Cervantes Rubí Brandon
No. cuenta. 316136741
Minería de Datos 2023-1, G.3, 28/09/2022
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.

Los datos se descargaron del portal de Huawei y están asociados a diversas lecturas con estadísticas de sesiones de ejercicio y monitoreo de sueño para identificar alguna relación entre estos dos apartados.

Fuente de datos

- duration: Duracion en minutos del ejercicio
- altitude: Altitud en metros
- distance: Distancia en metros
- calorie: Calorias quemadas
- steps: Numero de pasos
- recordDay: Dia en formato YYYYMMDD
- sportType: Exercise mode.
    - 1:Climbing 2:Mountain climbing
    - 3:Ride 4:Run 5:walk 6:Deep sleep
    - 7:Light sleep 8:wide awake 9:Swim
    - 10:Fitness 11:Children's footprint
- allSleepTime: Tiempo total de sueño en minutos
- awakeTime: Parte del tiempo de sueño que estamos despiertos
- deepSleepPart: Porcentaje de tiempo de sueño profundo
- deepSleepTime: Tiempo de sueño profundo
- dreamTime: Tiempo de sueño
- goBedTime: Tiempo que tarda en recostarse
- lightSleepTime: Tiempo de sueño ligero
- sleepEfficiency: Eficiencia de sueño
- sleepLatency: Latencia en el sueño
- sleepScore: Puntuacion del sueño
- snoreFreq: Frecuencia de ronquidos
- validData: Porcentaje de datos validos
- wakeupTime: Tiempo del dia despierto

Cervantes Rubí Brandon
No. cuenta. 316136741
Minería de Datos 2023-1, G.3, 28/09/2022
Practica N.4 Análisis de Componentes Principales (PCA)

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.

Cervantes Rubí Brandon
No. cuenta. 316136741

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 los cuales son lecturas de un reloj inteligente de Huawei relacionadas con las horas de sueño y actividades físicas del usuario (Caminar y Correr), los datos serán consumidos desde la plataforma de `github` leyendo 2 archivos diferentes para posteriormente hacer una intersección entre ambos teniendo la fecha como discriminante quedándonos solamente con lecturas de los mismos días. La forma en la que se hace es pasando la `url` del repositorio en formato raw al método `read_csv`, después decodificamos la columna donde vienen los datos en formato `json` iterando por todo el `dataframe` agregando las nuevas columnas con información útil.

