Actividad 3

Autor: Rubén Garrido Hidalgo

Fecha: 09/05/2025

Asignatura: Minería de Datos

Curso: 4º Grado en Matemáticas

Introducción

En este proyecto vamos a implementar y analizar diferentes métodos de clustering sobre un conjunto de datos extraidos del portal Kaggle. En este proyecto implementaremos el método de Clustering Jerárquico y de K-Means para agrupar nuestros datos no etiquetados, así como el método de reducción de componentes principales (PCA) para tomar los datos que expliquen mejor la varianza reduciendo su dimensionalidad.

Descripción de los datos

Como mencionamos los datos que vamos a emplear para implementar los modelos de agrupamiento o clustering han sido extraidos del portal Kaggle del cual adjunto el enlace para su consulta: https://www.kaggle.com/datasets/camnugent/california-housing-prices

Este dataset constan de diferentes datos sobre la vivienda en California. Nn nuetsro caso vamos a tener en cuenta las variables espaciales de 'Latitude' y 'Longitude' para realizar el Clustering o agrupamiento mediante Clustering Jerárquico y K-Means.

1. Clustering o agrupamiento con y sin PCA

Vamos a implementar el algoritmo de aprendizaje no supevisado K-Means con y sin PCA sobre los datos housing.csv

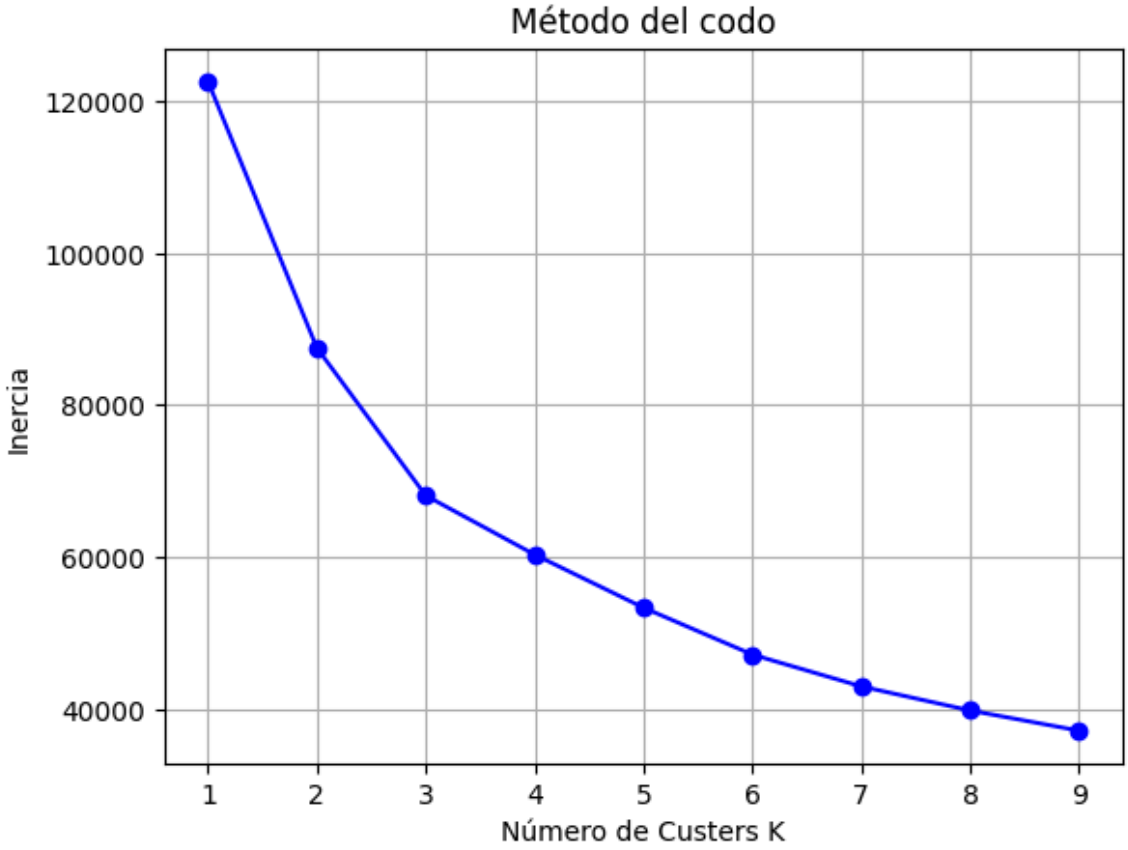
In [4]:
```
# ================================================================================
# Comenzamos importado las librerías necesarias
# ================================================================================
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

#================================================================================
# Importación y preprocesamiento de los datos
#================================================================================

#Importamos los datos
df = pd.read_csv("housing.csv")
```

```
# Eliminamos las filas con valores nulos en caso de que existieran
df = df.dropna()

# Seleccionamos las características que tendremos en cuenta para crear los clust
X = df.drop(columns=["ocean_proximity", "total_rooms", "total_bedrooms", "median

# Estandarizamos los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Como método de aprendizaje automático no supervisado para relizar el clustering

#Importamos el clasificador y estimamos el número de grupos o clusters que tendr

#=============================================================================
# Estimación del número de clusters
#=============================================================================
# Método del codo para calcular el número óptimo de clusters k mediante el uso d
inercias = []
K_range = range(1,10)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inercias.append(kmeans.inertia_)

plt.plot(K_range, inercias, "bo-")
plt.xlabel("Número de Custers K")
plt.ylabel("Inercia")
plt.title("Método del codo")
plt.grid(True)
plt.show()

#Este ejemplo es sin estandarizar los datos, por lo que contribuye a que no se a
#podemos ver un cambio en la curvatura a partir de k= 3, vamos a calcular el Sil

# Cálculo del Silhouette Score:
from sklearn.metrics import silhouette_score

silhouette_scores = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)


plt.plot(range(2, 11), silhouette_scores, "bo-")
plt.xlabel("Número de clústers (k)")
plt.ylabel("Silhouette score")
plt.title("Método del Silhouette Score")
plt.grid(True)
plt.show()


#Este nos permite ver que el valor más optimo para el número de clusters es k =


#=============================================================================
# Clustering K-Means sin PCA
```

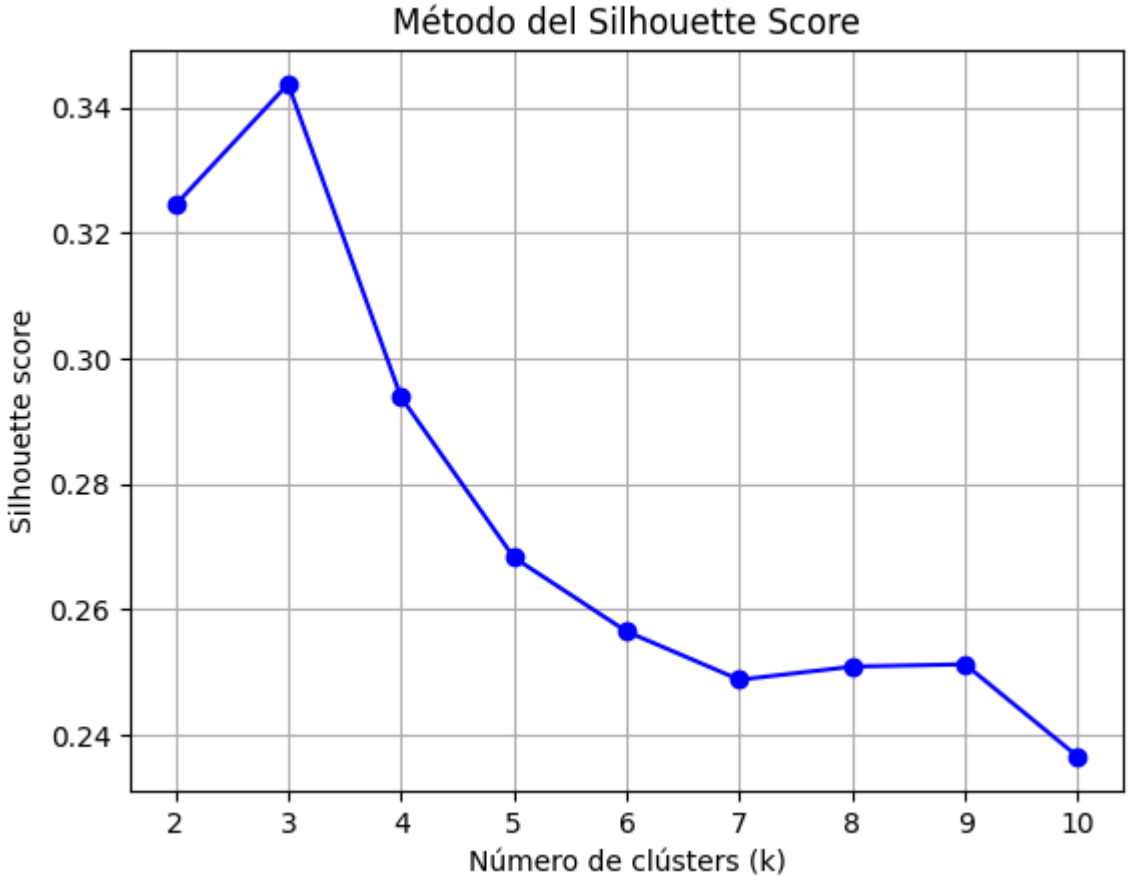
```
#================================================================================
#A continuación definimos el clasificador para realizar el clustering con K-Mean
kmeans_sin_pca = KMeans(n_clusters=3, random_state=42)
labels_sin_pca = kmeans_sin_pca.fit_predict(X_scaled)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels_sin_pca, cmap="rainbow", s=
plt.scatter(kmeans_sin_pca.cluster_centers_[:, 0], kmeans_sin_pca.cluster_center
plt.title("Clusters usando K-Means sin PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()

#================================================================================
# A continuación vamos a implementar la reducción de componentes principales (PC
# el K_Means
#================================================================================



#================================================================================
#Aplicamos la reducción de componentes principales (PCA)
#================================================================================
from sklearn.decomposition import PCA

# Creamos el objeto PCA
pca = PCA(n_components=3)
pca_result = pca.fit_transform(X_scaled)

#Imprimimos las varianzas en cada eje
print(pca.explained_variance_ratio_)
# El resultado es [0.36777967 0.29960122 0.17461867 0.13223773 0.01536309 0.0103
#Vemos como las cuatro primeras componentes principales explican casi un 97% de
#vamos a realizar la representación gráfica de la varianza explicada:

#Representación gráfica de la varianza explicada
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker = "o")
plt.title("Varianza explicada acumulada")
plt.xlabel("Número de Componentes")
plt.ylabel("Varianza acumulada")
plt.show()

#Podemos tomar 3 o 4 componentes principales, deberíamos de estudiar el rendienm
#puede sobreajustarse a los datos. En mi caso he tomado 3 vectores principales o

#A continuación vamos a implementar el modelo de K-Means:
#================================================================================
# Estimación del número de clusters
#================================================================================
#Método del codo para calcular el número óptimo de Clústers k mediante el uso de
inercias = []
K_range = range(1,10)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(pca_result)
    inercias.append(kmeans.inertia_)

plt.plot(K_range, inercias, "bo-")
plt.xlabel("Número de Custers con PCA K")
```

```
plt.ylabel("Inercia")
plt.title("Método del codo")
plt.grid(True)
plt.show()

print("Ocurre como en el caso anterior, puede observar como un candidato a ser e

#Calculamos el Silhouette Score:
from sklearn.metrics import silhouette_score

silhouette_scores = []

for k in range(2, 8):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(pca_result)
    score = silhouette_score(pca_result, labels)
    silhouette_scores.append(score)


plt.plot(range(2, 8), silhouette_scores, "bo-")
plt.xlabel("Número de clústers con PCA (k)")
plt.ylabel("Silhouette score")
plt.title("Método del Silhouette Score")
plt.grid(True)
plt.show()

print("Vemos como basándonos en el Silhouette Score el número de cluster es k =


#=============================================================================
# Aplicamos el K-Means con PCA
#=============================================================================
#A continuación definimos el clasificador para realizar el clustering con K-Mean
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=labels, cmap="rainbow", s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c="bla
plt.title("Clusters usando K-Means con PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()
```

