Universidad Nacional Autónoma de México

Inteligencia Artificial

Clúster Jerárquico

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 4

Profesor: Dr. Guillermo Gilberto Molero Castillo

16 de marzo de 2022

Introducción

Para esta práctica se revisará el concepto de clustering jerárquico que consiste en la segmentación y delimitación de grupos de objetos para su análisis. Además, se utilizarán conceptos de selección de características para modificar datos y obtener el mejor modelo posible. Se obtendrán clústeres de casos de usuarios que tienen características similares que fueron evaluados por un crédito hipotecario. Se obtendrán diferentes clústeres y se analizarán las características de cada uno.

Objetivo

Obtener clústeres de casos de usuarios, con características similares, evaluados para la adquisición de una casa a través de un crédito hipotecario con tasa fija a 30 años.

Desarrollo

Primero se utilizan las bibliotecas usuales y se añade seaborn para poder visualizar datos de matplotlib con diferentes funcionalidades. Se imprimen los datos que constan de 202 filas y se checa que no existan valores nulos.

```
Hipoteca = pd.read_csv("Hipoteca.csv")
Hipoteca
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	6000	1000	0	600	50000	400000	0	2	2	1
1	6745	944	123	429	43240	636897	1	3	6	0
2	6455	1033	98	795	57463	321779	2	1	8	1
3	7098	1278	15	254	54506	660933	0	0	3	0
4	6167	863	223	520	41512	348932	0	0	3	1
...	...	...	...	...	...	...	...	...	...	...
197	3831	690	352	488	10723	363120	0	0	2	0
198	3961	1030	270	475	21880	280421	2	3	8	0
199	3184	955	276	684	35565	388025	1	3	8	0
200	3334	867	369	652	19985	376892	1	2	5	0
201	3988	1157	105	382	11980	257580	0	0	4	0

202 rows × 10 columns

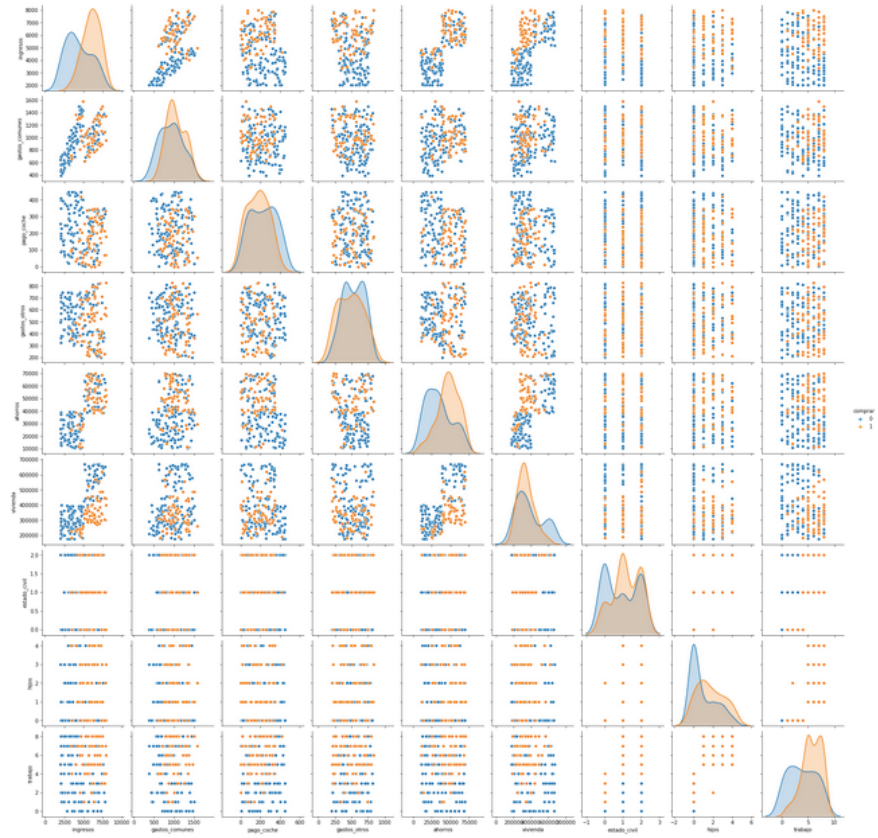
Se generan dos grupos a partir de la variable de si se compró o no la casa. Se agrupa y se obtiene el tamaño de cada uno de los grupos que se generan. A manera de referencia se tiene este etiquetado.

