

UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



CLUSTERING CON R Y PYTHON

ASIGNATURA: MINERÍA DE DATOS
DOCENTE: FERNANDEZ CHAMBI, MAYENKA
ESTUDIANTE: NEIRA MONTESINOS, ERIKSON
SEMESTRE X
GRUPO: U

ENERO DEL 2024

PUNO – PERÚ

ÍNDICE

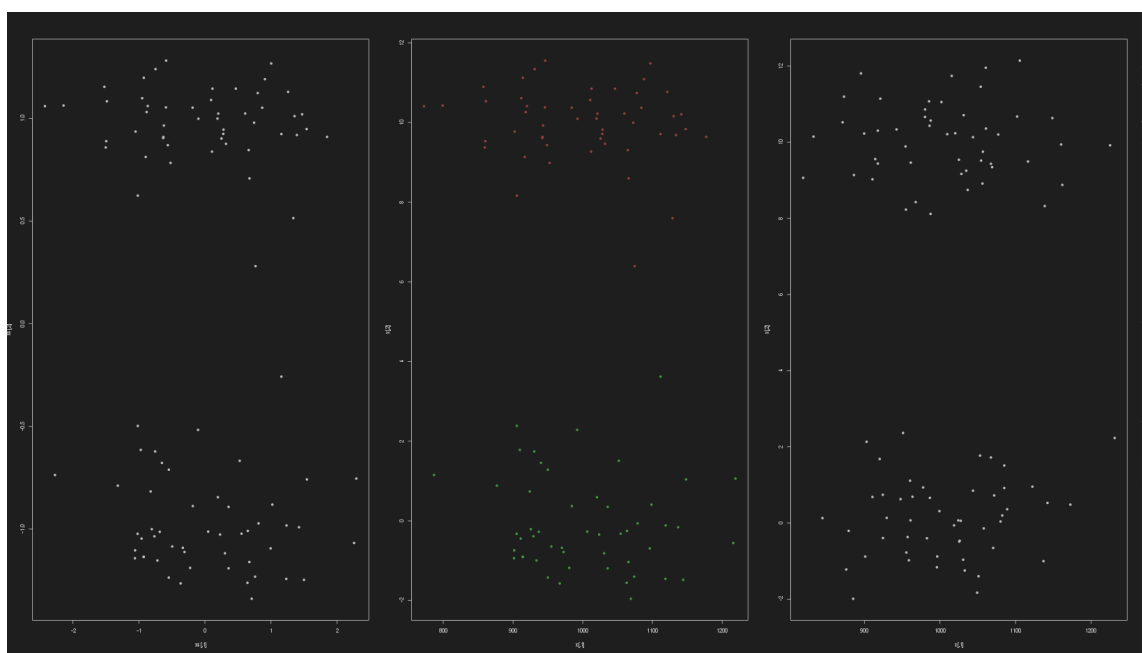
1. INTRODUCCIÓN.....	3
2. CLUSTERING CON R.....	3
3. CLUSTERING CON PYTHON.....	8
4. CONCLUSIONES.....	14
5. ANEXOS.....	15

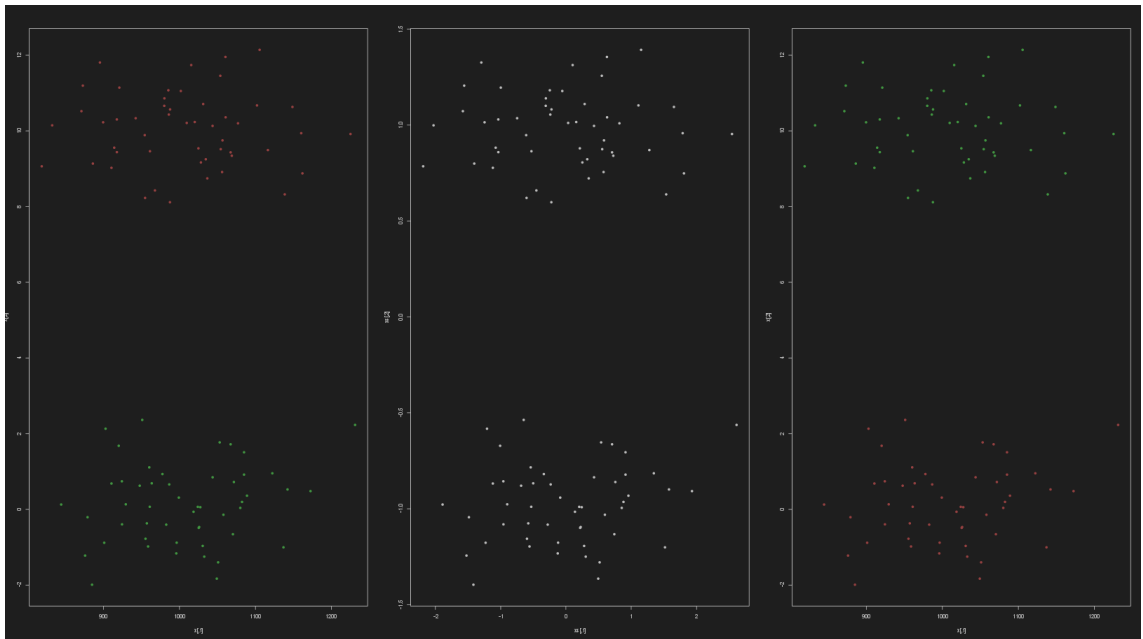
1. INTRODUCCIÓN

El clustering es una técnica clave en el análisis de datos que se centra en agrupar observaciones similares en conjuntos llamados "clusters". La idea central es que elementos con rasgos comunes pertenecen al mismo grupo, facilitando la identificación de patrones y estructuras en conjuntos de datos complejos. Esta herramienta tiene aplicaciones amplias, desde la investigación científica hasta la toma de decisiones empresariales, ya que ofrece un enfoque efectivo para entender la estructura de los datos y descubrir relaciones entre variables. Al segmentar datos en clusters, los analistas pueden identificar tendencias y patrones útiles para la toma de decisiones informada. En este informe nos enfocamos en cómo se implementa esta técnica en dos lenguajes de programación populares, R y Python.

