Cervantes Rubí Brandon

Minería de Datos 2022-2, G.3, 27/09/2022
Practica N.4 Análisis de Componentes Principales (PCA)

Objetivo.

Realizar una selección de características a través de Análisis de Componentes Principales (ACP) y Análisis Correlacional de Datos (ACD).

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

Análisis de componentes principales (PCA)

Es un método de reducción de variables que tiene como objetivo describir los datos en términos de nuevas variables a las cuales se les conoce como componentes principales, siendo una combinación lineal de las variables originales y donde a su vez estas están cada vez menos correlacionadas entre sí, lo cual en principio no reduce la dimensionalidad del modelo pero si transforma los datos de tal manera que el primer componente principal tenga la mayor cantidad de información al tener un mayor índice de varianza con respecto al segundo componente principal pudiendo despreciar los componentes con menor índice de varianza según nuestro análisis. Por ejemplo, que el primer componente principal contenga el 90% de la información, el segundo el 5% de la información, el tercero el 2% y así sucesivamente.

El primer paso consiste en encontrar evidencia de que las variables iniciales están correlacionadas, esto lo podemos hacer con gráficos de dispersión o matrices de correlación. Después se tienen que estandarizar o escalar los datos para una mayor optimización.

Después se crea una matriz de covarianzas entre las variables para identificar las relaciones que puedan existir entre éstas, dicha matriz será una matriz simétrica $P\ x\ P$, donde P es el número de dimensiones inicial del modelo, además de que la diagonal principal se compone por las varianzas de las variables debido a que $Cov(a,a) = Var(a)$.

$$A = \begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix}; \qquad Cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n}$$

Para el siguiente paso a partir de la matriz de covarianzas tendremos que calcular los eigenvalores (λ) que satisfacen la relación $(A - \lambda I) = 0$ y eigenvectores (x) que satisfacen la relación $(A - \lambda I)\, x = 0$, donde I es la matriz identidad. Seguido de esto tenemos que ordenar los eigenvalores en orden decreciente siendo estos mismos los componentes principales que describirán nuestros datos y obtener el porcentaje de varianza, es decir información, que estos representan.

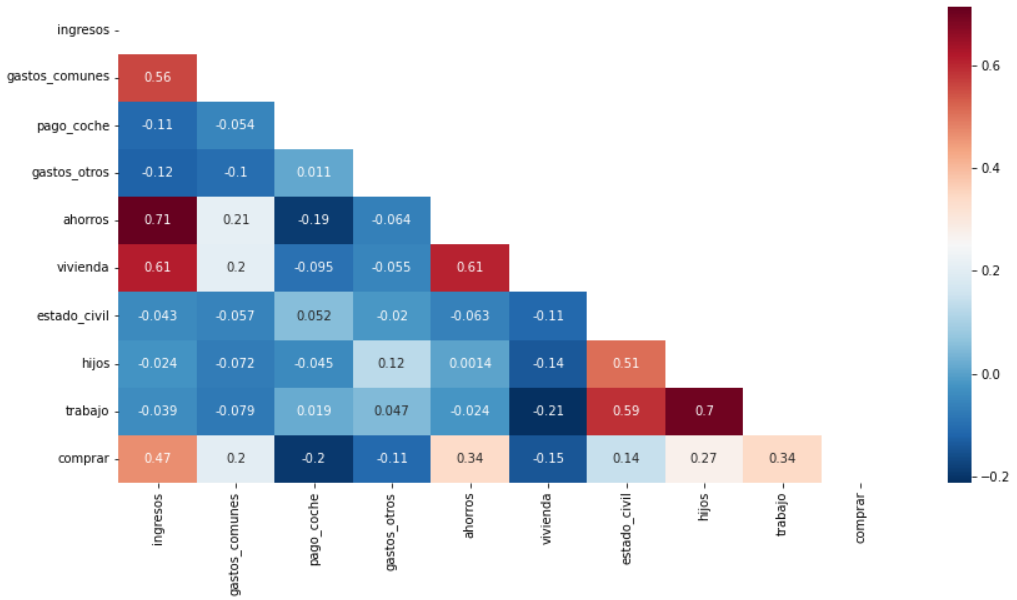
Después tenemos que hacer una suma de cada uno de estos porcentajes comenzando por el mayor hasta que lleguemos a un límite previamente establecido, comúnmente va del 70% hasta el 90%. Con base en el número de componentes que nos dieron el porcentaje adecuado tenemos que hacer la selección de características recorriendo renglones y columnas hasta obtener el número de variables que satisfagan la condición de cantidad mínima de información que pongamos como límite.