```
print(Hipoteca.groupby('comprar').size())
```

```
comprar
0    135
1     67
dtype: int64
```

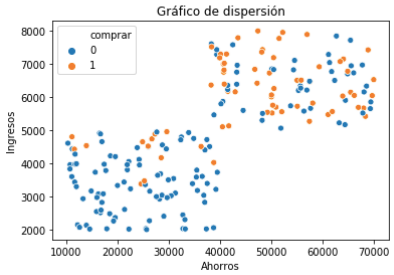
Se genera una gráfica de dispersión que existe entre las diferentes variables para identificar si existe una correlación muy grande y poca dispersión para poder eliminar dicha variable de los datos utilizados. En este caso se generó una matriz de gráficas de dispersión y a partir de ella se visualizó la dispersión de los datos.

```
sns.pairplot(Hipoteca, hue='comprar')       #Densidad y gráficas de dispersión
plt.show()
```



Si se necesita observar uno en particular se puede imprimir la gráfica de dispersión de esas variables en particular. Se muestra la gráfica de dispersión de Ahorros e Ingresos.

```
sns.scatterplot(x='ahorros', y ='ingresos', data=Hipoteca, hue='comprar')
plt.title('Gráfico de dispersión')
plt.xlabel('Ahorros')
plt.ylabel('Ingresos')
plt.show()
```



También, se puede utilizar una matriz de correlaciones para analizar cuáles son las relaciones que se pueden eliminar por tener una correlación muy alta o ser muy dependientes. La matriz de correlaciones se muestra a continuación.

```
CorrHipoteca = Hipoteca.corr(method='pearson')
CorrHipoteca
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
ingresos	1.000000	0.560211	-0.109780	-0.124105	0.712889	0.614721	-0.042556	-0.024483	-0.038852	0.467123
gastos_comunes	0.560211	1.000000	-0.054400	-0.099881	0.209414	0.204781	-0.057152	-0.072321	-0.079095	0.200191
pago_coche	-0.109780	-0.054400	1.000000	0.010602	-0.193299	-0.094631	0.052239	-0.044858	0.018946	-0.196468
gastos_otros	-0.124105	-0.099881	0.010602	1.000000	-0.064384	-0.054577	-0.020226	0.124845	0.047313	-0.110330
ahorros	0.712889	0.209414	-0.193299	-0.064384	1.000000	0.605836	-0.063039	0.001445	-0.023829	0.340778
vivienda	0.614721	0.204781	-0.094631	-0.054577	0.605836	1.000000	-0.113420	-0.141924	-0.211790	-0.146092
estado_civil	-0.042556	-0.057152	0.052239	-0.020226	-0.063039	-0.113420	1.000000	0.507609	0.589512	0.142799
hijos	-0.024483	-0.072321	-0.044858	0.124845	0.001445	-0.141924	0.507609	1.000000	0.699916	0.272883
trabajo	-0.038852	-0.079095	0.018946	0.047313	-0.023829	-0.211790	0.589512	0.699916	1.000000	0.341537
comprar	0.467123	0.200191	-0.196468	-0.110330	0.340778	-0.146092	0.142799	0.272883	0.341537	1.000000

