

```
In [41]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
```

```
In [42]: # Definir el path al documento CSV
file_path = 'datasets/clustering_database.csv'

# Leer el documento CSV
df = pd.read_csv(file_path)

#Imprimiendo Las 5 primeras filas
df.head()
```

Out[42]:

	fullVisitorID	channelgrouping	date	OS	Apparel	Office	Electronics	LimitedSupply
0	2886674552607729793	Referral	20160901	Linux	0	0	1	
1	876940283010756174	Referral	20160901	Macintosh	1	0	0	
2	1957458976293878100	Organic Search	20160901	Windows	0	0	2	
3	3044311821343900129	Organic Search	20160901	Macintosh	0	0	0	
4	3890747985732303742	Organic Search	20160901	Macintosh	1	0	0	

```
In [43]: # Observando el tipo de información del dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38670 entries, 0 to 38669
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   fullVisitorID    38670 non-null   uint64 
 1   channelgrouping  38670 non-null   object  
 2   date              38670 non-null   int64  
 3   OS                38670 non-null   object  
 4   Apparel           38670 non-null   int64  
 5   Office             38670 non-null   int64  
 6   Electronics        38670 non-null   int64  
 7   LimitedSupply     38670 non-null   int64  
 8   Accessories        38670 non-null   int64  
 9   ShopByBrand       38670 non-null   int64  
 10  Bags               38670 non-null   int64  
 11  totalSpent_USD   38670 non-null   float64 
dtypes: float64(1), int64(8), object(2), uint64(1)
memory usage: 3.5+ MB
```

La definición de las variables son las siguientes:

- fullVisitorID : Identificador del cliente
- channelgrouping : Medio por el cual llegó al sitio web
- date : dia
- OS : Tipo de dispositivo con el que entro al sitio web
- Apparel , Office , Electronics , LimitedSupply , Accessories , ShopByBrand , Bags : Categorías de los productos comprados
- totalSpent_USD : Total gastado

Por cada variable, tenemos una cantidad de valores, veamos:

- La cantidad de medios que se tienen en channelgrouping .
- Los días estudiados y el año date .
- El tipo de dispositivo con el que se entro al sitio web con OS .

```
In [44]: # Lista de columnas para ver elementos únicos
_list = ["channelgrouping", "date", "OS"]

for column in _list:
    unique_values = df[column].unique()
    if column == "date":
        list_year_unique_values = list(set([str(value)[:4] for value in unique_
        list_month_unique_values = list(set([str(value)[4:6] for value in unique_
        list_month_unique_values.sort()
        print(f"Para la columna `{column}` se tienen que los años de estudio son:")
        print(f"y para los meses {list_month_unique_values}.\n")
    else:
        print(f"Para la columna `{column}` se tienen que los valores únicos son:")
```

Para la columna `channelgrouping` se tienen que los valores únicos son:
['Referral' 'Organic Search' 'Paid Search' 'Direct' 'Social' 'Display'
'Affiliates']

Para la columna `date` se tienen que los años de estudio son (es):
['2016']
y para los meses ['08', '09', '10', '11', '12'].

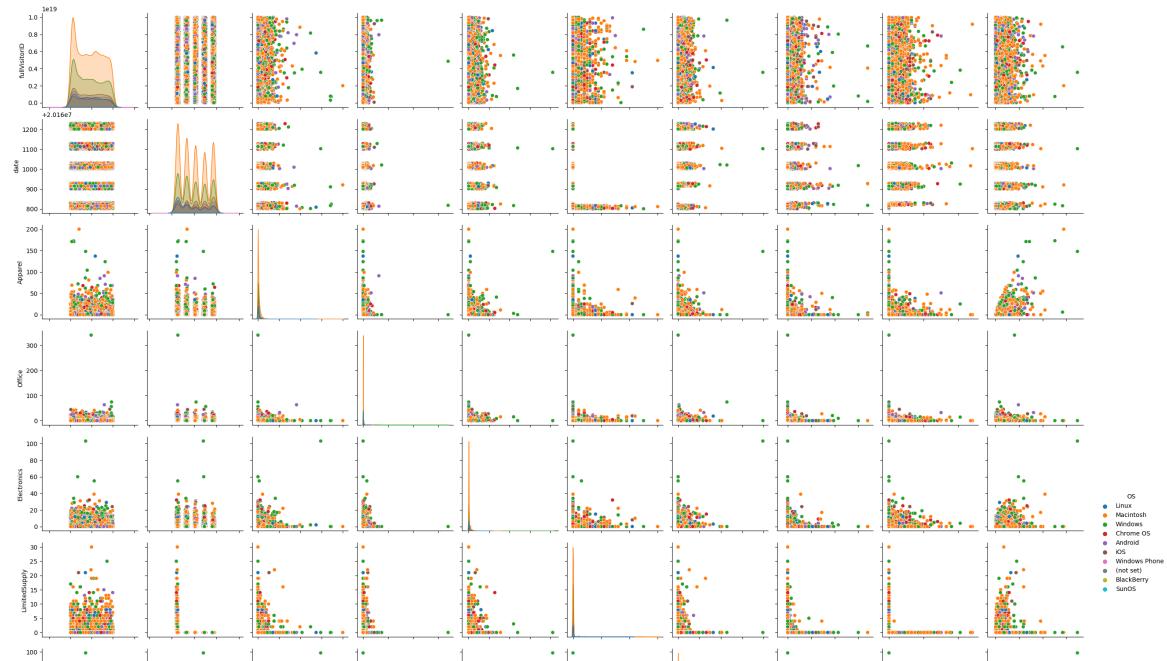
Para la columna `OS` se tienen que los valores únicos son:
['Linux' 'Macintosh' 'Windows' 'Chrome OS' 'Android' 'iOS' 'Windows Phone'
'(not set)' 'BlackBerry' 'SunOS']