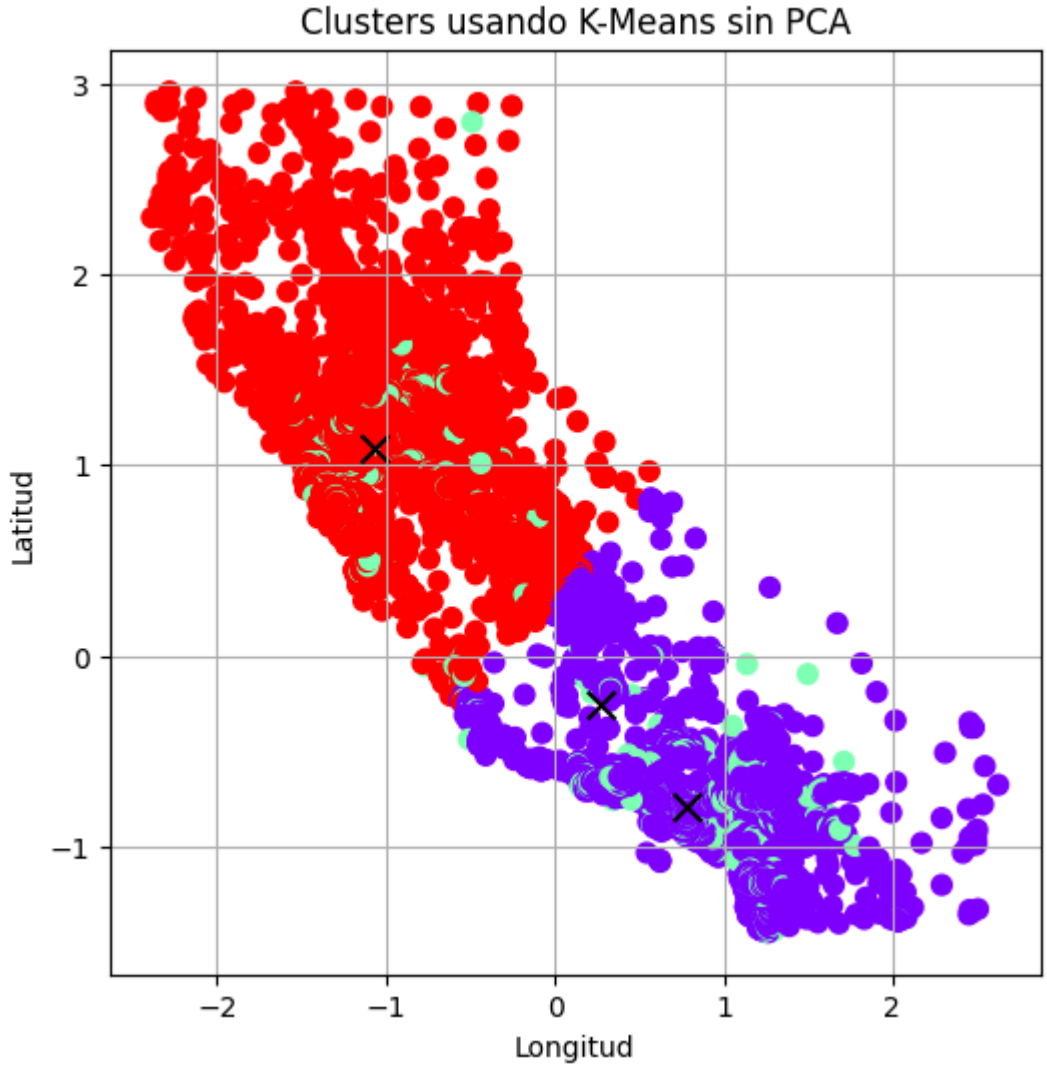
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Método del codo
120000
100000
80000
60000
40000
Inercia
1
2
3
4
5
6
7
8
9
Número de Custers K

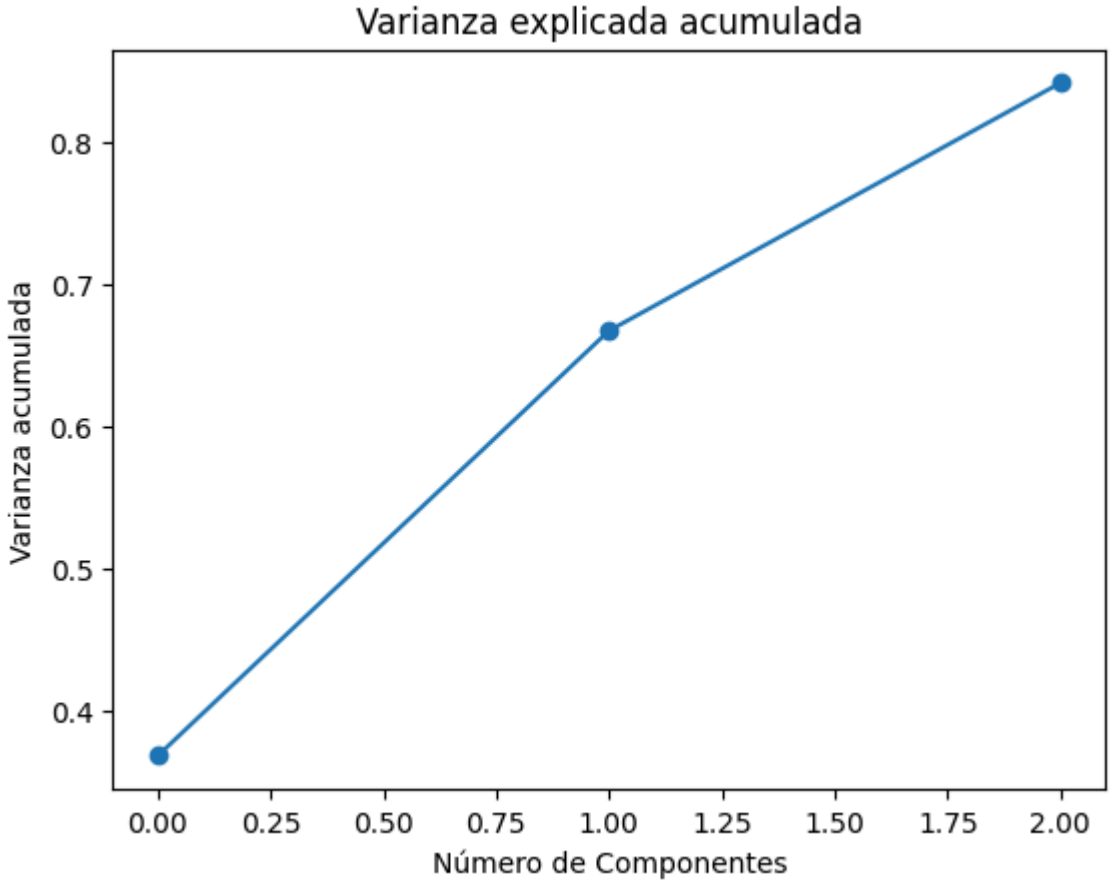
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



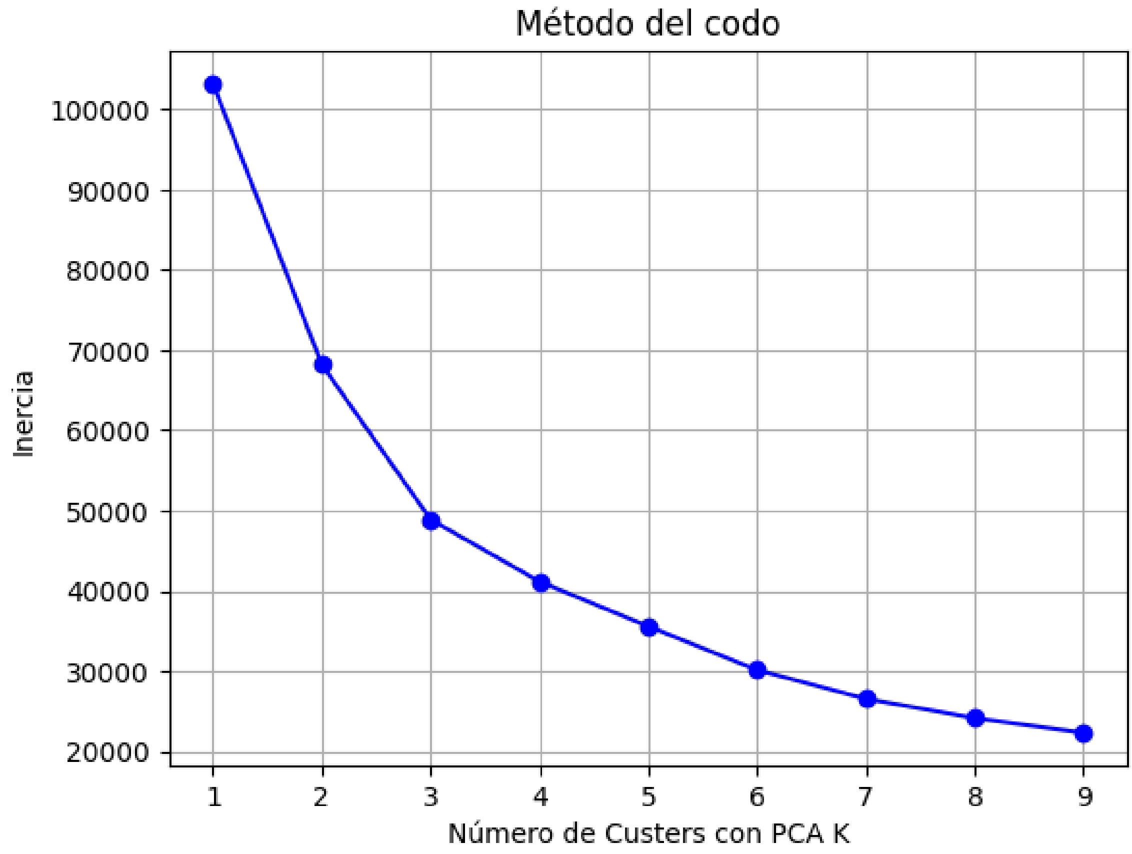

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
[0.36777967 0.29960122 0.17461867]
```

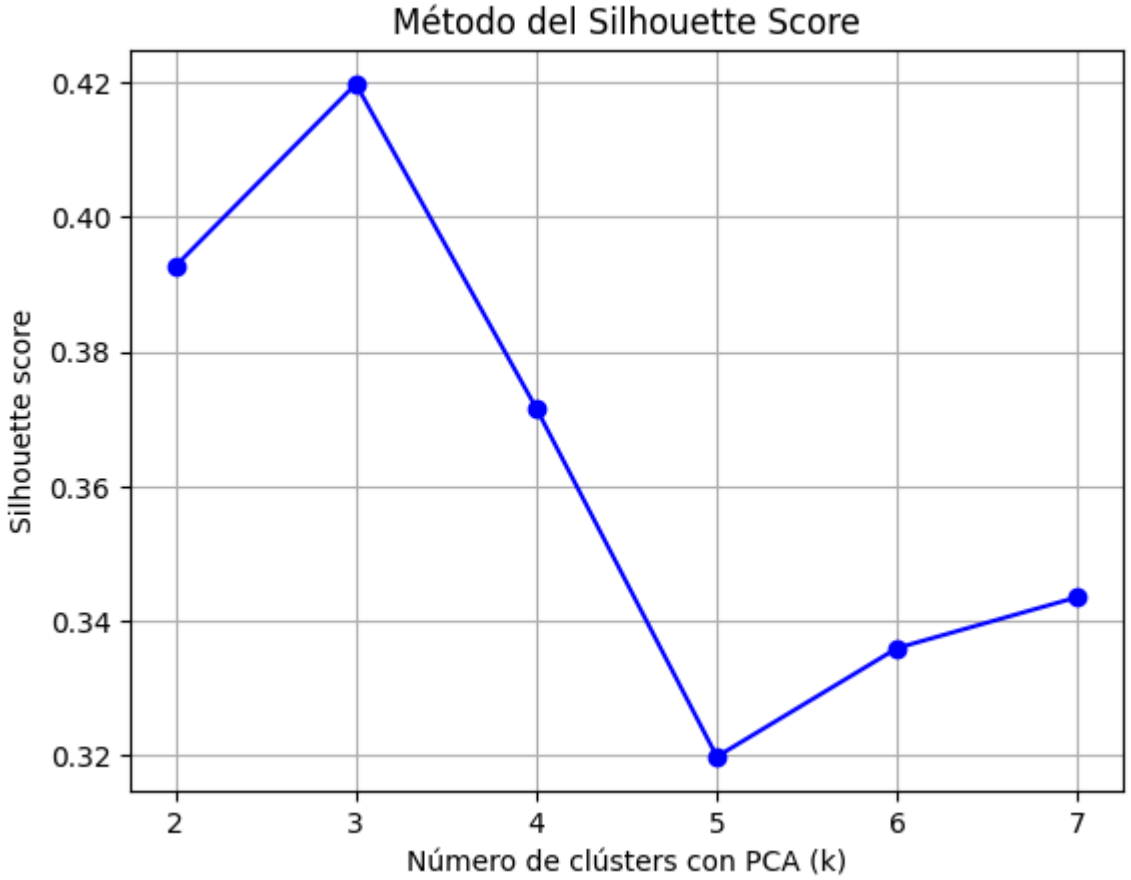