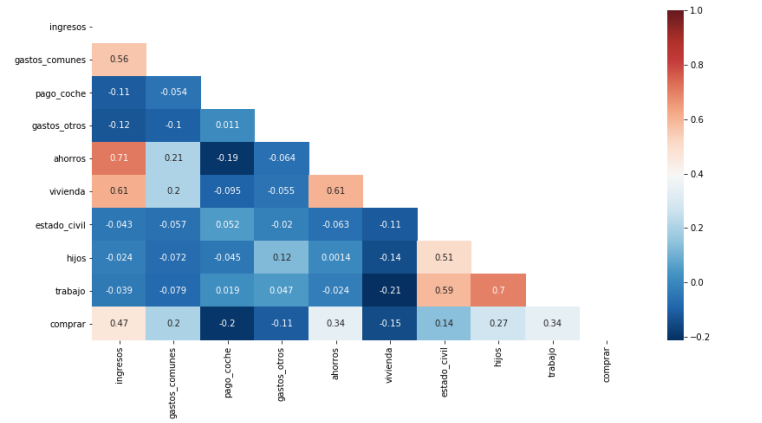
Esta matriz es difícil de analizar si se tienen muchos datos o si se tienen muchos decimales. Una manera de analizar cada una de las filas es ordenando los valores que se tienen de correlación y prestando atención a aquellos que tienen correlación de más de .7.

```
print(CorrHipoteca['ingresos'].sort_values(ascending=False)[:10], '\n')   #Top 10 valores
```

```
ingresos          1.000000
ahorros           0.712889
vivienda          0.614721
gastos_comunes    0.560211
comprar           0.467123
hijos            -0.024483
trabajo          -0.038852
estado_civil     -0.042556
pago_coche       -0.109780
gastos_otros     -0.124105
Name: ingresos, dtype: float64
```

Una manera más fácil de visualizar los datos se genera coloreando aquellos valores de la matriz que se encuentran más cercanos al uno para identificarlos.

```
plt.figure(figsize=(14,7))
MatrizInf = np.triu(CorrHipoteca)
sns.heatmap(CorrHipoteca, cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



En la gráfica anterior se muestran los valores de correlación que existen entre las variables de los datos. Se muestra que hijos con trabajo y ahorros con ingreso tienen una correlación muy alta. Aunque tengan una correlación muy alta no se eliminarán dichas variables por tener importancia en el modelo que se generará. Si fueran datos redundantes o dependientes se eliminarían. La variable comprar sí se elimina ya que representa un etiquetado hecho a partir de un análisis y esto afectaría el modelo que se generará. Se genera un DataFrame con las variables de interés.

```
MatrizHipoteca = np.array(Hipoteca[['ingresos', 'gastos_comunes', 'pago_coche', 'gastos_otros', 'ahorros', 'vivienda
pd.DataFrame(MatrizHipoteca)
#MatrizHipoteca = Hipoteca.iloc[:, 0:9].values     #iloc para seleccionar filas y columnas según su posición
```

	0	1	2	3	4	5	6	7	8
0	6000	1000	0	600	50000	400000	0	2	2
1	6745	944	123	429	43240	636897	1	3	6
2	6455	1033	98	795	57463	321779	2	1	8
3	7098	1278	15	254	54506	660933	0	0	3
4	6167	863	223	520	41512	348932	0	0	3
...	...	...	...	...	...	...	...	...	...
197	3831	690	352	488	10723	363120	0	0	2
198	3961	1030	270	475	21880	280421	2	3	8
199	3184	955	276	684	35565	388025	1	3	8
200	3334	867	369	652	19985	376892	1	2	5
201	3988	1157	105	382	11980	257580	0	0	4

Después de obtener todo este análisis de los datos, se debe estandarizar los datos como se ha hecho en prácticas pasadas. Para esto se utilizó el módulo de StandardScaler que se encuentra en sklearn. Utilizando este módulo es muy fácil estandarizar los datos utilizando la función que transforma los datos. Adicionalmente, se muestra la estandarización para entender qué es lo que realizó esta función.

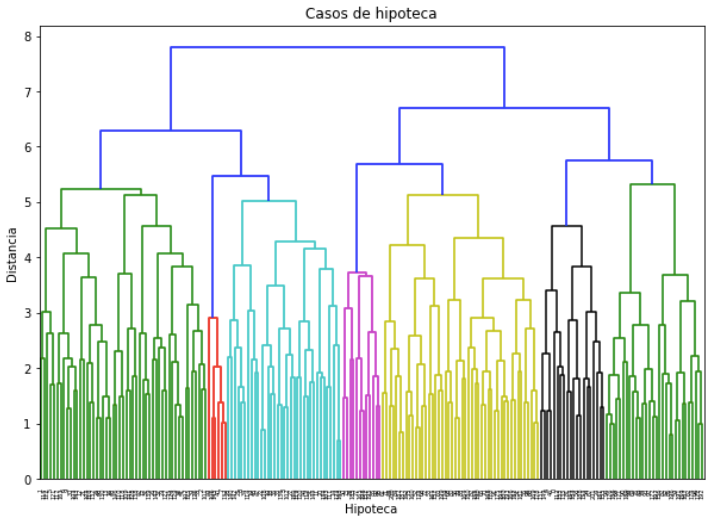
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
estandarizar = StandardScaler()                          # Se instancia el objeto Stand
MEstandarizada = estandarizar.fit_transform(MatrizHipoteca)   # Se calculan la media y desvi
```

```
pd.DataFrame(MEstandarizada)
```