In [45]: `df.describe()`

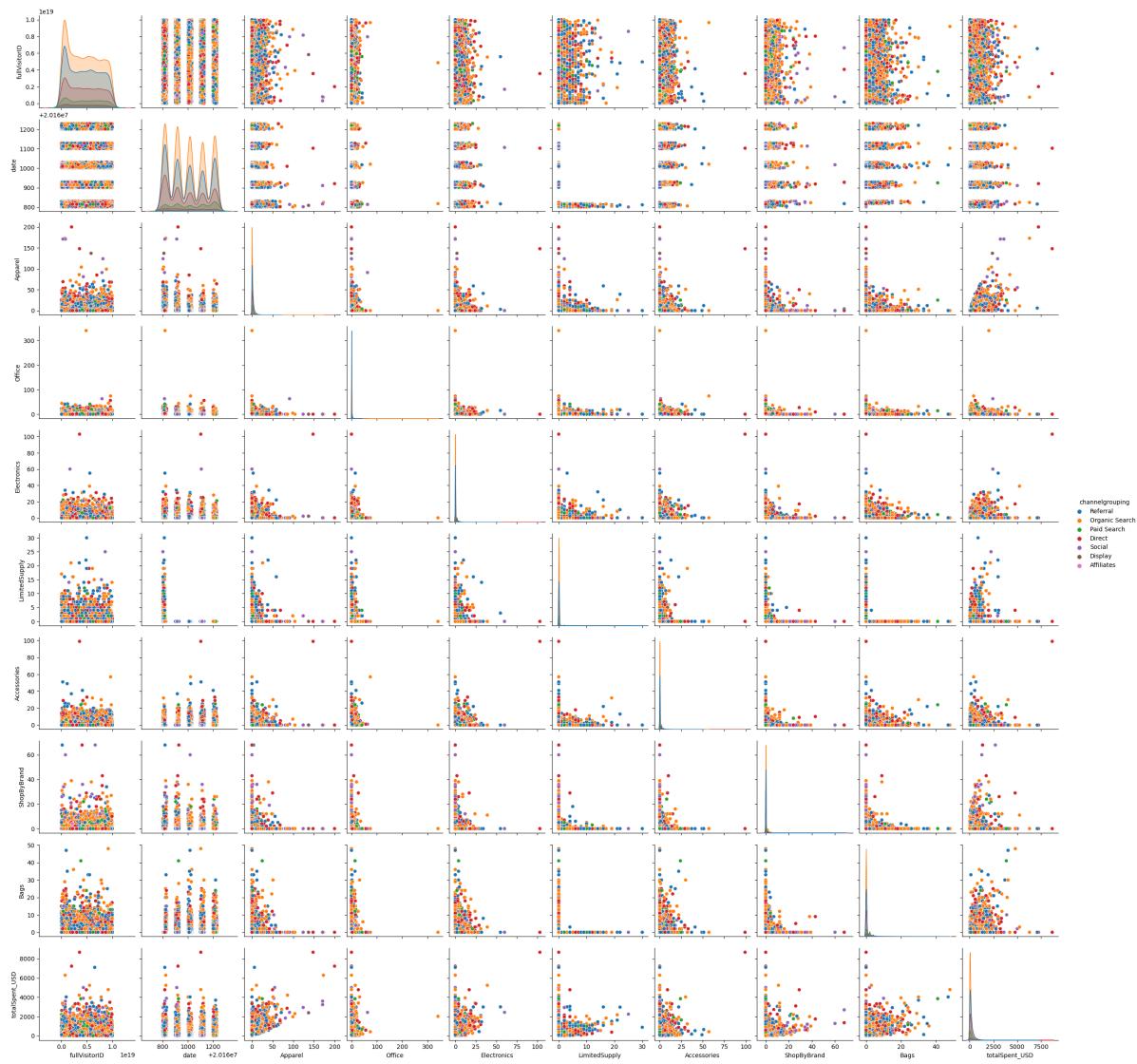
Out[45]:

	fullVisitorID	date	Apparel	Office	Electronics	LimitedSupply	A
count	3.867000e+04	3.867000e+04	38670.000000	38670.000000	38670.000000	38670.000000	38
mean	4.483221e+18	2.016100e+07	1.945074	0.531601	0.523946	0.107292	
std	3.058644e+18	1.458918e+02	4.751453	2.685681	1.875687	0.798183	
min	5.103959e+12	2.016080e+07	0.000000	0.000000	0.000000	0.000000	
25%	1.606532e+18	2.016090e+07	0.000000	0.000000	0.000000	0.000000	
50%	4.344901e+18	2.016101e+07	0.000000	0.000000	0.000000	0.000000	
75%	7.135485e+18	2.016112e+07	2.000000	0.000000	0.000000	0.000000	
max	9.999801e+18	2.016123e+07	200.000000	340.000000	103.000000	30.000000	

In [46]: `# Scatter plots de cada una de las variables numéricas clasificados por Tipo de OS`
`sns.pairplot(df, hue='OS')`
`plt.show()`



In [47]: # Scatter plots de cada una de las variables numéricas clasificados por Medio p
sns.pairplot(df, hue='channelgrouping')
plt.show()