```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



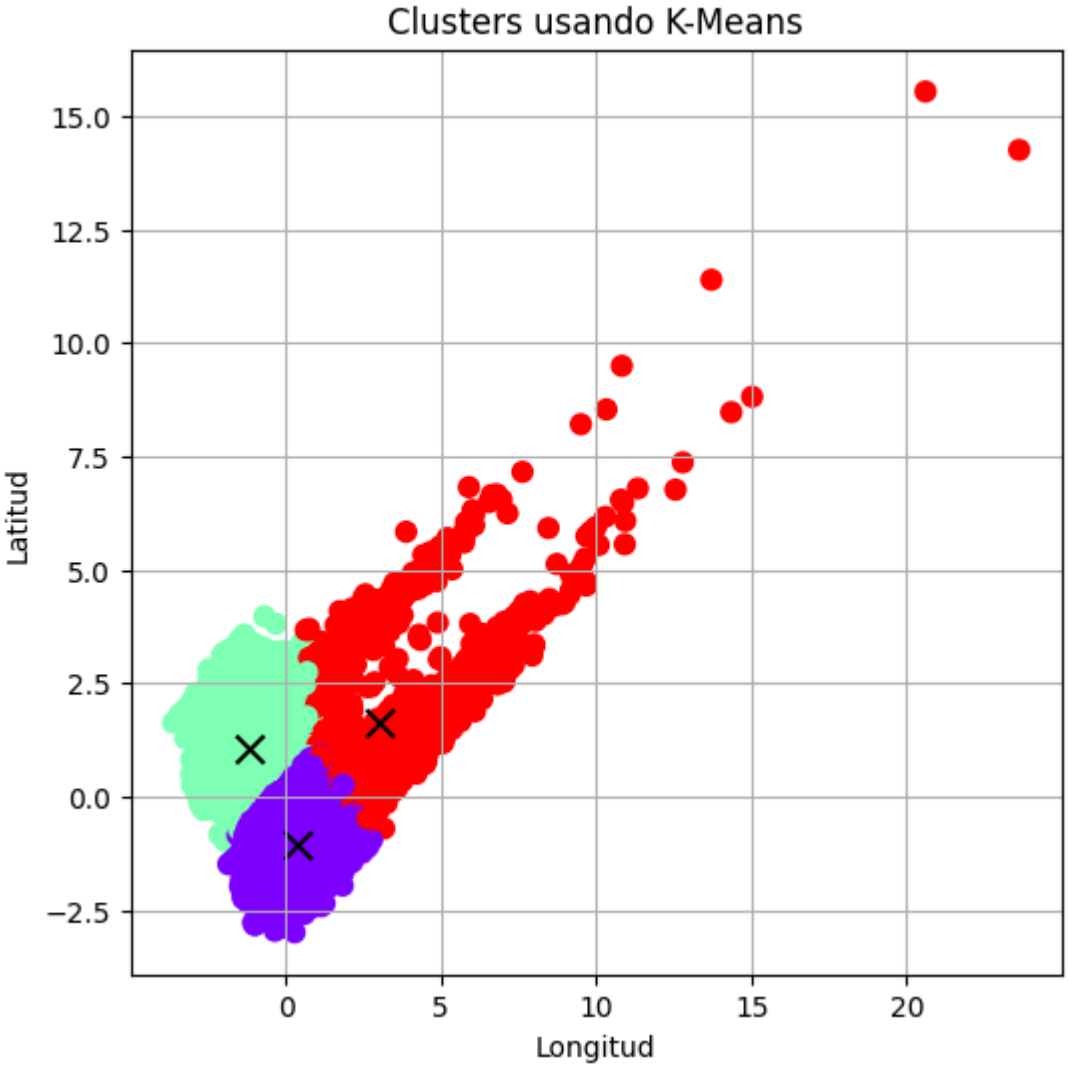
Ocurre como en el caso anterior, puede observar como un candidato a ser el número de clústers es 3 pero no se ve claramente , vamos a calcular el Silhouette Score paar verificarlo

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Vemos como basándonos en el Silhouette Score el número de clúster es k = 3, con un valor de 0.41 más elevado que en el método anterior sin PCA

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Conclusión y análisis de resultados:

En este apartado de la actividad hemos implementado un algoritmo de Clustering sobre los datos no etiquetados de las viviendad en California. El K-Means que hemos implementado ha sido en uno de los casos sin reducción de las componentes principales y en otro con reducción de componentes pincipales. En primera instancia debemos resaltar que todo ha sido relizado sobre datos estandarizados. Una vez realizada esta aclaración vamos a proceder al análisis de los resultados.

- En el primer caso, sin implementar PCA, vemos que la representación gráfica del Silhouette Score es buena, pues posee un valor que está por encima de todos y nos permite deducir el número de clusters, pero en el segundo caso donde hemos relizado una reducción de las componentes principales vemos como esta gráfica es más clara aún y además el valor del número de clúster que será 3 es el mismo que en el caso anterior pero con un valor más elevado, lo cual es lo que nos interesa pues el valor más optimo del Silhouette Score debe ser lo mas cercano a 1 posible.
- Como segunda observación y más evidente es el agrupamiento o clustering realizado con el modelo. En el primer caso sin aplicar PCA vemos como cuando relizamos el clustering los datos no se ven agrupados en clústers bien definidos,

siendo los centroides puntos sin aparentemente mucho sentido. En cambio, tras realizar PCA en el segundo ejemplo vemos como si se aprecia un agrupamiento mucho más coherente, con tres clusters diferenciados centrados en unos centroides bien definidos.

Conclusión: El uso de PCA sobre datos estandarizados nos permite que el algoritmo de aprendizaje no supervisado K-Means pueda segmentar mejor los datos sobre la viviendo en California en función de la longitud y la latitud de las mismas, permitiendo diferenciar tres grupos distintos de viviendas en función de estas variables ya mencionadas. A diferencia del clustering sin PCA el cual no nos permite establecer 3 grupos distintos de viviendas sino aparentemente solo 2 con datos de un tercer grupo dispersos entre esos dos clusters que se definen claramente.

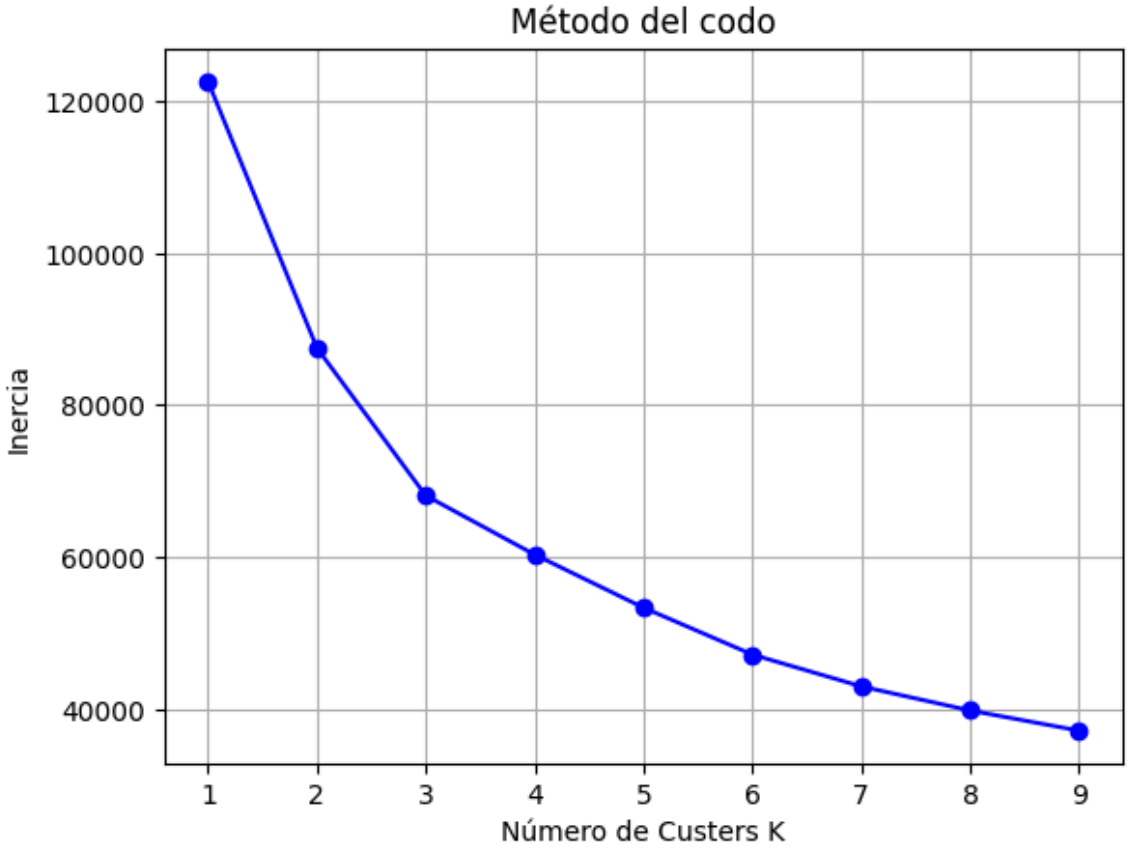
2. Implementar cada uno de los casos anteriores con y sin escalamiento:

In [5]:
```
# ================================================================================
# Comenzamos importado las librerías necesarias
# ================================================================================
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

#================================================================================
# Importación y preprocesamiento de los datos
#================================================================================

#Importamos los datos
df = pd.read_csv("housing.csv")

# Eliminamos las filas con valores nulos en caso de que existieran
df = df.dropna()

# Seleccionamos las características que tendremos en cuenta para crear los clust
X = df.drop(columns=["ocean_proximity", "total_rooms", "total_bedrooms", "median

# Estandarizamos los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Como método de aprendizaje automático no supervisado para realizar clsutering e

#Importamos el clasificador y estimamos el número de vecinos que tendremos en cu

#================================================================================
# Estimación del número de clústers
#================================================================================
# Método del codo para calcualr el número óptimo de Clústers k mediante el uso d
inercias = []
K_range = range(1,10)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inercias.append(kmeans.inertia_)
```

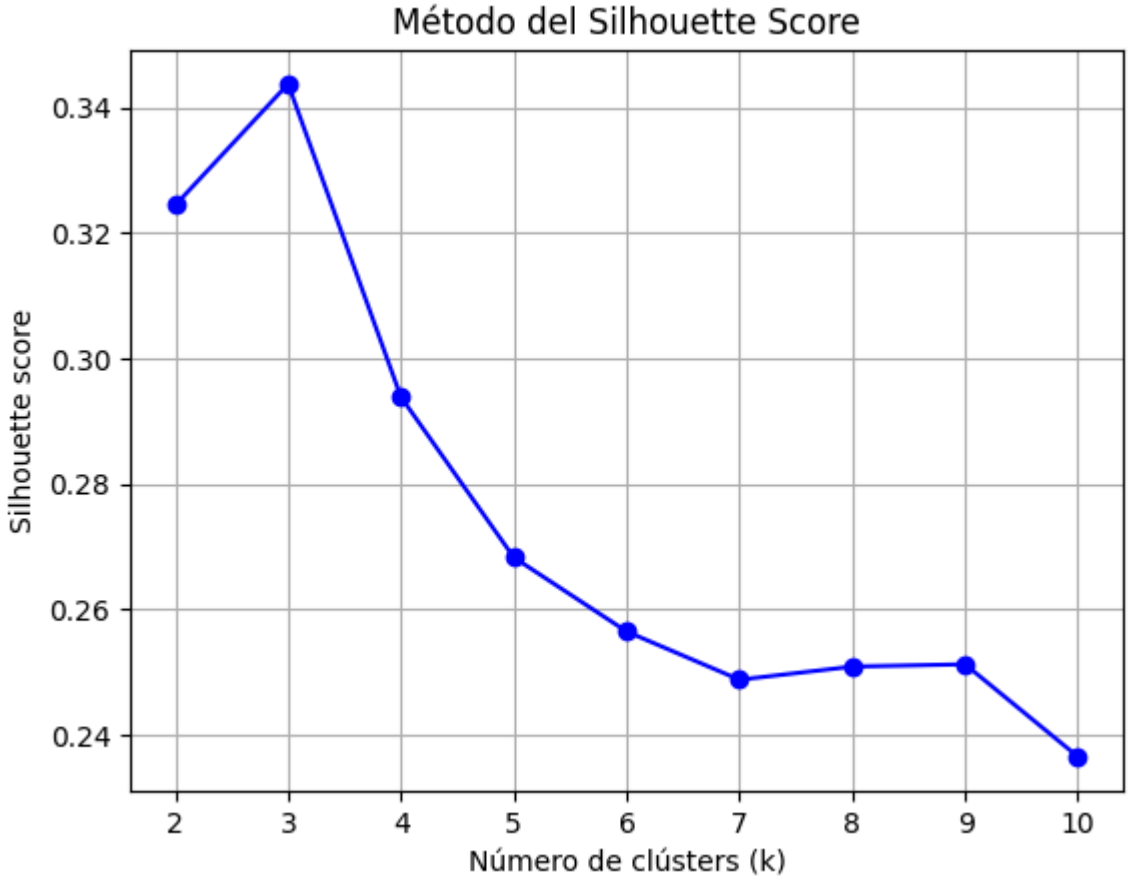
```
plt.plot(K_range, inercias, "bo-")
plt.xlabel("Número de Custers K")
plt.ylabel("Inercia")
plt.title("Método del codo")
plt.grid(True)
plt.show()

#Este ejemplo es sin estandarizar los datos, por lo que contribuye a que no se a
# que podemos ver un cambio en la curvatura a partir de k= 3, vamos a calcular e

# Calculamos el Silhouette Score:
from sklearn.metrics import silhouette_score

silhouette_scores = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)


plt.plot(range(2, 11), silhouette_scores, "bo-")
plt.xlabel("Número de clústers (k)")
plt.ylabel("Silhouette score")
plt.title("Método del Silhouette Score")
plt.grid(True)
plt.show()


#Este nos permite ver que el valor más optimo para el número de clusters es k =

#=============================================================================
# Clustering K-Means sin PCA con datos sin estandarizar
#=============================================================================

kmeans_sin_escl = KMeans(n_clusters=3, random_state=42)
labels_sin_escl= kmeans_sin_escl.fit_predict(X)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=labels_sin_escl, cmap="rainbow", s=50)
plt.scatter(kmeans_sin_escl.cluster_centers_[:, 0], kmeans_sin_escl.cluster_cent
plt.title("Clusters usando K-Means sin PCA y sin estandarizar los datos")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()

#=============================================================================
# Clustering K-Means sin PCA con datos estandarizados
#=============================================================================
#A continuación definimos el clasificador para realizar el clustering con K-Mean
kmeans_sin_pca = KMeans(n_clusters=3, random_state=42)
labels_sin_pca = kmeans_sin_pca.fit_predict(X_scaled)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels_sin_pca, cmap="rainbow", s=
plt.scatter(kmeans_sin_pca.cluster_centers_[:, 0], kmeans_sin_pca.cluster_center
```

```
plt.title("Clusters usando K-Means sin PCA con datos estandarizados")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()
```