	0	1	2	3	4	5	6	7	8
0	0.620129	0.104689	-1.698954	0.504359	0.649475	0.195910	-1.227088	0.562374	-0.984420
1	1.063927	-0.101625	-0.712042	-0.515401	0.259224	1.937370	-0.029640	1.295273	0.596915
2	0.891173	0.226266	-0.912634	1.667244	1.080309	-0.379102	1.167809	-0.170526	1.387582
3	1.274209	1.128886	-1.578599	-1.559015	0.909604	2.114062	-1.227088	-0.903426	-0.589086
4	0.719611	-0.400042	0.090326	0.027279	0.159468	-0.179497	-1.227088	-0.903426	-0.589086
...	...	...	...	...	...	...	...	...	...
197	-0.671949	-1.037402	1.125381	-0.163554	-1.617963	-0.075199	-1.227088	-0.903426	-0.984420
198	-0.594508	0.215214	0.467439	-0.241079	-0.973876	-0.683130	1.167809	1.295273	1.387582
199	-1.057368	-0.061099	0.515581	1.005294	-0.183849	0.107880	-0.029640	1.295273	1.387582
200	-0.968013	-0.385305	1.261783	0.814462	-1.083273	0.026040	-0.029640	0.562374	0.201581
201	-0.578424	0.683102	-0.856468	-0.795686	-1.545397	-0.851037	-1.227088	-0.903426	-0.193753

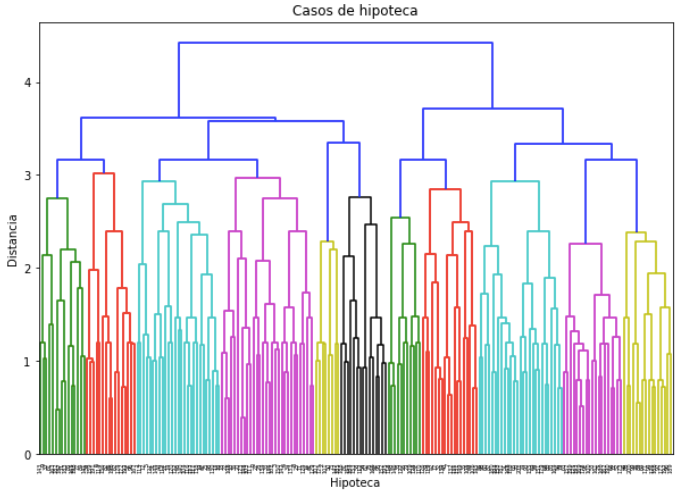
Se importan bibliotecas para generar el *clustering* jerárquico a partir de sklearn y scipy. Se genera un gráfico del árbol jerárquico de las agrupaciones que genera el algoritmo. La métrica que se utiliza es euclidiana, si esta se cambia los valores del árbol y de las agrupaciones cambiarán.

```
#Se importan las bibliotecas de clustering jerárquico para crear el árbol
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
plt.figure(figsize=(10, 7))
plt.title("Casos de hipoteca")
plt.xlabel('Hipoteca')
plt.ylabel('Distancia')
Arbol = shc.dendrogram(shc.linkage(MEstandarizada, method='complete', metric='euclidean'))
#plt.axhline(y=5.4, color='orange', linestyle='--')
#Probar con otras medciones de distancia (euclidean, chebyshev, cityblock)
```



