Universidad Nacional
Autónoma de México

Inteligencia Artificial

Clustering Particional - K-means

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 5

Profesor: Dr. Guillermo Gilberto Molero Castillo

29 de marzo de 2022

Introducción

En esta práctica se utilizará el algoritmo K-medias para agrupar un conjunto de datos que se trata de un crédito hipotecario. Esta práctica es similar a la anterior pero se utiliza un algoritmo diferente para generar los clústeres. Se analizará cada uno de los clústeres y se utilizará el método del codo para definir el número de grupos en los que se agrupará. Por medio de otros conceptos como la selección de datos, visto en prácticas pasadas, se obtendrá el mejor modelo posible.

Objetivos

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

Se importan las bibliotecas necesarias para la realización de la práctica.

```
import pandas as pd                 # Para la manipulación y análisis de datos
import numpy as np                  # Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt     # Para la generación de gráficas a partir de los datos
import seaborn as sns               # Para la visualización de datos basado en matplotlib
%matplotlib inline
```

Se genera una vista de los datos que se van a utilizar durante la práctica.

```
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

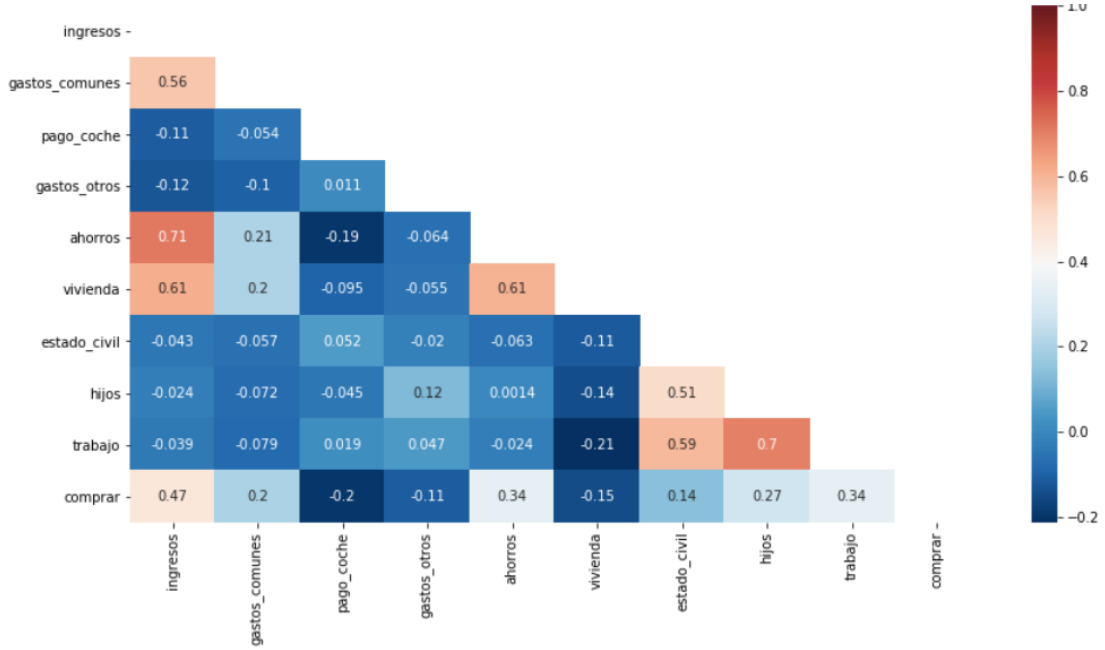
202 rows × 10 columns

Se obtuvo la información del conjunto de datos para corroborar que no existan valores nulos dentro de los datos.