Gráficas a revisar:

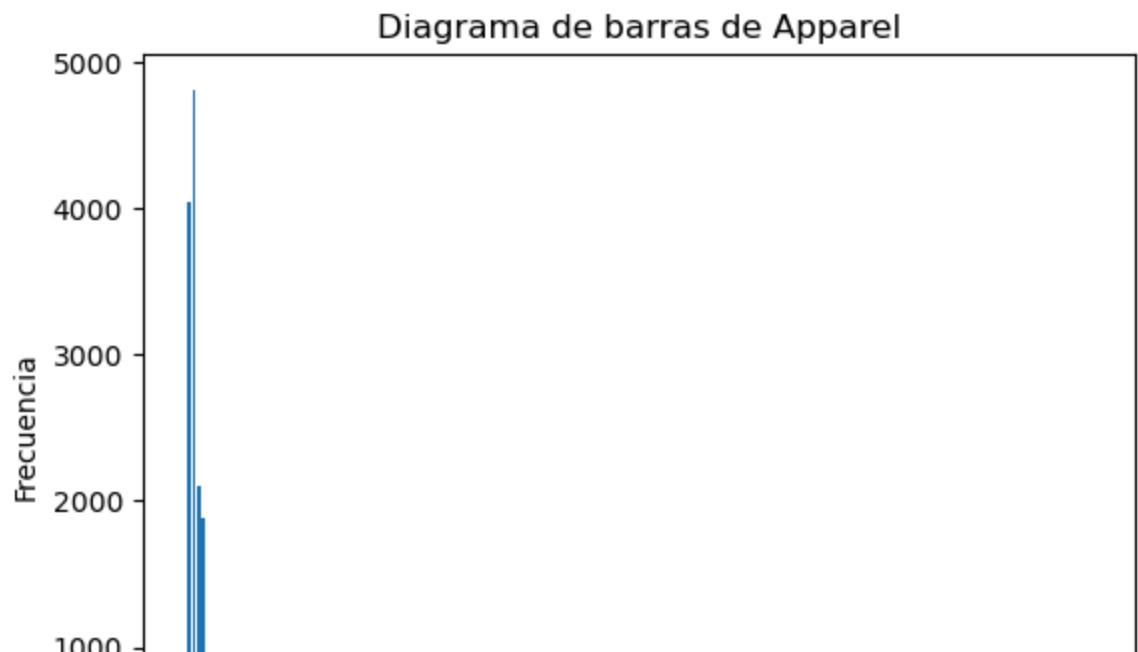
- Diagrama de frecuencias de cantidad de transacciones realizadas del medio por el cual se llegó a la solicitud.
- Diagrama de frecuencias de cantidad de transacciones realizadas del dispositivo por el cual se hizo la solicitud.
- Fechas vs Total gastado.
- Diagrama de barras de la cantidad de productos comprados por categoría.
- Histograma de la información del total gastado.
- Clasificar por Medio y ver digrama de barras por monto total gastado.
- Clasificar por Dispositivo utilizado y ver digrama de barras por monto total gastado.

```
In [48]: # Gráfico de frecuencias de compras por categoría
_list_columns_to_normalize = ['Apparel', 'Office', 'Electronics', 'LimitedSupply']

for column in _list_columns_to_normalize:
    data = df[(df[column] != 0)]
    # Crear grafica
    plt.bar(data[column].value_counts().index, data[column].value_counts().values)

    # Agregar Labels y titulo
    plt.xlabel(f'{column}')
    plt.ylabel('Frecuencia')
    plt.title(f'Diagrama de barras de {column}')

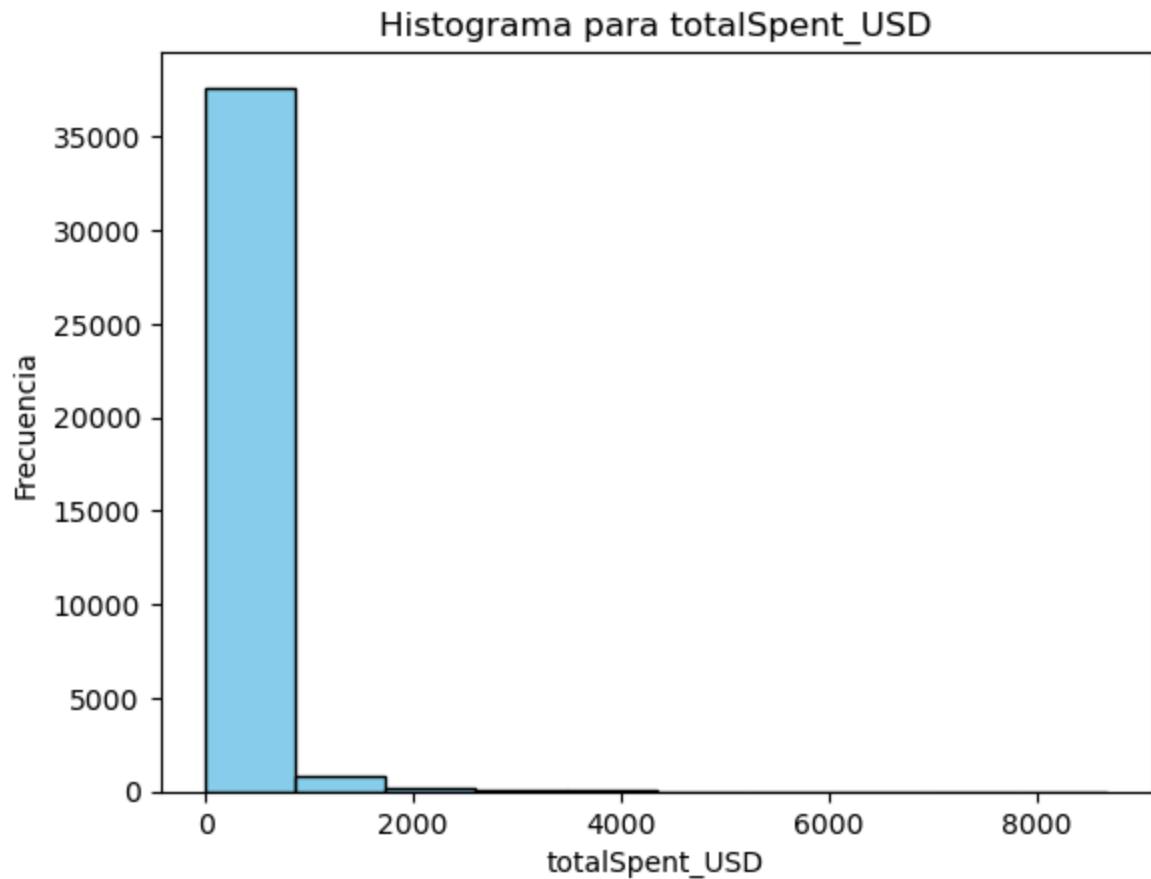
    # Show plot
    plt.show()
```



```
In [49]: # Histograma por Dolares totales gastados
data = df['totalSpent_USD']
# Crear el histograma
plt.hist(data, color='skyblue', edgecolor='black')

# Añadir etiquetas y título
plt.xlabel('totalSpent_USD')
plt.ylabel('Frecuencia')
plt.title('Histograma para totalSpent_USD')

# Mostrar el histograma
plt.show()
```

