Minería de Datos **Ibañez Guzman Osvaldo**
Grupo: 1 **Semestre 2023-1**
Dr. Guillermo Gilberto Molero Castillo **Práctica #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).

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

Desarrollo

Lo primero a realizar es la importación de las bibliotecas que nos ayudarán a la manipulación, creación de matrices, gráficas y formas de visualizar los datos que usaremos, en este caso serán los datos relacionados a las hipotecas de un grupo de 202 personas, donde hay datos como lo son sus ingresos, su tipo de viviendo, la cantidad de hijo, sus ahorros entre otros dándonos un total de diez datos a analizar.

Estos datos se encuentran en el archivo llamado Hipoteca.csv, la cual usando la biblioteca pandas y la función *read_csv* obtendremos los datos y los colocaremos en una table para su manejo, dicha tabla se muestra a continuación.

```
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

El análisis de componentes principales consta de seis pasos, los cuales son los siguientes y se irán describiendo a lo largo de esta práctica.

1. Hay evidencia de variables posiblemente correlacionadas.
2. Se hace una estandarización de los datos.
3. Con los datos estandarizados, se calcula la matriz de covarianzas o correlaciones.
4. Se calculan los componentes (eigen-vectores) y la varianza (eigen-valores) a partir de la matriz anterior.
5. Se decide el número de componentes principales.
    a. Se calcula el porcentaje de relevancia, es decir, entre el 75 y 90% de varianza total.
    b. Se identifica mediante una gráfica el grupo de componentes con mayor varianza.
6. Se examina la proporción de relevancias (cargas)

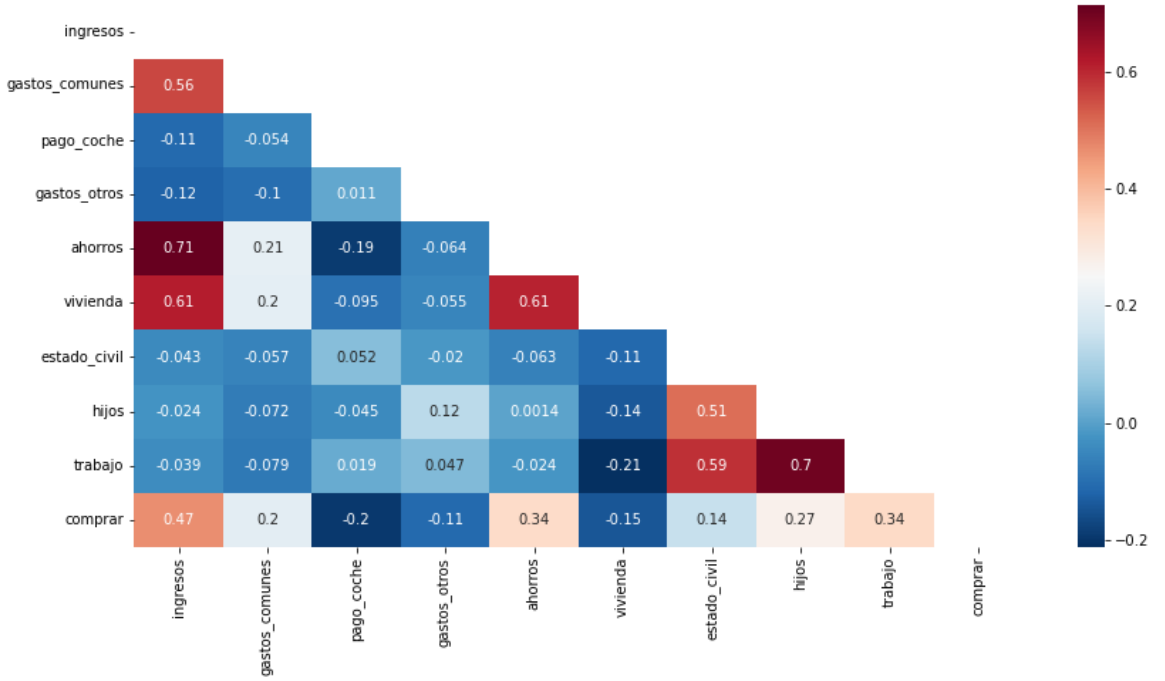
Una vez que tenemos los datos procederemos al análisis de componentes principales, donde el primer paso es **Verificar si hay evidencia de correlaciones entre variables**. Para realizar este proceso obtenemos una matriz de correlación usando la matriz de datos y la función *corr* e indicando el método que usaremos le cual será *pearson*, con la tabla de correlaciones hecha crearemos nuestro mapa de calor únicamente mostrando la matriz triangular inferior ya que es exactamente igual que la superior y la diagonal principal únicamente son valores s de 1.

Podremos identificar valores con correlaciones alta (mayores a 0.66 o menores a -0.66).

- Ingresos y Ahorros (0.71)
- Trabajo e Hijos (0.7)

Que podríamos eliminar alguna de las variables ya que su correlación es muy fuerte, pero en este paso únicamente es necesario el identificar dichas correlaciones fuertes.

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



El siguiente paso es **Estandarizar los datos** esto para que todos los datos tengan el mismo peso, ya que valores como ingresos son mucho más altos que por ejemplo los valores de la columna hijos. Para este proceso es necesario importar diferentes funciones entre ellas se encuentra *StandardScaler* la nos ayudara a escalar y otra es *MinMaxScaler* la cual sirve para normalizar los datos, podremos usar cualquiera de las dos funciones según queramos, aunque es más común usar los valores estandarizados que normalizados.

Procedemos a instanciar la función *StandardScaler* con dicha función usaremos la función *fit_transform* cuto parámetro será la matriz de datos original, dicha función calculará la media y la desviación estándar para posteriormente escalar los datos.

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

Mostraremos los datos obtenidos después del proceso de estandarización, donde cambiaremos los nombres de las columnas por los nombres de las columnas originales, mostrando los siguientes datos que se pueden observar que no se siguen teniendo las misma cantidad de datos que en un inicio.

