Universidad del Desarrollo
Universidad de Excelencia

Informe Tarea 1

Análisis de Componentes Principales

Bruno Fonseca, Valentín Grau, Valeria Rodríguez

Facultad de Ingeniería, Universidad del Desarrollo

IELE750A: Machine Learning

Profesor: Tomás Fontecilla

8 de septiembre de 2023

Índice

Introducción

El análisis de componentes principales (PCA) es una técnica que se utiliza en estadística multivariada y análisis de datos, que tiene como objetivo reducir la dimensionalidad de un conjunto de datos, manteniendo la mayor parte de su información original.

El presente informe tiene como finalidad evidenciar el procedimiento realizado en el programa Python para analizar las bases de datos “load_breast_cancer” y “pokemnon.csv” de forma detallada, mediante el uso de PCA (Principal Component Analysis), y comprender los pasos que hay detrás de esta función.

La primera base de datos “load_breast_cancer”, contiene variables relacionadas con el cáncer mamario y su objetivo es predecir si un tumor es benigno o maligno a partir de la correlación entre sus variables, por lo que se considera relevante para la comunidad médica. La segunda base de datos “pokemnon.csv”, contiene variables relacionadas con las características de personajes del juego. Ambos conjuntos de datos se analizarán mediante PCA, para poder reducir su dimensionalidad manteniendo la mayor cantidad de información.

Luego de realizar el método PCA para las bases de datos, se procederá a analizar las gráficas desarrolladas para ambas bases de datos y explicar mediante elementos matemáticos y visuales, los resultados y las conclusiones obtenidas.

¿Qué es PCA?

Como se mencionó anteriormente, el análisis de componentes principales (PCA) es una técnica de estadística hace referencia a un método que permite simplificar la complejidad de espacios muestrales con muchas dimensiones, pero conservando la información. Esta información se conserva maximizando la varianza retenida de los datos originales, en los componentes principales.

El primer componente principal (PC1) captura la dirección en la que los datos tienen la mayor varianza, el segundo componente principal (PC2) captura la segunda mayor varianza, y así sucesivamente. Por lo tanto, al seleccionar un número de componentes principales, se controla cuánta varianza de los datos originales se está explicando en la proyección reducida.

Los pasos para poder ejecutar correctamente este análisis son los siguientes:

1. Carga y limpieza de datos: Se cargan los datos al programa donde se hará el análisis y se corrigen valores que pueden aparecer como “NaN”, “Nulos” o incorrectos, en caso de que corresponda.
2. Estandarizar los datos: De esta forma, se facilitará la manera de observar los datos, ya que se estará eliminando el factor de que las variables estén en distintas escalas.
3. Calcular la matriz covarianza: Nos mostrará cómo las variables se relacionan entre sí y nos ayudará a calcular luego los valores y vectores propios.
4. Calcular valores y vectores propios: Los vectores propios representan la dirección en que se mueve la variabilidad de los datos y el valor propio representa el valor de la varianza sobre ese vector. Los valores propios se ordenan de forma descendiente para observar de mejor forma qué valores explican la mayor parte de la varianza de los datos.
5. Seleccionar componentes principales: Se calcula la varianza explicada acumulada de los componentes y se escoge la cantidad de componentes que permita un valor de la varianza que se adecúe a las necesidades.
6. Proyectar los datos en los componentes principales: Mostrar los datos en un espacio de dimensionalidad reducida, si es posible.

El problema principal de este método es su validación, ya que no se tiene una variable respuesta con la cual contrastar los resultados de la metodología. Es decir, es un método no supervisado ya que no toma como referencia resultados conocidos para funcionar, sino que se centra en explorar y resumir la estructura de los datos en función de las relaciones entre las variables originales.

Parte 1: Base de datos “load_breast_cancer”

En esta sección se realizó un análisis de los componentes principales de la base de datos load_breast_cancer obtenida de la librería de scikit-learn.