Desarrollo.

Como primer paso es necesario importar las librerías necesarias, `numpy` para el manejo de matrices, pandas para la manipulación/análisis de los datos, `matplotlib` y `seaborn` para la visualización de los datos con diferentes tipos de gráficas, algunos módulos de `sklearn` como `StandardScaler` y `MinMaxScaler` para los procesos de estandarización y normalización de la matriz de datos, PCA del módulo descomposition.

Seguido de esto ahora pasaremos a la etapa de la lectura de los datos para los usuarios candidatos al préstamo hipotecario usando pandas para crear un `dataframe` obteniendo nuestra matriz con 202 registros donde cada uno de estos tiene 10 variables o atributos que los caracterizan.

Como primer paso para implementar el algoritmo tenemos que obtener evidencia de correlaciones las cuales podemos apreciar de una forma más clara en el siguiente mapa de calor.



En el mapa de calor podemos notar que existe una correlación fuerte entre el número de hijos y el tipo de trabajo que desempeñan los posibles candidatos al crédito, además entre la cantidad de ahorros y el número de ingresos. Por otra parte, se tiene una correlación moderada entre el tipo de vivienda que solicitan y sus ingresos, los gastos comunes que llevan con sus ingresos, su estado civil con el número de hijos, etc. Por lo cual podemos decir que tenemos presencia de correlaciones en las variables del modelo, con esto en mente podemos proceder al siguiente paso.

El siguiente paso consiste en realizar una estandarización de los datos con la función `StandardScaler` que se mencionó en un inicio de la siguiente forma.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, MinMaxScaler
Estandarizar = StandardScaler()
#Estandarizar = MinMaxScaler() #Escalado
MEstandarizada = Estandarizar.fit_transform(Hipoteca)
```

Lo cual nos entrega el dataframe con los datos estandarizados (media = 0 y desviación estándar = 1) de la siguiente forma.

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	0.620129	0.104689	-1.698954	0.504359	0.649475	0.195910	-1.227088	0.562374	-0.984420	1.419481
1	1.063927	-0.101625	-0.712042	-0.515401	0.259224	1.937370	-0.029640	1.295273	0.596915	-0.704483
2	0.891173	0.226266	-0.912634	1.667244	1.080309	-0.379102	1.167809	-0.170526	1.387582	1.419481
3	1.274209	1.128886	-1.578599	-1.559015	0.909604	2.114062	-1.227088	-0.903426	-0.589086	-0.704483
4	0.719611	-0.400042	0.090326	0.027279	0.159468	-0.179497	-1.227088	-0.903426	-0.589086	1.419481
...	...	...	...	...	...	...	...	...	...	...
197	-0.671949	-1.037402	1.125381	-0.163554	-1.617963	-0.075199	-1.227088	-0.903426	-0.984420	-0.704483
198	-0.594508	0.215214	0.467439	-0.241079	-0.973876	-0.683130	1.167809	1.295273	1.387582	-0.704483
199	-1.057368	-0.061099	0.515581	1.005294	-0.183849	0.107880	-0.029640	1.295273	1.387582	-0.704483
200	-0.968013	-0.385305	1.261783	0.814462	-1.083273	0.026040	-0.029640	0.562374	0.201581	-0.704483
201	-0.578424	0.683102	-0.856468	-0.795686	-1.545397	-0.851037	-1.227088	-0.903426	-0.193753	-0.704483

Seguido de esto tenemos que calcular los eigenvalores ayudándonos de la matriz de correlaciones que acabamos de obtener. Esto lo hacemos con el módulo `PCA` pasando como argumento el número de componentes principales P con el que vamos a trabajar el cual debe de coincidir con el número de variables a considerar en el análisis, en este caso será de 10. Posterior a esto es necesario indicar con el método `fit` que queremos aplicar el algoritmo a esa matriz.