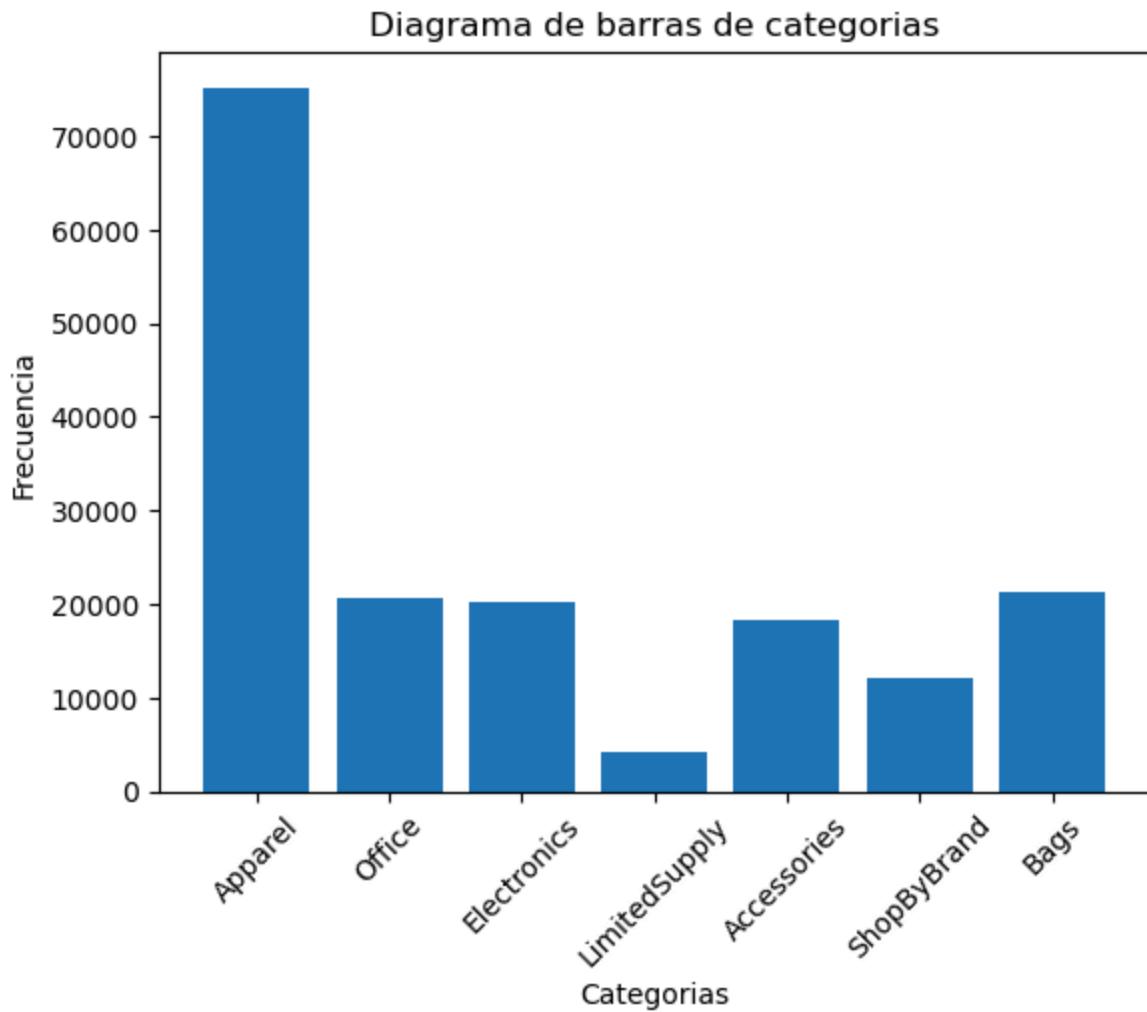


```
In [50]: _list_categories = ['Apparel', 'Office', 'Electronics', 'LimitedSupply', 'Accessories', 'ShopByBrand', 'Bags']
data = df[_list_categories].sum()
# Create bar plot
plt.bar(data.index, data.values)

plt.xticks(rotation=45)

plt.xlabel('Categorías')
plt.ylabel('Frecuencia')
plt.title('Diagrama de barras de categorías')

plt.show()
```



Descripción del problema inicial:

Con el fin de caracterizar algunos grupos de acuerdo a la información brindada, se plantean algunos modelos de Machine Learning para encontrar los grupos y sus características.