```
Hipoteca.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202 entries, 0 to 201
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ingresos        202 non-null    int64
 1   gastos_comunes  202 non-null    int64
 2   pago_coche      202 non-null    int64
 3   gastos_otros    202 non-null    int64
 4   ahorros         202 non-null    int64
 5   vivienda        202 non-null    int64
 6   estado_civil    202 non-null    int64
 7   hijos           202 non-null    int64
 8   trabajo         202 non-null    int64
 9   comprar         202 non-null    int64
dtypes: int64(10)
memory usage: 15.9 KB
```

Se generó gráficas de las relaciones que existen entre todas las variables para generar una evaluación visual de qué variables están altamente correlacionadas. Además, se obtuvo la matriz de correlaciones utilizando el método de Pearson. A partir de un mapa de calor de esta matriz de correlaciones se puede ver dónde generar una reducción de dimensionalidad. Existen cuatro variables altamente correlacionadas: ingresos con ahorros y trabajos con hijos; sin embargo, estos datos son relevantes para el modelo y la correlación no es tan alta. Por esto solo se elimina la variable de comprar ya que esta variable representa un agrupamiento.



Se generó una variable a partir de las variables que se seleccionaron.

```
MatrizHipoteca = np.array(Hipoteca[['ingresos', 'gastos_comunes', 'pago_coche', 'gastos_otros', 'ahorros', 'vivien
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

202 rows × 9 columns

Para este tipo de algoritmos se necesita que los datos contribuyan de igual manera y para esto se genera una estandarización de los datos. La estandarización se genera como en prácticas pasadas.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
estandarizar = StandardScaler()                              # Se instancia el objeto StandardScaler o MinMaxScaler
MEstandarizada = estandarizar.fit_transform(MatrizHipoteca)   # Se calculan la media y desviación y se escalan los d
```

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

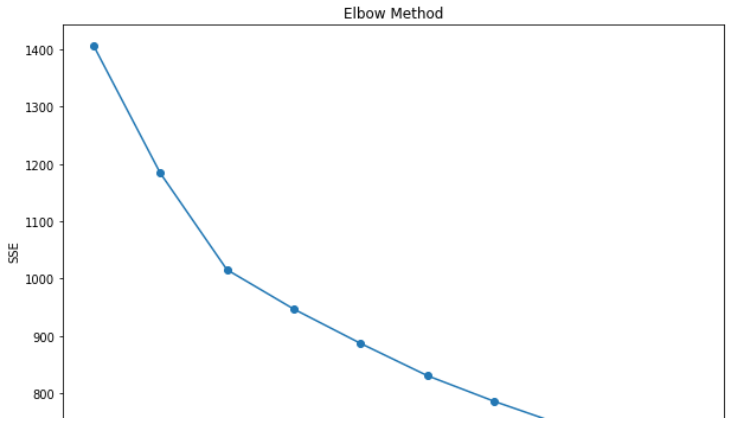
	0	1	2	3	4	5	6	7	8
0	0.620129	0.104689	-1.698954	0.504359	0.649475	0.195910	-1.227088	0.562374	-0.984420
1	1.063927	-0.101625	-0.712042	-0.515401	0.259224	1.937370	-0.029640	1.295273	0.596915
2	0.891173	0.226266	-0.912634	1.667244	1.080309	-0.379102	1.167809	-0.170526	1.387582
3	1.274209	1.128886	-1.578599	-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

Lo primero que se realiza es el método del codo para determinar el número de agrupaciones que se deberá seleccionar para el algoritmo. Para esto se utiliza el método de *inertia* para generar las mediciones de SSE y a partir de ellas hacer la gráfica del codo donde el punto de inflexión representa el número de K.

```
#Se importan las bibliotecas
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
```

```
#Definición de k clusters para K-means
#Se utiliza random_state para inicializar el generador interno de números aleatorios
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()
```

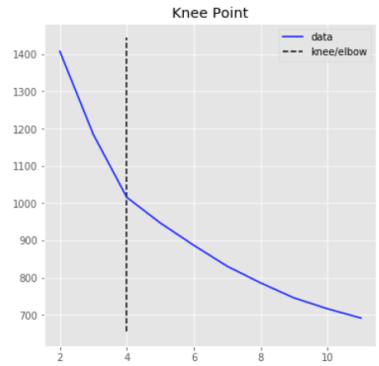


La gráfica que se genera no es totalmente clara de donde se da el punto de inflexión por lo que se utiliza una biblioteca para calcular este número. Utilizando Kneed se determinó que el número de clústeres debe de ser de 4.

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

```
4
```

```
plt.style.use('ggplot')
kl.plot_knee()
```



Después, se generaron las etiquetas para un K-medias de 4 clústeres a partir de la matriz estandarizada. Se utilizó un método que fue importado antes para generar dichas etiquetas.

