Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 03/26/2022
Practica No.5 Clustering Particional

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.

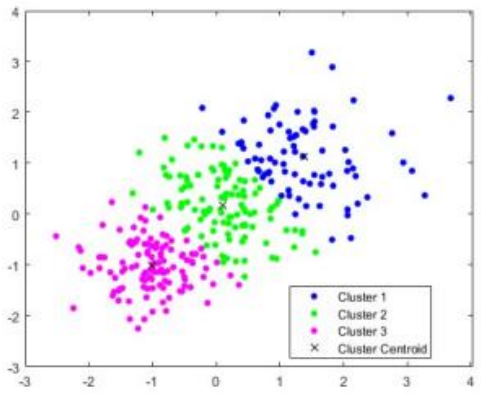
Características.

La fuente de datos tiene las siguientes características.

- 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 autónomo, 8-empresarios o empresario y autónomo
- comprar: 0-alquilar, 1-comprar casa a través de crédito hipotecario con tasa fija a 30 años.

El clustering lo que busca es hacer una segmentación de los registros en el conjunto de datos, con esto se descubren una serie de patrones que normalmente estarían ocultos los cuales relacionan a cada uno de ellos consiguiendo agruparlos, sin embargo, no se ofrece una descripción clara y es tarea de nosotros darles una interpretación adecuada.

En particular el clustering particional organiza a los elementos determinando n centroides (Punto que ocupa la posición media en un clúster), donde n es un numero que tenemos que definir nosotros, se usara el algoritmo de k-means el cual va a estar moviendo iterativamente los centroides tomando la media de cada clúster que se forme hasta un punto de convergencia donde ya no se muevan más.



Un punto importante de este algoritmo es que tenemos que definir el número de clústeres con los cuales queremos terminar y para ello es necesario utilizar otras herramientas como el método del codo, de la rodilla o el clustering jerárquico de la practica anterior los cuales nos sugieren un buen numero por el cual podríamos iniciar con nuestro análisis.

Cervantes Rubí Brandon

Desarrollo.

Como ya se ha venido haciendo, el primer paso dentro de nuestra practica siempre es importar las bibliotecas necesarias para el análisis; `pandas` para la manipulación de los datos, `numpy` para el manejo de matrices, `matplotlib` y `seaborn` para la visualización de estos datos mediante gráficos de diferente tipo dependiendo del análisis, `StandardScaler` y `MinMaxScaler` para escalar o normalizar el conjunto de datos evitando sesgos en el análisis y aumentando también el rendimiento de los algoritmos, `kneed` para el método de la rodilla el cual nos da una aproximación adecuada de cual deberá de ser el número de clústeres para el algoritmo de k-means y por último la librería de `KMeans` que se encuentra dentro de los paquetes que ofrece `sklearn` para estos desarrollos.