```
pca = PCA(n_components=10)      #Se instancia el objeto
pca.fit(MEstandarizada)         #Se obtiene los componentes
print(pca.components_)
```

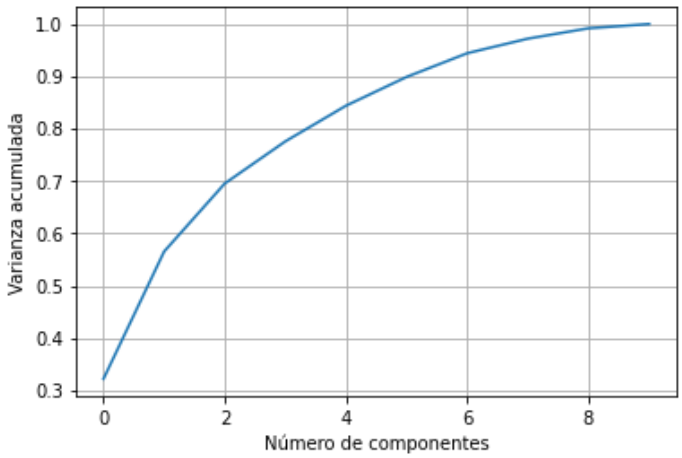
Lo cual nos regresara una lista de listas con la información de los componentes principales de la siguiente manera.

```
[[ 1.51969166e-01  4.32189650e-02 -6.46476525e-02 -1.96673143e-02
   1.23875559e-01 -6.38182230e-02  4.75775378e-01  4.17739343e-01
   4.16859930e-01  6.13524965e-01]
 [-4.38092157e-01 -1.97516012e-01  1.48420616e-01  9.59582998e-02
  -4.04347230e-01 -2.72534487e-01  4.17390240e-01  2.45466124e-01
   2.37180459e-01 -4.57527576e-01]
 [-2.96792310e-01 -1.01277160e-01 -4.56366086e-02 -2.94854960e-04
  -3.56131272e-01 -5.70197120e-01 -4.19580667e-01 -1.00451550e-01
  -4.05417015e-02  5.09743225e-01]
 [ 6.56903462e-02  1.46380791e-01  3.85786050e-01 -6.23765243e-01
  -1.46401625e-01 -1.08269581e-01  4.04364347e-01 -4.52428727e-01
  -1.51458262e-01  1.09753664e-01]
```

El siguiente paso consiste en hacer la suma de los porcentajes que corresponden a cada uno de los componentes principales. Estos valores los podemos obtener del atributo `explained_variance_ratio_` que se encuentra en el módulo de `PCA` que instanciamos anteriormente.