```
#Se crean las etiquetas de los elementos en los clusters
MParticional = KMeans(n_clusters=4, random_state=0).fit(MEstandarizada)
MParticional.predict(MEstandarizada)
MParticional.labels_
```

```
array([0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2,
       0, 2, 0, 0, 2, 0, 0, 2, 2, 0, 2, 0, 0, 2, 0, 2, 2, 2, 0, 2, 2, 2,
       0, 0, 3, 2, 2, 0, 1, 1, 1, 1, 3, 1, 3, 3, 3, 3, 1, 1, 3, 1, 3, 1,
       1, 3, 1, 3, 1, 1, 1, 1, 3, 1, 3, 1, 1, 3, 1, 0, 3, 3, 1, 1, 3, 1,
       1, 3, 3, 1, 1, 3, 3, 1, 3, 3, 1, 3, 1, 2, 0, 2, 2, 0, 0, 2, 0, 2,
       2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0,
       0, 0, 0, 0, 0, 2, 2, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0, 0, 3, 1, 3,
       0, 3, 0, 1, 1, 3, 1, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1, 3, 3, 1, 3, 1,
       3, 3, 1, 3, 1, 1, 1, 1, 3, 1, 3, 1, 0, 3, 1, 3, 3, 1, 1, 1, 3, 3,
       1, 1, 1, 3], dtype=int32)
```

Se reemplazó la columna de comprar que tenía las etiquetas originales de los datos por las etiquetas que generó el algoritmo.

```
Hipoteca = Hipoteca.drop(columns=['comprar'])
Hipoteca['clusterP'] = MParticional.labels_
Hipoteca
```

	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

Se agruparon los diferentes clústeres generados para ver cuántos datos contiene cada una de las agrupaciones.

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

```
clusterP
0    49
1    56
2    54
3    43
Name: clusterP, dtype: int64
```

Se genera una tabla de los promedios de cada una de las agrupaciones que se generaron. A partir de esta tabla se puede ver las características de cada una de las agrupaciones. Se generó un análisis de cada uno de los clústeres.