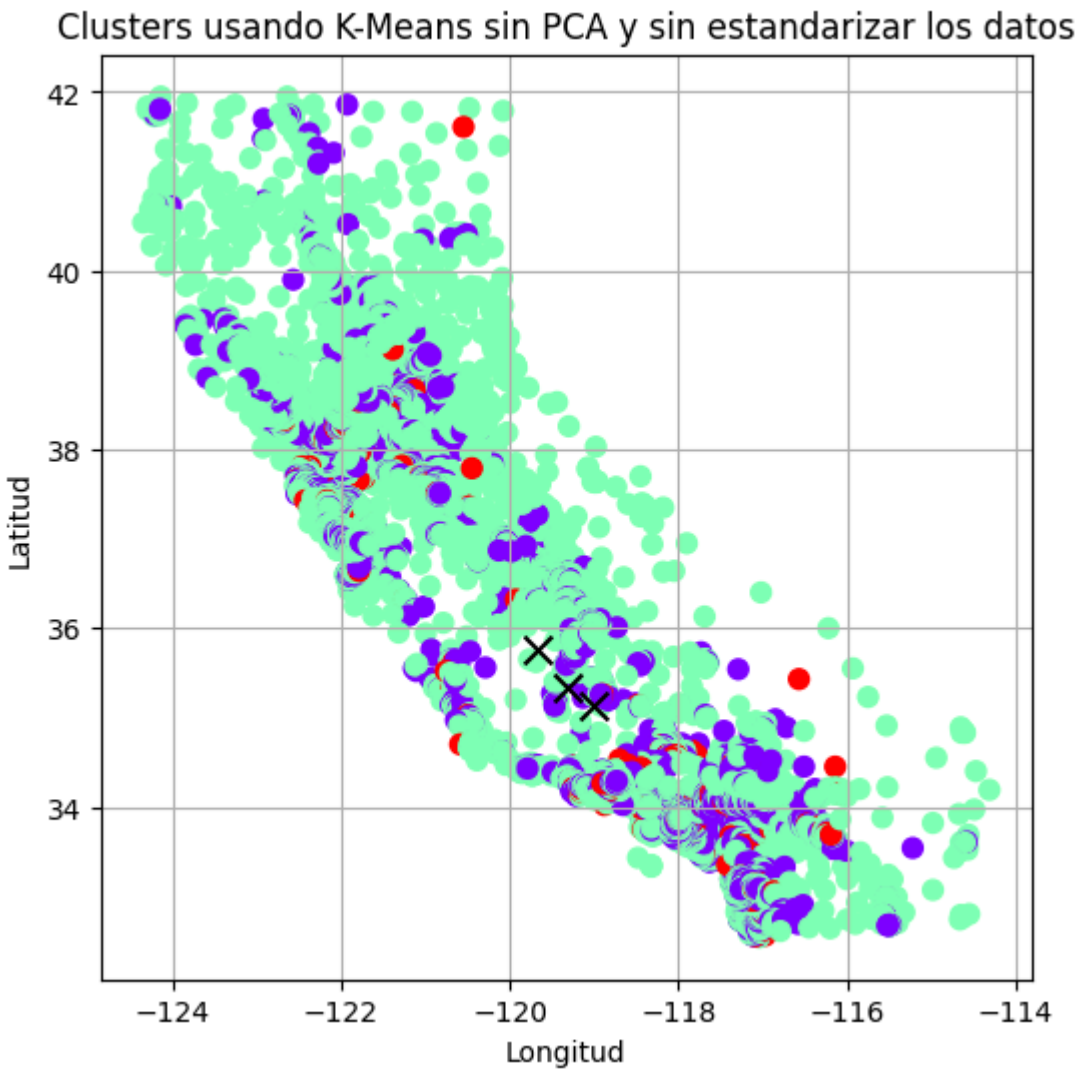
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Método del codo
120000
100000
80000
60000
40000
Inercia
1
2
3
4
5
6
7
8
9
Número de Custers K

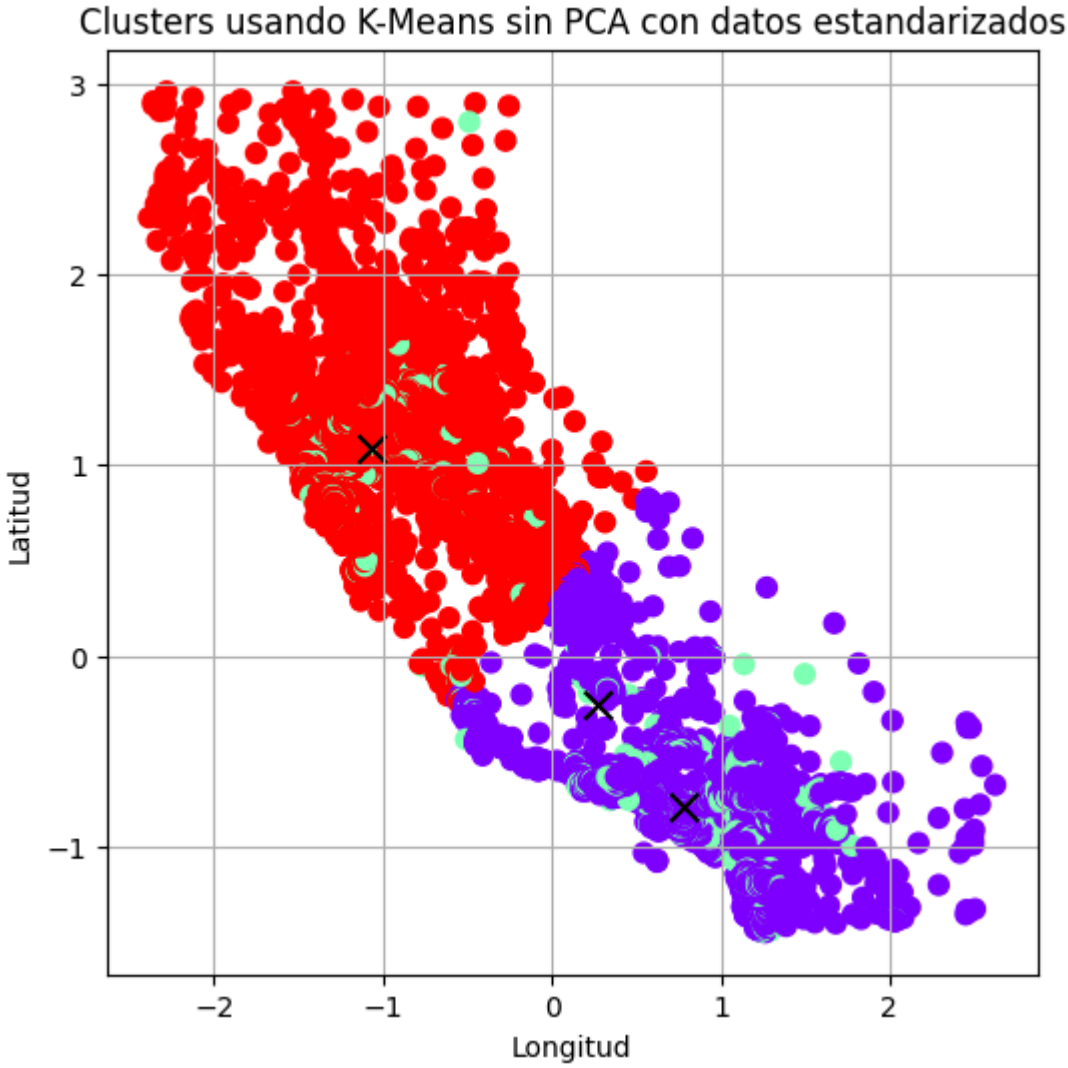
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Conslusión y Análisis de Resultados

- Vemos en la representación gráfica que aunque el método de aprendizaje automático no supervisado de K-Means no es muy eficiente sin estandarizar los datos y sin aplicar la reducción de componentes principales. Pero vemos como sobre estos casos donde no se aplica reducción de componentes principales estandarizar los datos es crucial, pues en la representación gráfica vemos como en el primer caso sin estandarizar los datos no se distinguen ningun tipo de cluster. Además los centroides se encuentran muy cercanos entre sí sin capturar los datos.

- Mientras que cuando estandarizamos los datos vemos como aunque el agrupamiento no es perfecto los datos si se agrupan en dos centroides prncipalmente y no se condensan los mismos centroides uno al lado de otros. Hay que resaltar que nos interesa que los centroides se encuentren los más alejados posibles y con los datos agrupados sin que se solapen unos con otros como en este caso.

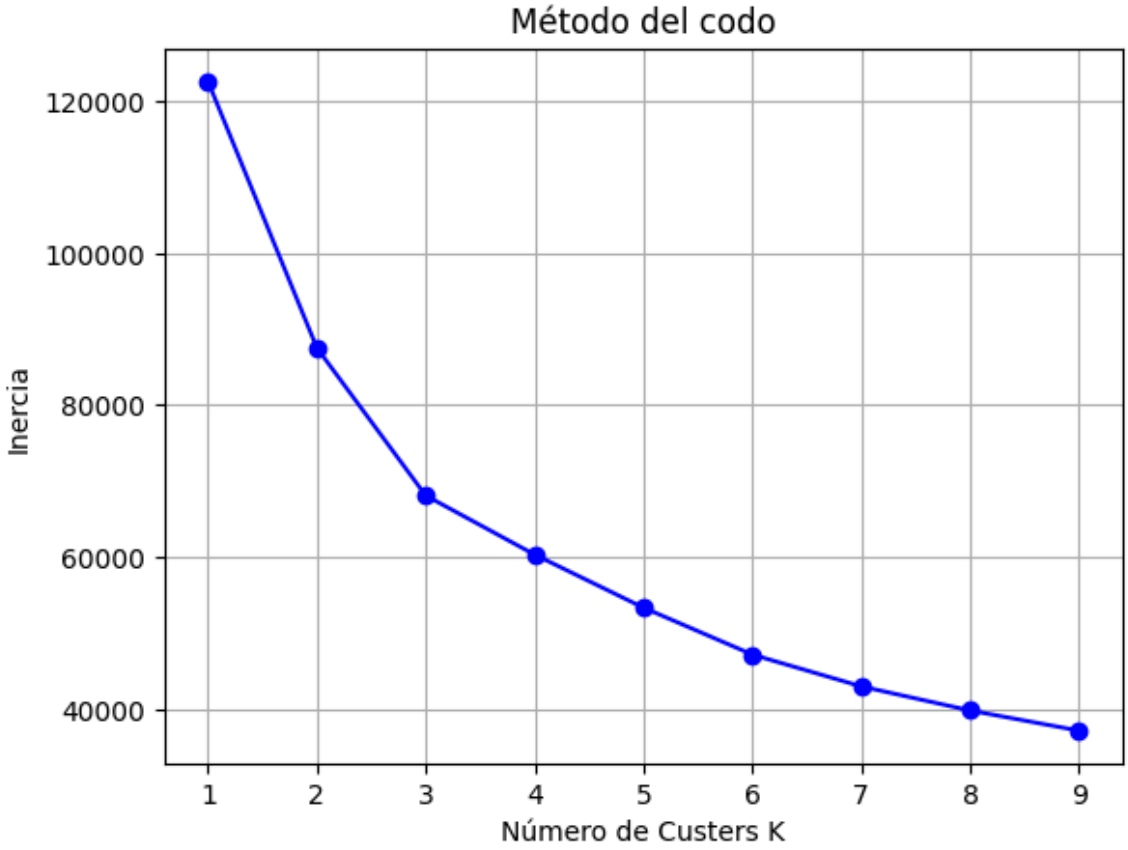
Conslusión: Estandarizar los datos siempre será conveniente y en nuestro caso al estandarizar los datos vemos como podemos diferenciar dos grupos de viviendas en California en función de la latitud y longitud de las viviendas de forma clara.

