico

-Factor 2: Salud física y estrés emocional

- Comunalidad:

-El valor de la mayoría de las variables fue entre 0.55 y 0.81, lo que significa que los factores seleccionados resumen bien la información del conjunto de datos

Regresión Multiple

Antes de comenzar la regresión se hicieron dos tablas de correlación, una general y una en la que nos enfocabamos en nuestras posibles y, es decir, "anxiety_level" y "stress_level". Los resultados fueron importantes, ya que la mayoría de las variables tenían una correlación bastante fuerte tanto con "anxiety_level". Así mismo, observamos que variables como "selfEsteem", "sleep_quality", "academic_performance" y otras, tienen una correlación correlativa significativa con nuestra y.

Coeficientes que son estadísticamente significativos:

- blood_pressure : -0.1026
- sleep_quality : -0.1135
- future_carreer_concerns : 0.1292
- social_support : -0.905
- depression : 0.0863
- mental_heaklth_history : 0.0715

Con los datos sin entrenamiento ni prueba obtuvimos los siguiente resultados:

- R^2 : 0.690
- Adj. R^2 : 0.685
- F-statistic: 120.2
- Prob (F-statistic): 9.88e-258

Para el modelo con datos de entrenamiento obtuvimos los siguientes valores:

- R^2 : 0.686
- Adj. R^2 : 0.679
- F-statistic: 93.83
- Prob (F-statistic): 4.41e-200

Error Cuadratico Medio (MSE): 11.4595 sugiere que el modelo generaliza bien los datos no y no hay sobreajuste.

Raíz Cuadrada del Error Cuadrático (RMSE): 3.3852 y esto nos indica que el error promedio en la predicción del nivel de ansiedad en los datos de prueba es de 3.3852, lo que refuerza la idea de una buena generalización.

Error Medio Absoluto (MAE): 2.4371 Esto nos sugiere que el modelo tiene una buena capacidad del modelo para predecir.

Conglomerados

Se usa el conjunto de datos ya estandarizados para evita que variables con valores altos dominen sobre otras en el agrupamiento, determinamos el número de grupos como 4 a partir del método del codo y el puntaje de silueta, para posteriormente aplicar K-Means lo que agrupo los estudiantes de la siguiente manera:

- Grupo 0 → 300 estudiantes
- Grupo 1 → 192 estudiantes
- Grupo 2 → 301 estudiantes
- Grupo 3 → 307 estudiantes

Para la interpretación de clústeres (valores originales invertidos de la estandarización)

Grupo 0 (300 estudiantes): Ansiedad y depresión promedio, autoestima moderada. Historia de salud mental variada. Condiciones de vida y rendimiento académico en la media. Estrés nivel 1 (intermedio). Estudiantes con datos equilibrados con una situación promedio.

Grupo 1 (192 estudiantes):

- Ansiedad y depresión moderada, autoestima más baja.
- Más dolores de cabeza y presión arterial alta.
- Menor apoyo social, relaciones alumno-profesor más débiles.
- Estrés alrededor de 1, pero con alta variabilidad.
- Estudiantes vulnerables sociales con autoestima baja y poco apoyo social.

Grupo 2 (301 estudiantes):

- Menor ansiedad y depresión, mayor autoestima.
- Buenas condiciones de vida, alto rendimiento académico.
- Alto apoyo social, baja presión de pares, poco bullying.
- Estrés nivel 0 (bajo).
- Estudiantes resilientes con mejor salud mental y condiciones de vida además de estrés bajo.

Grupo 3 (307 estudiantes):

- Altos niveles de ansiedad y depresión, baja autoestima.
- Antecedentes de salud mental presentes.
- Más dolores de cabeza, presión arterial alta, mala calidad de sueño.
- Condiciones de vida pobres, bajo rendimiento académico, alta carga de estudio.
- Estrés nivel 2 (alto).
- Estudiantes en riesgo con altos niveles de ansiedad, depresión y estrés.

El grupo más numeroso es el 3 con 307 estudiantes, lo cual alerta sobre la necesidad de estrategias de intervención psicosocial y académica. El grupo 2 puede servir de modelo para identificar factores protectores y de mayor rendimiento académico.

Conclusión

El análisis realizado permite concluir que la ansiedad en los estudiantes es un fenómeno multifactorial donde influyen tanto variables académicas como sociales y personales. Los modelos estadísticos aplicados confirmaron que la autoestima, la calidad del sueño, el apoyo social y la carga académica son factores críticos en la predicción de los niveles de ansiedad. Además, la segmentación por conglomerados evidenció la existencia de perfiles diferenciados de estudiantes, desde aquellos con buena salud mental hasta los que enfrentan mayores dificultades emocionales y académicas. Estos resultados no solo contribuyen a la comprensión del problema, sino que también ofrecen bases para implementar programas de prevención y estrategias de apoyo que promuevan un mejor desempeño académico y bienestar integral en la comunidad estudiantil.

Inteligencia Artificial Generativa

El uso de las herramientas de inteligencia artificial generativa en el proyecto fue de ayuda en la resolución de dudas y mejora de la eficiencia al programar. Principalmente lo generado fue investigado para corroborar su veracidad, factibilidad y utilidad en el proyecto, lo que permitió trabajar casi sin estancamientos. Además, gracias a la investigación de las propuestas de solución aprendimos nuevas formas de superar errores de código, nuevas funciones y sus características, además de formas más compactas y eficiente de programar.

Referencias

- Amat Rodrigo, J. (2023). *Regresión lineal múltiple con Python*. Ciencia de Datos. Recuperado el 9 de septiembre de 2025, de <https://cienciadedatos.net/documentos/py10b-regresion-lineal-multiple-python>
- IBM. (s. f.). *Análisis discriminante*. IBM Documentation. Recuperado el 9 de septiembre de 2025, de <https://www.ibm.com/docs/es/spss-statistics/cd?topic=features-discriminant-analysis>
- Ortega, C. (s. f.). *Prueba de hipótesis: Qué es, pasos y ejemplos*. QuestionPro. Recuperado el 9 de septiembre de 2025, de <https://www.questionpro.com/blog/es/prueba-de-hipotesis/>
- Universitat de València. (s. f.). *Introducción al análisis factorial*. CEACES. Recuperado de <https://www.uv.es/ceaces/multivari/factorial/intro.htm>