Este conjunto de datos proviene del Hospital de la Universidad de Wisconsin y se basa en características extraídas de imágenes de biopsias, lo que corresponde a información sobre diagnósticos de cáncer de mama.

I. Carga y limpieza de datos

a. **Carga:** Se cargaron los datos mediante la librería “sklearn” y se importaron las librerías necesarias para el análisis.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_breast_cancer
```

```
#Cargamos datos
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	wor smoothne
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0	0.162
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0	0.123
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0	0.144
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7	0.209
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0	0.137
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0	0.141
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0	0.116
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0	0.113
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0	0.165
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6	0.089

569 rows × 30 columns

Las librerías pertinentes para el análisis PCA, son las siguientes:

Pandas (pd): Se utiliza para el análisis y manipulación de datos. En este caso, para poder tabular la base de datos, y así, tener una mejor visualización de ellos. Como se puede apreciar, se tiene 569 datos para 30 columnas (las variables del dataset).

Numpy (np): Se utiliza para operaciones numéricas y soporte para arreglos multidimensionales y matrices. Esto nos ayudará más adelante para obtener la varianza de los datos, los vectores y valores propios.

Matplotlib.pyplt (plt): Se utiliza para la visualización de datos en forma de gráficos, que nos ayudará más adelante para obtener conclusiones sobre los componentes principales.

Scikit-learn (sklearn): Es una biblioteca de aprendizaje automático para el lenguaje de programación de Pyhton. Proporciona herramientas para realizar algoritmos como regresión, clustering, entre otros. Nos ayudará a cargar el primer dataset, ya que corresponde a esta librería. También a estandarizar los datos cargados y verificar que el análisis de componentes principales realizado paso a paso haya sido correcto, importando PCA.

Seaborn (sns): Se utiliza para la visualización de datos, atractivas e informativas. Su propósito en este análisis será generar un mapa de calor para la visualización de los datos de la covarianza.

b. **Limpieza**

Luego de cargar la base de datos “load_breast_cancer”, se procede a verificar la existencia de datos nulos o erróneos:

```
nulos = df.isna().sum()
nulos
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
dtype: int64
```

Como no hay existencia de datos nulos, no es necesario limpiar la base de datos por lo que se podrá continuar con los siguientes pasos

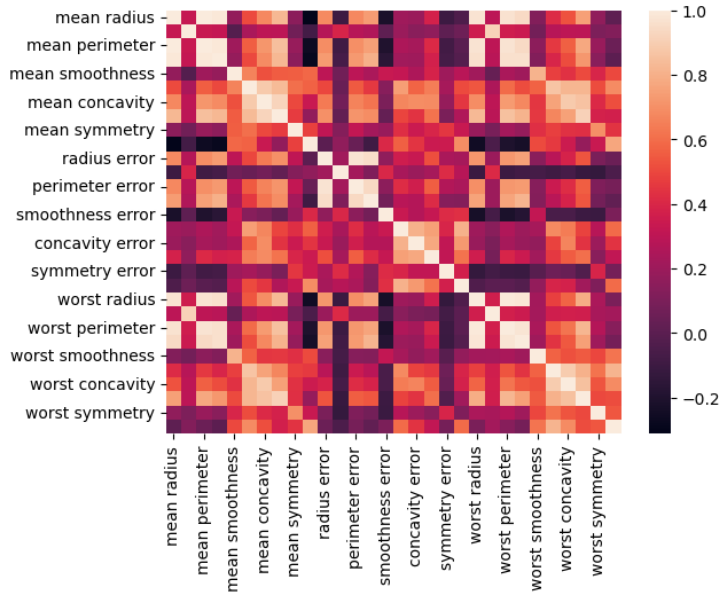
II. Estandarización de datos

Para eliminar las distintas escalas que puedan tener los datos.

```
df_std = StandardScaler().fit_transform(df)
df_std= pd.DataFrame(df_std, columns=df.columns)
```

III. Matriz de covarianza

Se presentará de forma de heatmap, para analizar visualmente cómo se correlacionan las variables.