```
Varianza = pca.explained_variance_ratio_
print('Porporción de varianza:', Varianza)
print('Varianza acumulada:', sum(Varianza[0:6]))
```

En este caso tuvimos que sumar en total 7 componentes principales lo cual nos regresó un porcentaje del 89.9% el cual es el que más se acerca al límite de 90% de información. También podemos graficar estos porcentajes y observar la curva que se dibuja, donde se aprecie un cambio repentino en la dirección de esta podemos decir que el porcentaje de información será cada vez menor. Además, podemos trazar una intersección entre el límite de nuestro problema, en este caso 0.9 y ver cuál es el índice del componente que más se acerca para no comenzar a sumar sin conocer los valores.



Por último, ya que se tiene el número de componentes principales tenemos que evaluar las cargas para saber cuáles serán las variables con las que trabajaremos. Estas se encuentran en el atributo `components_` del objeto de `PCA` que creamos.

	0	1	2	3	4	5	6	7	8	9
0	0.549343	0.341460	0.150491	0.117465	0.489396	0.441293	0.153442	0.140073	0.161600	0.204091
1	0.159098	0.055850	0.076581	0.005565	0.135533	0.073277	0.453844	0.524006	0.551721	0.396210
2	0.015479	0.270363	0.169250	0.585070	0.225149	0.495919	0.147870	0.186289	0.071546	0.444294
3	0.116159	0.249112	0.789679	0.401542	0.093508	0.124575	0.260397	0.041123	0.065694	0.203828
4	0.133790	0.525504	0.292144	0.660085	0.167576	0.281993	0.196322	0.002478	0.030011	0.194513
5	0.057321	0.585072	0.470722	0.039776	0.315515	0.139044	0.306505	0.079488	0.008611	0.460459
6	0.060758	0.109744	0.044617	0.206281	0.057104	0.113755	0.712110	0.581456	0.210120	0.184877
7	0.062014	0.060982	0.057275	0.032287	0.031747	0.068922	0.196652	0.568478	0.780884	0.103614
8	0.303171	0.227278	0.024930	0.038606	0.738158	0.461003	0.043414	0.039218	0.028632	0.304429
9	0.733855	0.243417	0.080184	0.015805	0.076869	0.457569	0.025773	0.016795	0.065805	0.418296

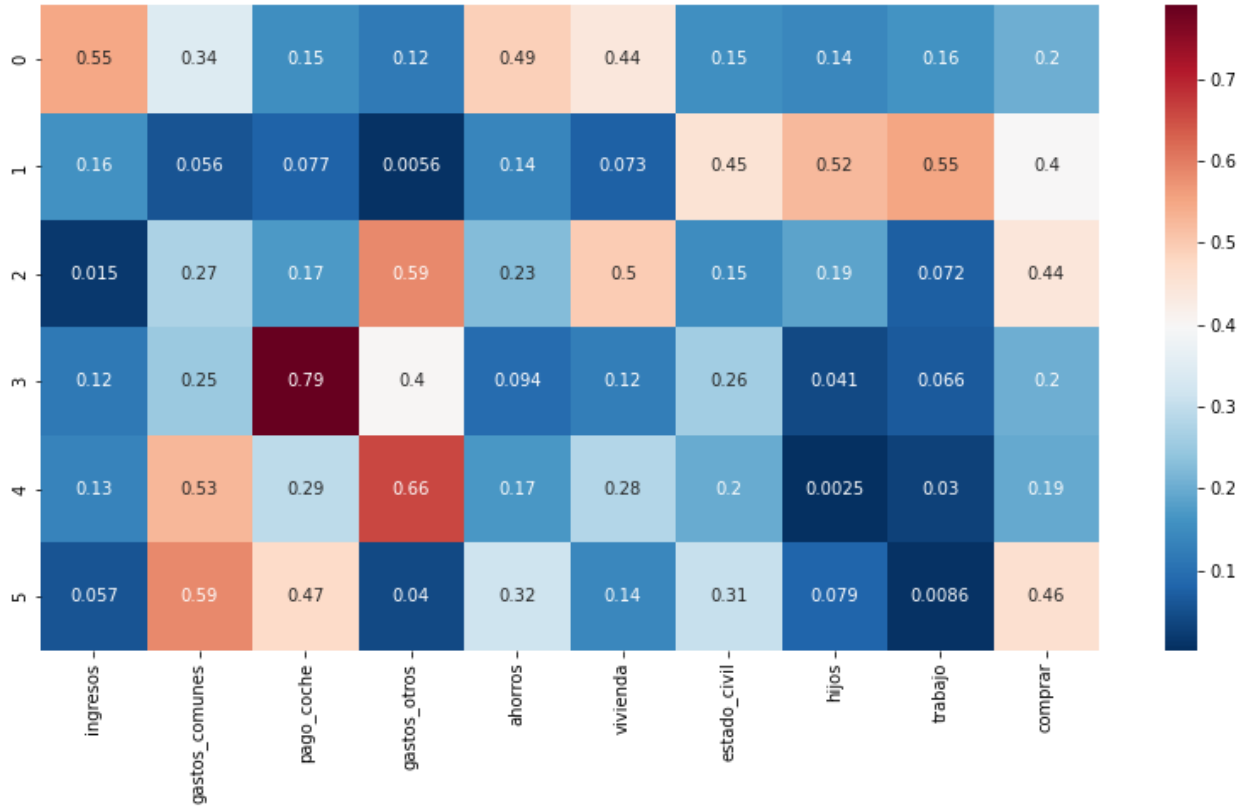
Podemos cambiar el nombre de las columnas por el de las variables correspondientes para que se vuelva aún más claro el análisis considerando además solamente el valor absoluto de los 7 componentes principales que corresponden del índice 0 al 6 con el siguiente bloque de Código.