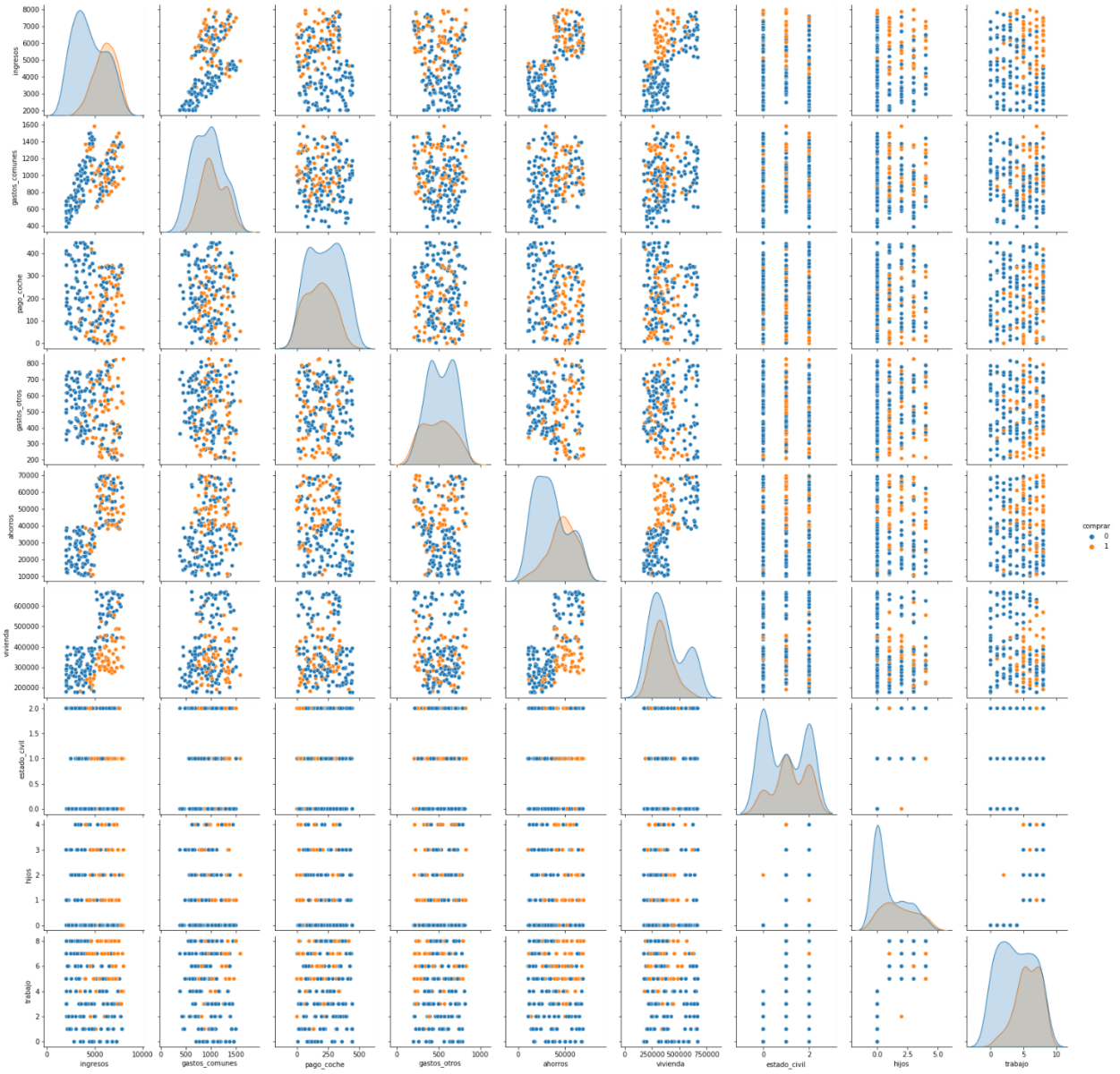
Se leyó el conjunto de datos con ayuda de `pandas` el cual es un `csv` con información de los principales gastos que presenta un usuario el cual solicitara un crédito hipotecario con el objetivo de comprar o rentar una casa, información como el numero de hijos, pagos, estado civil, etc. este conjunto de datos se compone por 202 registros inicialmente. Podemos observar además con ayuda de `pandas` el tipo de dato de cada una de las 10 columnas, variables o características que tenemos inicialmente y nos podemos dar cuenta de que todas ellas son variables numéricas por lo cual están listas para trabajar con las métricas de distancias pertinentes.

	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

Una primera agrupación la podríamos hacer por aquellos clientes que buscan comprar (comprar = 1) o alquilar (comprar = 0) una casa con ese crédito hipotecario, observando que son mayoría los usuarios que buscan alquilar el lugar con ese crédito, en total 135, en comparación con aquellos que buscan comprar este espacio, en total 67.