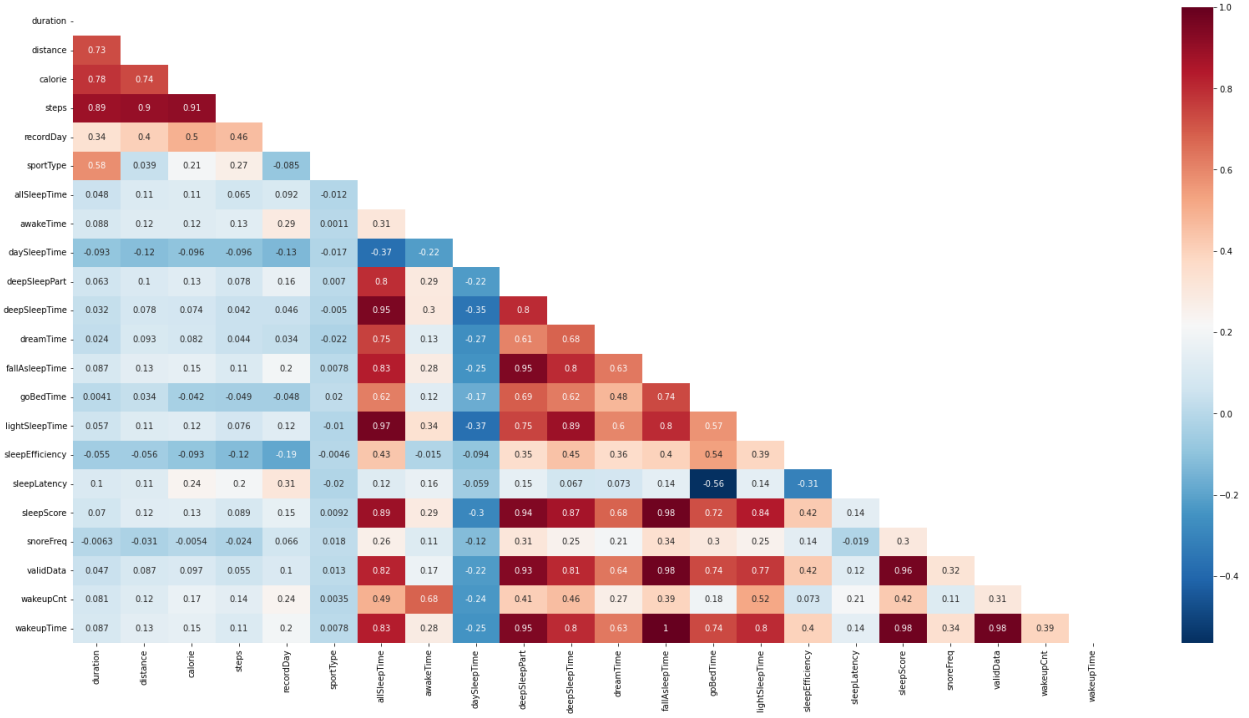
```
sport_url =
'https://raw.githubusercontent.com/Slevkro/DataSets/main/Deportes/MySportsHealth-
Data.csv'

sport_total_data = pd.read_csv(sport_url)
sportBasicInfoData = sport_total_data['sportBasicInfoData']
sportBasicInfoData

sportData = pd.DataFrame()

import json
for record in sportBasicInfoData:
    recordSportData = json.loads(record)
    sportData = sportData.append(recordSportData, ignore_index=True)
    #print(recordSportData)

sportData['recordDay'] = sport_total_data['recordDay']
sportData['sportType'] = sport_total_data['sportType']
sportData
```

Para el segundo `dataframe` se sigue un procedimiento similar, por último, para finalizar con la lectura de datos se utilizó el método `merge` la cual hace la intersección entre dos `dataframes` considerando una columna la cual deberá de llevar el mismo nombre.

```
health_data = pd.merge(sportData,sportSleepData,on='recordDay')
health_data
```

Con esto obtenemos una matriz de 567 registros y 25 columnas, de las cuales 3 de ellas se encuentran con solamente el valor 0 como registro; `altitude`, `count`, `floor` por lo cual se procede a eliminar esos registros de la matriz ya que son datos que no se capturaron del todo bien.

Como primer paso para implementar el algoritmo de análisis de componentes principales (PCA) tenemos que obtener evidencia de correlaciones en el conjunto de datos, para esto es necesario crear una matriz de correlaciones con el método `corr` y con el objetivo de apreciar de una forma más clara podemos trazar un mapa de calor el cual pinta la matriz de colores cálidos (Rojos) y fríos (Azules), mientras más fuerte el color más fuerte será el nivel de correlación.



En el mapa de calor podemos apreciar que existe una correlación fuerte entre las variables de cada `dataframe`, es decir, las variables de sueño se correlacionan algunas entre si y lo mismo pasa con las variables de actividad física durante el día sin embargo las variables de sueño no se correlacionan del todo con las variables de actividad física. Por lo anterior es factible hacer un análisis de componentes principales para reducir la dimensionalidad de estos datos ya sea por separado o junto como se realizará en esta ocasión.