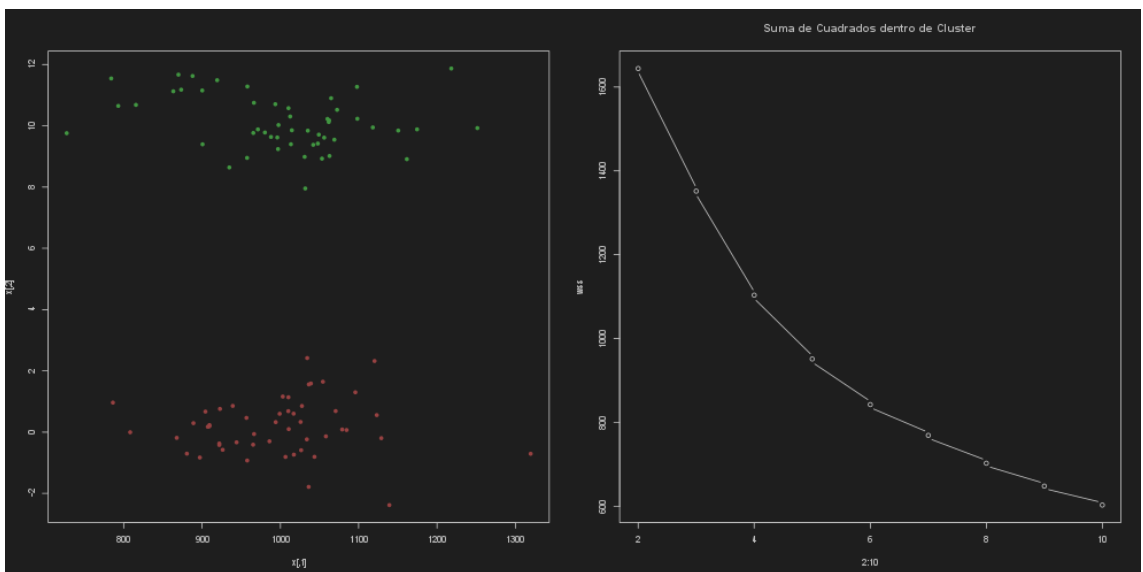
2. CLUSTERING CON R

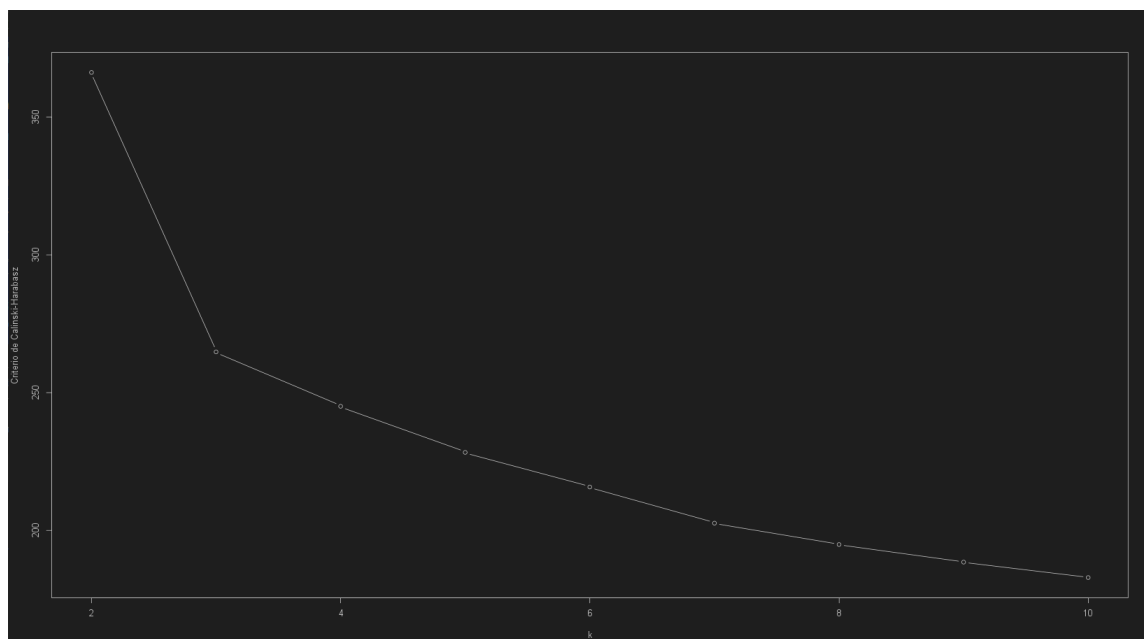
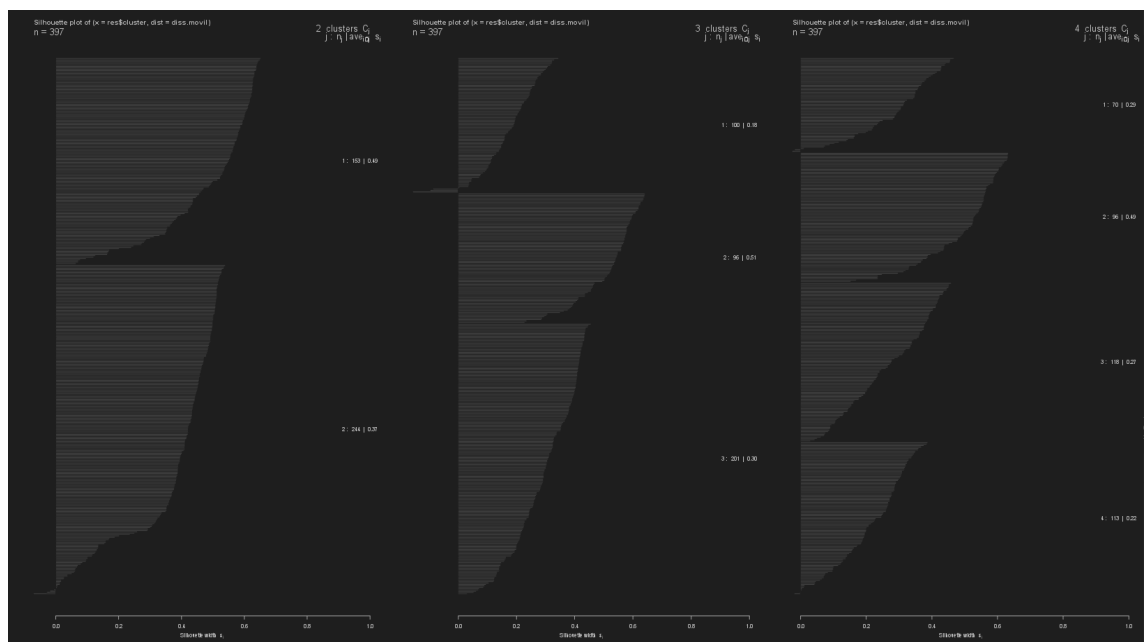
El código comienza con una simulación de datos aleatorios (x) y la aplicación del algoritmo K-Means para dos conglomerados. Se visualizan los resultados mediante gráficos. Posteriormente, los datos se estandarizan y se aplica K-Means nuevamente, presentando otra visualización.





