

- Quiroz Rodriguez Alejandro
- Zavaleta Carranza Lenner

## 1. Nombre del caso

Segmentación Estudiantil para Intervenciones Académicas Personalizadas.

## 2. Respuestas a Cuestionario

### a. Problemática

Una institución educativa ha notado una alta tasa de deserción y rendimiento irregular en los cursos de ciencias básicas. Actualmente, el área de tutoría aplica las mismas estrategias de apoyo (charlas generales) para todos los estudiantes. Sin embargo, sospechan que existen perfiles muy distintos: estudiantes que asisten poco pero rinden bien, estudiantes que estudian mucho pero tienen bajas notas, y aquellos con un equilibrio estándar. Al no tener identificados estos "perfiles" ocultos, los recursos de tutoría se desperdician. El objetivo es aplicar modelos de aprendizaje no supervisado para encontrar agrupaciones naturales (clusters), permitiendo a la institución diseñar programas de nivelación o mentoría específicos para cada grupo.

### b. Determinar el algoritmo(s) que aplicará

Para este problema de segmentación, aplicaremos dos enfoques de clustering diferentes para comparar cuál agrupa mejor a los estudiantes:

**K-Means:** Es el estándar de la industria. Agrupará a los estudiantes en  $k$  grupos definidos calculando la distancia matemática entre sus calificaciones y hábitos de estudio hacia un punto central (centroide). Es rápido y excelente cuando sabemos aproximadamente cuántos perfiles queremos buscar (por ejemplo, 3: Alto, Medio, Bajo).

**Clustering Jerárquico (Agglomerative):** A diferencia de K-Means, este algoritmo no necesita que le digamos cuántos grupos queremos al principio. Construirá un árbol (dendrograma) uniendo a los estudiantes más similares paso a paso. Es ideal para explicarle a los directivos cómo se relacionan los distintos perfiles de alumnos.

**Jerárquico (Agglomerative):** A diferencia de K-Means, este algoritmo no necesita que le digamos cuántos grupos queremos al principio. Construirá un árbol (dendrograma) uniendo a los estudiantes más similares paso a paso. Es ideal para explicarle a los directivos cómo se relacionan los distintos perfiles de alumnos.

### c. Defina el dataset de trabajo

Para este caso, el dataset utilizado es el "**Student Performance Data Set**" de la plataforma kaggle

Para que el modelo funcione perfecto, filtraremos el CSV para quedarnos con variables numéricas clave que definen el rendimiento:

- i. **nota-final** (Nota final del curso de matemáticas, de 0 a 20).
- ii. **inasistencias** (Cantidad de ausencias a clases).
- iii. **horas-estudio** (Horas de estudio semanal).
- iv. **cursos-reprobados** (Cantidad de materias reprobadas anteriormente).

#### d. Intentar aplicar en código con Python

##### Leyendo datos

```
[2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings("ignore")

# Para que los gráficos se vean dentro del notebook
%matplotlib inline

sns.set(style="whitegrid")
```

```
[18]: # Si ya tienes un archivo con tus datos de alumnos
df = pd.read_csv("estudiantes.csv")
df.head()
```

```
[18]:   escuela sexo edad dirección famsize Pstatus Medu Fedu Mjob Fjob ... famrel freetime salidas Dalc Walc health inasistencias nota1 nota2 n
    0   GP   F   18      U    GT3     A     4     4 at_home teacher ...     4     3     4     1     1     3     4     0     11
    1   GP   F   17      U    GT3     T     1     1 at_home other ...     5     3     3     1     1     3     2     9     11
    2   GP   F   15      U    LE3     T     1     1 at_home other ...     4     3     2     2     3     3     6    12     13
    3   GP   F   15      U    GT3     T     4     2  health services ...     3     2     2     1     1     5     0    14     14
    4   GP   F   16      U    GT3     T     3     3   other   other ...     4     3     2     1     2     5     0    11     13
```

5 rows × 33 columns

```
[20]: # Resumen estadístico
df.describe()
```

##### Datos descriptivos y calidad de nulos

```
[20]:
```

	edad	Medu	Fedu	traveltime	horas-estudio	cursos-reprobados	famrel	freetime	salidas	Dalc	Walc	health	inasistencias
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880	3.930663	3.180277	3.184900	1.502311	2.280431	3.536210	3.659476
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235	0.955717	1.051093	1.175766	0.924834	1.284380	1.446259	4.640759
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	2.000000	0.000000
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	2.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	6.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	32.000000

```
[21]: df.isnull().sum()
```

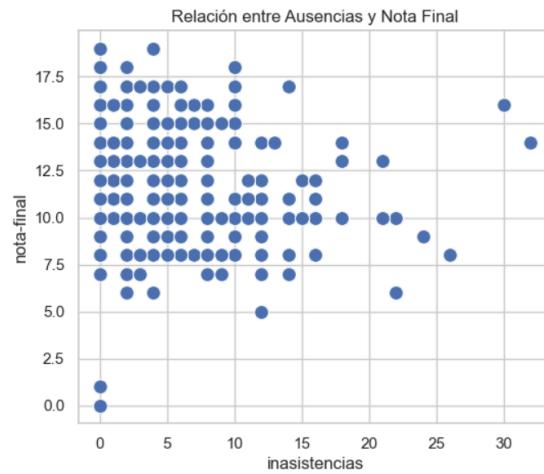
escuela	0
sexo	0
edad	0
direccion	0
famsize	0
Pstatus	0
Medu	0
Fedu	0
Mjob	0
Fjob	0
reason	0
guardian	0