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)
```

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

	duration	distance	calorie	steps	recordDay	sportType	allSleepTime	awakeTime	daySleepTime	deepSleepPart	...
0	-1.081719	-1.200227	-0.986878	-1.090326	-1.147321	-1.408620	0.938610	-0.601425	-1.128142	0.916481	...
1	-1.081719	-1.200227	-0.986878	-1.090326	-1.147321	-1.408620	1.586875	0.789951	-0.037839	0.647142	...
2	0.295073	0.470492	-0.221260	0.236744	-1.147321	0.709915	0.938610	-0.601425	-1.128142	0.916481	...
3	0.295073	0.470492	-0.221260	0.236744	-1.147321	0.709915	1.586875	0.789951	-0.037839	0.647142	...
4	-0.715736	0.325776	0.125468	-0.123558	-1.147111	-1.408620	-0.005356	-0.661920	1.255740	-0.115985	...
...	...	...	...	...	...	...	...	...	...	...	...
562	-0.942297	-1.203221	-0.986383	-1.079820	0.990044	0.709915	0.483687	1.818359	0.091519	1.006261	...
563	-0.942297	-1.203221	-0.986383	-1.079820	0.990044	0.709915	-2.001330	-0.661920	1.921009	-2.270698	...
564	-0.942297	-1.203221	-0.986383	-1.079820	0.990044	0.709915	0.603104	1.697370	0.424153	0.467583	...
565	-0.942297	-1.203221	-0.986383	-1.079820	0.990044	0.709915	-0.579696	-0.238458	2.493879	0.467583	...
566	-0.942297	-1.203221	-0.986383	-1.079820	0.990044	0.709915	0.369956	-0.661920	0.590470	0.377803	...

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 22. Posterior a esto es necesario indicar con el método `fit` que queremos aplicar el algoritmo a esa matriz.

```
pca = PCA(n_components=22)      #Se instancia el objeto
pca.fit(MEstandarizada)         #Se obtiene los componentes
print(pca.components_)
```

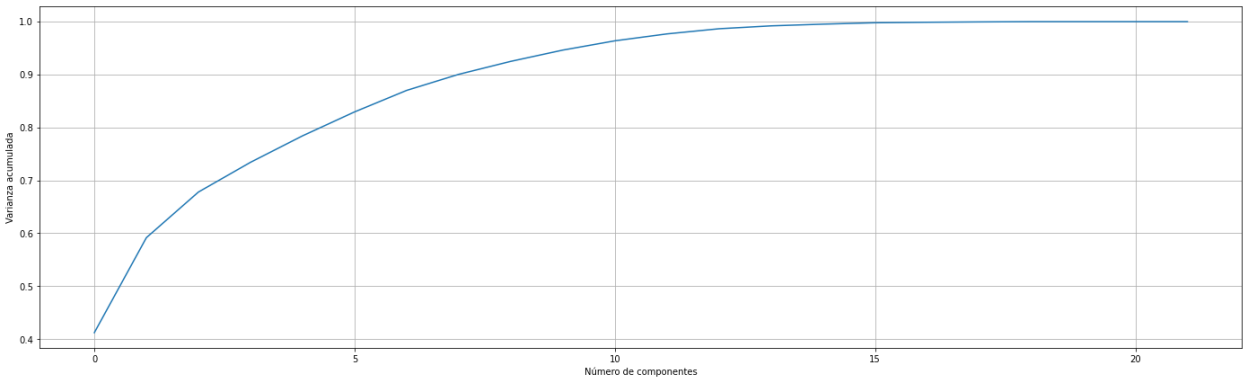
Lo cual nos regresara una lista de listas con la información de los componentes principales de la siguiente manera.