Después, el código carga datos reales desde un archivo CSV (data_cluster.csv) y selecciona las columnas relevantes (movil1). Estos datos se estandarizan y se aplican múltiples ejecuciones de K-Means con diferentes cantidades de conglomerados. Se evalúa la calidad del clustering mediante métricas como la suma de cuadrados dentro de los clusters, la silueta y el índice de Calinski-Harabasz.

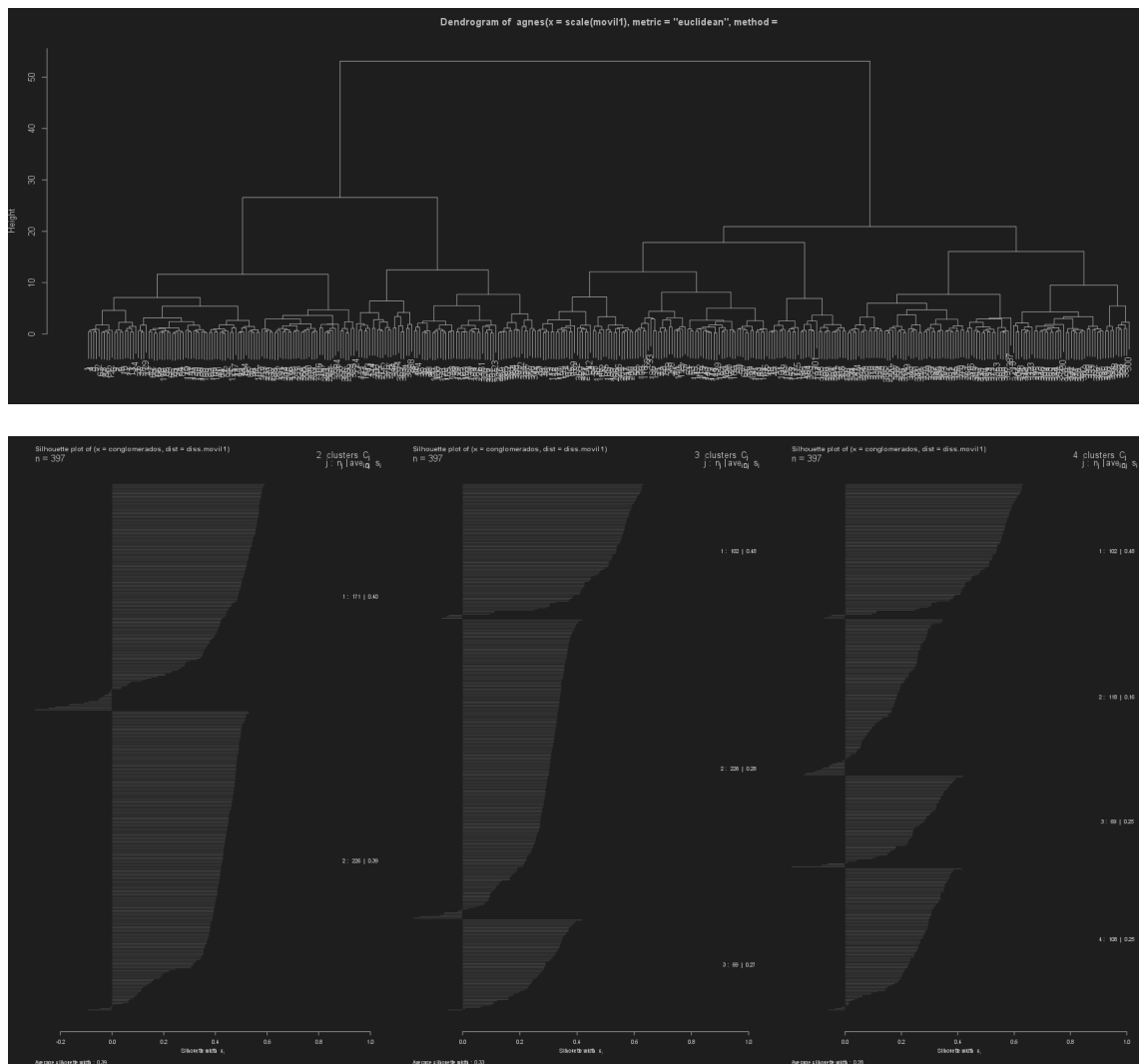




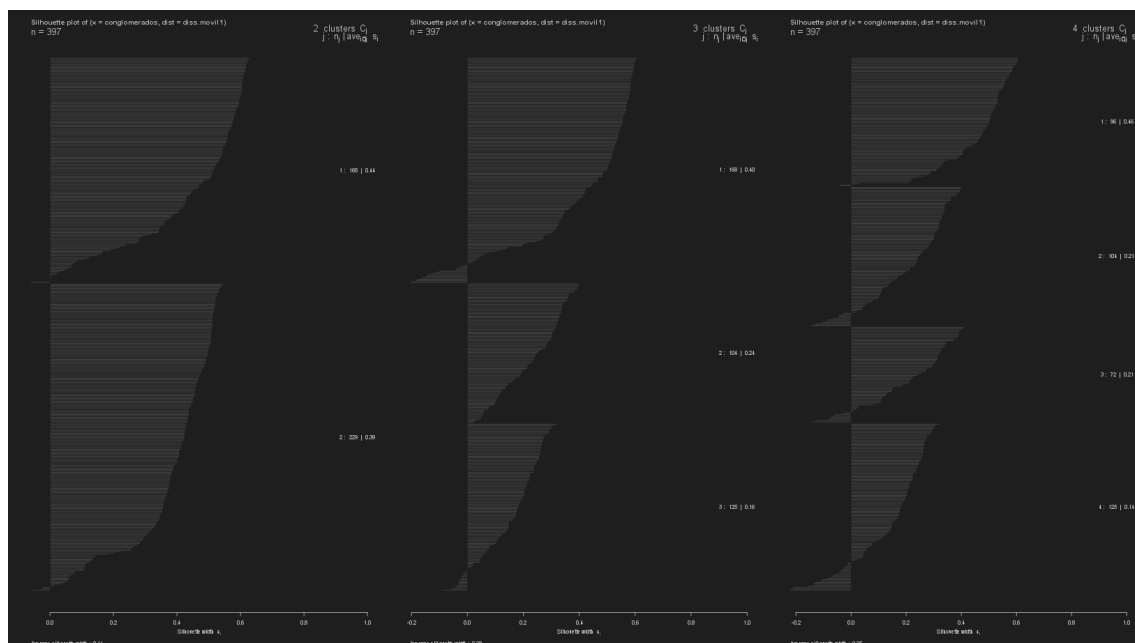
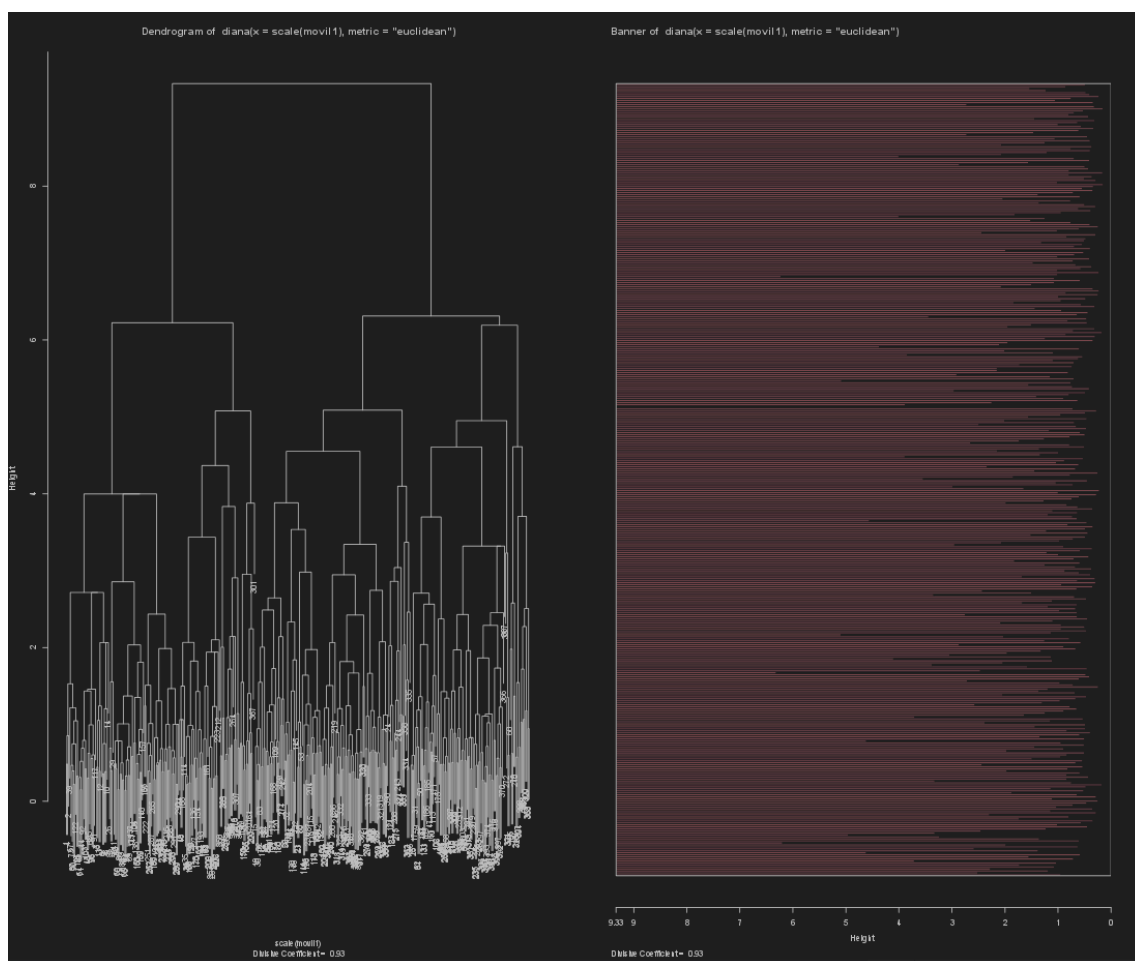
A continuación, se realiza un perfilamiento de los conglomerados formados, calculando el promedio de diversas variables para cada conglomerado. La biblioteca sqldf se utiliza para realizar consultas SQL en el conjunto de datos.