```
pd.DataFrame(MEstandarizada, columns=Hipoteca.columns)
```

	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

202 rows × 10 columns

Los siguientes dos pasos son **Calcular la matriz de covarianzas y calcular sus componentes y su varianza**, estos procesos primeramente debemos instanciar al objeto *PCA* importado anteriormente cuyo parámetro será el número de componentes que queremos, usualmente se usan todos los componentes (el número de columnas en nuestros datos) por lo que en este caso usaremos diez componentes, aunque podrían ser menos según sea el caso.

Posteriormente obtendremos todos los componentes de nuestra matriz ya estandarizada y los mostraremos en pantalla, de igual forma podemos desplegar la varianza que nos ayudara a desarrollar el siguiente paso.

```
pca = PCA(n_components=10)
pca.fit(MEstandarizada)
print(pca.components_)
```

```
[[-0.54934302 -0.34146007  0.15049138  0.11746503 -0.48939638 -0.4412933
   0.15344247  0.14007281  0.16160045 -0.20409138]
 [ 0.15909755  0.05584992 -0.07658077  0.00556506  0.13553262 -0.07327675
   0.45384378  0.52400611  0.55172097  0.39620952]
 [ 0.01547889 -0.27036268  0.1692496   0.58506991  0.2251488   0.49591877
   0.1478699   0.18628931  0.07154605 -0.44429353]
 [ 0.11615868  0.2491118   0.78967879 -0.40154183 -0.09350787  0.12457483
   0.2603972  -0.04112256  0.0656939  -0.20382827]
 [-0.13379015 -0.52550381 -0.2921437  -0.66008455  0.16757599  0.28199311
   0.19632196  0.00247784  0.03001071 -0.19451342]
 [-0.05732063  0.58507176 -0.47072239 -0.03977609 -0.31551545  0.13904418
   0.30650475  0.07948763 -0.00861148 -0.46045932]
 [ 0.06075777 -0.10974406 -0.04461737  0.20628126  0.05710368 -0.11375472
   0.7121099  -0.58145566 -0.21011973  0.18487727]
 [-0.06201356  0.06098212 -0.05727528  0.03228667  0.03174719  0.0689216
  -0.19665212 -0.56847793  0.78088384 -0.10361354]
 [ 0.30317093 -0.22727848 -0.02493028  0.03860608 -0.73815788  0.46100264
  -0.04341353 -0.03921835  0.02863228  0.30442869]
 [ 0.73385456 -0.243417   -0.08018422 -0.01580487 -0.07686939 -0.45756887
  -0.02577314  0.01679541  0.06580472 -0.41829603]]
```

Posteriormente debemos **Decidir el número de componentes principales**, para este proceso usaremos la varianza calculada en el paso anterior mostrándola con la función *explaines_variance_ratio_*, dicha función nos desplegara un vector de números las cuales son las porciones de la varianza, este vector funciona forma acumulativa quiere decir que si únicamente tomamos una variable el valor será del 27.36%, pero si tomamos dos seria el valor anterior (27.36) sumado con el segundo dato (23.95) siendo el porcentaje de 51.31%.