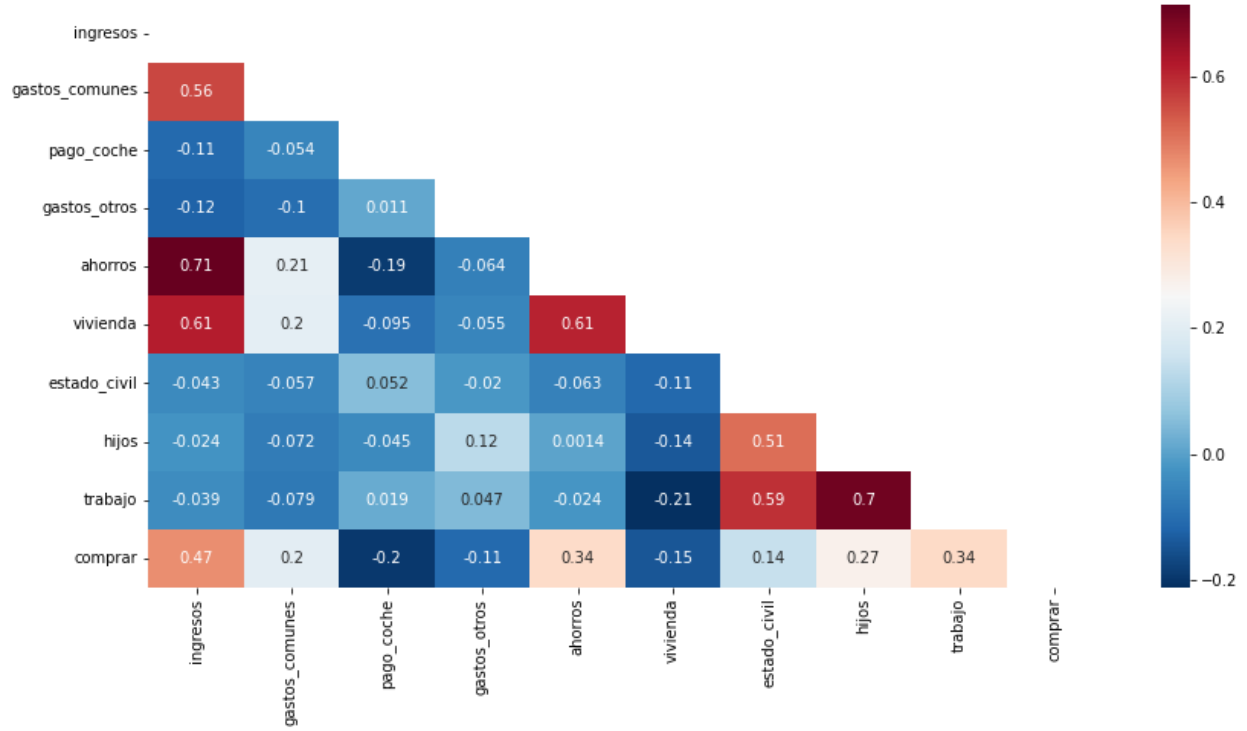
Con ayuda de `seaborn` y su método `pairplot` podemos comenzar trazando los gráficos de dispersión para estas variables con el objetivo de identificar relaciones fuertes (>0.67 en su índice de correlación) entre el conjunto de variables.

A continuación, se muestra el grafico de dispersión para las variables.



Con esta grafica nos podemos dar una idea de que pudiera existir una relación fuerte entre ahorros e ingresos, para asegurarnos de esto es más recomendable utilizar algún método que mida con exactitud esta relación como lo es trazar una matriz de correlaciones con el método de Pearson y buscar aquellos índices mayores que 0.67 o menores que -0.67. Sin embargo, al ser matrices grandes llenas de números se vuelve complicado identificar estos índices cuando el número de variables asciende, es por ello que se utiliza `pandas` para ordenar los valores de cada columna o `seaborn` para imprimir el mapa de calor.

A continuación, se muestra el mapa de calor de la matriz de correlaciones para la reducción de características.



Nos podemos dar cuenta que hijos con trabajo y ahorros con ingresos son 2 relaciones fuertes dentro del modelo, sin embargo, se consideran necesarias para el análisis por lo cual se decidió que se quedarían, caso contrario con la variable comprar la cual se quitara del modelo ya que independientemente de para que se vaya a usar el crédito los usuarios seguirán siendo candidatos al mismo por lo cual terminamos con un modelo de 9 variables.