```
[[-4.38101763e-02 -6.10907572e-02 -6.32893564e-02 -5.07752602e-02
  -6.26018422e-02 -7.16499900e-03 -3.10938371e-01 -1.19230215e-01
   1.18677185e-01 -3.06943197e-01 -3.01455918e-01 -2.39136597e-01
  -3.18485115e-01 -2.43254432e-01 -2.96338712e-01 -1.52104221e-01
  -3.27486850e-02 -3.23543088e-01 -1.15490174e-01 -3.10880883e-01
  -1.66754778e-01 -3.18485813e-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[0:8]))
```

En este caso tuvimos que sumar en total 8 componentes principales (Del 0 al 7) lo cual nos regresó un porcentaje del 90.04% 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 conforme sube el número de componentes principales. 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.



Con el numero de componentes principales ya bien definido 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. A continuación, se muestra una parte del valor absoluto de los 8 componentes.

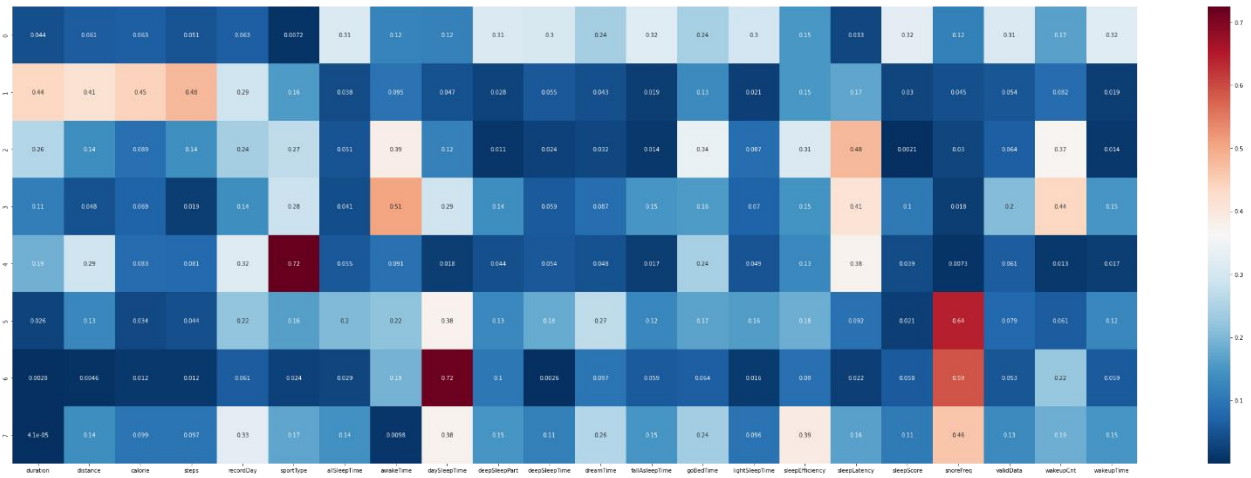
	duration	distance	calorie	steps	recordDay	sportType	allSleepTime	awakeTime	daySleepTime	deepSleepPart	...
0	0.043810	0.061091	0.063289	0.050775	0.062602	0.007165	0.310938	0.119230	0.118677	0.306943	...
1	0.437385	0.413516	0.446195	0.478316	0.288821	0.155818	0.037898	0.095122	0.046545	0.027717	...
2	0.255233	0.136179	0.089412	0.141098	0.239701	0.271211	0.050633	0.386189	0.118625	0.010999	...
3	0.112945	0.047757	0.069122	0.019083	0.136110	0.284612	0.040756	0.505535	0.291421	0.136732	...
4	0.188733	0.289733	0.082974	0.080717	0.315495	0.724335	0.054768	0.091201	0.018309	0.043815	...
5	0.026237	0.126426	0.034446	0.044144	0.219993	0.161735	0.202292	0.215779	0.377726	0.129964	...
6	0.002786	0.004567	0.011590	0.011887	0.060633	0.023678	0.029295	0.191026	0.717507	0.101711	...
7	0.000041	0.136811	0.098726	0.096630	0.325159	0.170271	0.141695	0.009843	0.376196	0.148092	...

La forma en la que se obtuvo el dataframe de la parte superior se muestra a continuación.

```
CargasComponentes = pd.DataFrame(abs(pca.components_[0:8]), columns=health_data.columns)
CargasComponentes
```

Para establecer cuales son las variables que contienen la mayor cantidad de información para el modelo tenemos que seguir el siguiente procedimiento. Primero establecer cuál será el límite en cuanto al porcentaje de información que una variable tiene que cumplir para ser considerada, en este análisis será del 40%. Después en el primer componente, el renglón 0, se busca el índice con el mayor valor de porcentaje y que además este por encima del 0.4 de valor. Como no hay una variable que cumpla con este índice procedemos con segundo componente, para el segundo componente tenemos `duration`, `distance`, `calorie` y `steps` cumpliendo con esta condición por lo cual se consideran 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[:8], cmap='RdBu_r', annot=True)
plt.show()
```



- En el tercer componente solamente cumple la variable `sleepLatency`.
- Para el cuarto componente cumple `awakeTime`, nuevamente `sleepLatency` y `wakeupCtn`.
- Para el quinto componente cumple `sportType`.
- En el sexto componente cumple `snoreFreq`.
- Para el séptimo componente cumple `daySleepTime` y de nueva cuenta `snoreFreq`.
- En el octavo y ultimo componente solamente cumpliría `snoreFreq` y quedaría cerca `sleepEfficiency` por lo cual se podría considerar.

