Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 26/10/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 5: Clustering Particional (Segmentación de Clientes)

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.

Características Generales:

En esta práctica, utilizaremos el clustering particional. Éste es un tipo de aprendizaje no supervisado en donde, a partir de una fuente de datos, se organizan los elementos dentro de *k* clústeres, dentro de los cuales los elementos son unidos por características que éstos tienen en común. El objetivo de este tipo de aprendizaje es dividir una población heterogénea de elementos en un número de grupos homogéneos, de acuerdo con sus similitudes.

Dentro de este tipo de aprendizaje, utilizaremos otros algoritmos de IA vistos previamente, como las métricas de distancia, las cuales nos serán útiles para conocer las similitudes entre objetos.

En el clustering particional, se establecen *k* centroides para la formación de *k* grupos. Estos centroides se eligen de forma aleatoria. Cada elemento de la fuente de datos es asignado al centroide más cercano. Luego, se actualiza la posición del centroide con base en la media de los elementos asignados en el clúster. Se repiten los dos pasos anteriores de manera iterativa hasta que los centroides no cambien de lugar. Se hablará más sobre el algoritmo dentro del desarrollo de la práctica, conforme se analicen los bloques de código utilizados.

Se utilizarán datos que ya habíamos utilizado para otras prácticas. La fuente de datos es el archivo *hipoteca.csv*, el cual cuenta con 202 registros con los siguientes datos:

- ingresos: son ingresos mensuales de 1 o 2 personas, si están casados.
- gastos_comunes: son gastos mensuales de 1 o 2 personas, si están casados.
- pago_coche
- gastos_otros
- ahorros
- vivienda: valor de la vivienda.
- estado_civil: 0-soltero, 1-casado, 2-divorciado
- hijos: cantidad de hijos menores (no trabajan).

- trabajo: 0-sin trabajo, 1-autonomo, 2-asalariado, 3-empresario, 4-autonomos, 5-asalariados, 6-autonomo y asalariado, 7-empresario y autonomo, 8-empresarios o empresario y autónomo
- comprar: 0-alquilar, 1-comprar casa a través de crédito hipotecario con tasa fija a 30 años.

Desarrollo

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas *Hipoteca.csv*. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de *n* dimensiones y *seaborn*, para la visualización de datos basado en *matplotlib.*

Más tarde, importamos el archivo *Hipoteca.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir del explorador de archivos que abrimos utilizando el comando *files.upload()*. Una vez importados los datos, mostramos el DataFrame que los contiene:

	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

Figura 1: DataFrame *Hipoteca*.

Como los datos ya se encuentran preparados, comenzaremos mostrando información sobre ellos, para después poder hacer una selección de variables para evitar la alta dimensionalidad de los datos. Observamos que al imprimir información sobre la variable *comprar*, ésta ya se encontraba previamente agrupada (entre los que sí tienen posibilidad de compra y los que no) en un análisis anterior.

Antes de eliminarla, utilizamos la biblioteca *seaborn* para mostrar un *pairplot* (el cual nos permitirá visualizar rápidamente gráficas de dispersión que muestran la relación entre las variables de nuestro *dataframe*). Dentro de este *pairplot*, utilizamos la variable *compra* como *hue*, destacando su valor con respecto de las demás variables que se están graficando. En la siguiente

imagen, de color azul se muestra a aquellas personas que no tienen la posibilidad de comprar (y por lo tanto rentarán) y en naranja a aquellas que sí tienen dicha posibilidad:

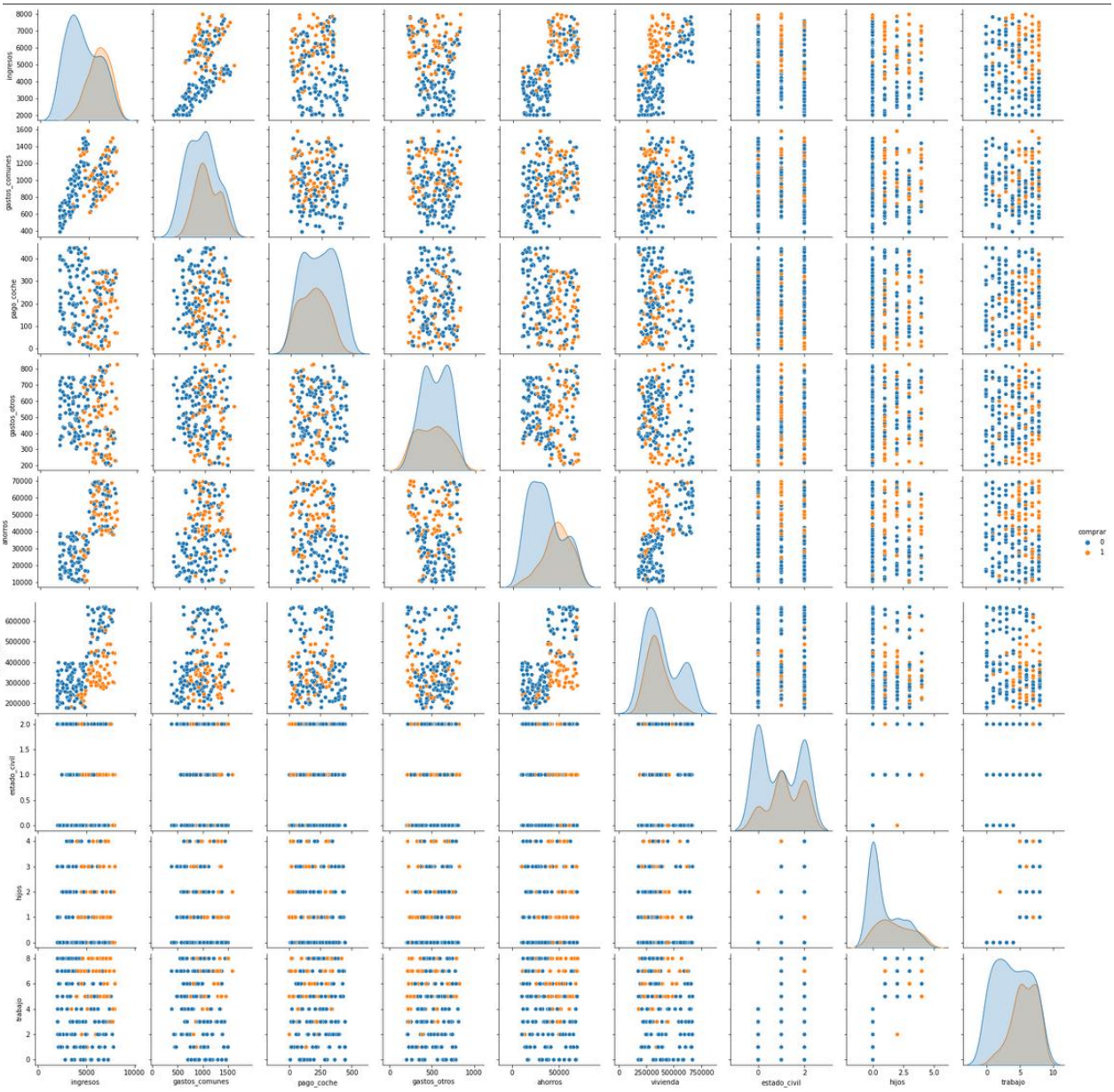

Figura 2: Resultado de utilizar la función *pairplot* sobre el *dataframe.*

Las gráficas de tipo *pairplot* son una gran manera de identificar tendencias para un análisis posterior de los datos.