Se puede concluir que son más variables que se correlacionan negativamente (por ejemplo, smoothness error con mean radius) que las variables que se correlacionan positivamente (por ejemplo, mean radius con mean perimeter).

IV. Cálculo de vectores y valores propios

Serán desplegados en forma de pares ordenados de manera descendiente para poder asociar de mejor manera cada valor propio con su vector propio respectivo y además, visualizar de mejor forma qué valor propio representa la mayor cantidad de varianza explicada de los datos originales.

```
#Calculamos valores y vectores propios de la matriz de correlacion
values, vectors = np.linalg.eig(corr)
```

```
#Juntamos en pares y ordenamos de manera descendente segun valor propio
pares = [(np.abs(values[i]), vectors[:, i]) for i in range(len(values))]
pares.sort(key=lambda x: x[0], reverse=True)
pares
```

```
[(13.281607682257917,
  array([0.21890244, 0.10372458, 0.22753729, 0.22099499, 0.14258969,
         0.23928535, 0.25840048, 0.26085376, 0.13816696, 0.06436335,
         0.20597878, 0.01742803, 0.21132592, 0.20286964, 0.01453145,
         0.17039345, 0.15358979, 0.1834174 , 0.04249842, 0.10256832,
         0.22799663, 0.10446933, 0.23663968, 0.22487053, 0.12795256,
         0.21009588, 0.22876753, 0.25088597, 0.12290456, 0.13178394])),
 (5.691354613209925,
  array([-0.23385713, -0.05970609, -0.21518136, -0.23107671,  0.18611302,
          0.15189161,  0.06016536, -0.0347675 ,  0.19034877,  0.36657547,
         -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
          0.2327159 ,  0.19720728,  0.13032156,  0.183848  ,  0.28009203,
         -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
          0.14359317,  0.09796411, -0.00825724,  0.14188335,  0.27533947])),
 (2.8179489772294146,
  array([-0.00853124,  0.0645499 , -0.00931422,  0.02869953, -0.1042919 ,
         -0.07409157,  0.00273384, -0.02556354, -0.04023994, -0.02257409,
          0.26848139,  0.37463367,  0.26664537,  0.21600653,  0.30883898,
          0.15477972,  0.17646374,  0.22465757,  0.28858429,  0.21150376,
         -0.04750699, -0.04229782, -0.04854651, -0.01190232, -0.25979761,
         -0.23607563, -0.17305734, -0.17034408, -0.27131264, -0.23279131])),
```

Cabe destacar que cada uno de los 30 valores propios, contiene 30 vectores propios que definen su dirección en un espacio de 30 dimensiones, que corresponde a la cantidad de variables.

V. Selección de componentes principales

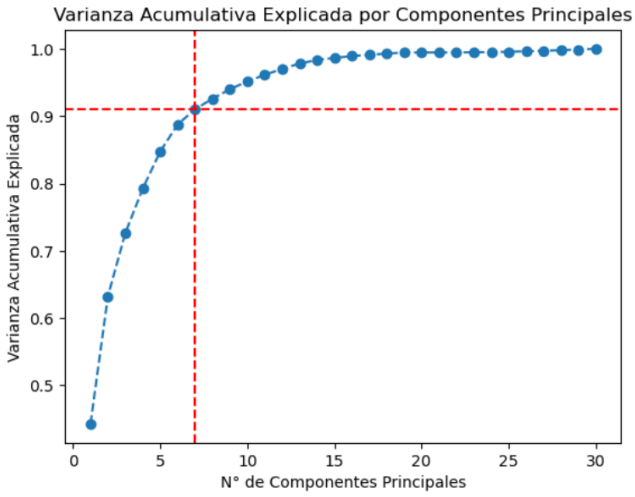
En primer lugar, se calcula la varianza acumulada para visualizar la cantidad de varianza total que explica cada componente.

```
#Calculamos la varianza exlicada por cada componente, y la acumulada.
var_total = sum(values)
var_expl = [value / var_total for value in values]
var_cum = np.cumsum(var_expl)
var_cum
```

```
array([0.44272026, 0.63243208, 0.72636371, 0.79238506, 0.84734274,
       0.88758796, 0.9100953 , 0.92598254, 0.93987903, 0.95156881,
       0.961366  , 0.97007138, 0.97811663, 0.98335029, 0.98648812,
       0.98915022, 0.99113018, 0.99288414, 0.9945334 , 0.99453783,
       0.99456279, 0.99461577, 0.99484578, 0.99511837, 0.99563442,
       0.99623625, 0.99704761, 0.99796226, 0.9990009 , 1.        ])
```