El siguiente paso es estandarizar (media = 0 y desviación estándar = 1) y normalizar (escalado de 0 a 1) los datos para ello se utilizó el método `fitTransform` de la clase `StandardScaler` y `MinMaxScaler` respectivamente, en la siguiente pagina se muestran las matrices de datos estandarizadas y normalizadas.

Estandarización. (Media = 0, Desviación Estándar = 1 y utilizando StandardScaler.fit_transform).

	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

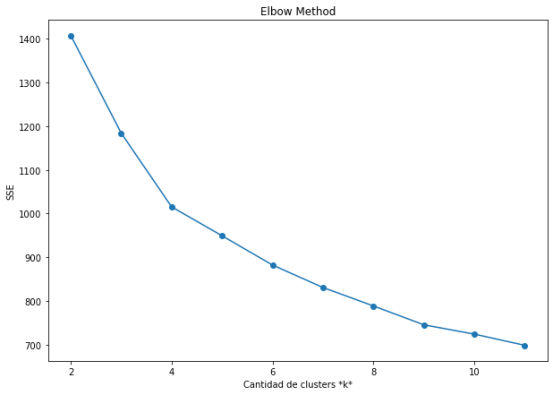
Normalización. (Escala de valores entre 0 y 1 utilizando MinMaxScaler.fit_transform).

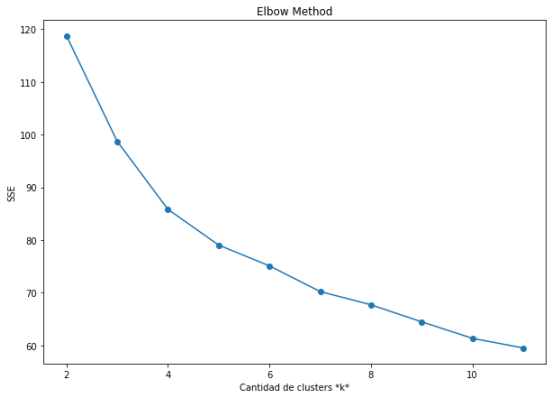
	0	1	2	3	4	5	6	7	8
0	0.668005	0.512906	0.000000	0.636364	0.665621	0.453251	0.0	0.50	0.250
1	0.792671	0.466278	0.274554	0.363636	0.552227	0.933785	0.5	0.75	0.750
2	0.744143	0.540383	0.218750	0.947368	0.790808	0.294584	1.0	0.25	1.000
3	0.851740	0.744380	0.033482	0.084530	0.741206	0.982541	0.0	0.00	0.375
4	0.695950	0.398834	0.497768	0.508772	0.523241	0.349662	0.0	0.00	0.375
...	...	...	...	...	...	...	...	...	...
197	0.305054	0.254788	0.785714	0.457735	0.006777	0.378442	0.0	0.00	0.250
198	0.326807	0.537885	0.602679	0.437002	0.193928	0.210691	1.0	0.75	1.000
199	0.196787	0.475437	0.616071	0.770335	0.423484	0.428961	0.5	0.75	1.000
200	0.221888	0.402165	0.823661	0.719298	0.162140	0.406378	0.5	0.50	0.625
201	0.331325	0.643630	0.234375	0.288676	0.027862	0.164359	0.0	0.00	0.500

Con nuestros datos ya en forma podemos comenzar con el algoritmo de clustering particional, para ello el primer paso es definir cuál será el número de clusters adecuado. La primera aproximación a este número será con el método del codo calculando para configuraciones con 2, 3, …, 12 clusters la suma de las distancias cuadradas de cada elemento con su centroide. La clase de KMeans que importamos en un inicio nos ayudara a hacer más fácil este proceso.

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

Primero para cada configuración de clusters se calcula el algoritmo de K-Means con el método fit y después se agrega la suma de sus distancias al cuadrado con su atributo inertia_ a la lista SSE la cual se va a graficar con ayuda de matplotlib para observar el punto de infleccion. A continuación, se muestra la gráfica del método del codo para los datos estandarizados a la izquierda y a la derecha para la matriz con datos normalizados.