Más tarde, a manera de pormenorizar el análisis, obtenemos nada más algunas gráficas (para un par de variables) utilizando la función *scatterplot* de *seaborn*. A continuación, se muestra un ejemplo con las variables *ahorros* e *ingresos*, utilizando como *hue* la variable *comprar*:

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

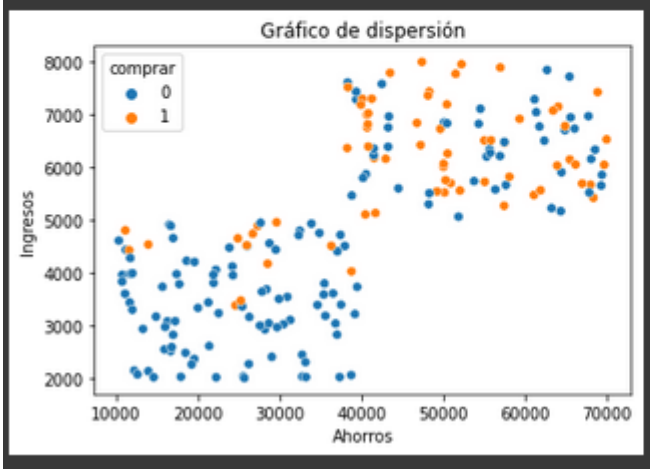

Figura 3: Resultado de utilizar la función *scatterplot* sobre el *dataframe.*

En la gráfica anterior, notamos una dispersión moderada de los datos, sin tendencias demasiado claras para definir la relación entre *ahorros* e *ingresos.*

Más tarde, utilizamos un método más preciso para definir la correlación entre las variables, la *matriz de correlaciones*, con el método *corr* de Python, pasando como parámetro que queremos utilizar (en este caso utilizaremos el método Pearson para calcular la correlación). Este método asigna un valor entre -1 y 1, donde 0 implica que no hay correlación, 1 es una correlación total positiva y -1 es una correlación total negativa. Una correlación con valor de 0.7, por ejemplo, implicaría que dos variables tienen una relación significativa y positiva, donde si la variable A sube, la variable B también subirá. Si la correlación fuera negativa y la variable A sube, la variable B bajaría.

Para nuestros datos, obtuvimos lo siguiente de la matriz de correlación:

	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

Figura 4: Matriz de correlación entre los datos. Vemos que, por ejemplo, existe una fuerte relación entre los ingresos y los ahorros, con un valor de 0.712 positivo.

Al imprimir los 10 mayores valores, obtenemos lo siguiente:

```
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
```

Figura 5: 10 mayores valores de correlación.

Otra forma de obtener una representación gráfica de la correlación entre las variables es utilizando un *heatmap*, como se muestra a continuación:

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

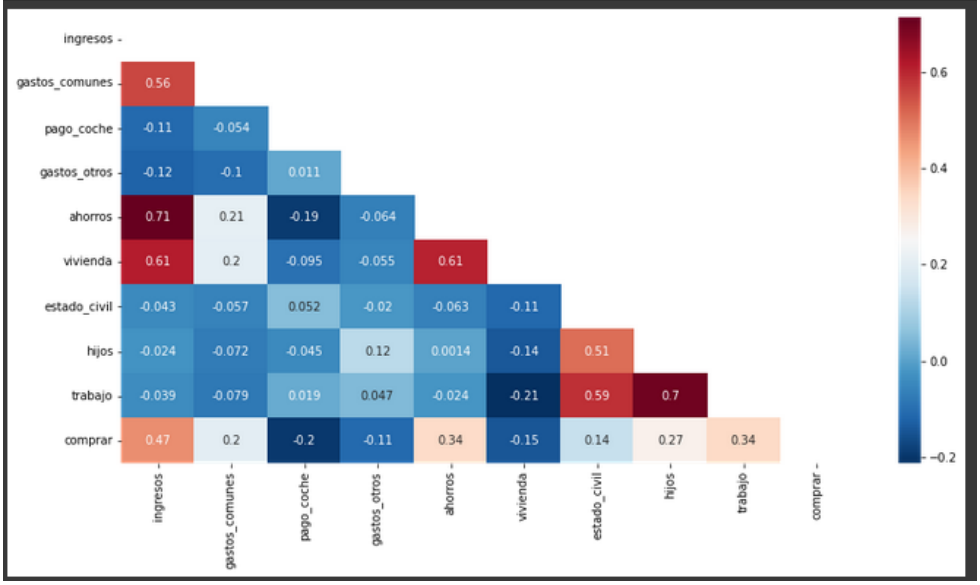

Figura 6: *Heatmap* de los valores de correlación.

Vemos aquí que existe una correlación alta entre *trabajo* e *hijos*, también. Hecho el análisis anterior, decidimos eliminar la variable *comprar* ya que este fue un campo calculado con base en un análisis hipotecario anterior, ya representaba una condición.

Posteriormente, se hace la aplicación del algoritmo *k-means* particional y, ya que son algoritmos basados en distancias, es fundamental hacer un escalamiento de los datos para que cada una de las variables que utilizamos contribuya de manera igual en el análisis.

Importamos *StandardScaler* y *MinMaxScaler* de *sklearn.preprocessing* para poder hacer el escalamiento de la matriz de datos con el método *StandardScaler*. El método *MinMax* podrá ser útil cuando se identifica que ninguna variable tenga un dato fuera de rango (por ejemplo, una edad de 150 años). Por defecto, utilizaremos la matriz euclidiana para la matriz de distancias.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler  #min max scaler tien
estandarizar = StandardScaler()                                 # Se instancia el obje
MEstandarizada = estandarizar.fit_transform(MatrizHipoteca)   # Se calculan la media

#MINMAXSCALER puede ser útil cuando se identifica que ninguna variable tenga un dato
# por defecto se utiliza distancia euclideana para matriz de distancias.
```

```
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

202 rows × 9 columns

Figura 7: Matriz estandarizada utilizando el método *StandardScaler*.

Luego, importamos la biblioteca *KMeans* de *skleanr.cluster* y *pairwise_distances_argmin*_min de *sklearn.metrics.* Utilizamos *random_state* para inicializar el generador interno de números aleatorios. Para conocer cuántos clústers tenemos que formar, utilizamos el método del codo. Este método es una herramienta para estimar el número adecuado de grupos a formar. Para este método, debemos calcular la suma de las distancias al cuadrado (SSE) entre cada elemento del clúster y su centroide, para varias configuraciones de *k*. Luego, trazamos la curva de SSE de acuerdo al número de grupos *k* y debemos de visualizar un punto de inflexión en la curva para calcular el número adecuado de grupos. Mostramos el algoritmo y el resultado a continuación:

```
SSE = []
for i in range(2, 12):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(MEstandarizada)
    SSE.append(km.inertia_)

#Se grafica SSE en función de k
plt.figure(figsize=(10, 7))
plt.plot(range(2, 12), SSE, marker='o')
plt.xlabel('Cantidad de clusters *k*')
plt.ylabel('SSE')
plt.title('Elbow Method')
plt.show()
```