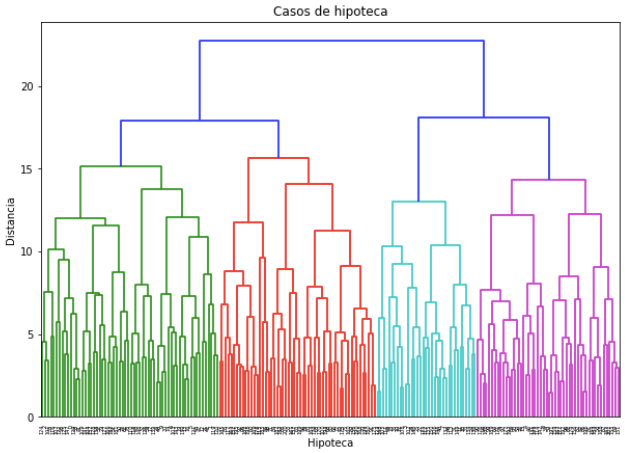
Utilizando Chebyshev se obtuvieron más agrupaciones que en la métrica anterior. Se muestra la imagen con las agrupaciones creadas con esta métrica.

```
plt.figure(figsize=(10, 7))
plt.title("Casos de hipoteca")
plt.xlabel('Hipoteca')
plt.ylabel('Distancia')
Arbol01 = shc.dendrogram(shc.linkage(MEstandarizada, method='complete', metric='chebyshev'))
```



Utilizando Manhattan se obtuvieron menos agrupaciones que con la métrica Euclidiana. A continuación se muestra el diagrama generado con esta métrica.

```
plt.figure(figsize=(10, 7))
plt.title("Casos de hipoteca")
plt.xlabel('Hipoteca')
plt.ylabel('Distancia')
Arbol02 = shc.dendrogram(shc.linkage(MEstandarizada, method='complete', metric='cityblock'))
```



Utilizando Agglomerative Clustering se generan 7 clusters utilizando la distancia euclidiana y se le asigna una etiqueta a cada uno de los datos que se tiene.

```
#Se crean las etiquetas de los elementos en los clústeres
MJerarquico = AgglomerativeClustering(n_clusters=7, linkage='complete', affinity='euclidean')
MJerarquico.fit_predict(MEstandarizada)
MJerarquico.labels_
```

```
array([4, 1, 1, 2, 4, 1, 1, 6, 2, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 2, 1,
       2, 1, 4, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 6, 1, 1, 1,
       2, 2, 4, 2, 1, 6, 5, 3, 3, 3, 4, 3, 3, 0, 4, 0, 3, 3, 0, 3, 0, 3,
       3, 4, 3, 0, 3, 3, 3, 5, 0, 3, 0, 5, 5, 3, 3, 4, 0, 3, 3, 5, 0, 3,
       3, 0, 0, 3, 5, 0, 0, 5, 0, 0, 3, 0, 3, 1, 2, 1, 1, 2, 6, 1, 2, 1,
       1, 2, 4, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 4,
       6, 4, 2, 4, 2, 1, 1, 1, 2, 1, 2, 1, 2, 6, 1, 1, 2, 4, 2, 4, 5, 4,
       4, 4, 0, 3, 3, 0, 3, 3, 3, 1, 3, 5, 3, 0, 3, 3, 3, 0, 0, 3, 0, 3,
       0, 0, 3, 3, 3, 3, 3, 3, 4, 5, 0, 3, 4, 0, 3, 0, 0, 3, 3, 5, 0, 0,
       5, 3, 3, 4])
```