Para el item inicial se solicita estudiar 3 clusters para segmentar la información. Seguido vamos a utilizar varios algoritmos de Machine Learning no supervisados y técnicas de feature engineer para determinar si 3 cluster son los ideales para generar el estudio.

K-means

Para el planteamiento de un modelo usando K-Means, es importante tener en cuenta que la información con la que trabajemos debe estar normalizada para que no se generen sesgos en las conclusiones del modelo, con esto aseguramos que todas las variables trabajen sobre la misma escala. De igual modo es importante trabajar con variables numéricas para que se pueda entrenar el modelo.

In [51]: df.columns

Out[51]: Index(['fullVisitorID', 'channelgrouping', 'date', 'OS', 'Apparel', 'Office', 'Electronics', 'LimitedSupply', 'Accessories', 'ShopByBrand', 'Bags', 'totalSpent_USD'],
dtype='object')

In [89]: df_kmeans = df.copy()
features = ['Apparel', 'Office', 'Electronics', 'LimitedSupply', 'Accessories',
df_kmeans = df_kmeans[features]

_list_columns_to_normalize = ['Apparel', 'Office', 'Electronics', 'LimitedSupply']
X = df_kmeans

Normalizando Los datos
scaler = MinMaxScaler()

Normalizamos Las columnas
X = scaler.fit_transform(X)
X = pd.DataFrame(X, columns = features)
X

Out[89]:

	Apparel	Office	Electronics	LimitedSupply	Accessories	ShopByBrand	Bags	totalSpe
0	0.000	0.000000	0.009709	0.000000	0.000000	0.000000	0.0	0
1	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
2	0.000	0.000000	0.019417	0.000000	0.000000	0.000000	0.0	0
3	0.000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
4	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
...
38665	0.000	0.000000	0.000000	0.000000	0.000000	0.073529	0.0	0
38666	0.075	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
38667	0.010	0.008824	0.058252	0.066667	0.000000	0.000000	0.0	0
38668	0.065	0.008824	0.019417	0.133333	0.010101	0.088235	0.0	0
38669	0.000	0.008824	0.077670	0.200000	0.060606	0.000000	0.0	0

38670 rows × 8 columns

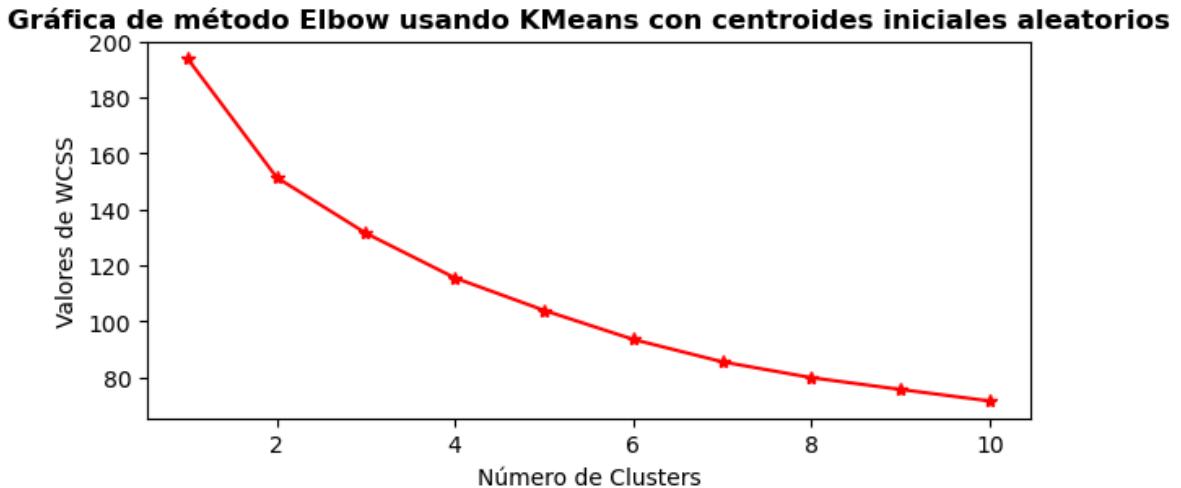
Usando el método Elbow

Vamos a utilizar "the elbow method" para determinar cual es la cantidad de cluster que mejor describen el problema. Para esto generamos una lista llamada `wcss` que significa "Within-Cluster Sum of Square", el cual nos permite determinar el error de acuerdo a cada uno de los cluster obtenidos. Este determina la suma de los errores al cuadrado de cada punto en el cluster con respecto al centroide del cluster.

```
In [64]: wcss = []
number_cluster = []

#Entrenando el modelo para k-clusters
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init='random', random_state=10213)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
    number_cluster.append(k)

# Gráfico de Elbow
plt.figure(figsize=(7, 3))
plt.plot(number_cluster, wcss, 'r*-')
plt.xlabel('Número de Clusters')
plt.ylabel('Valores de WCSS')
plt.title('Gráfica de método Elbow usando KMeans con centroides iniciales aleatorios')
plt.show()
```



Podemos observar que la tasa de decremento a partir de cada cluster no es significativa cuando esta entre 3 y 4. De este modo vamos a considerar el tanto una clasificación con los 3 clusters y la de 4 clusters.