	Id	Sexo	Estad_Civil	Ingreso_mensual_cliente	N_llamada_recib	N_recargas	Tot_Min_consum	N_mens_tex_env	N_llamadas_realiz	Antig_cliente_meses	N_reclamos	clust
1	1	Masculino	Casado	3504	18.0000	8	307.000	17	12	70	1	4
2	2	Femenino	Soltero	3693	15.0000	8	350.000	35	12	70	1	4
3	3	Masculino	Soltero	3436	18.0000	8	318.000	29	11	70	1	4
4	4	Masculino	Casado	3433	16.0000	8	304.000	29	12	70	1	4
5	5	Masculino	Soltero	3449	17.0000	8	302.000	24	10	70	1	4
6	6	Femenino	Casado	4341	15.0000	8	429.000	42	10	70	1	4
7	7	Masculino	Soltero	4354	14.0000	8	454.000	47	9	70	1	4
8	8	Masculino	Soltero	4312	14.0000	8	440.000	46	8	70	1	4
9	9	Masculino	Soltero	4425	14.0000	8	455.000	48	10	70	1	4
10	10	Femenino	Casado	3850	15.0000	8	390.000	40	8	70	1	4

El código continúa con el análisis de clustering jerárquico, tanto aglomerativo como divisivo, en los mismos datos. Se construye un dendrograma y se determina el número óptimo de conglomerados. Se realiza también un análisis de silhouette para evaluar la calidad del clustering jerárquico.



En una sección posterior, se escalan las columnas numéricas de los datos y se realiza nuevamente el análisis de clustering jerárquico. Se presenta una alternativa para realizar el análisis de silhouette después del escalado. Finalmente, el código concluye mostrando los resultados finales, incluyendo el número óptimo de conglomerados y la asignación de observaciones a conglomerados en ambos métodos, K-Means y Clustering Jerárquico.



3. CLUSTERING CON PYTHON

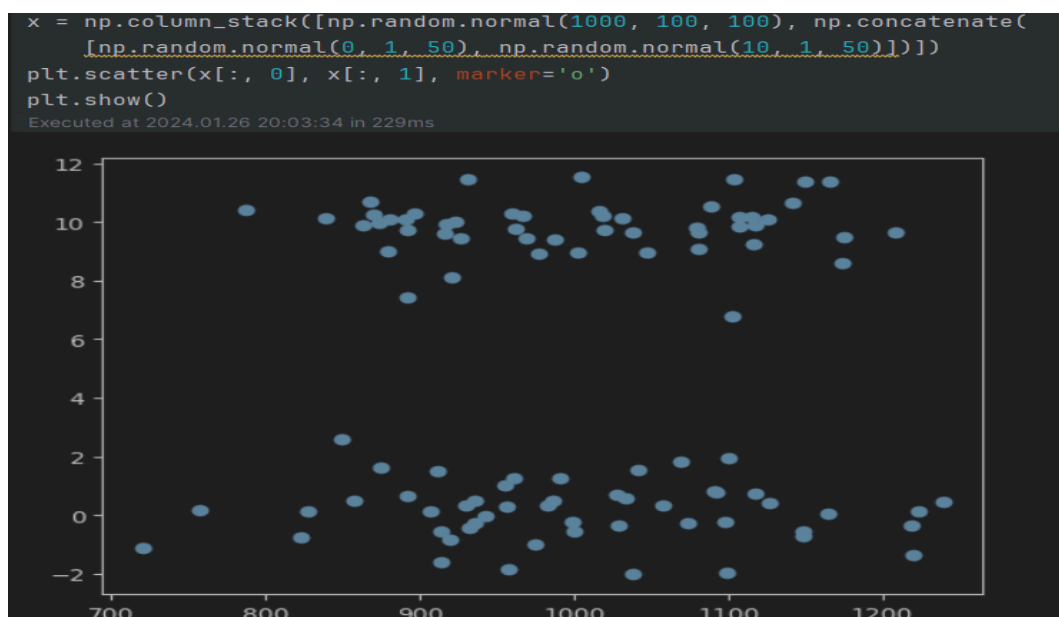
El código proporcionado realiza un análisis de clustering utilizando el algoritmo K-Means y técnicas de clustering jerárquico en un principio se usa un conjunto de datos simulados y en un conjunto de datos leído desde un archivo CSV. Aquí se presenta una explicación formal del código, en primer lugar, se importan las librerías necesarias para el análisis, como os, pandas, numpy, matplotlib, y las clases y funciones relacionadas con clustering de scikit-learn y scipy. Además, se configura el número de hilos de OpenMP y se realiza una simulación con datos generados aleatoriamente.

```
import os
os.environ['OMP_NUM_THREADS'] = '1'
!set OMP_NUM_THREADS=1
Executed at 2024.01.26 20:03:30 in 60ms

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import scale
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import calinski_harabasz_score
```

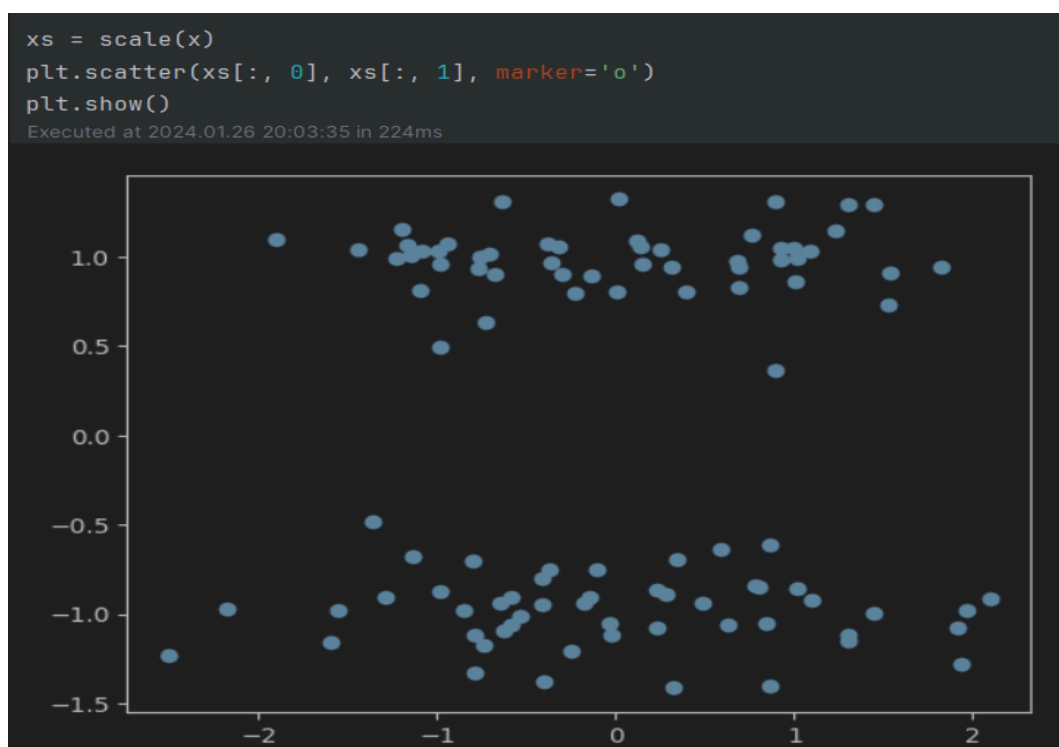
En la simulación, se crea un conjunto de datos x con dos columnas, donde la primera columna contiene datos normalmente distribuidos con media 1000 y desviación estándar 100, y la segunda columna combina dos conjuntos de datos normalmente distribuidos con medias 0 y 10, respectivamente.