Lo que buscamos es que la suma de dichas varianza este en entre un 75% y 90% de la varianza total, a continuación, se muestran algunos valores de la varianza acumulado usando de cuatro a siete variables.

- 4 - 72%
- 5 - 81%
- 6 - 89%
- 7 - 93%

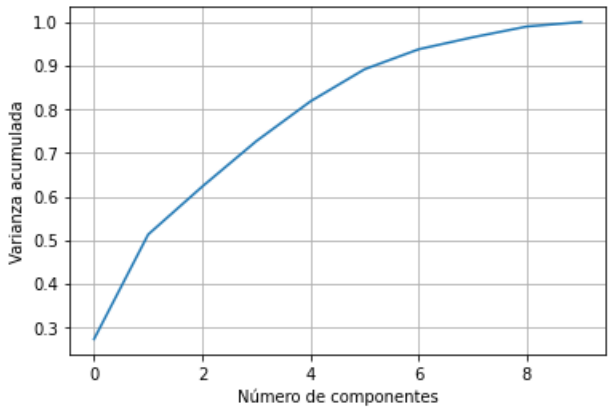
Donde los valores que buscamos serian usando cinco o seis variables, en este caso usaremos seis componentes, pero se usa el valor que necesitemos al momento de realizar el análisis de componentes principales.

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

```
Proporción de varianza: [0.27368381 0.23958688 0.10991099 0.10411098 0.09105662 0.07352523
 0.0457761  0.02745036 0.02469122 0.01020781]
Varianza acumulada: 0.8918745043774534
```

Otra forma de visualizarlos seria graficando la varianza acumulada con respecto al número de componentes, aunque esto dos da una vista general y posiblemente identificar más rápido cuantas componentes debemos utilizar, es mas complicado tener el valor exacto para cada varíanza.

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Número de componentes')
plt.ylabel('Varianza acumulada')
plt.grid()
plt.show()
```



El ultimo paso es la **Examinar la proporción de relevancias (cargas)**, para esto usaremos los componentes calculados entre mayor sea el valor absoluto delo componente mayor será su importancia, entre mayor sea el valor mayor será la importancia de esa variable.

Para este proceso recordaremos que decidimos elegir únicamente seis componentes es decir los registros del 0 al 5, lo primero a definir será un valor mínimo que deberá tener en ese caso un valor mínimo de 50% para que la consideremos la variable más fuerte.

El proceso para encontrar las variables mas fuertes debemos ir registro por registro, es decir comenzaremos en el registro 0, y buscaremos los valores mas alto que supere el 50% (0.50) que es el caso de **ingresos**, una vez hallado el valor en el renglón se busca si existe un valor más alto en la columna de dicho dato, lo cual en este caso no hay un dato mayor.

Para el registro numero 1, contamos con dos valores mayores al 50%, los cuales son **hijos** y **trabajo**, ambos del registro 1 ya que en sus respectivas columnas no hay valores mayores a los de dicho registro.

Al pasar al registro 2 encontramos que el valor de **gastos_otros** tiene un valor del 58.5% por lo que lo seleccionaremos, pero si buscamos en sus misma columna observaremos que el valor en el registro 4 es mayor, con un valor de 66%.