Al final nos quedaremos con 10 variables que demostraron tener la mayor cantidad de información.

1. `Duration`
2. `Distance`
3. `Calorie`
4. `Steps`
5. `sleepLatency`
6. `awakeTime`
7. `wakeupCtn`
8. `sportType`
9. `snoreFreq`
10. `daySleepTime`

Las 12 variables que se descartarían según el análisis serían.

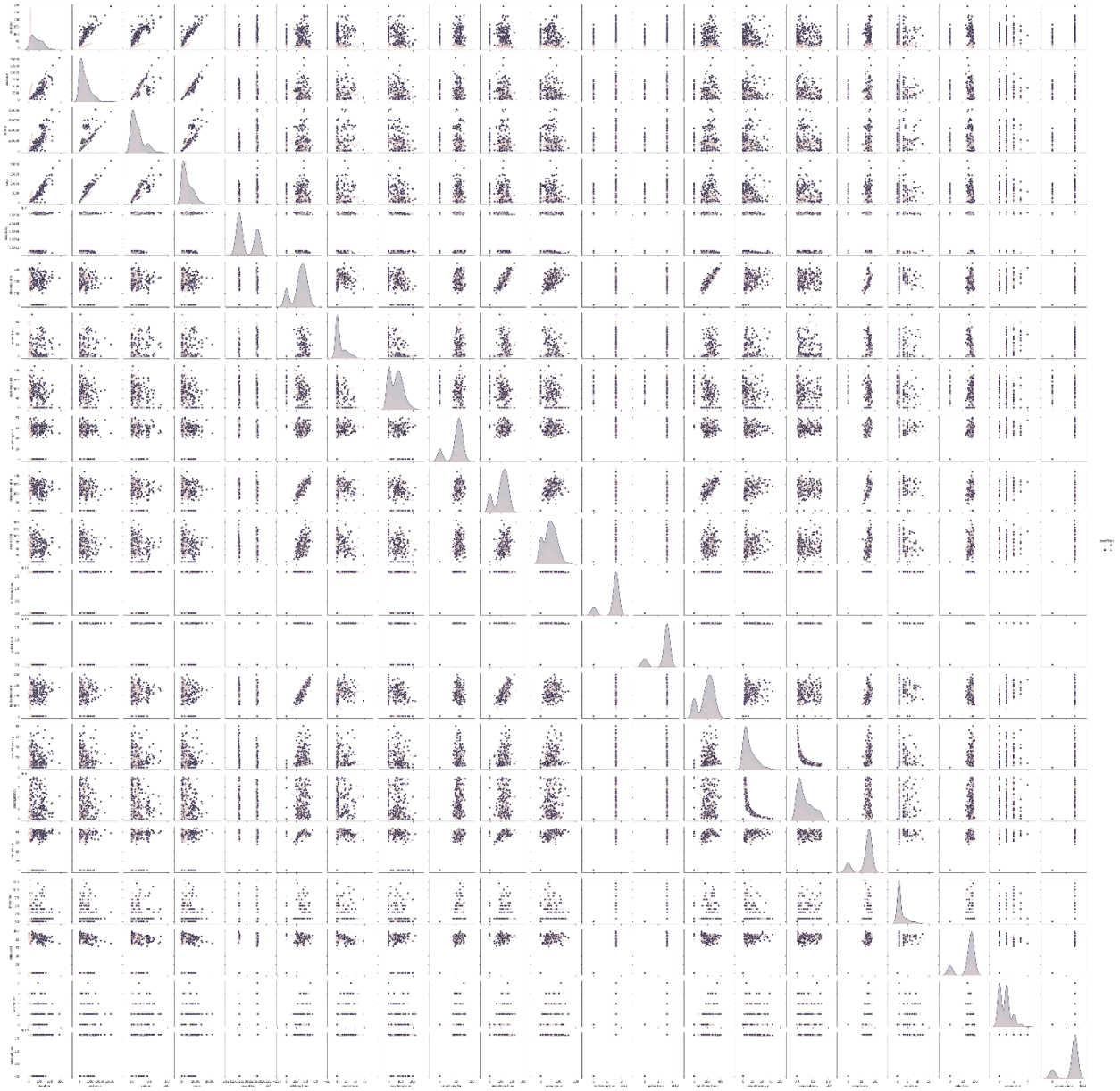
1. `recordDay`
2. `allSleepTime`
3. `deepSleepPart`
4. `deepSleepTime`
5. `dreamTime`
6. `fallAsleepTime`
7. `goBedTime`
8. `lightSleepTime`
9. `sleepScore`
10. `validData`
11. `wakeupTime`
12. `sleepEfficiencyA`