```
CargasComponentes = pd.DataFrame(abs(pca.components_), columns=Hipoteca.columns)
CargasComponentes[:7]
```

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	0.549343	0.341460	0.150491	0.117465	0.489396	0.441293	0.153442	0.140073	0.161600	0.204091
1	0.159098	0.055850	0.076581	0.005565	0.135533	0.073277	0.453844	0.524006	0.551721	0.396210
2	0.015479	0.270363	0.169250	0.585070	0.225149	0.495919	0.147870	0.186289	0.071546	0.444294
3	0.116159	0.249112	0.789679	0.401542	0.093508	0.124575	0.260397	0.041123	0.065694	0.203828
4	0.133790	0.525504	0.292144	0.660085	0.167576	0.281993	0.196322	0.002478	0.030011	0.194513
5	0.057321	0.585072	0.470722	0.039776	0.315515	0.139044	0.306505	0.079488	0.008611	0.460459
6	0.060758	0.109744	0.044617	0.206281	0.057104	0.113755	0.712110	0.581456	0.210120	0.184877

Para el primer componente, el renglón 0, se busca el índice con el mayor valor de porcentaje y que además este por encima del 0.5 de valor. Este es `ingresos` con el 0.549343 por lo cual es una variable que se considera para el análisis. Este mismo proceso se realiza para los demás componentes, sin embargo, para que sea un poco más claro podemos trazar el mapa de calor de la matriz y con esto en mente centrarnos en los colores más cálidos y fríos por cada una de las columnas.