In [6]:
```
#K-Means con PCA con datos estandarizados y sin datos estandarizados
# ================================================================================
# Comenzamos importado las librerías necesarias
# ================================================================================
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

#================================================================================
# Importación y preprocesamiento de los datos
#================================================================================

#Importamos los datos
df = pd.read_csv("housing.csv")

# Eliminamos las filas con valores nulos en caso de que existieran
df = df.dropna()

# Seleccionamos las características que tendremos en cuenta para crear los clust
X = df.drop(columns=["ocean_proximity", "total_rooms", "total_bedrooms", "median

# Estandarizamos los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Como método de aprendizaje automático no supervisado para relizar clustering em

#Importamos el clasificador y estimamos el número de vecinos que tendremos en cu

#================================================================================
# Estimación del número de clusters
#================================================================================
# Método del codo para calcualr el número óptimo de Clústers k mediante el uso d
inercias = []
K_range = range(1,10)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inercias.append(kmeans.inertia_)

plt.plot(K_range, inercias, "bo-")
plt.xlabel("Número de Custers K")
plt.ylabel("Inercia")
plt.title("Método del codo")
plt.grid(True)
plt.show()

#Este ejemplo es sin estandarizar los datos, por lo que contribuye a que no se a
# que podemos ver un cambio en la curvatura a partir de k= 3, vamos a calcular e

# Calculamos el Silhouette Score:
from sklearn.metrics import silhouette_score

silhouette_scores = []
```

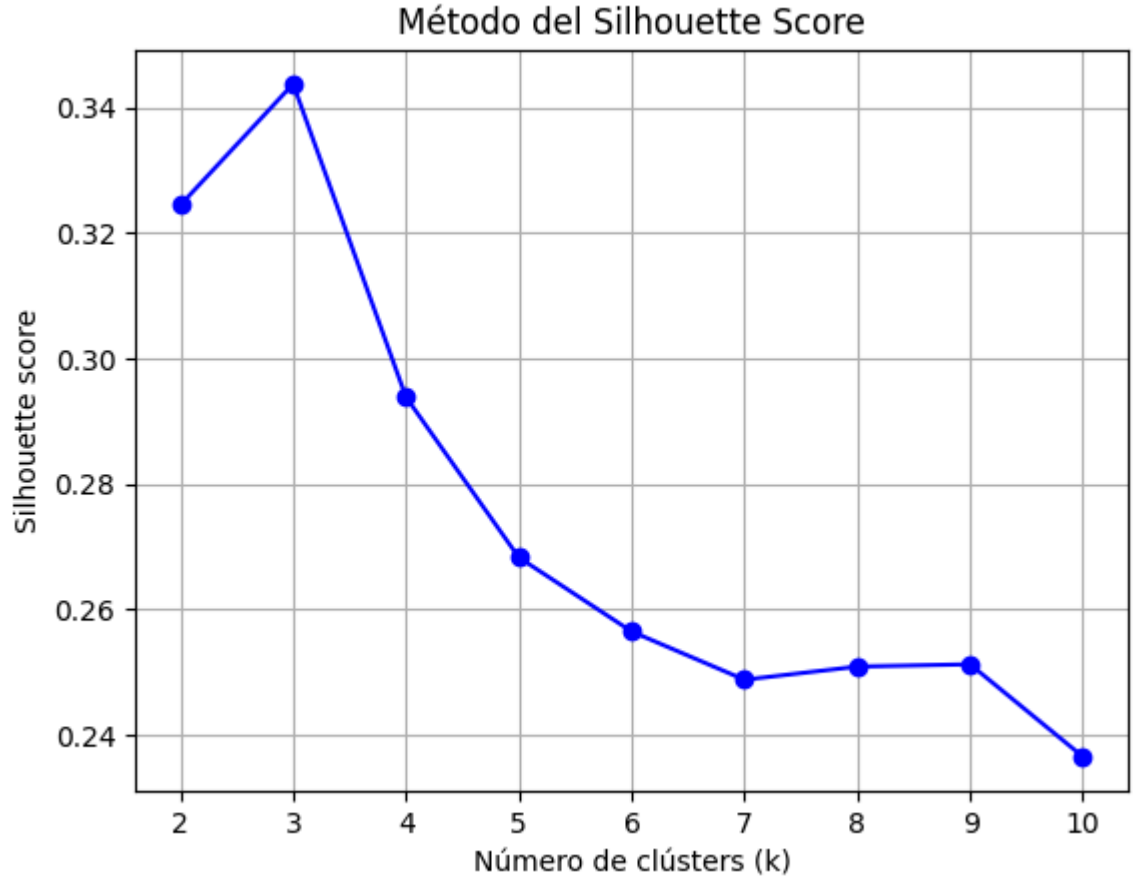
```
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels)
    silhouette_scores.append(score)


plt.plot(range(2, 11), silhouette_scores, "bo-")
plt.xlabel("Número de clústers (k)")
plt.ylabel("Silhouette score")
plt.title("Método del Silhouette Score")
plt.grid(True)
plt.show()


#Este nos permite ver que el valor más optimo para el número de clusters es k =

#=============================================================================
#Aplicamos la reducción de componentes principales (PCA) sobre datos no estandar
#=============================================================================
from sklearn.decomposition import PCA

# Creamos el objeto PCA
pca_1 = PCA(n_components=3)
pca_result_1 = pca_1.fit_transform(X)

#Imprimimos las varianzas en cada eje
print(pca_1.explained_variance_ratio_)

# El resultado es [0.36777967 0.29960122 0.17461867 0.13223773 0.01536309 0.0103
#Vemos como las cuatro componentes primeras componentes principales explican cas
#vamos a relizar la representación gráfica de la varianza explicada:

#Varianza explicada
plt.plot(np.cumsum(pca_1.explained_variance_ratio_), marker = "o")
plt.title("Varianza explicada acumulada")
plt.xlabel("Número de Componentes")
plt.ylabel("Varianza acumulada")
plt.show()


#=============================================================================
# Aplicamos el K-Means con PCA y con datos sin estandarizar
#=============================================================================

kmeans_pca_sin_est = KMeans(n_clusters=3, random_state=42)
labels = kmeans_pca_sin_est.fit_predict(pca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(pca_result_1[:, 0], pca_result_1[:, 1], c=labels, cmap="rainbow", s=
plt.scatter(kmeans_pca_sin_est.cluster_centers_[:, 0], kmeans_pca_sin_est.cluste
plt.title("Clusters usando K-Means con PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()
```

```
#=============================================================================
#Aplicamos la reducción de componentes principales (PCA)
#=============================================================================
from sklearn.decomposition import PCA

# Creamos el objeto PCA
pca = PCA(n_components=3)
pca_result = pca.fit_transform(X_scaled)

#Imprimimos las varianzas en cada eje
print(pca.explained_variance_ratio_)

# El resultado es [0.36777967 0.29960122 0.17461867 0.13223773 0.01536309 0.0103
#Vemos como las cuatro componentes primeras componentes principales explican cas
#vamos a relizar la representación gráfica de la varianza explicada:

#Varianza explicada
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker = "o")
plt.title("Varianza explicada acumulada")
plt.xlabel("Número de Componentes")
plt.ylabel("Varianza acumulada")
plt.show()


#A continuación definimos el clasificador para realizar el clustering con K-Mean
#=============================================================================
# Aplicamos el K-Means con PCA y con datos estandarizados
#=============================================================================
#A continuación definimos el clasificador para realizar el clustering con K-Mean
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=labels, cmap="rainbow", s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c="bla
plt.title("Clusters usando K-Means con PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()
```

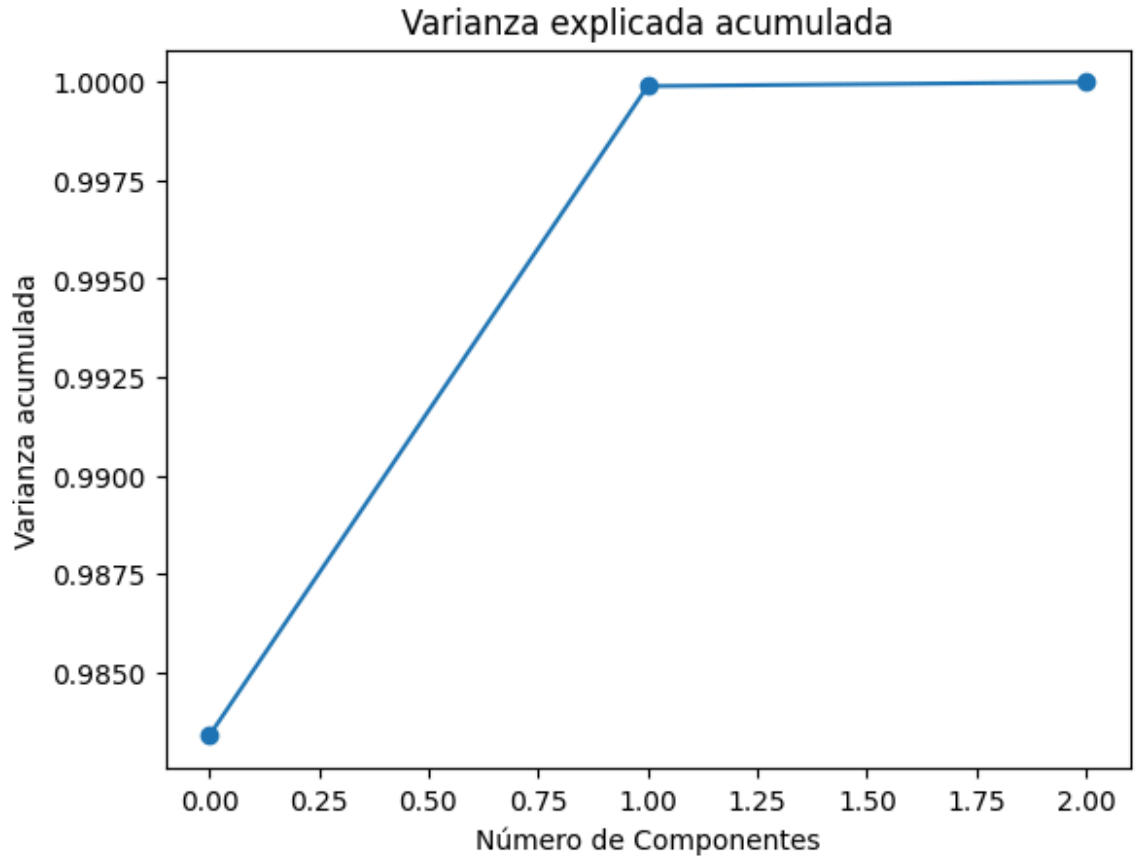
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Método del codo
120000
100000
80000
60000
40000
Inercia
1
2
3
4
5
6
7
8
9
Número de Custers K