Por lo cual el dataframe con el que comenzaríamos a desarrollar pruebas para el modelo quedaría de la siguiente forma.

	duration	distance	calorie	steps	sportType	awakeTime	daySleepTime	sleepLatency	snoreFreq	wakeupCnt
0	1.0	112.0	6621.0	144.0	4	1.0	0.0	4.011000e+06	1.0	1.0
1	1.0	112.0	6621.0	144.0	4	24.0	59.0	4.285000e+06	1.0	1.0
2	80.0	3460.0	148855.0	4818.0	5	1.0	0.0	4.011000e+06	1.0	1.0
3	80.0	3460.0	148855.0	4818.0	5	24.0	59.0	4.285000e+06	1.0	1.0
4	22.0	3170.0	213269.0	3549.0	4	0.0	129.0	4.267000e+06	5.0	0.0
...	...	...	...	...	...	...	...	...	...	...
562	9.0	106.0	6713.0	181.0	5	41.0	66.0	2.147484e+09	1.0	3.0
563	9.0	106.0	6713.0	181.0	5	0.0	165.0	0.000000e+00	0.0	0.0
564	9.0	106.0	6713.0	181.0	5	39.0	84.0	2.147484e+09	1.0	1.0
565	9.0	106.0	6713.0	181.0	5	7.0	196.0	2.147484e+09	1.0	1.0
566	9.0	106.0	6713.0	181.0	5	0.0	93.0	2.147484e+09	1.0	0.0

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.

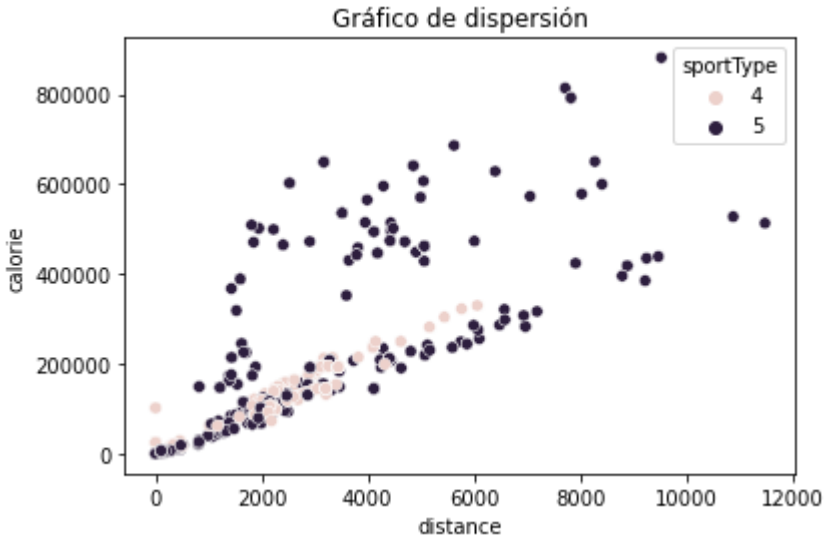
```
sns.pairplot(health_data, hue='comprar')
plt.show()
```



Al ser muchas variables se vuelve complicado leer este tipo de diagramas, pero es factible analizar las relaciones por separado con gráficos de dispersión o pasar al siguiente paso el cual es emplear la matriz de correlaciones para medir el grado de similitud entre las variables.