A continuación, se realiza el análisis K-Means en los datos originales (kmeans_orig) y se muestra un gráfico de dispersión con colores que representan los clusters encontrados por el algoritmo.



Después, se estandarizan los datos utilizando la función scale de scikit-learn y se realiza el análisis K-Means en los datos estandarizados (kmeans_std), mostrando nuevamente un gráfico de dispersión.





Luego, se leen datos desde un archivo CSV (data_cluster.csv) utilizando pandas. Se seleccionan ciertas columnas del DataFrame (movil1), se estandarizan los datos, y se realiza el clustering K-Means en estos datos estandarizados (kmeans_clusters). Se imprime la asignación de clusters y la inercia del modelo.

```

kmeans_clusters = KMeans(n_clusters=2, n_init=50)
kmeans_clusters.fit(zmovil1)
print(kmeans_clusters.labels_)

```

Executed at 2024.01.26 20:03:39 in 2s 870ms

```

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1
 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0
 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 1
 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1
 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0
 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

```

```

print(kmeans_clusters.inertia_)

```

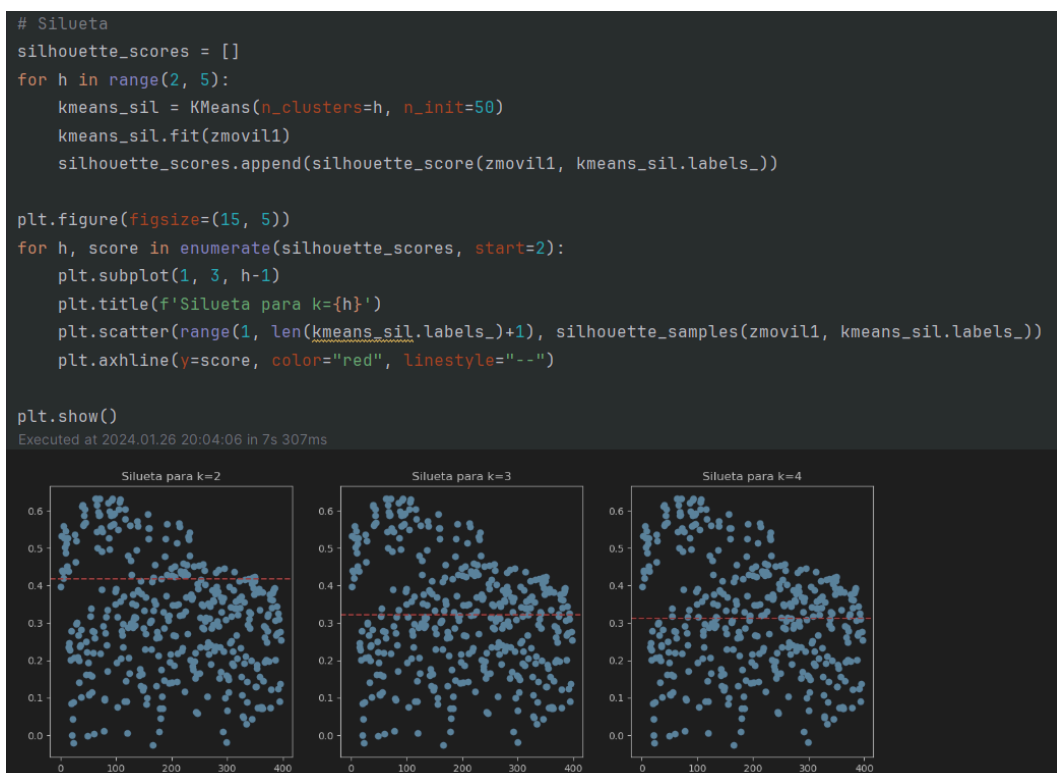
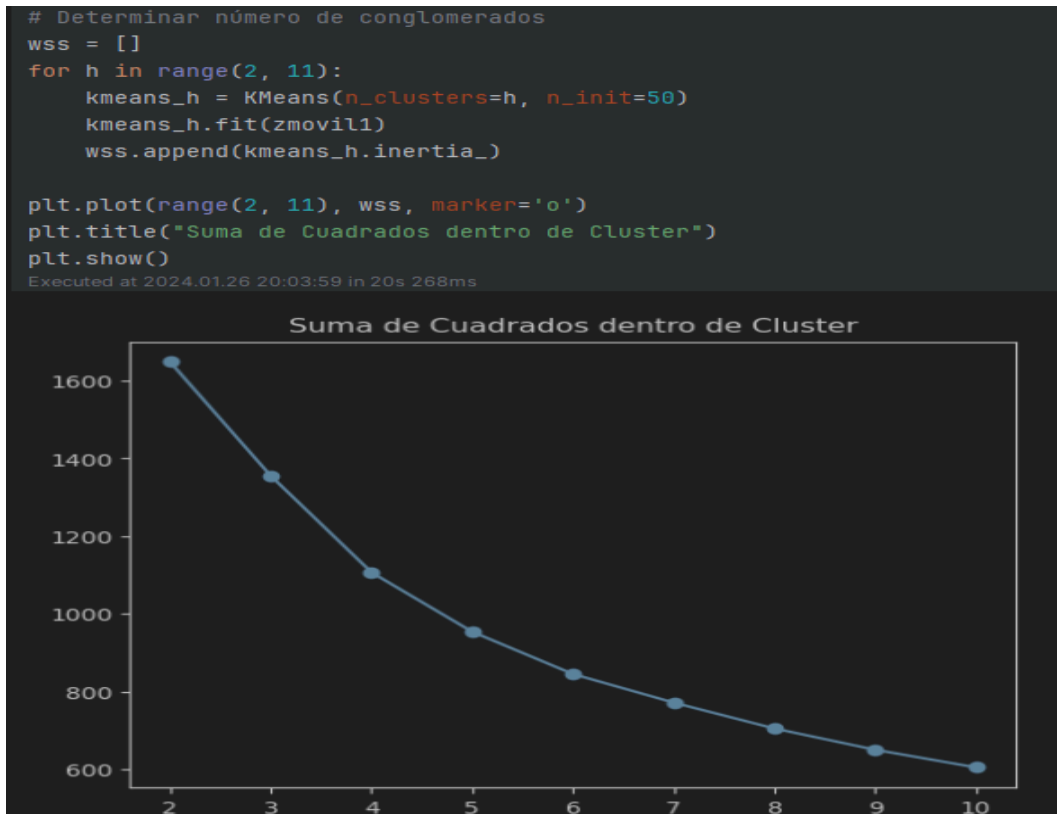
Executed at 2024.01.26 20:03:39 in 34ms

```

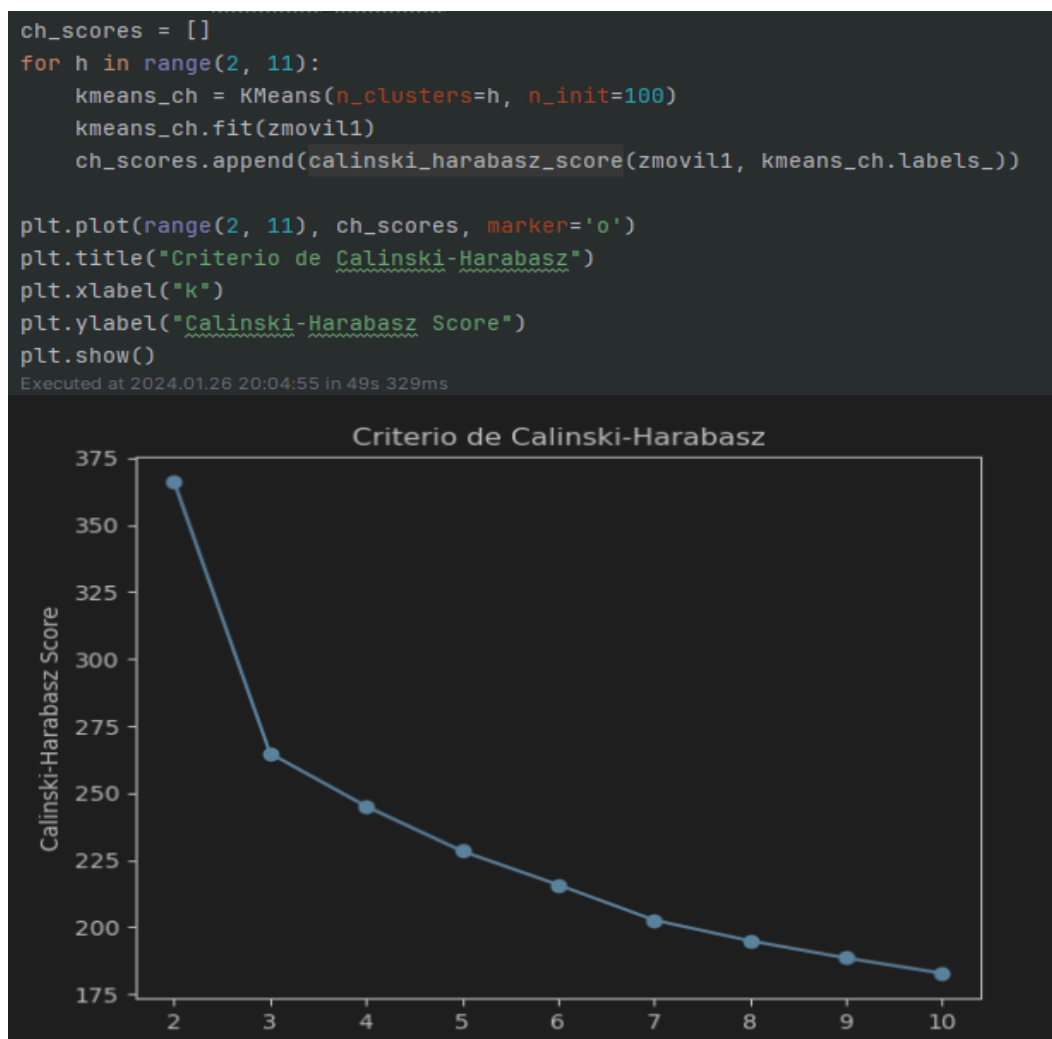
1648.1648461958634

```

Posteriormente, se determina el número óptimo de conglomerados utilizando el método del codo, calculando la suma de cuadrados dentro de los clusters para diferentes valores de k. También se calcula el índice de silueta para diferentes valores de k y se grafican los resultados.



Se aplica el criterio de Calinski-Harabasz para evaluar la calidad de los clusters en función de la dispersión intra-cluster y la dispersión entre clusters.