Se elimina la columna de comprar y en su lugar se agregan las etiquetas generadas por el algoritmo. La columna de clusterH representa la etiqueta que se le dio.

```
Hipoteca = Hipoteca.drop(columns=['comprar'])
Hipoteca['clusterH'] = MJerarquico.labels_
Hipoteca
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	clusterH
0	6000	1000	0	600	50000	400000	0	2	2	4
1	6745	944	123	429	43240	636897	1	3	6	1
2	6455	1033	98	795	57463	321779	2	1	8	1
3	7098	1278	15	254	54506	660933	0	0	3	2
4	6167	863	223	520	41512	348932	0	0	3	4
...	...	...	...	...	...	...	...	...	...	...
197	3831	690	352	488	10723	363120	0	0	2	0
198	3961	1030	270	475	21880	280421	2	3	8	5
199	3184	955	276	684	35565	388025	1	3	8	3
200	3334	867	369	652	19985	376892	1	2	5	3
201	3988	1157	105	382	11980	257580	0	0	4	4

Ahora se agrupan a los *clusters* por su etiqueta y se cuentan los datos que pertenecen a cada una de las agrupaciones.

```
#Cantidad de elementos en los clusters
Hipoteca.groupby(['clusterH'])['clusterH'].count()
```

```
clusterH
0    30
1    51
2    35
3    48
4    20
5    12
6     6
Name: clusterH, dtype: int64
```

Para mostrar los datos de cada *cluster* se puede imprimir el conjunto de datos donde la columna clusterH sea igual a la etiqueta que se desea ver. A partir de los datos se puede generar una interpretación de lo que representa cada una de las etiquetas.

```
Hipoteca[Hipoteca.clusterH == 6]
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	clusterH
7	6470	1035	39	782	57439	606291	0	0	1	6
40	6822	1296	81	786	50433	669054	0	0	0	6
49	6959	1392	333	818	67714	571076	0	0	3	6
106	6205	1179	240	729	56904	661009	0	0	2	6
132	6325	1139	102	754	68527	588004	0	0	0	6
145	5646	1016	215	747	69276	655399	0	0	1	6

Se generó una interpretación de cada uno de los *clusters* que se generaron gracias al algoritmo. Para el *cluster 0* se tienen 30 casos con un ingreso de 3241USD y gastos comunes bajos de 846USD. Este agrupación tiene de los ahorros más bajos de todas las agrupaciones y también de los ingresos más bajos. Además, no tienen hijos y el precio de su vivienda es de las más bajas.

Para el *cluster* 1 se tienen 51 casos con ingresos de 6394USD y gastos comunes de 1021USD. Tienen la tercer cantidad más alta de ingresos. La mayoría de estas

personas está casado y con dos hijos. La vivienda de esta agrupación es de las más altas.

Para el *cluster* 2 se tienen 35 casos con ingresos de 6599USD y gastos comunes de 1087USD. Es la segunda agrupación con ingresos más altos. Se tiene que más de la mitad está casada y en general no tienen hijos. El precio de la vivienda de esta agrupación es la segunda más alta.

Para el *cluster* 3 se tienen 48 casos con ingresos de 3189USD y gastos comunes de 785USD. Es la agrupación con ingresos y gastos más baja de todos los datos. La mayoría está casada y tiene dos hijos. Cuentan con el segundo precio de vivienda más baja de todos los datos.

Para el *cluster* 4 se tienen 20 casos con ingresos de 4843USD y gastos comunes de 1009 USD. Tanto gastos como ingresos se encuentran a la mitad de las diferentes medias de las diferentes agrupaciones. La mayoría tiene trabajo y no tienen ni hijos ni están casados. Su vivienda tampoco es de las más altas ni de las más bajas de los datos.

Para el *cluster* 5 se tienen 12 casos con ingresos de 4466USD y 1315USD de gastos comunes. Tienen la vivienda con el valor más bajo de todas las agrupaciones. La mayoría se encuentran casados, con más de dos hijos y con trabajo asalariado.