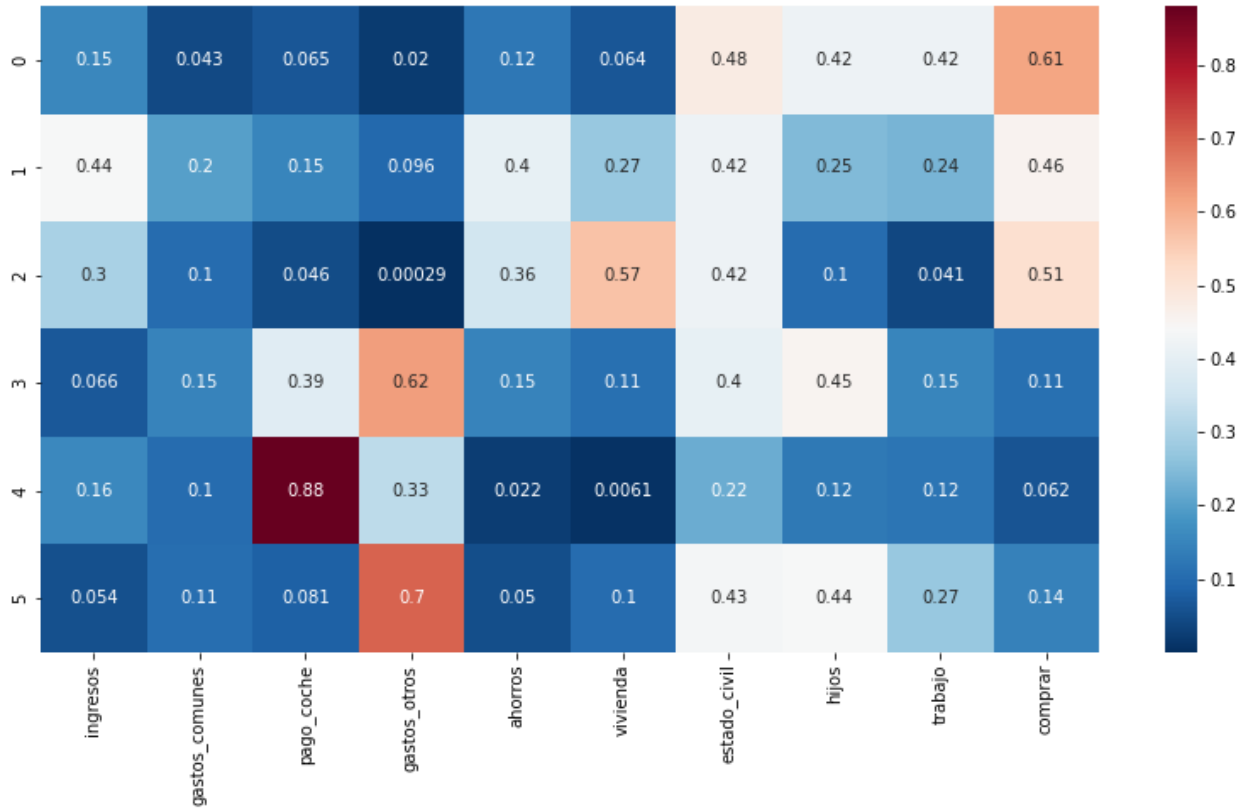
```
plt.figure(figsize=(14,7))
sns.heatmap(CargasComponentes[:7], cmap='RdBu_r', annot=True)
plt.show()
```



Por ejemplo, en el componente con índice 1 la variable que cumple con la condición es `trabajo` e `hijos` por lo cual también se consideran, en el componente con índice 2 cumple `gastos_otros` sin embargo en este caso el componente con índice 4 tiene un mayor índice por lo cual la variable también se considerara. En el componente 3 la variable con más información es `pago_coche` por lo cual también se considera y por último en el componente 5 la variable con más información es `gastos_comunes` por lo cual también se considera.

Al final nos quedaremos con 6 variables que demostraron tener la mayor cantidad de información; `ingresos`, `gastos_comunes`, `pago_coche`, `gastos_otros`, `hijos` y `trabajo` eliminando `ahorros`, `vivienda`, `estado_civil` y `comprar`.

Para el caso de tratar los datos con un escalamiento (De 0 a 1) también se tienen 6 componentes principales con el siguiente mapa de calor.



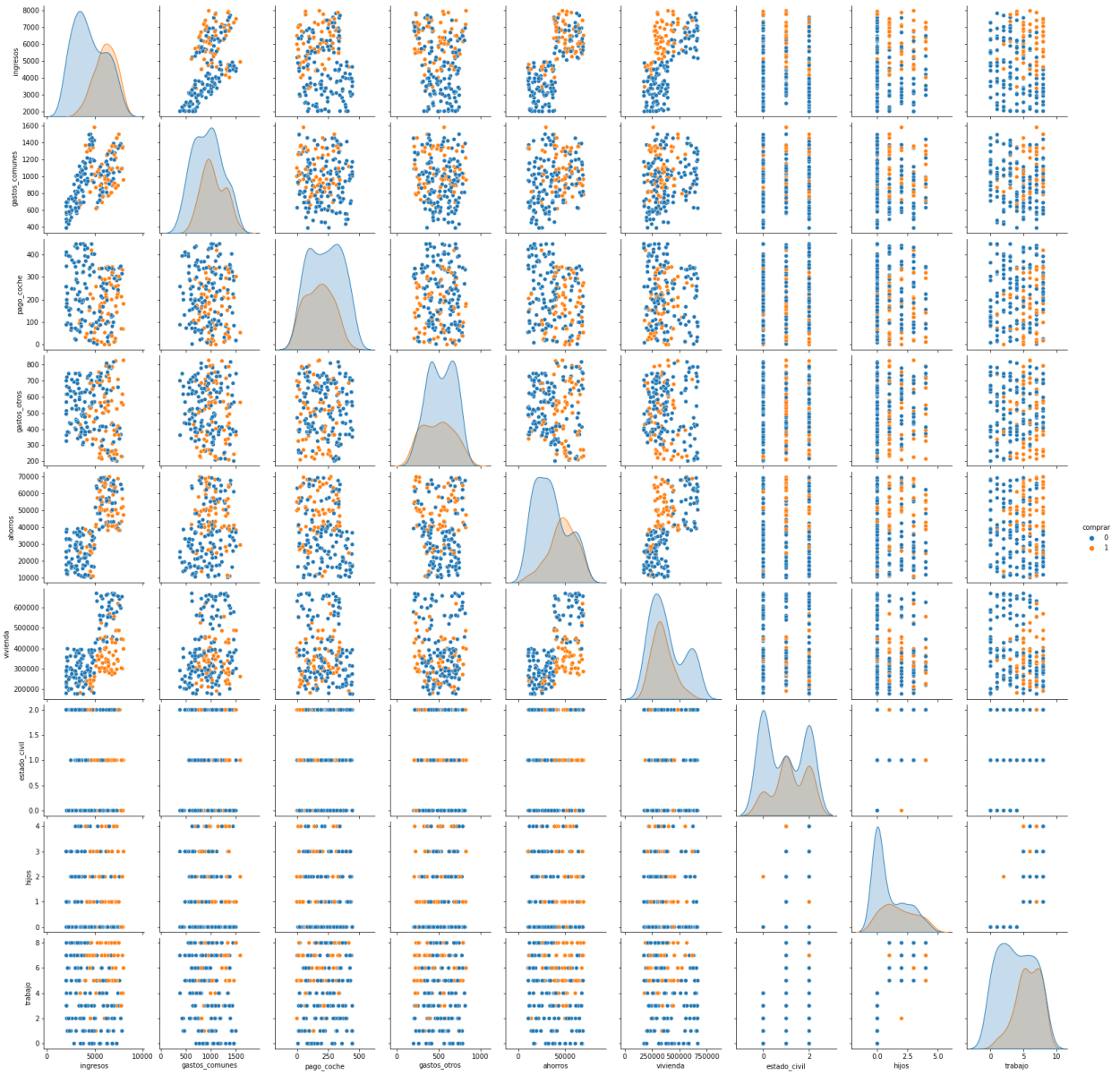