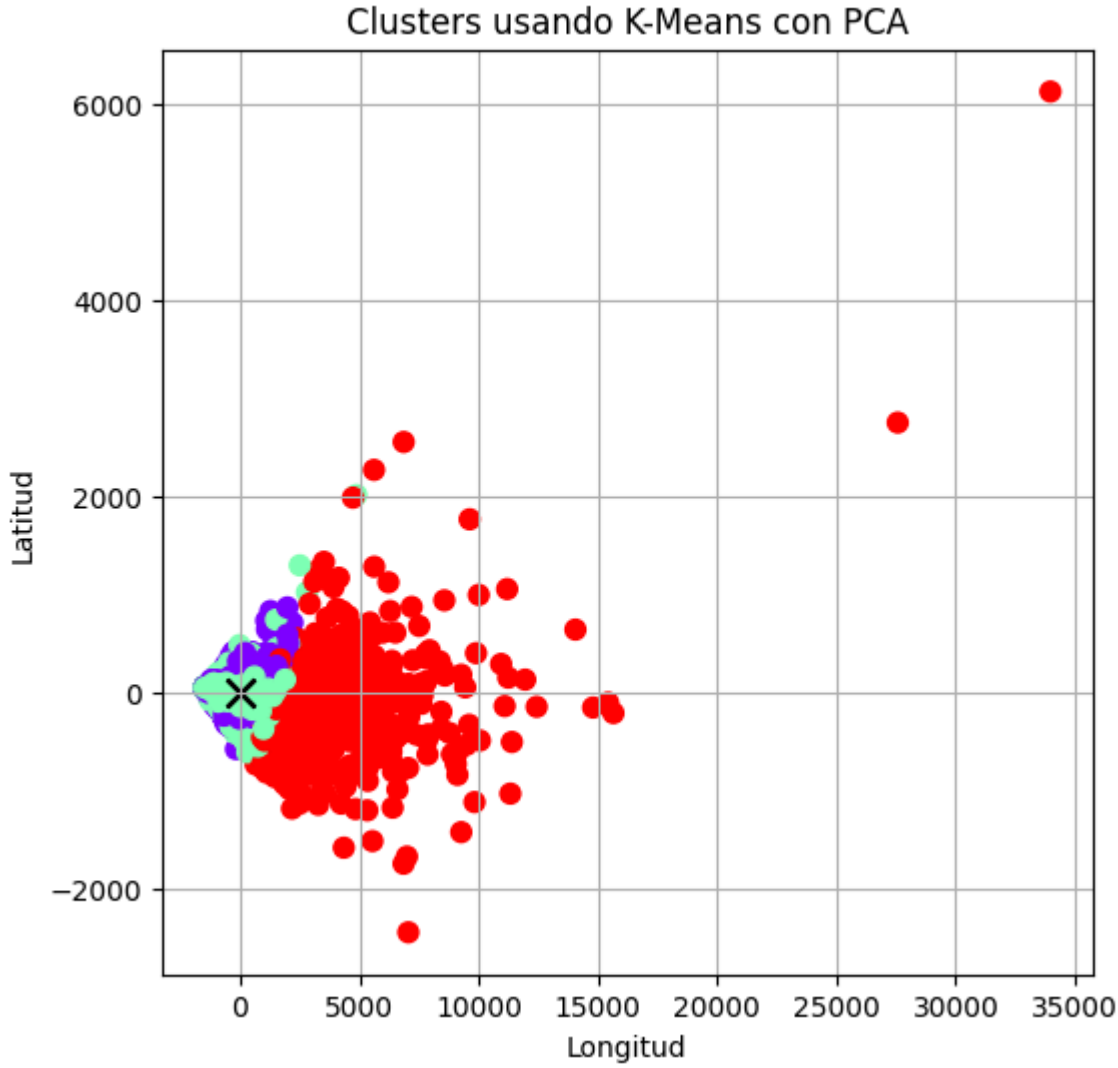
```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



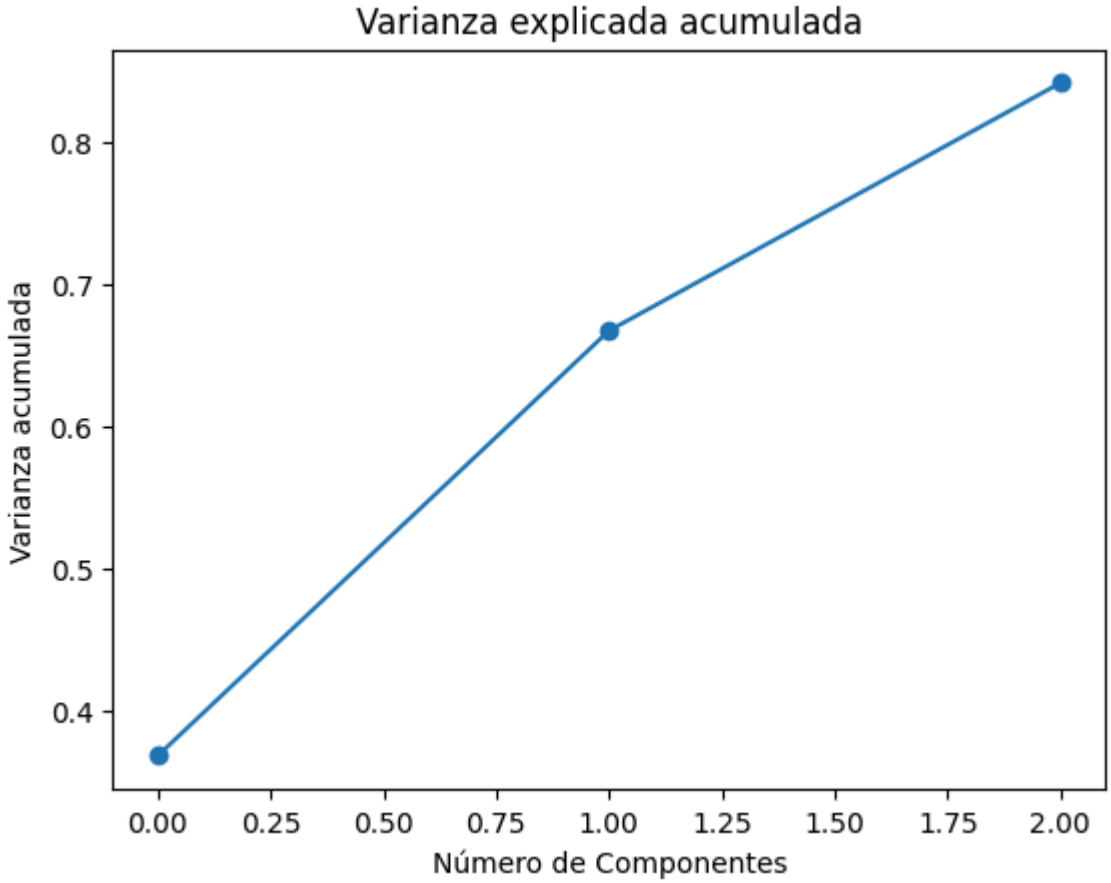

```
[9.83380505e-01 1.65106806e-02 1.00457062e-04]
```



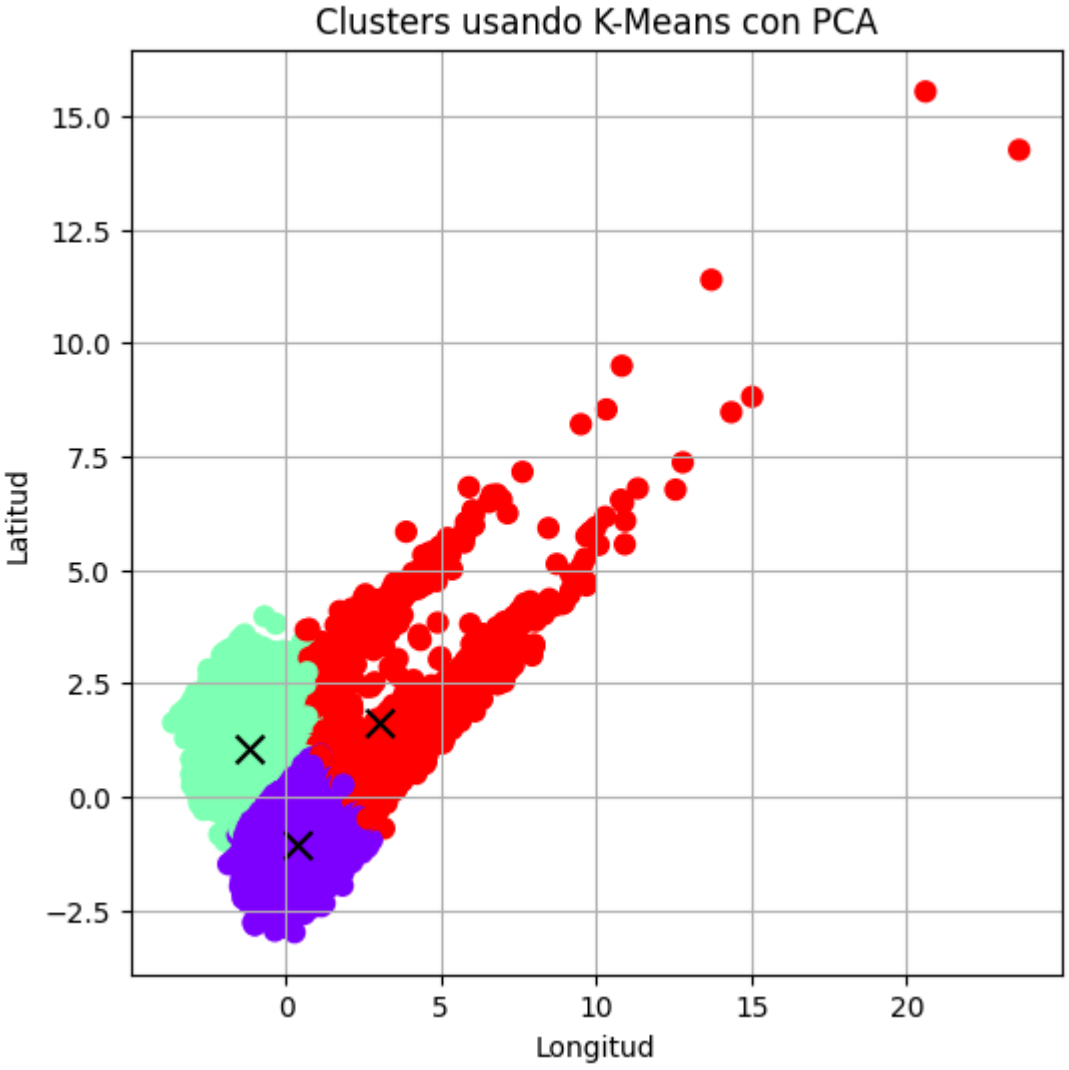

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
[0.36777967 0.29960122 0.17461867]
```



```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Conclusión y Análisis de resultados:

- Al igual que en los caoss anteriores vemos como al estandarizar los datos la gráfica del clustering es más coherente. Al aplicar en el primer caso el K-Means con PCA sin estandarizar los datos vemos como nos devuelve una gráfica con un clúster en cero y datos superpuestos unos con otros de los cuales no podemos deducir nada.
- En cambio vemos como cuando aplicamos la estandarización de los datos sobre el msimo caso es evidente la segmentación de la vivienda en California, con tres clusters diferenciados claramente y con tres centroides claramente representados.

Conclusión: Estandarizar los datos cuando aplicamos un algoritmo de aprendizaje no supervisado como K-Means es fundamental para poder

diferenciar lso clústers de las viviendas en California en función de las variables "Longitud" y "Latitud".

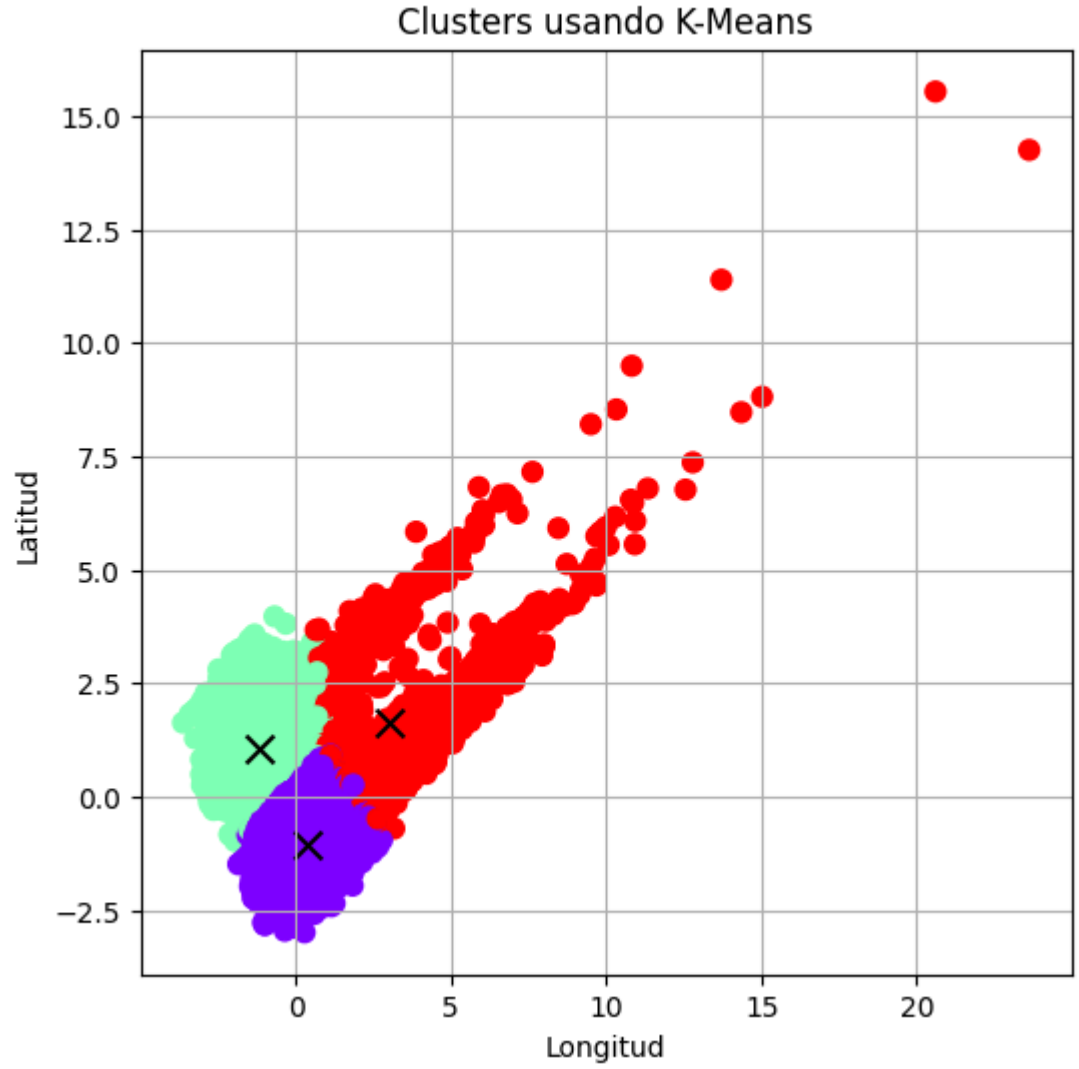
3. Comparar los dos métodos de clustering que vimos en clase y al menos dos de PCA (también de los que vimos en clase).

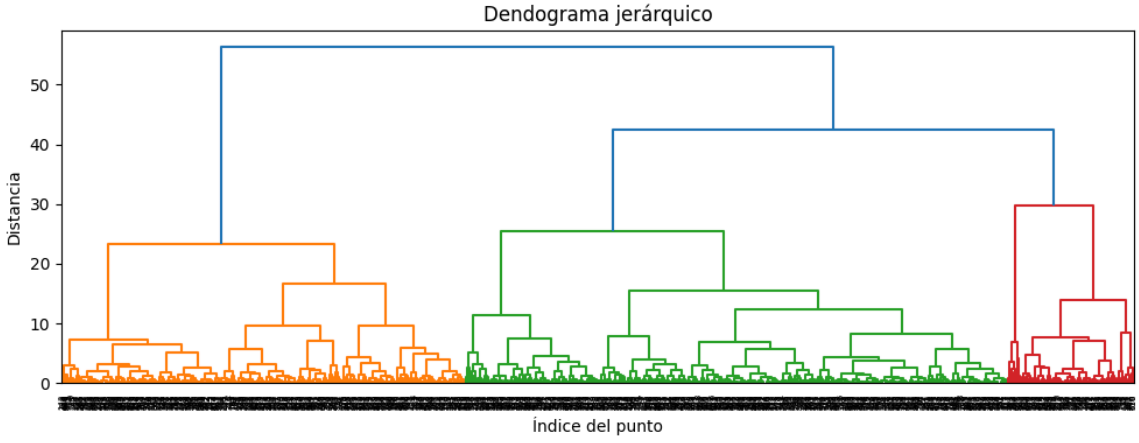
- En este apaartado vamos a comparar los dos métodos de Clustering como son el K-Means y el Clustering Jerárquico asi como los dos métodos de reducción de componentes principales el PCA y el Sparse PCA

In [7]:
```
#=============================================================================
#Comparación de Clustering Jerárquico y de K-Means con PCA
#=============================================================================
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