Se puede apreciar que las primeras variables podrían presentar una relación fuerte por el patrón que se dibuja, lo podemos ver más de cerca con la función `scatterplot` de `seaborn`, graficando `distance` contra `calorie` del `dataframe` por ejemplo.

```
sns.scatterplot(x='distance', y ='calorie', data=health_data, hue='sportType')
plt.title('Gráfico de dispersión')
plt.xlabel('distance')
plt.ylabel('calorie')
plt.show()
```

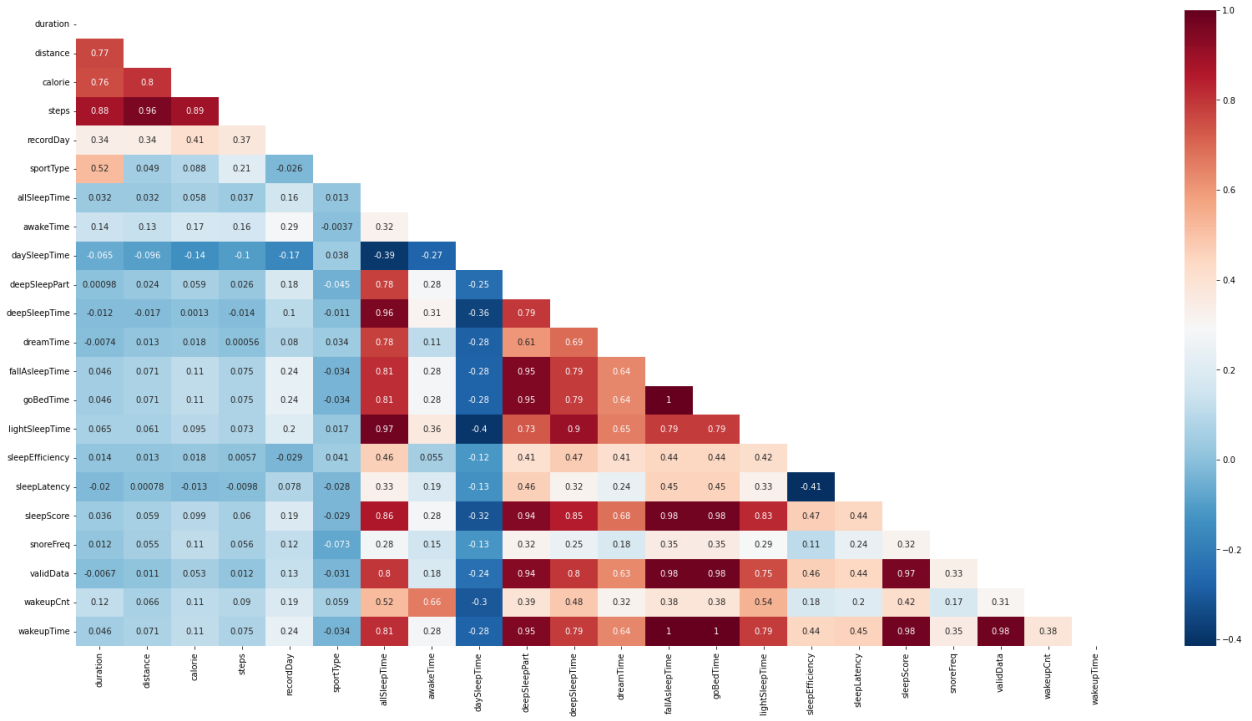


Podemos confirmar el grado de la relación exacto que existe entre las variables creando nuevamente una matriz de correlaciones con el método `corr` obteniendo el siguiente mapa de calor.

```
plt.figure(figsize=(28,14))
MatrizInf = np.triu(corr_health_data)
sns.heatmap(corr_health_data, cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```

En la siguiente página podemos apreciar más a detalle el mapa de calor de las variables en cuestión.

Cervantes Rubí Brandon
No. cuenta. 316136741
Minería de Datos 2023-1, G.3, 28/09/2022
Practica N.4 Análisis de Componentes Principales (PCA)



En el grafico podemos identificar rápidamente el nivel de correlación que existe entre las variables poniendo atención en los tonos más rojos o azules. En ella podemos ver claramente que existe una correlación fuerte (correlación > 0.66) entre las variables distance, calorie, duration y steeps, sportType y duration, allSleepTime y la mayoría de las variables que describen el sueño, daySleepTime y las variables de sueño, deepSleepPart y las variables de sueño, deepSleepPart y las variables de sueño, y el mismo fenómeno lo podemos ver con dreamTime, fallAsleepTime, goBrdTime, lightSleepTime, sleepScore, validData y wakeUpTime. Por lo cual las variables que se quedarían serían:

Se quedan	Se van
1. duration 2. distance 3. calorie 4. steps 5. recordDay 6. sportType 7. awakeTime 8. daySleepTime 9. sleepEficiency 10. sleepLatency 11. snoreFreq 12. wakeupCnt	1. allSleepTime 2. deepSleepPart 3. deepSleepTime 4. dreamTime 5. fallAsleepTime 6. goBedTime 7. lightSleepTime 8. sleepScore 9. validData 10. wakeupTime