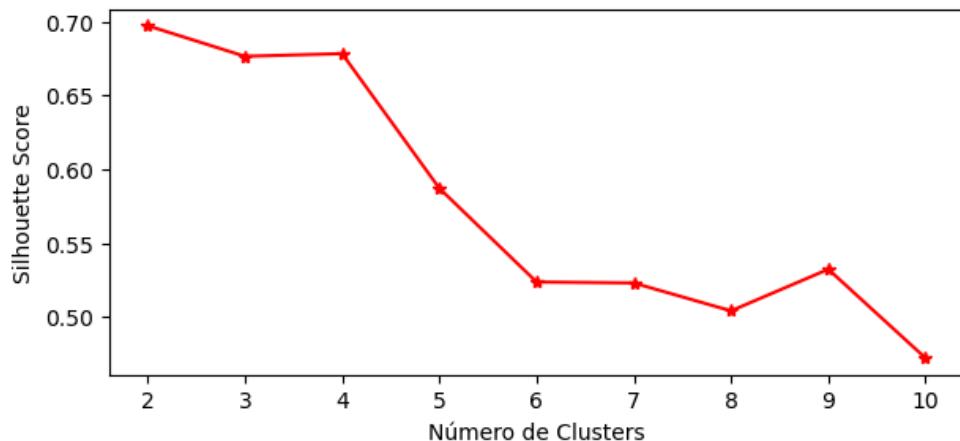
Usando el método coeficiente Silhouette

```
In [65]: silhouette_avg = []
number_cluster = []

#Entrenando el modelo para k-clusters
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, init='random', random_state=10213)
    kmeans.fit(X)
    cluster_labels = kmeans.labels_
    silhouette_avg.append(silhouette_score(X, cluster_labels))
    number_cluster.append(k)

# Gráfico usando Silhouette Score
plt.figure(figsize=(7, 3))
plt.plot(number_cluster, silhouette_avg, 'r*-')
plt.xlabel('Número de Clusters')
plt.ylabel('Silhouette Score')
plt.title('Gráfica de Silhouette Score usando KMeans con centroides iniciales aleatorios')
plt.show()
```

Gráfica de Silhouette Score usando KMeans con centroides iniciales aleatorios



4 clusters

Combinando tanto la descripción utilizando el método Elbow, como Silhouettes Score, podemos determinar 4 clusters son la mejor cantidad de clusters para implementar el método K-Means.

In [90]: # Inicializando K-Means

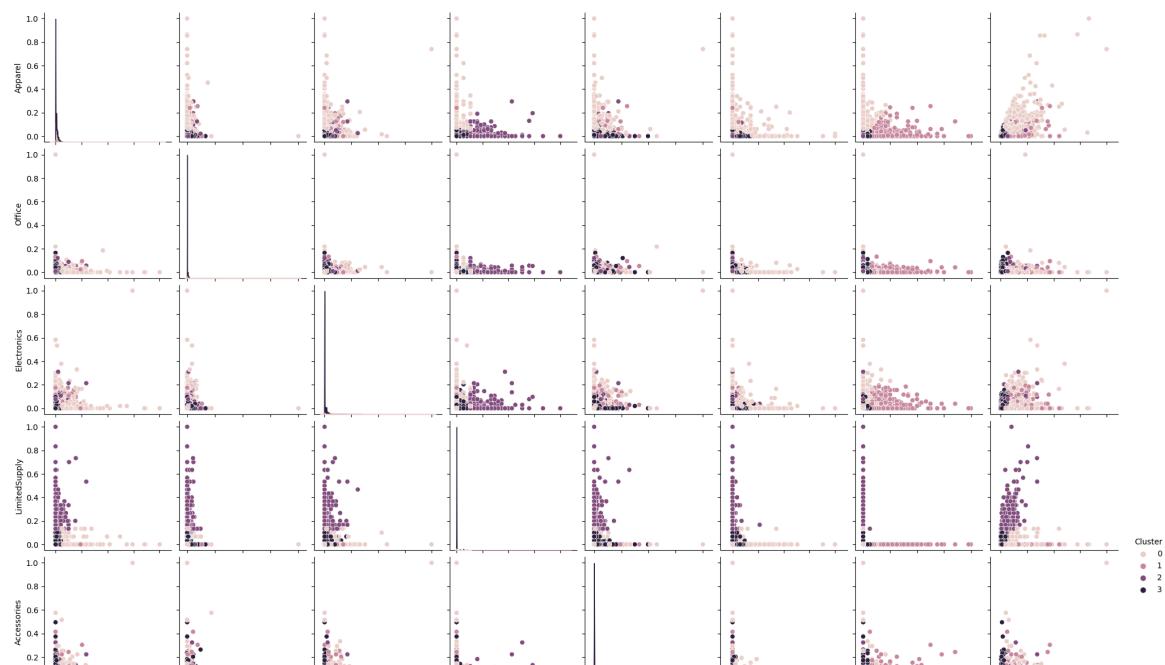
```
kmeans = KMeans(n_clusters=4, init='random', random_state=10213)
dataframe4 = X
# Entrenando el modelo
kmeans.fit(X)
# Agregando los valores de los clusters
dataframe4['Cluster'] = kmeans.labels_
dataframe4
```