```
CentroidesP = Hipoteca.groupby('clusterP').mean()
CentroidesP
```

clusterP	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo
0	6358.959184	1117.306122	190.755102	465.653061	50687.081633	497262.265306	0.448980	0.061224	2.122449
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	3502.930233	857.209302	245.790698	533.627907	24129.139535	291900.953488	0.348837	0.000000	2.093023

Clúster 0: Es un segmento de clientes conformado 49 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).

Clúster 1: Es un segmento de clientes conformado 56 usuarios, con un ingreso promedio mensual de 3472 USD, con gastos comunes de 905 USD, otros gastos de 536 USD y un pago mensual de coche de 224 USD. Por otro lado, este grupo de usuarios tienen un ahorro promedio de 23957 USD, y un valor promedio de vivienda (a comprar o hipotecar) de 272010 USD. Además, en su mayoría son solteros (2-divorciado), con dos hijos y tienen un tipo de trabajo asalariado (7-autónomo y empresario).

Clúster 2: Es un segmento de clientes conformado 54 usuarios, con un ingreso promedio mensual de 6389 USD, con gastos comunes de 998 USD, otros gastos de 524 USD y un pago mensual de coche de 190 USD. Por otro lado, este grupo de usuarios tienen un ahorro promedio de 54899 USD, y un valor promedio de vivienda (a comprar o hipotecar) de 430860 USD. Además, en su mayoría son solteros (1-casado), con dos hijos y tienen un tipo de trabajo asalariado (6-autónomo y asalariado).

Clúster 3: Conformado por 43 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

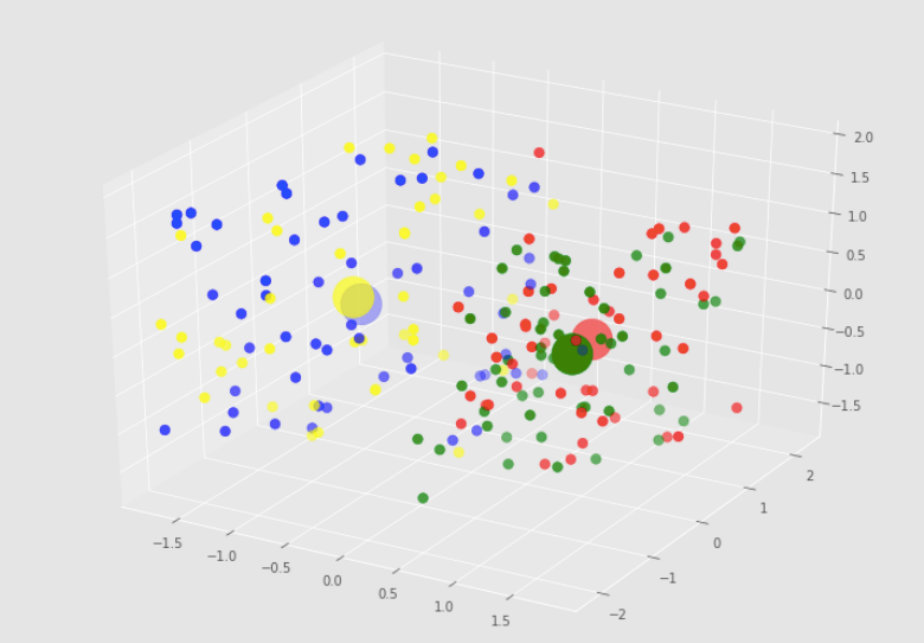
representan menos de la tercera parte del salario mensual (1772 USD). Por otro lado, este grupo de usuarios tienen un ahorro promedio de 50687 USD, y un valor promedio de vivienda (a comprar o hipotecar) de 497262 USD. Además, en su mayoría son solteros (0-soltero), casi sin hijos menores y tienen un tipo de trabajo, en su mayoría, asalariado (2-asalariado).

Por último, se generó una gráfica con tres ejes de las primeras tres variables de la tabla donde se colorea a cada uno de los diferentes grupos. En esta gráfica se puede observar la manera en que se generó la agrupación en una forma gráfica. El código para generar la gráfica fue el siguiente.

```
# Gráfica de los elementos y los centros de los clusters
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (10, 7)
plt.style.use('ggplot')
colores=['red', 'blue', 'green', 'yellow']
asignar=[]
for row in MParticional.labels_:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(MEstandarizada[:, 0],
           MEstandarizada[:, 1],
           MEstandarizada[:, 2], marker='o', c=asignar, s=60)
ax.scatter(MParticional.cluster_centers_[:, 0],
           MParticional.cluster_centers_[:, 1],
           MParticional.cluster_centers_[:, 2], marker='o', c=colores, s=1000)
plt.show()
```

Los puntos más grandes representan a los centroides que se generaron.



Conclusiones

A partir de un conjunto de datos de usuarios que fueron evaluados a través de un crédito hipotecario se generaron clústeres de agrupaciones utilizando el algoritmo de K-medias. Se utilizó la matriz de correlaciones y el método de Pearson para realizar la selección de variables. También, se utilizó el método del codo para determinar el número de agrupaciones que se iban a generar. A partir de estos métodos y de una matriz con los datos estandarizados se pudo obtener un conjunto de cuatro clústeres que fueron analizados para saber las características de cada una de las agrupaciones. Este algoritmo sirve para generar agrupaciones de a quién se le puede otorgar el crédito y a quién no. Con esta práctica se pudo aplicar lo que se vio en teoría sobre el algoritmo de K-medias y se complementó con los conocimientos vistos en prácticas anteriores.