#=============================================================================
#Preprocesamiento y escalado de los datos
#=============================================================================
#Importamos los datos
df = pd.read_csv("housing.csv")

# Eliminamos filas con valores nulos
df = df.dropna()

# Seleccionamos las características que tendremos en cuenta para crear los clust
X = df.drop(columns=["ocean_proximity", "total_rooms", "total_bedrooms", "median

# Escalamos los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#=============================================================================
# Aplicamos PCA
#=============================================================================
from sklearn.decomposition import PCA

# Creamos el objeto PCA
pca = PCA(n_components=3)
pca_result = pca.fit_transform(X_scaled)

#=============================================================================
# K-Means
#=============================================================================
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=labels, cmap="rainbow", s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c="bla
```

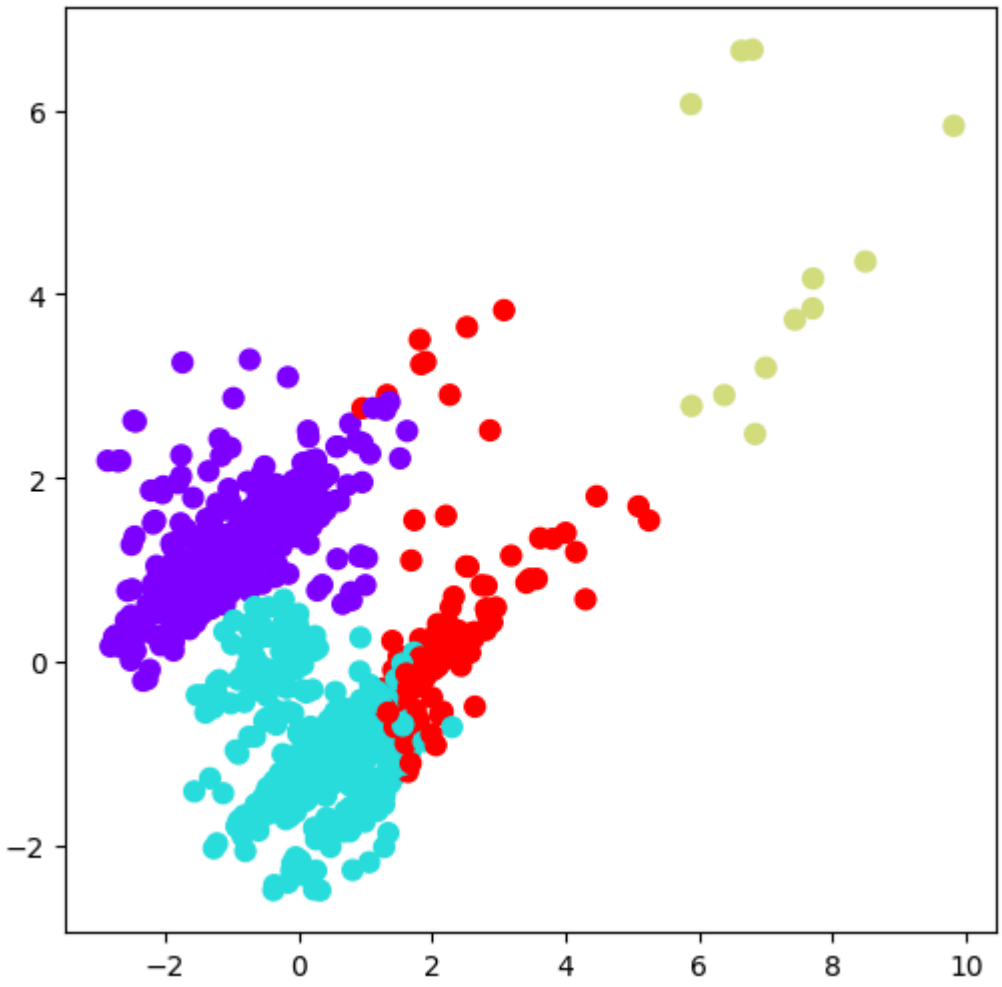
```
plt.title("Clusters usando K-Means")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()

#=============================================================================
#Clustering Jerárquico (Método de Varianza mínima de Ward)
#=============================================================================
#Reducimos le numero de muestras porque no es eficiente para garndes cantidades
X_sample = pca_result[np.random.choice(pca_result.shape[0], size=1000, replace=F

from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Clustering jerárquico
Z = linkage(X_sample, method="ward")

# Mostramos el dendrograma
plt.figure(figsize=(10, 4))
dendrogram(Z)
plt.title("Dendograma jerárquico")
plt.xlabel("Índice del punto")
plt.ylabel("Distancia")
plt.tight_layout()
plt.show()

#Obtenemos las etiquetas del cluster
labels = fcluster (Z, t=3, criterion = "maxclust")

#Visualización simple en 2D
plt.figure(figsize =(6,6))
plt.scatter(X_sample[:,0],X_sample[:,1], c= labels, cmap = "rainbow", s= 50)

# NOTA: Hace un agrupamiento bueno, pero hemos tenido que limitar el número de m
```

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```





Out[7]: <matplotlib.collections.PathCollection at 0x2823340da10>



Conclusión y análisis de resultados:

- En cuanto a la representaciones de ambos, vemos como la represnetacón del modelo de K-Means es más clara a la hora de diferenciar los clusters que el Clustering Jerárquico. Podemos ver gracias la representación en 2 dimensiones del clustering jerárquico que proporciona una agrupación que no es del todo incorrecta, diferenciando tres clusters aparentemente, pero no quedan muy bien definidos porque hay muchos datos que se solapan e incluso que se clasifican erróneamente. A diferencia del modelo de K-Means que como hemos visto en ejemplos anteriores diferencia tres clusters y los datos están separados diferenciando claramente los centroides.

- Otro punto a tener en cuenta es la eficiencia comptacional, pues el K-Means se ha ejecutado al momento, con un tiempo de ejecución muy bueno. Pero el Clustering Jerárquico no ha cargado con todos los datos e incluso he tenido que limitar el número de muestras para que me construya el dendograma el cual es muy profundo y no permite distinguir claramente el número óptimo de clusters.

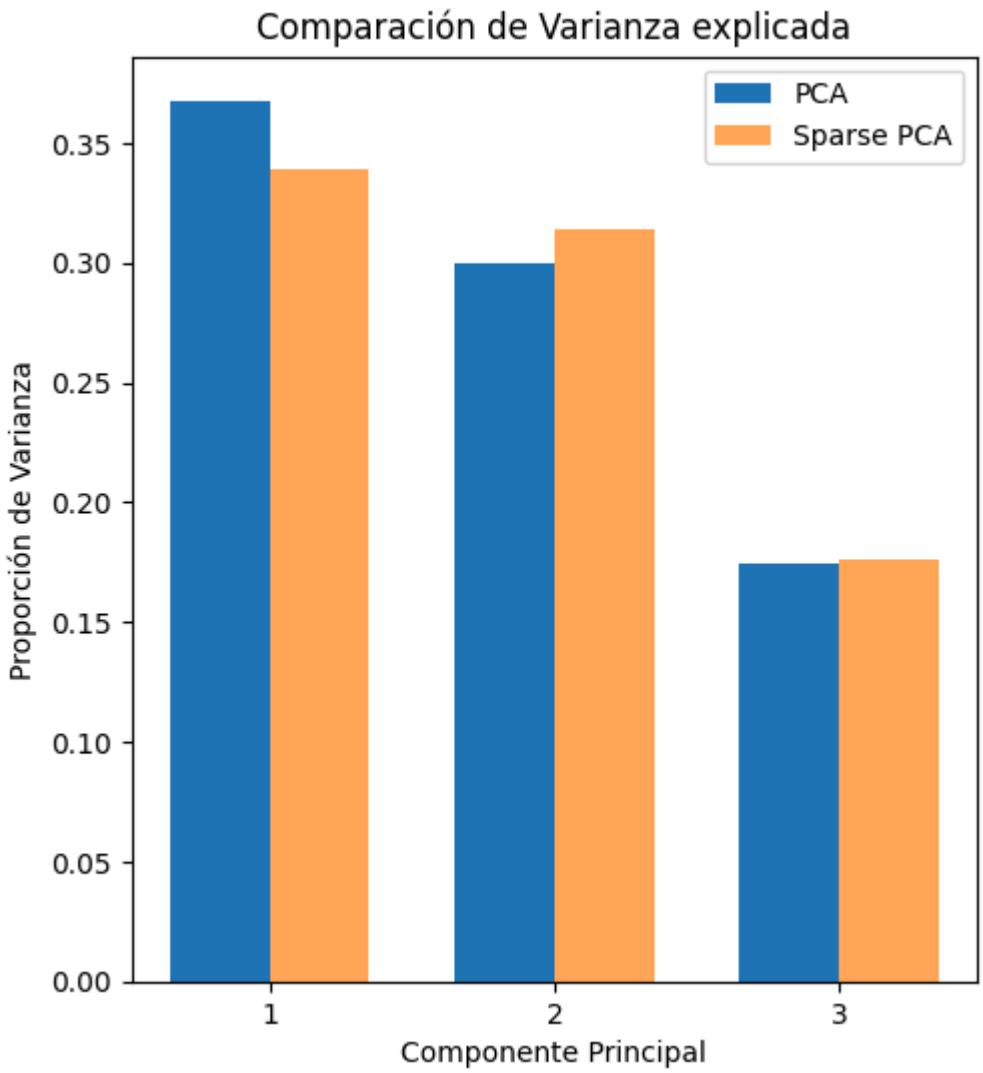
Conclusión: El método de K-Means es más eficiente que el Cluster Jerárquico para conjuntos de datos grandes. En nuestro problema de agrupamiento el K-Means nos permitirá establecer mejor los diferentes grupos de viviendas en California según su longitud y latitud frente al Clustering jerárquico.

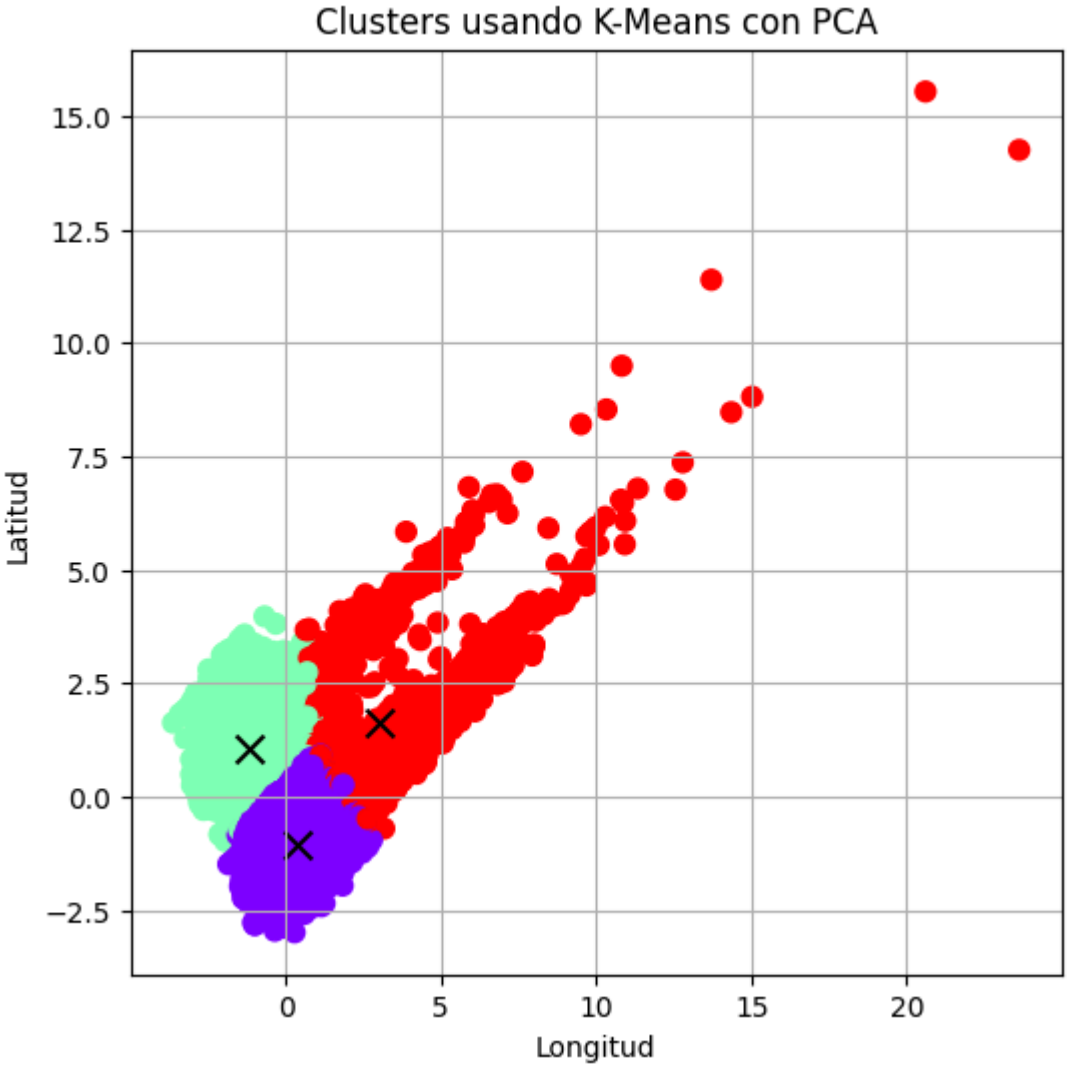
In [15]:
```
#=============================================================================
#Implementación de K-Means empleando PCA y Sparse PCA
#=============================================================================