Para el *cluster* 6 se tienen 6 casos con ingresos de 6404USD y gastos comunes de 1176USD. Son de los ingresos más altos que se tienen en los datos y los gastos no son de los más altos de las diferentes agrupaciones. Son el grupo con la mayor cantidad de ahorros de todos, no están casados y tampoco tienen hijos.

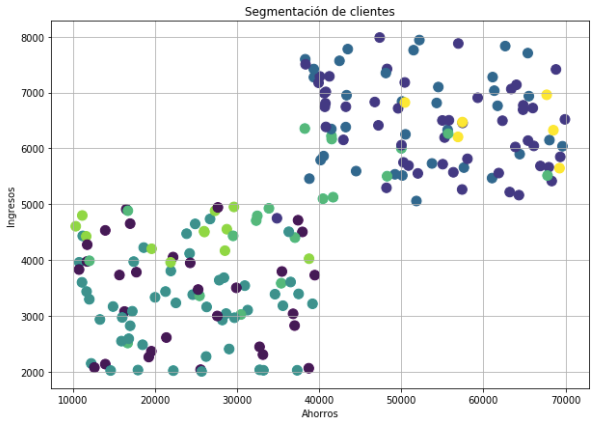
Se genera una vista con la media de cada una de las variables de cada uno de los *clusters*. Se agrupan en una fila por agrupación y se genera la media de cada variable para poder observar en promedio qué es lo que representa cada uno de los grupos. Con esta tabla se puede generar una interpretación de buena manera ya que se tiene una vista global de cada una de las agrupaciones y de las diferentes agrupaciones que se tienen.

```
CentroidesH = Hipoteca.groupby('clusterH').mean()
CentroidesH
```

clusterH	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo
0	3421.133333	846.466667	309.933333	527.233333	24289.633333	295590.700000	0.233333	0.000000	2.000000
1	6394.019608	1021.627451	192.274510	533.039216	54382.529412	421178.764706	1.490196	2.254902	6.313725
2	6599.542857	1087.428571	204.771429	362.600000	51863.028571	515494.257143	0.685714	0.228571	2.885714
3	3189.687500	785.020833	243.208333	548.270833	23616.854167	277066.687500	1.645833	1.979167	6.208333
4	4843.750000	1009.200000	122.200000	572.850000	36340.650000	337164.850000	0.050000	0.100000	1.900000
5	4466.416667	1315.083333	114.416667	502.750000	23276.166667	269429.916667	1.666667	2.416667	6.750000
6	6404.500000	1176.166667	168.333333	769.333333	61715.500000	625138.833333	0.000000	0.000000	1.166667

Por último, se muestra cómo crear gráficos de dispersión donde se muestran dos variables para que se muestre su relación y los diferentes grupos a los que pertenece cada uno de los datos. Se muestra la gráfica generada.

```
plt.figure(figsize=(10, 7))
plt.scatter(Hipoteca.ahorros, Hipoteca.ingresos , c=Hipoteca.clusterH, s=100)
plt.title('Segmentación de clientes')
plt.xlabel('Ahorros')
plt.ylabel('Ingresos')
plt.grid()
plt.show()
```



Conclusión

Se utilizó el concepto de clustering jerárquico para segmentar y agrupar casos de usuarios evaluados para la adquisición de una casa a través de un crédito hipotecario. Con esta práctica se pudieron observar conceptos sobre la selección de características de los datos para obtener solo las variables relevantes e impactantes para el modelo. Se generaron diferentes gráficos que permitieron la visualización de la relación entre los diferentes datos y se utilizaron las métricas de distancias vistas en prácticas anteriores para generar el modelo de clustering jerárquico. Con los diferentes modelos generados se pudieron observar las diferentes agrupaciones que se generaron gracias al algoritmo utilizado. Posteriormente, se analizaron los modelos

y se obtuvo la media de los datos de cada grupo para observar porque cada grupo se segmentó de dicha manera. Con todo este análisis se pudo obtener una buena visualización del clustering jerárquico y de todas las implicaciones que se necesitan para este modelo.