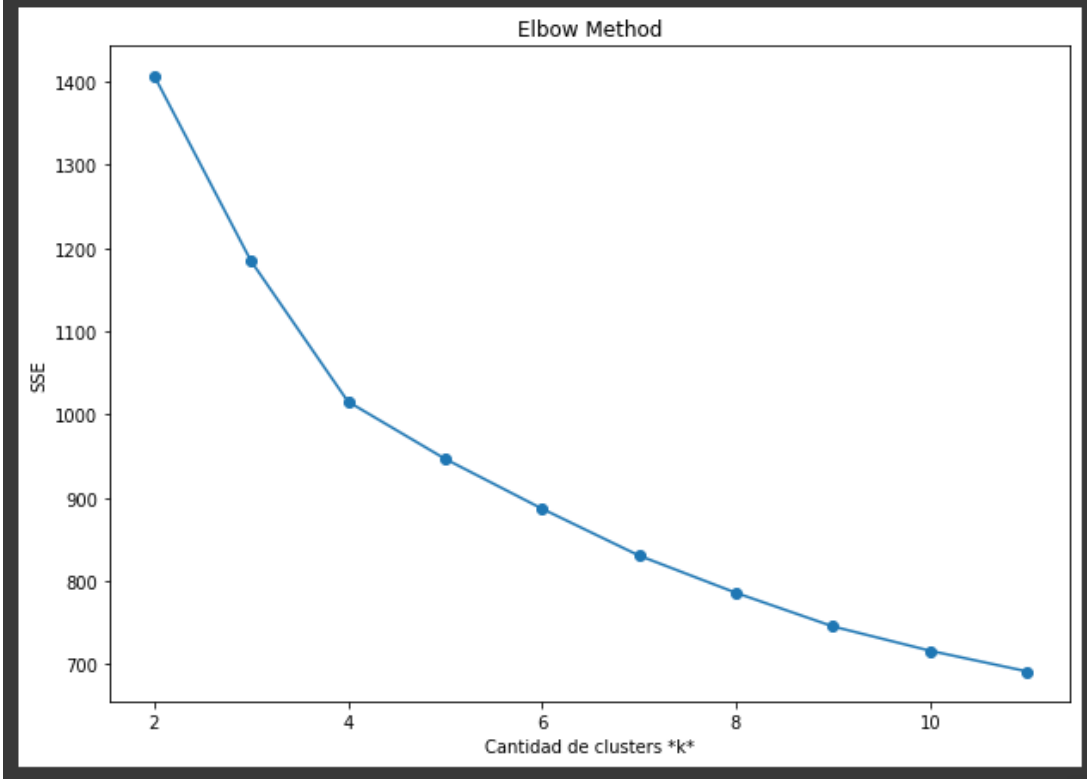

Figura 8: Gráfica de SSE, donde, a simple vista, podríamos decir que *k=4* representa un punto de inflexión de la curva.

Si bien en este caso es relativamente fácil observar que *k=4* representa el punto de inflexión de la curva, en la práctica no siempre es tan fácil de encontrar. En Python, podemos utilizar la biblioteca *kneed* para usar el método *KneeLocator*, el cual facilita la tarea de encontrar el "codo" de la curva pasando como parámetro los valores de SSE, el tipo de curva (en este caso convexa) y la dirección (en este caso decreciente).

El método *KneeLocator* nos muestra lo siguiente de forma gráfica:

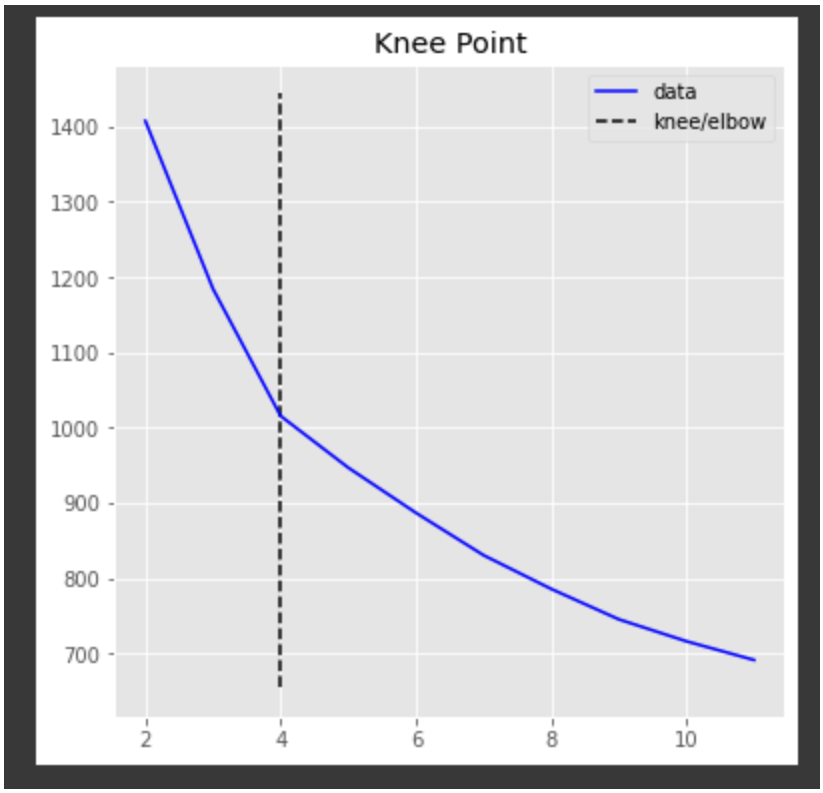

Figura 9: Localización del codo de la curva utilizando *KneeLocator*.

Conociendo el número de *k* clústeres que debemos de crear, utilizamos *KMeans* para dividir los datos en clústeres y luego los imprimimos:

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

202 rows × 10 columns

Figura 10: División de los datos en *k* clústeres.

donde utilizando el método *count()*, vemos que el clúster 0 tiene 49 elementos, el clúster 1, 56, el clúster 2, 54 y el clúster 3, 43.

Más tarde, obtenemos los centroides para cada uno de los clústeres en los que dividimos los datos. Estos centroides nos permitirán observar las características promedio de los elementos en cada uno de los clústers. Analizaremos cada uno de ellos en las conclusiones.

Finalmente, graficamos los elementos de cada clúster en una impresión tridimensional, mostrando los centros de los clústers. Esta impresión nos permitirá tener una noción general de cómo se colocan los datos respecto a la cercanía con los clústeres.

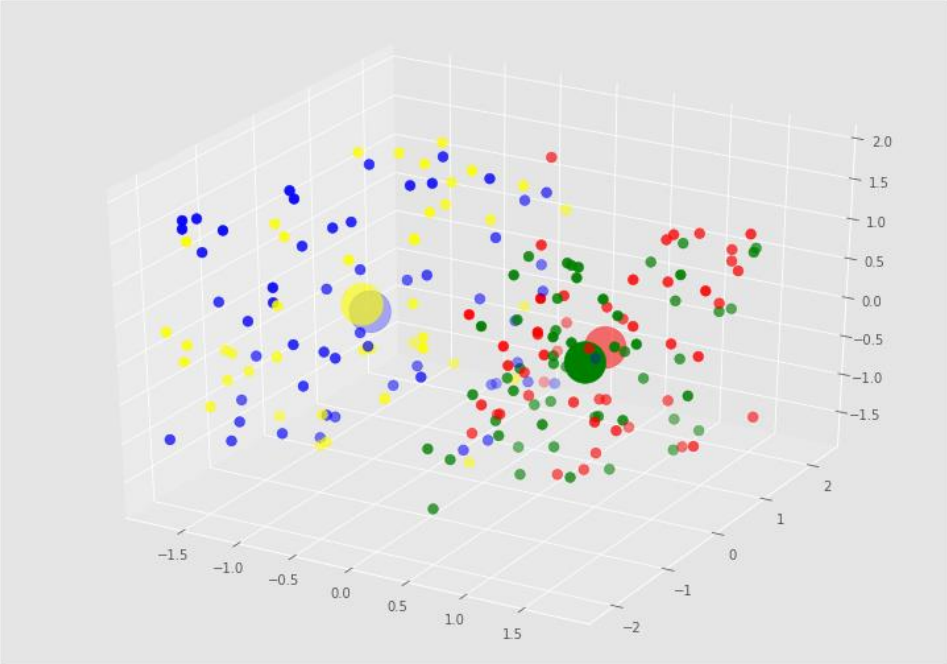

Figura 11: Impresión de los centroides y los elementos de cada clúster en una gráfica tridimensional.

Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo obtener clústeres de usuarios con características similares, evaluados para la adquisición de una casa a través de un crédito hipotecario con tasa fija a 30 años. Para lograr tal objetivo, se utilizó el algoritmo de clústering particional.

Haciendo una descripción de los clústeres obtenidos con el algoritmo y el número de clústeres indicado por el método del codo, tenemos que:

Clúster 0: Conformado por 49 casos de una evaluación hipotecaría, con un ingreso promedio mensual de 6358 USD, con gastos comunes de 1117 USD, otros gastos de 465 USD y un pago mensual de coche de 190 USD. Estos gastos en promedio representan menos de la tercera parte del salario mensual (1772 USD). Por otro lado, este grupo de usuarios tienen un ahorro promedio de 50687 USD, y un valor promedio de vivienda (a comprar o hipotecar) de 497262 USD. Además, en su mayoría son solteros (0-soltero), casi sin hijos menores y tienen un tipo de trabajo, en su

mayoría, asalariado (2-asalariado). Al representar menos de la tercera parte de los ingresos, los gastos no son suficientes como para declinar la posibilidad de una renta inmobiliaria o de un préstamos hipotecario.

Clúster 1: Es un segmento de clientes conformado por 56 usuarios, con un ingreso promedio anual de 3472 dólares, con gastos comunes de 905 dólares, otros gastos por 536 dólares y un pago de coche de 224 dólares. Los ahorro en este grupo son de 23957 dólares en promedio con un valor promedio de vivienda (a comprar o hipotecar) de 430860 dólares. El estado civil de este grupo es, en su mayoría casado o divorciado, con 2 hijos y con trabajo autónomo y asalariado o empresario y autónomo. Es un grupo en el cual los gastos representan prácticamente la mitad de los ingresos (1665 USD).

Clúster 2: Es un segmento de clientes conformado por 54 usuarios, con un ingreso promedio anual de 6389 USD, gastos comunes de 998 USD, otros gastos de 524 USD y pago de coche en promedio de 190 USD. Los ahorros de este grupo son de 54899 USD en promedio, con costo de vivienda de 430860 USD. El estado civil de estos usuarios es casado o divorciado, con dos hijos en promedio. Además, es un grupo que tiene trabajo autónomo y asalariado o empresario y autónomo, en promedio. Los gastos de este grupo de usuarios (1712 USD) representan menos de una tercera parte de sus ingresos.

Clúster 3: Es un segmento de clientes conformado 43 usuarios, con un ingreso promedio mensual de 3502 USD, con gastos comunes de 857 USD, otros gastos de 533 USD y un pago mensual de coche de 245 USD. Estos gastos en promedio representan casi la mitad del salario mensual (1635 USD). Por otro lado, este grupo de usuarios tienen un ahorro promedio de 24129 USD, y un valor promedio de vivienda (a comprar o hipotecar) de 291900 USD. Además, en su mayoría son solteros (0-soltero), sin hijos y tienen un tipo de trabajo asalariado (2-asalariado).

A través de la creación de clústeres, logramos definir propiedades o características comunes a grupos de usuarios dentro del conjunto de datos para créditos hipotecarios con tasa fija a 30 años. Para esta práctica, utilizamos el algoritmo de clustering particional, el cual puede ser más eficaz (computacionalmente) para grandes cantidades de datos (comparado con el algoritmo jerárquico).

A comparación con el algoritmo jerárquico, donde se produjeron 7 clústeres para los mismos datos, en este caso únicamente se produjeron 4. En ambos casos, los grupos representan de buena manera características específicas de los clientes. Quedará a interpretaciones posteriores el definir qué hacer con los clústeres ya formados.