Out[90]:

	Apparel	Office	Electronics	LimitedSupply	Accessories	ShopByBrand	Bags	totalSpe
0	0.000	0.000000	0.009709	0.000000	0.000000	0.000000	0.0	0
1	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
2	0.000	0.000000	0.019417	0.000000	0.000000	0.000000	0.0	0
3	0.000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
4	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
...
38665	0.000	0.000000	0.000000	0.000000	0.000000	0.073529	0.0	0
38666	0.075	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
38667	0.010	0.008824	0.058252	0.066667	0.000000	0.000000	0.0	0
38668	0.065	0.008824	0.019417	0.133333	0.010101	0.088235	0.0	0
38669	0.000	0.008824	0.077670	0.200000	0.060606	0.000000	0.0	0

38670 rows × 9 columns

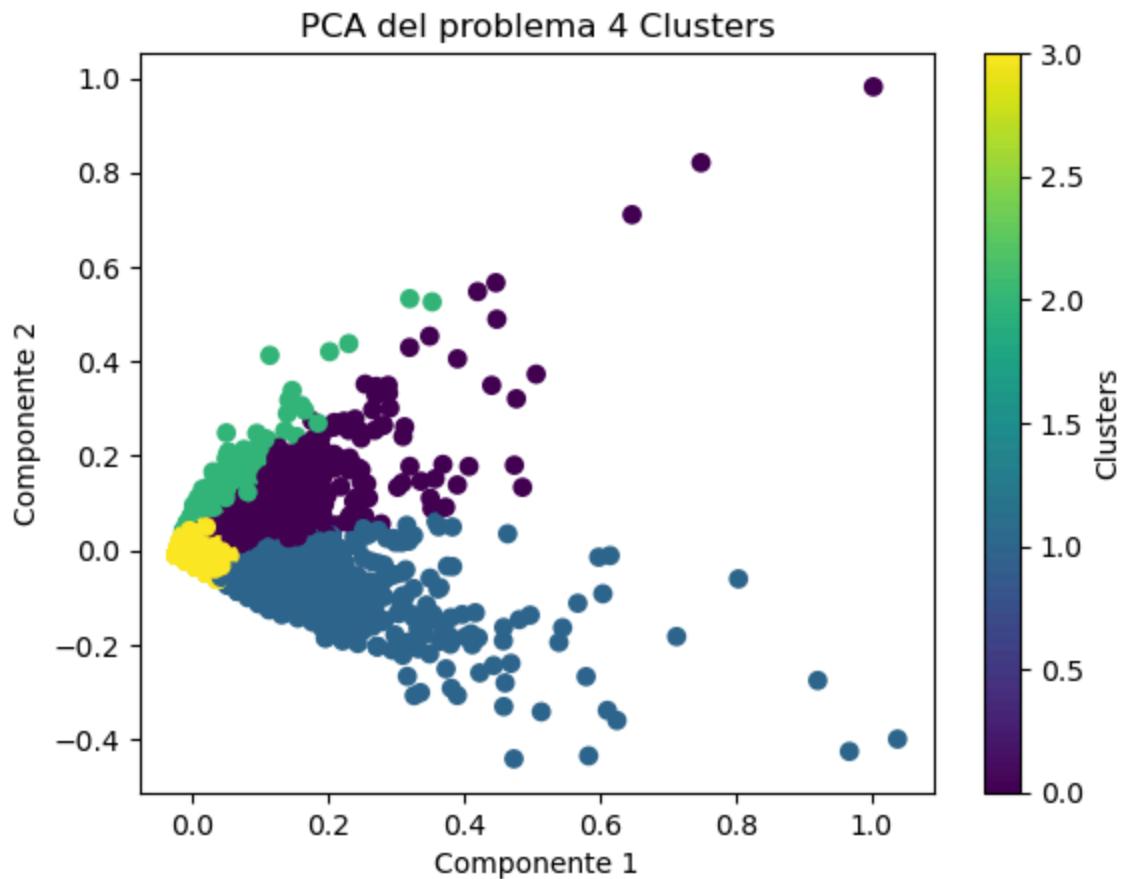
In [67]: sns.pairplot(dataframe4, hue='Cluster')
plt.show()



Observando PCA sobre la clasificación para ver como esta distribuido a partir de una nueva base con varianza maxima y mínima.

```
In [91]: df_dropped = dataframe4.drop(columns=['Cluster'])
X4 = df_dropped
y4 = dataframe4['Cluster']
# Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X4)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y4, cmap='viridis')
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('PCA del problema 4 Clusters')
plt.colorbar(label='Clusters')
plt.show()
```



Se utilizan los datos originales para la revisión mediante scatter plots y ver si existe una segmentación evidente:

```
In [ ]: original_data =
```

3 clusters

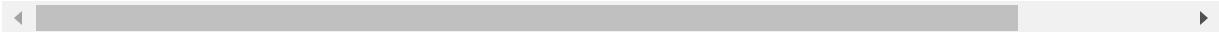
De acuerdo a lo pedido en el documento, se requiere primero observar 3 clusters, de este modo se realiza también el planteamiento del problema.

```
In [92]: # Inicializando K-Means
kmeans = KMeans(n_clusters=3, init='random', random_state=10213)
dataframe3 = X
# Entrenando el modelo
kmeans.fit(dataframe3)
# Agregando los valores de los clusters
dataframe3['Cluster'] = kmeans.labels_
dataframe3
```