A continuación, se realiza el perfilamiento de los clusters utilizando K-Means en los datos del archivo. Se crea un DataFrame (movil_cl) que incluye las etiquetas de cluster asignadas por K-Means.

```
# Perfilamiento de los clusters
movil_cl.reset_index(drop=True, inplace=True)
movil_cl['Clust'] = kmeans_movil.labels_

Executed at 2024.01.26 20:04:56 in 24ms

#movil_cl = movil_cl.drop(396)
movil_cl2 = movil_cl.drop(['Sexo', 'Estado_Civil'], axis = 1)
# Calcular el promedio por cluster
cluster_means = movil_cl2.groupby('Clust').mean()
cluster_means
```

Executed at 2024.01.26 20:04:56 in 48ms

Clust	Id	Ingreso_mensual_cli	N_llamada_recib	N_recargas	Tot_Min_consum	N_mens_tex_env
0	231.596708	2423.378601	28.070189	4.283951	122.594650	68.263374
1	145.934641	3840.477124	16.220915	7.333333	306.686275	24.745098

Luego, se realiza un análisis de clustering jerárquico utilizando los métodos AGNES (Agglomerative Nesting) y DIANA (DIvisive ANALysis). Se aplican estos métodos y se muestra la silueta para AGNES y el dendrograma para DIANA.

```
# AGNES
agnes_clusters = AgglomerativeClustering(n_clusters=2, linkage='ward', metric='euclidean')
agnes_labels = agnes_clusters.fit_predict(zmovil1)
print(agnes_labels)
```

Executed at 2024.01.26 20:04:56 in 99ms

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1
 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0
 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 0
 1 0 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1
 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1
 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1]
```

```
# Silueta para AGNES
silhouette_scores_agnes = []

# Update 'affinity' to 'metric'
for h in range(2, 5):
    agnes_sil = AgglomerativeClustering(n_clusters=h, metric='euclidean', linkage='ward')
    agnes_labels_sil = agnes_sil.fit_predict(zmovil1)
    silhouette_avg = silhouette_score(zmovil1, agnes_labels_sil)
    silhouette_scores_agnes.append(silhouette_avg)

plt.figure(figsize=(15, 5))
for h, score in enumerate(silhouette_scores_agnes, start=2):
    plt.subplot(1, 3, h-1)
    plt.title(f'Silueta para k={h}')
    plt.scatter(range(1, len(agnes_labels_sil)+1), silhouette_samples(zmovil1, agnes_labels_sil))
    plt.axhline(y=score, color="red", linestyle="--")
    plt.xlabel('Sample Index')
    plt.ylabel('Silhouette Score')

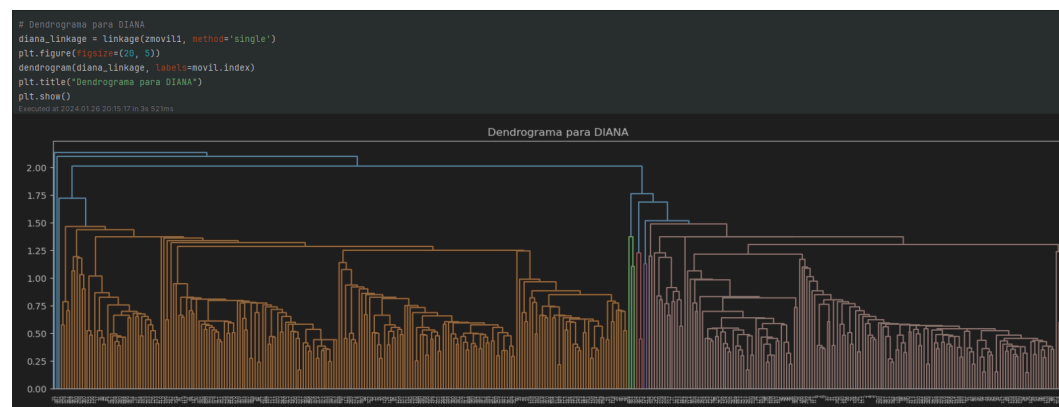
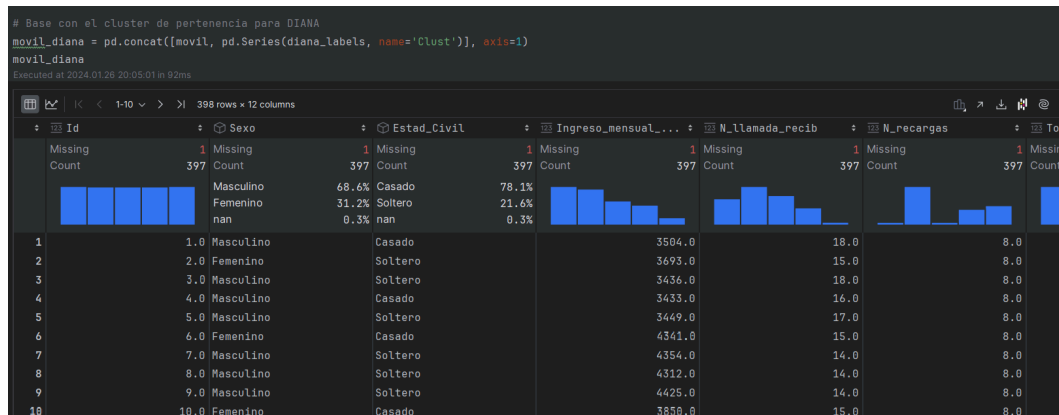
# Adjust subplot parameters
plt.subplots_adjust(wspace=0.5)

plt.show()
```

Executed at 2024.01.26 20:04:57 in 719ms




Finalmente, se presentan funciones auxiliares y visualizaciones de resultados, como la asignación de clusters para AGNES y DIANA, y se muestra el dendrograma para DIANA.



4. CONCLUSIONES

En resumen, este estudio subraya la importancia fundamental del clustering en el análisis de datos, proporcionando una técnica esencial para agrupar observaciones similares y revelar patrones en conjuntos complejos. La aplicación práctica de esta técnica en dos lenguajes de programación ampliamente utilizados, R y Python, expone enfoques efectivos para comprender la estructura de los datos y desentrañar relaciones entre variables. Por tanto podemos decir que tanto en R como en Python el clustering es una herramienta esencial para analizar datos complejos y extraer conocimientos significativos.

5. ANEXOS

- Archivo de Google Colab:  Clustering.ipynb
- Repositorio del curso:

<https://github.com/eldammar/unap-md-preprocessing>