import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA,SparsePCA
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

#=============================================================================
#Preprocesamiento y estandarizacón de los datos
#=============================================================================
#Importamos los datos
df = pd.read_csv("housing.csv")

# Eliminamos filas con valores nulos
df = df.dropna()

# Seleccionamos las características que tendremos en cuenta para crear los clust
X = df.drop(columns=["ocean_proximity", "total_rooms", "total_bedrooms", "median

# Estandarizamos los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#=============================================================================
# Aplicamos PCA sobre los datos estandarizados
#=============================================================================
from sklearn.decomposition import PCA

# Creamos el objeto PCA
pca = PCA(n_components=3)
pca_result = pca.fit_transform(X_scaled)

#=============================================================================
# Aplicamos Sparse PCA sobre los datos estandarizados
#=============================================================================
spca = SparsePCA (n_components = 3, alpha = 1, random_state = 42)
spca_result = spca.fit_transform(X_scaled)

#Calculamos varianza explicada aproximada para Sparse PCA
spca_varianza = np.var(spca_result,axis = 0)
spca_varianza_total = np.var(X_scaled,axis=0).sum()
spca_varianza_ratio = spca_varianza / spca_varianza_total

print("\nVarianza explicada (aproximada) por cada compoennete (Sparse PCA):")
print(spca_varianza_ratio)
print("Suma total de varianza explicada (aproximada):", np.sum(spca_varianza_rat

#Gráfico de comparación de varianza explicada
x = np.arange(1,4)
width = 0.35
plt.figure(figsize =(12,6))
```

```
#=============================================================================
#Comparación de ambas reducciones de componentes principales
#=============================================================================
# Subplot izquierda: barras comparativas
plt.subplot(1,2,1)
plt.bar(x - width / 2, pca.explained_variance_ratio_, width = width, label = "PC
plt.bar(x + width / 2, spca_varianza_ratio, width = width, alpha = 0.7, label =
plt.title("Comparación de Varianza explicada")
plt.xlabel("Componente Principal")
plt.ylabel("Proporción de Varianza")
plt.xticks(x)
plt.legend()

#=============================================================================
#Implementamos el K-Means con ambas reducciones de componentes principales
#=============================================================================

#=============================================================================
# K-Means con PCA
#=============================================================================
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=labels, cmap="rainbow", s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c="bla
plt.title("Clusters usando K-Means con PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()

#=============================================================================
# K-Means con Sparse PCA
#=============================================================================
kmeans_1 = KMeans(n_clusters=3, random_state=42)
labels_1 = kmeans_1.fit_predict(spca_result)

# Visualización usando las dos primeras características
plt.figure(figsize=(6,6))
plt.scatter(spca_result[:, 0], spca_result[:, 1], c=labels_1, cmap="rainbow", s=
plt.scatter(kmeans_1.cluster_centers_[:, 0], kmeans_1.cluster_centers_[:, 1], c=
plt.title("Clusters usando K-Means con Sparse PCA")
plt.xlabel("Longitud")
plt.ylabel("Latitud")
plt.grid(True)
plt.show()
```

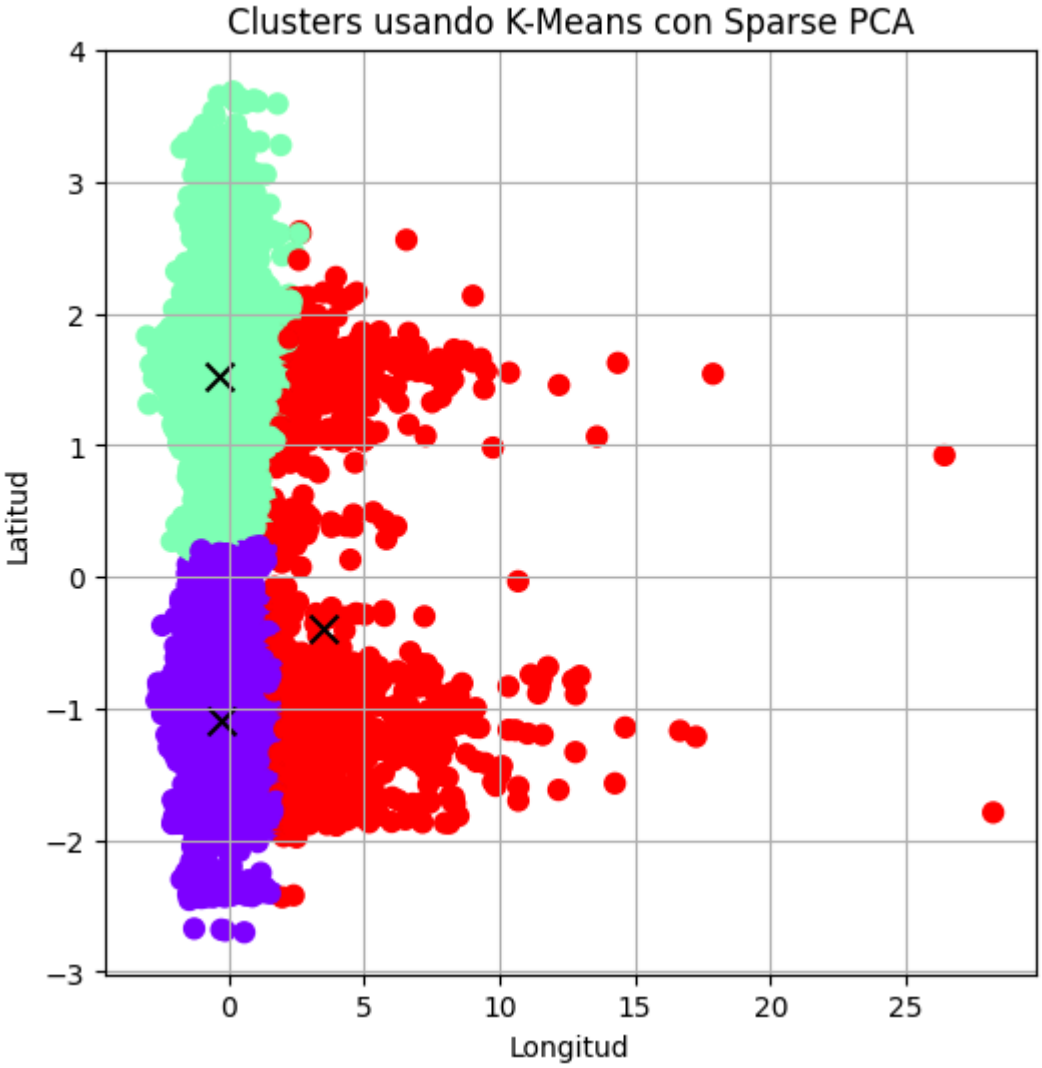
```
Varianza explicada (aproximada) por cada compoennete (Sparse PCA):
[0.33906347 0.31423394 0.17595048]
Suma total de varianza explicada (aproximada): 0.8292478866739799
```

```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to s
uppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Comparación de Varianza explicada
PCA
Sparse PCA
Proporción de Varianza
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
1
2
3
Componente Principal




```
C:\Users\Rubén Garrido\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Conclusión y análisis de los datos:

- Viendo los resultados que hemos obtenido en primera isntancia debemos resaltar la comparación entre la varianza explicada por PCA y Sparse PCA, vemos que con la gráfica podemos pensar que la varianza explicada es casi similar entre PCA y Sparse PCA. Pero en términos numéricos la varianza explicada del PCA es de un 96% aproximadamente a diferencia de la de Sparse PCA que es de un 82%. Podemos deducir por lo tanto que PCA en este caso explica mejor la varianza de los datos de latitud y logitud de la vivienda en California que Sparse PCA.

- En cuanto a las represnetaciones gráficas del K-Means con los diferentes métodos de reducción de componentes principales vemos como usar K-Means con PCA será mejor que con Sparse PCA, vemos como la represnetacion de K-Means con PCA permite diferenciar mejor los clusters y estudiar como se agrupan los datos. Mientras que Sparse PCA no divide tan bien los datos y además nos sugiere al explicar la varianza en un 82% que necesitamos otro vector principal, es decir que necesitaremos 4 clusters. Aún asi la varainza de los datos es mayor en el Sparse PCA que en modelo donde hemos usado PCA, como podemos ver en la última gráfica que hemos imprimido.

Conclusión: en nuetsro caso es mejor emplear PCA para reducir las componentes principales de nuestros datos y así obtener un método de agrupamiento mediante K-Means más eficiente y claro a la hora de representar los resultadsos. Con datos que capturan mejor la varianza de los datos y clusters mejor definidos.