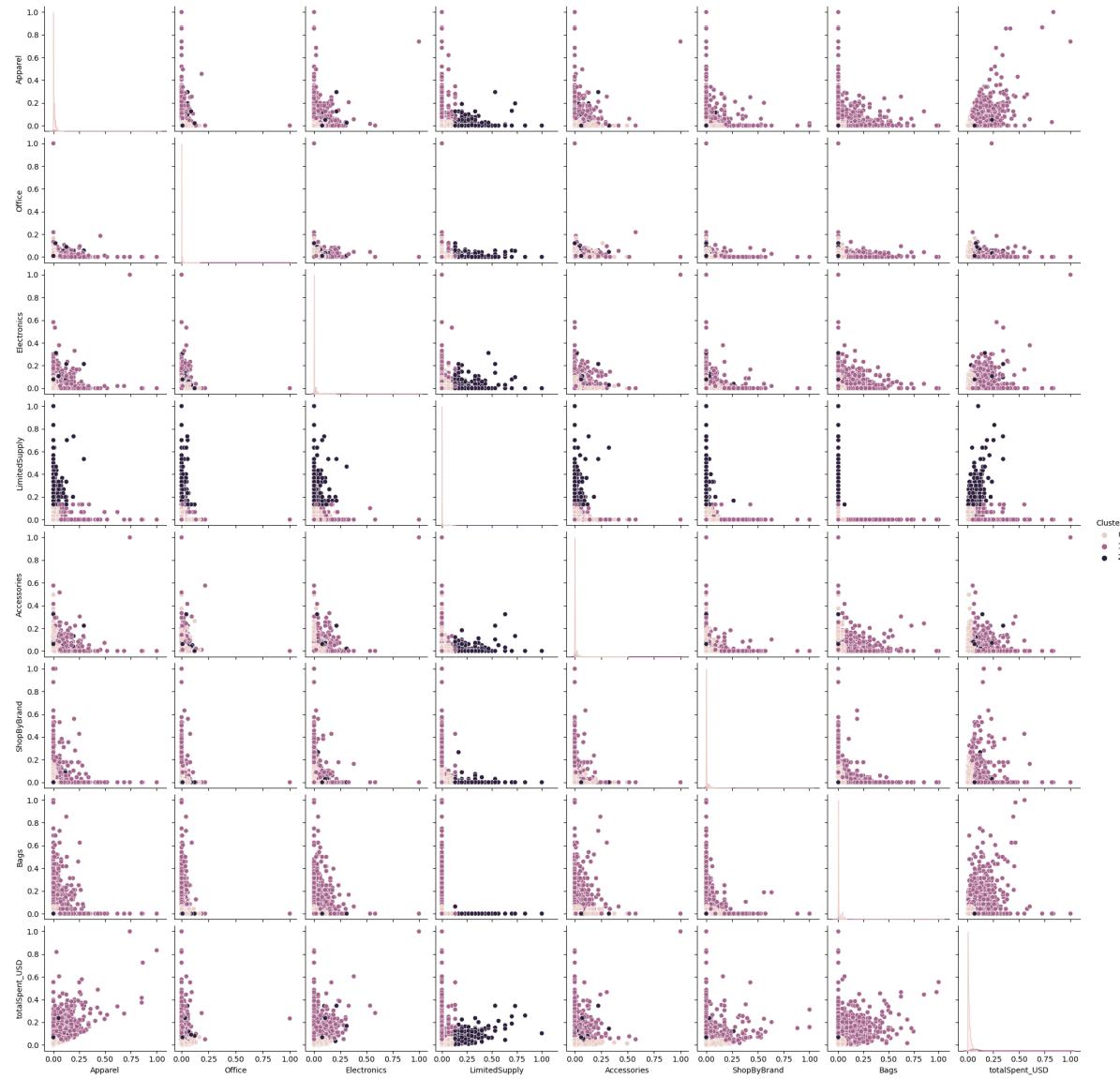
Out[92]:

	Apparel	Office	Electronics	LimitedSupply	Accessories	ShopByBrand	Bags	totalSpe
0	0.000	0.000000	0.009709	0.000000	0.000000	0.000000	0.0	0
1	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
2	0.000	0.000000	0.019417	0.000000	0.000000	0.000000	0.0	0
3	0.000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
4	0.005	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
...
38665	0.000	0.000000	0.000000	0.000000	0.000000	0.073529	0.0	0
38666	0.075	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0
38667	0.010	0.008824	0.058252	0.066667	0.000000	0.000000	0.0	0
38668	0.065	0.008824	0.019417	0.133333	0.010101	0.088235	0.0	0
38669	0.000	0.008824	0.077670	0.200000	0.060606	0.000000	0.0	0

38670 rows × 9 columns



```
In [71]: sns.pairplot(dataframe3, hue='Cluster')
plt.show()
```



```
In [93]: df_dropped = dataframe3.drop(columns=['Cluster'])
```

```
X3 = df_dropped
```

```
y3 = dataframe3['Cluster']
```

```
# Perform PCA
```

```
pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X3)
```

```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y3, cmap='viridis')
```

```
plt.xlabel('Componente 1')
```

```
plt.ylabel('Componente 2')
```

```
plt.title('PCA del problema 3 Clusters')
```

```
plt.colorbar(label='Clusters')
```

```
plt.show()
```