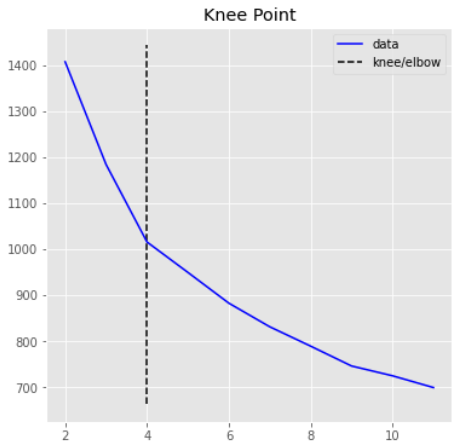
Como se puede apreciar es un poco mas complicado identificar el punto en el que la pendiente cambia de dirección mas bruscamente con datos normalizados en comparación con los datos estandarizados donde podemos ver un claro cambio de dirección para la configuración de 4 clústeres pues para los datos normalizados la gráfica se va torciendo un poco más conforme va bajando y pudiera ser que el cambio de dirección más brusco de da en 6 clústeres.

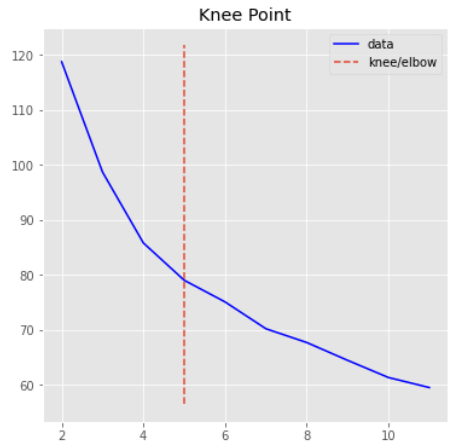
Para evitar ambigüedades es conveniente recurrir al método de la rodilla el cual nos entrega como resultado el número de clusters recomendado para hacer la partición, para ello es necesario primero instalar la librería Kneed la cual contiene la implementación de este método en su clase KneeLocator enviando como parámetros el rango de clusters, la lista de distancias SSE, la forma de la curva y la dirección decendente.

```
!pip install kneed
```

```
from kneed import KneeLocator
kl = KneeLocator(range(2, 12), SSE, curve="convex", direction="decreasing")
kl.elbow
```

Lo cual nos entregara el índice 4 para la matriz con datos estandarizados (grafica de la izquierda) y el índice 5 para la matriz con los datos normalizados (grafica de la derecha).





Tomando como base la configuración de 4 clústeres para los datos estandarizados ahora si podemos aplicar nuestro algoritmo de KMeans y etiquetar a los registros conforme a su clúster con el atributo `labels_` después de utilizar el método `predict` el cual predice el clúster más cercano para cada registro.

```
MParticional = KMeans(n_clusters=4, random_state=0).fit(MEstandarizada)
MParticional.predict(MEstandarizada)
MParticional.labels_
```

Este arreglo de etiquetas fácilmente lo podemos incorporar como una columna más al `DataFrame` de nuestra matriz resultante de la selección de características. Con esta columna podemos hacer una agrupación de los registros por el clúster al que pertenecen donde nuestro resultado final tiene 43 registros para el clúster 0, 56 registros para el clúster 1, 54 registros para el clúster 2 y 49 registros para el clúster 3 observando que el rango de usuarios por clúster va de los 43 a los 56 registros.

clusterP	
0	43
1	56
2	54
3	49

Para los datos normalizados al ser un mayor numero de clusters el rango en el que se dividen los registros evidentemente incrementa pues va d los 23 a los 53 elementos por cluster teniendo 49 registros para el primer cluster, 53 registros para el segundo, 38 registros para el tercero, 39 registros para el cuarto y 23 registrps para el quinto y ultimo cluster.