Para el registro 3 el valor del renglón mas alto esta en **pago_coche** y es el mas alto en su columna.

Para el registro 4, tenemos dos valores mayores al límite tomado, **gastos_comunes** con un 52.2% pero al buscar en la misma columna vemos que el valor de **gastos_otros** son mayor en el registro 5 y el otro valor más alto del 50% es **gastos_otros** que ya hemos seleccionado anteriormente.

Y al buscar en el registro 5, el valor mas alto es de **gastos_comunes** que ya había anteriormente, entonces ya concluimos con todos los registros según las componentes que tomamos, por lo que podemos decir como componentes principales son

- ingresos
- gastos_comunes
- pago_coche
- gastos_otros
- hijos
- trabajo

Una vez realizado este análisis es recomendable eliminar las otras variables, aunque esto depende del tipo de análisis que estemos realizando.

```
CargasComponentes = pd.DataFrame(abs(pca.components_), columns=Hipoteca.columns)
CargasComponentes
```

	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
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

Siguiendo el proceso deberemos eliminar las columnas de **ahorros**, **vivienda**, **estado_civil** y **comprar**. Así que usaremos la función *drop* e indicaremos que columnas queremos eliminar, pasando de una matriz de diez columnas a seis columnas.

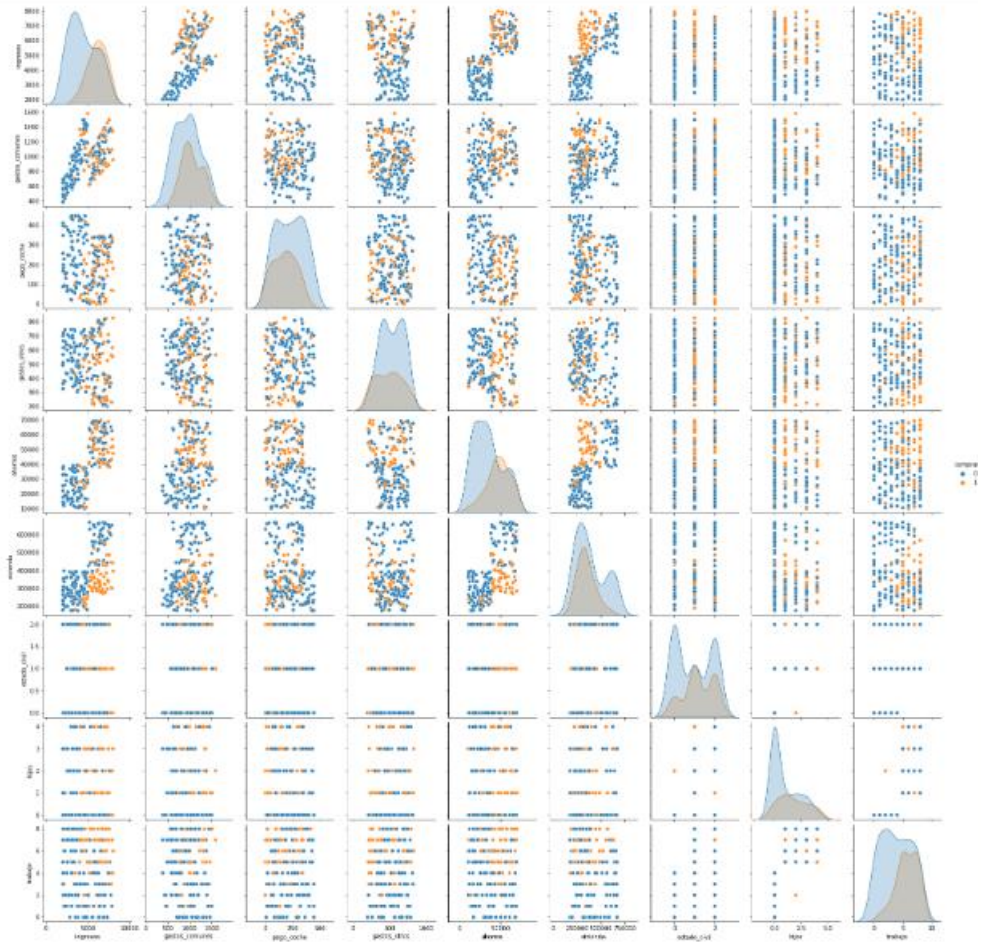
```
DatosHipotecaACP = Hipoteca.drop(columns=['ahorros', 'vivienda', 'estado_civil', 'comprar'])
DatosHipotecaACP
```