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

	duration	distance	calorie	steps	recordDay	sportType	awakeTime	daySleepTime	sleepEfficiency	sleepLatency	snoreFreq
0	1.0	112.0	6621.0	144.0	20210528	4	1.0	0.0	10.0	4.011000e+06	1.0
1	1.0	112.0	6621.0	144.0	20210528	4	24.0	59.0	12.0	4.285000e+06	1.0
2	80.0	3460.0	148855.0	4818.0	20210528	5	1.0	0.0	10.0	4.011000e+06	1.0
3	80.0	3460.0	148855.0	4818.0	20210528	5	24.0	59.0	12.0	4.285000e+06	1.0
4	22.0	3170.0	213269.0	3549.0	20210529	4	0.0	129.0	7.0	4.267000e+06	5.0
...	...	...	...	...	...	...	...	...	...	...	...
562	9.0	106.0	6713.0	181.0	20220726	5	41.0	66.0	0.0	2.147484e+09	1.0
563	9.0	106.0	6713.0	181.0	20220726	5	0.0	165.0	0.0	0.000000e+00	0.0
564	9.0	106.0	6713.0	181.0	20220726	5	39.0	84.0	0.0	2.147484e+09	1.0
565	9.0	106.0	6713.0	181.0	20220726	5	7.0	196.0	0.0	2.147484e+09	1.0
566	9.0	106.0	6713.0	181.0	20220726	5	0.0	93.0	0.0	2.147484e+09	1.0

Conclusiones

A lo largo de la practica se revisaron dos técnicas para reducir la dimensionalidad de los datos, la primera es una análisis correlacional el cual se basa en crear una matriz de correlaciones que mida la relación que existe entre los pares de variables poniendo especial atención entre las que tienen un índice de >0.66 o <-0.66 ya que se consideran relaciones fuertes lo cual indicaría que las dos variables describen la misma información de alguna manera por lo cual nos podríamos deshacer de una de ellas según el contexto.

Además, se revisó el análisis de componentes principales el cual se basa en crear una serie de vectores con el nivel de información de cada una de las variables partiendo igualmente de una matriz de correlaciones y estableciendo un cierto porcentaje que todas las variables tienen que cumplir para ser consideradas dentro del modelo.

Con ambas técnicas se logró reducir la dimensionalidad de los datos cerca de un 50%, es decir nos quedamos con la mitad de las variables con las que comenzamos, en algunos casos coincidían, pero en otros las variables consideradas eran diferentes. Es por esta razón que es importante explorar varias posibilidades para llegar al mejor modelo.