clusterP	
0	49
1	53
2	38
3	39
4	23

Para el desarrollo del análisis se utilizarán los datos estandarizados. El siguiente y ultimo paso seria la obtencion de los centroides para su posterior analisis, esto lo hacemos con pandas agrupando cada registro por el cluster al que pertenece y obteniendo el promedio con el metodo `mean` a continuacion se muestran los resultados de estos centroides.

clusterP	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo
0	3502.930233	857.209302	245.790698	533.627907	24129.139535	291900.953488	0.348837	0.000000	2.093023
1	3472.482143	905.607143	224.732143	536.589286	23957.642857	272010.535714	1.625000	2.250000	6.660714
2	6389.685185	998.851852	190.203704	524.148148	54899.722222	430860.092593	1.462963	2.222222	6.296296
3	6358.959184	1117.306122	190.755102	465.653061	50687.081633	497262.265306	0.448980	0.061224	2.122449

Clúster 0.

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).

Se descartan como candidatos al crédito por su alto índice de gastos en comparación con sus ingresos.

Clúster 1.

Es un grupo conformado por 56 usuarios buscando el crédito hipotecario los cuales tienen un ingreso promedio mensual de 3472 USD, con gastos comunes de 906 USD, otros gastos de 537 USD y un pago mensual de su coche de 225 USD. Todos estos gastos representan aproximadamente la mitad de su salario promedio mensual (1668 USD). Este grupo de usuarios también cuenta con un ahorro de 23957 USD y un valor promedio de la vivienda a comprar o hipotecar de 272010 USD. Por último, en su mayoría son casados (1-casado), con 2 hijos y un trabajo autónomo de empresario o asalariado (6 autónomo asalariado y 7 autónomo empresario).

Se descartan como candidatos al crédito por su alto índice de gastos en comparación con sus ingresos.

Clúster 2.

Es un grupo conformado por 54 usuarios buscando el crédito hipotecario los cuales tienen un ingreso promedio mensual de 6390 USD, con gastos comunes de 999 USD, otros gastos de 524 USD y un pago mensual de su coche de 190 USD. Todos estos gastos representan poco más del 25% de su salario promedio mensual (1713 USD). Este grupo de usuarios también cuenta con un ahorro de 54900 USD y un valor promedio de la vivienda a comprar o hipotecar de 430860 USD. Por último, en su mayoría son casados (1-casado), con 2 hijos y un trabajo autónomo de asalariado (6 autónomo asalariado).

Se conservan como candidatos al crédito por su bajo índice de gastos y su alto índice de ahorros.

Clúster 3.

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).

Se conservan como candidatos al crédito por su bajo índice de gastos y su alto índice de ahorros.

Conclusiones.

La técnica de clustering particional con el algoritmo de K-Means es muy útil para segmentar los registros debido a que gracias a su integración a librerías de Python se vuelve relativamente simple de implementar, sin embargo, debemos de tener en cuenta que primero tenemos que definir bien el número de clústeres con el que queremos terminar ya que esta métrica nos pudiera sesgar los análisis si ingresamos un numero de clústeres inadecuado.

Para poder conocer este número adecuado de clústeres se suelen utilizar los métodos del codo o la rodilla, el primero de ellos es meramente visual calculando las distancias desde cada registro hasta su centroide para obtener una grafica y tratar de identificar cual es el punto donde la gráfica cambia de dirección mas abruptamente sin embargo este cambio no siempre es evidente y por ello se recomienda corroborar también con el método de la rodilla el cual además de mostrar esta grafica también entrega el índice recomendado. Otra técnica que se podría utilizar es el clustering jerárquico cuando el numero de registros son menores ya que este tipo de segmentación entrega el número de clústeres a usar.

Por último, debido a que es un algoritmo el cual basa su funcionamiento en el calculo de las distancias es fundamental que utilicemos técnicas de normalización o estandarización de los datos para evitar sesgos en los mismos y mejorar el rendimiento de los algoritmos.

Fuentes.

sklearn.cluster.KMeans. (2022). Scikit-Learn. 2022, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html