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

202 rows × 6 columns

Otra forma de reducir la dimensión de los datos es con un análisis correlacional de datos (ACD), donde una de las formas de hacerlo es con un análisis visual, mostrando los gráficos de dispersión para intentar ver la correlación que existe entre las diferentes variables, aunque es algo complicado de verlo ya que buscamos relaciones fuertes.

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

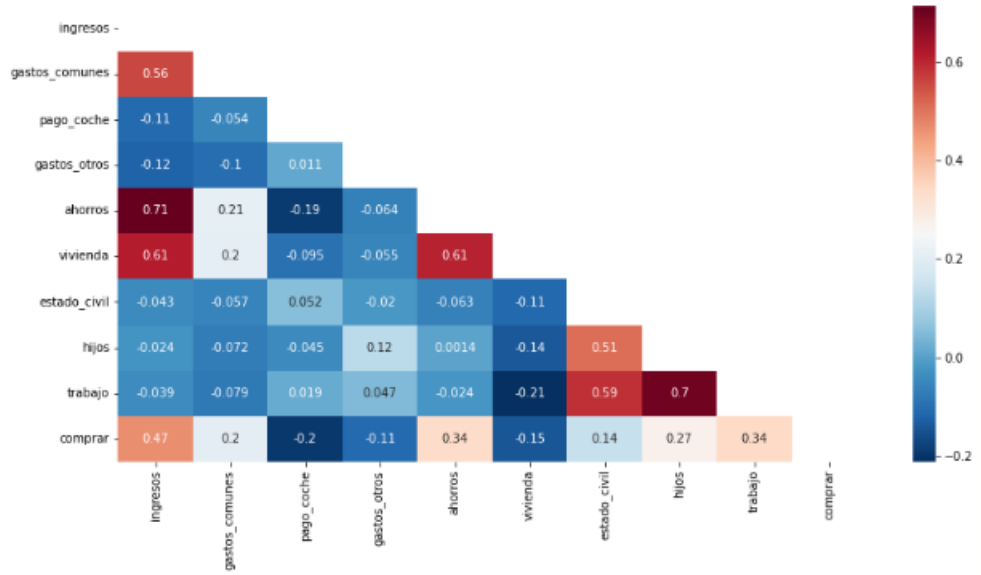


Podemos usar los valores de la correlación y mostrarlo en orden descendente o ascendente ara visualizar si dicha variable cuata con variables fuertes o débiles, pero una de las mejores formas de visualizarlo es un mapa de calor, donde se identifican dos correlaciones fuertes (ahorros e ingresos; trabajo e hijos)

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

```
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
```

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



Así que para reducir la dimensionalidad de nuestra matriz podremos eliminar ahorros e hijos quedando la siguiente tabla de datos.

```
DatosHipotecaACD = Hipoteca.drop(columns=['ahorros', 'hijos'])
DatosHipotecaACD
```

	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

202 rows × 8 columns

Conclusión

Es interesante el ver como existen una variedad de procesos para la reducciones de dimensiones de nuestros datos y evitar tener variables posiblemente correlacionadas fuertemente y así solamente tener un conjunto de variables que podríamos considerar independientes.

Se pudo observar la diferencia existente entre ambos procesos como uno únicamente se basa en un mapa de calor y otro tiene un proceso mas extenso, y no solo es el proceso si no los resultados, con el análisis de componentes principales logramos quitar de nuestros datos un total de cuatro datos, mientras que con el análisis de correlación únicamente eliminamos dos de las columnas.

Liga GitHub Practica 4

Cuaderno de Python de la practica 4

https://github.com/OsvaldoIG/MineriaDatos/tree/main/P4