Para el primer componente (El primer renglón) la única variable que cumple con este criterio es `comprar`, para el segundo componente no se tiene ninguna variable que supere el 0.5 de valor, en el tercer componente se tiene `vivienda` y nuevamente `comprar`, para el siguiente componente tenemos `gastos_otros` cumpliendo este criterio, en el quinto componente tenemos `pago_coche` y por último en el último componente nuevamente nos encontramos con `gastos_otros`.

Por lo cual las variables con las que nos quedamos son `pago_coche`, `comprar`, `vivienda` y `gastos_otros`. Y las variables que se quedarían fuera del modelo son `ingresos`, `gastos_comunes`, `ahorros`, `estado_civil`, `hijos` y `trabajo`.

Es factible volver a replantear el valor mínimo que se tiene que cumplir para que la variable sea considerada, en lugar de 0.5 proponer 0.4 para dar pie a que mas variables puedan integrarse al modelo y hacer pruebas para ver con que limite se comporta mejor el modelo.

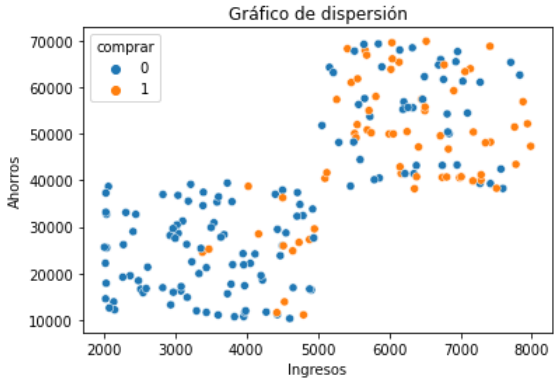
Además, es posible hacer un análisis correlacional de los datos como ya se ha revisado en prácticas anteriores. El primer paso para esto es hacer gráficos de dispersión comparando a todas las variables, con la función `pairplot` se grafica una especie de matriz con todos estos gráficos teniendo en el centro la distribución de las variables separadas por el hecho de que si lo quieren comprar o alquilar.