Finalmente, se graficaron los resultados de la varianza acumulada y se trazaron líneas horizontales y verticales para señalar el punto de intersección entre la cantidad de componentes que se escogió, con la cantidad de información (%) que explica respecto a los datos originales:

```
#Graficamos la varianza explicada segun componentes principales
plt.plot(range(1, len(var_cum) + 1), var_cum, marker='o', linestyle='--')
plt.xlabel('N° de Componentes Principales')
plt.ylabel('Varianza Acumulativa Explicada')
plt.title('Varianza Acumulativa Explicada por Componentes Principales')
posicion_vertical = 7
plt.axvline(x=posicion_vertical, color='red', linestyle='--', label='Línea Vertical')
posicion_horizontal = 0.9100953
plt.axhline(y=posicion_horizontal, color='red', linestyle='--', label='Línea Horizontal')
plt.show()
```



Analizando el gráfico, como grupo se pudo decidir con escoger siete componentes principales, debido a que si bien, sigue creciendo la cantidad de varianza explicada a medida que aumentan los componentes principales, este crecimiento empieza a ser muy pequeño a partir de este punto. Además, con siete componentes ya se explica aproximadamente el 91% de la varianza, lo que como grupo se decidió que es un alto porcentaje, por lo tanto, suficiente para la explicación de los datos originales.

Parte 2: Base de datos "pokemon.csv"

En esta sección se realizó un análisis de los componentes principales de la base de pokemon.csv entregada por el profesor del curso, Tomás Fontecilla, con la finalidad de reducir la dimensionalidad de los datos de manera que este nuevo conjunto de variables capture la mayor cantidad de información de los datos originales.

I. Carga y limpieza de datos

a. ***Carga***: Se importaron las mismas librerías que para el análisis de "load_breast_cancer" y cargamos el conjunto de datos en un data frame con pandas.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
```
✓ 0.0s

```
df = pd.read_csv('C:/Users/fonse/Desktop/ICI 2023/MACHINE LEARNING/CSV/pokemon.csv')
df.head()
```
✓ 0.0s

b. ***Limpieza:*** Se comenzó revisando los valores nulos del data frame, para poder decidir qué variables serán trabajadas y otras descartadas. También se revisaron los formatos para ver incongruencias.

```
print(df.dtypes)
```
✓ 0.0s

```
nulos = df.isna().sum()

print(nulos)
```
✓ 0.0s

Se observa que la columna "weight_kg" y "height_m" tienen 20 N/A (que hace referencia a datos nulos), por lo que se reemplazó los datos faltantes con el promedio de cada variable, de tal forma que no se eliminaran tantos datos. Se sabe que esta forma puede sesgar un poco el resultado al asumir que los datos faltantes distribuyen de igual forma que los datos que se encuentran en la variable, pero se consideró que era la mejor alternativa.