## Analizando datos por variable



## Análisis de variables en conjunto

```
[32]: plt.figure(figsize=(6, 5))
sns.scatterplot(x="inasistencias", y="nota-final", data=df, s=100)
plt.title("Relación entre Ausencias y Nota Final")
plt.show()
```



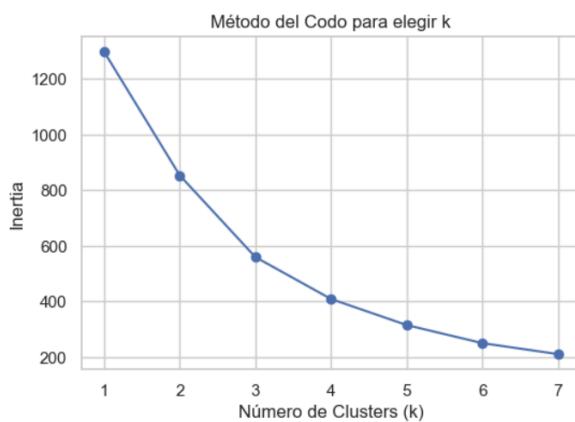
```
[37]: features = ["nota-final", "inasistencias"]
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])
```

## Encontrando número de clusters

```
[33]: inertia = []
K_range = range(1, 8) # prueba de 1 a 7 clusters

for k in K_range:
    kmeans_temp = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans_temp.fit(df_scaled)
    inertia.append(kmeans_temp.inertia_)

plt.figure(figsize=(6, 4))
plt.plot(K_range, inertia, "o-")
plt.xlabel("Número de Clusters (k)")
plt.ylabel("Inertia")
plt.title("Método del Codo para elegir k")
plt.show()
```



```
[34]: # K-Means
kmeans = KMeans(n_clusters=6, random_state=42, n_init=10)
df["Cluster_KMeans"] = kmeans.fit_predict(df_scaled)

# Clustering Jerárquico
jerarquico = AgglomerativeClustering(
    n_clusters=3, metric="euclidean", linkage="ward"
)
df["Cluster_Jerarquico"] = jerarquico.fit_predict(df_scaled)

df.head()
```

	escuela	sexo	edad	direccion	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	salidas	Dalc	Walc	health	inasistencias	nota1	nota2	nota-final	Cluster_KMeans
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	1	1	3	4	0	11	11	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	1	1	3	2	9	11	11	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	2	3	3	6	12	13	12	4
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	1	1	5	0	14	14	14	3
4	GP	F	16	U	GT3	T	3	3	other	other	...	2	1	2	5	0	11	13	13	2

5 rows × 35 columns

Datos para gráficas clusters determinados por modelo

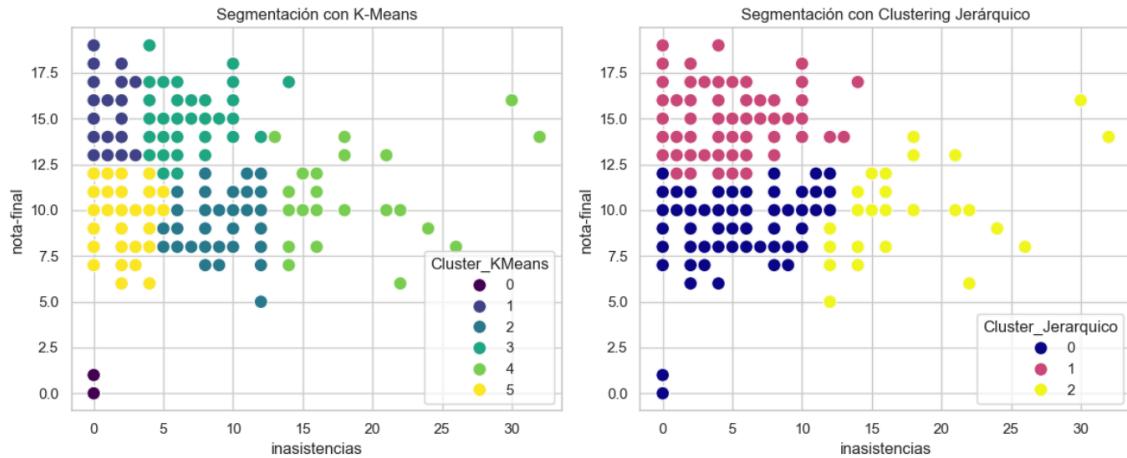
Con 2 algoritmos

```
[35]: fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# K-Means
sns.scatterplot(
    x="inasistencias",
    y="nota-final",
    hue="Cluster_KMeans",
    data=df,
    palette="viridis",
    s=100,
    ax=axes[0],
)
axes[0].set_title("Segmentación con K-Means")

# Jerárquico
sns.scatterplot(
    x="inasistencias",
    y="nota-final",
    hue="Cluster_Jerarquico",
    data=df,
    palette="plasma",
    s=100,
    ax=axes[1],
)
axes[1].set_title("Segmentación con Clustering Jerárquico")

plt.tight_layout()
plt.show()
```



```
[36]: print("Resultados de la asignación de grupos (K-Means):")
df[["nota-final", "inasistencias", "Cluster_KMeans"]]
```

Resultados de la asignación de grupos (K-Means):

	nota-final	inasistencias	Cluster_KMeans
0	11	4	5
1	11	2	5
2	12	6	3
3	14	0	1
4	13	0	1
...	...	...	...
644	10	4	5
645	16	4	3
646	9	6	2
647	10	6	2
648	11	4	5

649 rows × 3 columns