```
sns.pairplot(Hipoteca, hue='comprar')
plt.show()
```



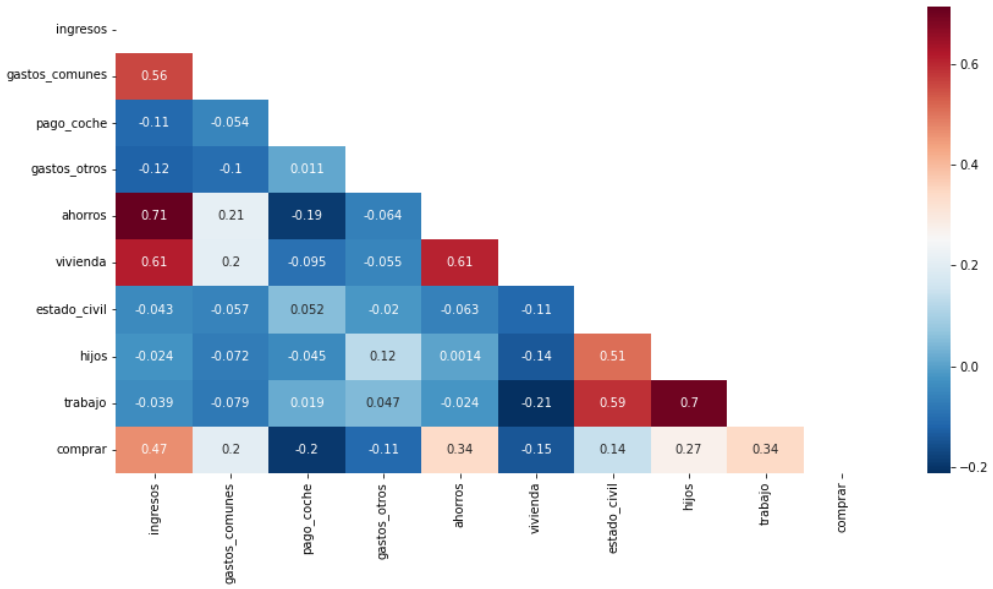
En ella podemos ver que ingresos y ahorros podrían presentar una relación fuerte por el patrón que se dibuja, lo podemos ver mas de cerca con la función `scatterplot` de `seaborn`, graficando ingresos contra ahorros del `dataframe` de hipoteca.

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



Podemos confirmar el grado de la relacion creando nuevamente una matriz de correlaciones con el método corr obteniendo el siguiente mapa de calor.

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



En el grafico podemos identificar rápidamente el nivel de correlación que existe entre las variables poniendo atención en los tonos más rojos o azules. Con esto en consideración se concluye que `ahorros` e `ingresos` tienen una correlación fuerte y `trabajo` con `hijos` por lo cual podemos deshacernos de una de las variables por cada par de relaciones, se optara por eliminar `ahorros` e `hijos` del análisis.

Al final el `dataframe` que consideraremos para el modelo quedara con 8 variables como se muestra en la siguiente imagen.

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

Conclusiones.

En conclusión, a mayor numero de variables en el modelo se elevará también el grado de complejidad computacional y los recursos necesarios para desarrollarlo. Es importante por esta razón tener en consideración varias técnicas para reducir la dimensionalidad del modelo, en una primera instancia podemos hacer un análisis correlacional simple identificando grados de correlación fuerte entre pares de variables (correlación > 0.66) y deshacernos de una de ellas.

En una segunda instancia podemos considerar también el algoritmo de análisis de componentes principales (PCA – Principal Component Analisis) el cual busca extraer e identificar cuales son las variables que tienen un mayor número de información creando vectores ponderados que representan un porcentaje de importancia el cual al sumarse debe cumplir con un criterio mínimo el cual suele ser del 70% al 90% dependiendo de que tanto necesitemos eliminar variables del modelo.