```
promedio = df['height_m'].mean()
df['height_m'].fillna(promedio, inplace=True)
```
✓ 0.0s

```
promediokg = df['weight_kg'].mean()
df['weight_kg'].fillna(promediokg, inplace=True)
```
✓ 0.0s

Siguiendo con la limpieza hay 98 datos faltantes en la variable “percentage_male”. Este valor muestra la proporción de masculinos de esa especie de Pokemon que se puede encontrar en el juego. Asumimos que estos valores faltantes son Pokemon sin género por lo que utilizamos un valor de 0.5 que representa la misma proporción de masculinos y femeninos.

```
df['percentage_male'].fillna(0.5, inplace=True)
male = df['percentage_male'].values
print(male)
```
✓ 0.0s

Revisando la variable “capture_rate” se encontró que todos los datos eran numéricos exepto “30 (Meteorite)255 (Core)”. Esta variable representa la tasa de captura, pero en este pokemon en particular pareciera tener diferente tasa según la condición en la que se encuentra, por lo que se optó mantener el dato “30” asumiendo que es la principal, extrayendo el texto sobrante.

```
df['capture_rate'] = df['capture_rate'].str.extract(r'(\d+)')
valores_capture_rate = df['capture_rate'].values
print(valores_capture_rate)
```
✓ 0.0s

```
df['capture_rate'] = df['capture_rate'].astype(float)
print(df.dtypes)
```
✓ 0.0s

Terminando con la limpieza de los datos, se eliminarán las variables: “abilities”, “class”, “japanesse_name”, “name”, “pokedex_number”, “type1”, “type2”, “generation” y “is_legendary”, por ser variables categóricas (tipo object), las cuales no podrían ser parte del análisis. Las variables “pokedex_number”, “name”, “japan_name” son variables propias de cada pokemon por lo que no me aportan información significante.

```
eliminar = ['abilities', 'classfication', 'japanese_name', 'name', 'pokedex_number', 'type1', 'type2', 'generation', 'is_legendary']
df.drop(columns=eliminar, inplace=True)
```
✓ 0.0s

II. Estandarización de los datos

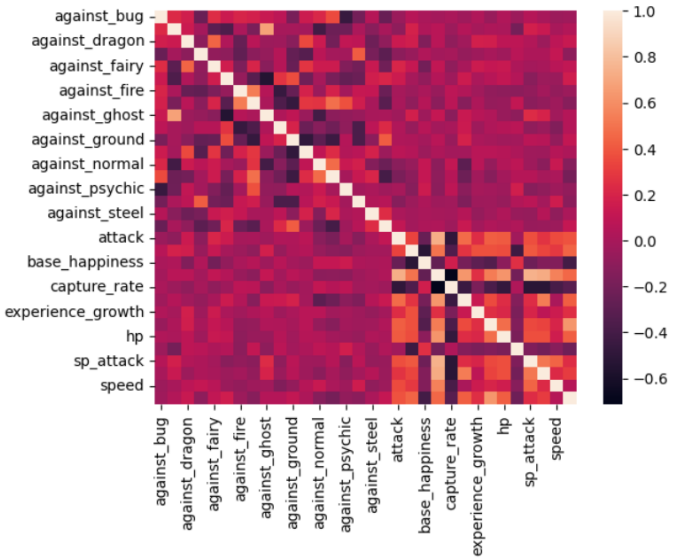
Como trabajamos con PCA la cual es un algoritmo que trabaja con la varianza y la covarianza de los datos, es muy importante estandarizar nuestros datos para que todas las variables tengan la misma importancia relativa en el análisis, asegurándonos que las variables tienen todas media cero y desviación estándar uno.

```
#Es necesario normalizar los datos, ya que pca trabaja mejor con datos centrados y escalados.
df_std = StandardScaler().fit_transform(df)
df_std= pd.DataFrame(df_std, columns=df.columns)
```
✓ 0.0s

III. Matriz de covarianzas

Calculamos la matriz de correlación la que nos muestra la correlación entre pares de variables, de esta matriz calcularemos los valores y vectores propios.

```
corr = df_std.corr()
corr
✓ 0.0s
```



Se puede concluir que la mayoría de las variables tienen una correlación cercana al cero, lo que quiere decir que no covarían ni positiva ni negativamente. Sí hay excepciones de variables que están altamente correlacionadas positivamente (como por ejemplo, “against_dark” con “against ghost”) y negativamente (como por ejemplo “Capture_rate” con “sp_attack”.

IV. Calcular valores y vectores propios

Luego de tener la matriz de correlación, podemos calcular los valores propios que indican la cantidad de variabilidad representada por cada componente, y los vectores propios los cuales son la dirección en la que se encuentra la máxima variabilidad de los datos.

Se creó una lista de pares, donde cada par es un valor propio y su correspondiente vector propio asociado. Se ordenó de manera descendente en función de los valores propios. Dejando el componente que explica la mayor varianza en la parte superior de la lista.

```
#Calculamos valores y vectores propios de la matriz de correlacion
values, vectors = np.linalg.eig(corr)
```

✓ 0.0s

```
#Juntamos en pares y ordenamos de manera descendente segun valor propio
pares = [(np.abs(values[i]), vectors[:, i]) for i in range(len(values))]
pares.sort(key=lambda x: x[0], reverse=True)
pares
```

✓ 0.0s

V. Seleccionamos componentes principales

Con el fin de seleccionar nuestros componentes que maximicen la varianza acumulada, ahora calculamos la varianza total de los datos, es la suma de todos los valores propios, que nos servirá para calcular la proporción de varianza explicada por cada componente principal.

Proseguimos con el cálculo de la varianza explicada por cada componente principal, dividiendo cada valor propio entre la varianza total. También se calculó la varianza acumulada explicada por los componentes principales.

Con estos cálculos luego realizamos un gráfico de línea que represente la varianza acumulativa explicada por componentes principales. En el eje de la abscisa se encuentra el número de componentes principales y en el eje de las ordenadas, la varianza acumulativa explicada.

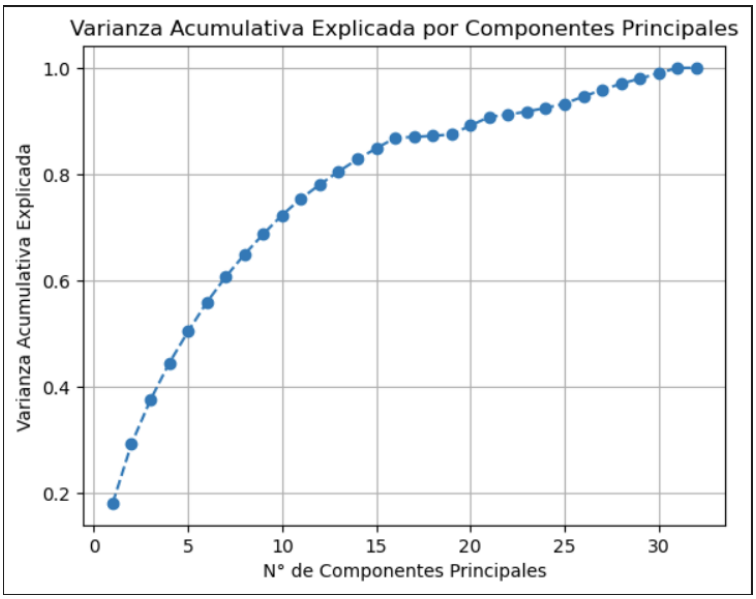
```
#Calculamos la varianza exlicada por cada componente, y la acumulada.
var_total = sum(values)
var_expl = [value / var_total for value in values]
var_cum = np.cumsum(var_expl)
var_cum
```

✓ 0.0s

Se grafica:

```
#Graficamos la varianza explicada segun componentes principales
plt.plot(range(1, len(var_cum) + 1), var_cum, marker='o', linestyle='--')
plt.grid()
plt.xlabel('N° de Componentes Principales')
plt.ylabel('Varianza Acumulativa Explicada')
plt.title('Varianza Acumulativa Explicada por Componentes Principales')
plt.show()
```

✓ 0.1s

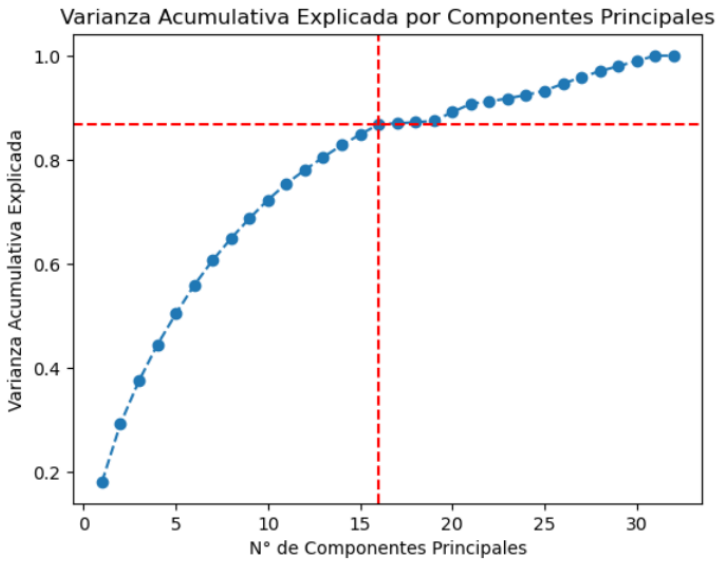



```
var_cum[15]

✓ 0.0s

0.8682865871512668
```

```python
#Graficamos la varianza explicada segun componentes principales
plt.plot(range(1, len(var_cum) + 1), var_cum, marker='o', linestyle='--')
plt.grid()
posicion_vertical = 16
plt.axvline(x=posicion_vertical, color='red', linestyle='--', label='Línea Vertical')
posicion_horizontal = 0.8682865871512668
plt.axhline(y=posicion_horizontal, color='red', linestyle='--', label='Línea Horizontal')
plt.xlabel('N° de Componentes Principales')
plt.ylabel('Varianza Acumulativa Explicada')
plt.title('Varianza Acumulativa Explicada por Componentes Principales')
plt.show()
```



```
primeros_datos = var_expl[:16]
varianza_acumulada = np.cumsum(primeros_datos)
varianza_acumulada
```

```
array([0.18167783, 0.29335081, 0.37538527, 0.44493105, 0.50428547,
       0.55929169, 0.60714296, 0.64926293, 0.68772306, 0.7233078 ,
       0.75426112, 0.78057397, 0.80516564, 0.82816201, 0.84871432,
       0.86828659])
```

Analizando el gráfico de varianza acumulada explicada, se optó por seleccionar los primeros 16 componentes principales se logra explicar aproximadamente un 86.82% de la varianza total de los datos, lo que se considera adecuado para un análisis posterior, ya que se logra reducir significativamente la dimensionalidad de los datos y al mismo tiempo, se logra mantener una representación efectiva de la estructura de los datos.

Conclusiones

Se puede concluir que el método de análisis PCA es una herramienta efectiva, que ayuda a la reducción de dimensiones para distintas bases de datos que poseen múltiples variables, de forma que se pueda reducir su complejidad y poder analizar los datos más rápido y manteniendo la mayor cantidad de información.

Para las dos bases de datos trabajadas, se pudo verificar que este método consta de cinco pasos principales y fundamentales que ayudaron a la comprensión matemática que hay detrás de este análisis estadístico y junto con esto, poder reducir las dimensiones de los datos para un posterior análisis.

Para personas que deseen obtener un análisis más profundo o concluir de manera más precisa sobre estas o cualquier base de datos, el análisis PCA ayuda significativamente al proceso, ya que, al reducir su complejidad, queda dispuesto a ser manipulado según el análisis posterior que se desee realizar, ya sea regresión lineal, clustering, entre otros.