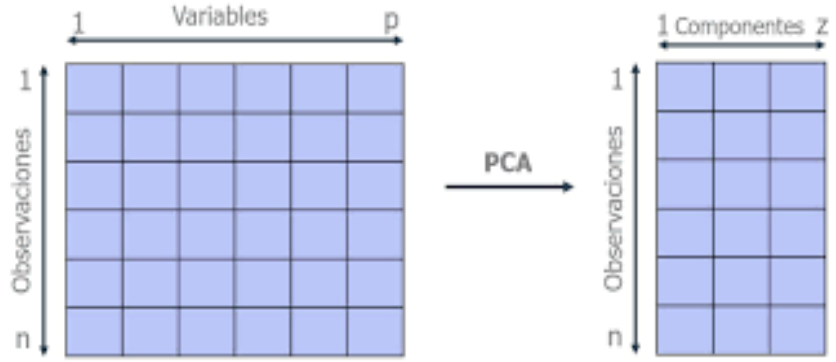
Introducción

El Análisis de Componentes Principales (ACP) – o en su versión del inglés, Principal Component Analysis (PCA) – es un método estadístico cuya utilidad radica en la reducción de la dimensionalidad de la base de datos (BDD) con la que se esté trabajando. El análisis de componentes principales (ACP) es una técnica estadística multivariante de simplificación, que permite transformar un conjunto de variables originales correlacionadas entre sí, en un conjunto sintético de variables no correlacionados denominados factores o componentes principales. En esta transformación no se establecen jerarquías entre variables y se elimina la información repetida. Es un método de proyección, ya que proyecta las observaciones de un espacio p-dimensional con p variables a un espacio k-dimensional (donde k < p) para conservar la máxima cantidad de información (la información se mide aquí a través de la varianza total del conjunto de datos) de las dimensiones iniciales. Las nuevas variables son combinaciones linealmente independientes de las variables originales, ordenadas de acuerdo con la representación de dispersión respecto a la nube total de información recogida en las muestras. Las dimensiones del ACP también se denominan ejes o factores. Si la información asociada a los 2 o 3 primeros ejes representa un porcentaje suficiente de la variabilidad total del diagrama de dispersión, las observaciones podrían representarse en un gráfico de 2 o 3 dimensiones, facilitando así su interpretación.



En esta práctica se hará uso del banco de datos de la flor iris que está formado por 3 clases o especies diferentes, las cuales tienen 4 atributos en total, por lo tanto, no se pueden graficar de esa forma, entonces se aplicará el método de análisis de componentes principales con el fin de identificar sus patrones principales y reducir su dimensión de 4 a 2 para que de esa forma se puedan visualizar en una gráfica.

Código

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
import pandas as pd

iris = datasets.load_iris() #Se importa el dataset de la flor iris
mediante sklearn
fig,axes=plt.subplots(figsize=(15, 8)) #Se define el gráfico a emplear

"""Se convierte el dataset en un dataframe para poder manipular mejor los
atributos de las clases"""
df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                  columns= iris['feature_names'] + ['Species'])

"""Mediante el siguiente ciclo se realiza el proceso de estandarizacion
de los valores con el fin de que se obtenga una distribución gaussiana en
donde la media de los valores sea igual a 0 y su varianza igual a 1 (este
procedimiento es opcional)"""

"""for name in df.columns:
    df[name] = (df[name]-df[name].mean())/df[name].std()"""


"""Se obtienen los valores de los 4 atributos del banco de datos y se
guardan en un arreglo para cada atributo"""
a = df.iloc[:,0]
b = df.iloc[:,1]
c = df.iloc[:,2]
d = df.iloc[:,3]

data = np.array([a, b, c, d]) #Se crea un arreglo que contenga los datos
de todos los atributos

cov=np.cov(data) #Se obtiene la matriz de covarianza de los 4 atributos
print('Matriz de covarianza: \n', cov)

eig_vals, eig_vecs =np.linalg.eig(cov) #Con la matriz de covarianza se
obtienen los valores y vectores propios en forma de matriz

print('Valores propios: \n', eig_vals)
print('Vectores propios: \n', eig_vecs)

v_pc1 = eig_vecs[:,0] #Se obtienen los valores de la primera columna de
la matriz de vectores propios
v_pc2 = eig_vecs[:,1] #Se obtienen los valores de la segunda columna de
la matriz de vectores propios
print(v_pc1)
```

```
print(v_pc2)

"""Se crean las listas para guardar los valores de las 2 componentes"""
pc1 = [] #Lista para la componente 1
pc2 = [] #Lista para la componente 2

"""Se separan los datos de cada clase para identificarlos en la gráfica
con diferente color"""
species = df["Species"].tolist()
species_unique = list(set(species))
species_colores = ["blue","orange","green"]

"""Mediante el siguiente ciclo se multiplican los valores de cada
atributo por los valores de las 2 primeras columnas de vectores propios,
generando así los valores para la componente 1 y 2"""
for i in range(150):
  pc1.append(v_pc1[0]*a[i]+v_pc1[1]*b[i]+v_pc1[2]*c[i]+v_pc1[3]*d[i])
  pc2.append(v_pc2[0]*a[i]+v_pc2[1]*b[i]+v_pc2[2]*c[i]+v_pc2[3]*d[i])

"""Finalmente se grafican los puntos con las 2 componentes generadas"""
for i, spec in enumerate(species):
        plt.scatter(pc1[i], pc2[i], s = 15,
c=species_colores[species_unique.index(spec)])


"""Se agregan los nombres de cada clase a la gráfica"""
for nombre in ['Setosa', 'Versicolor', 'Virginica']:
     plt.scatter(pc1[i], pc2[i], s = 15, label = nombre)

plt.scatter(pc1[i], pc2[i], s = 15, c = 'orange')

plt.xlabel('CP1')
plt.ylabel('CP2')

plt.legend(loc='best')
plt.show()
```

Desarrollo

A continuación, se presentan los resultados obtenidos al aplicar el análisis de componentes principales con los valores no estandarizados y estandarizados.

Valores no estandarizados

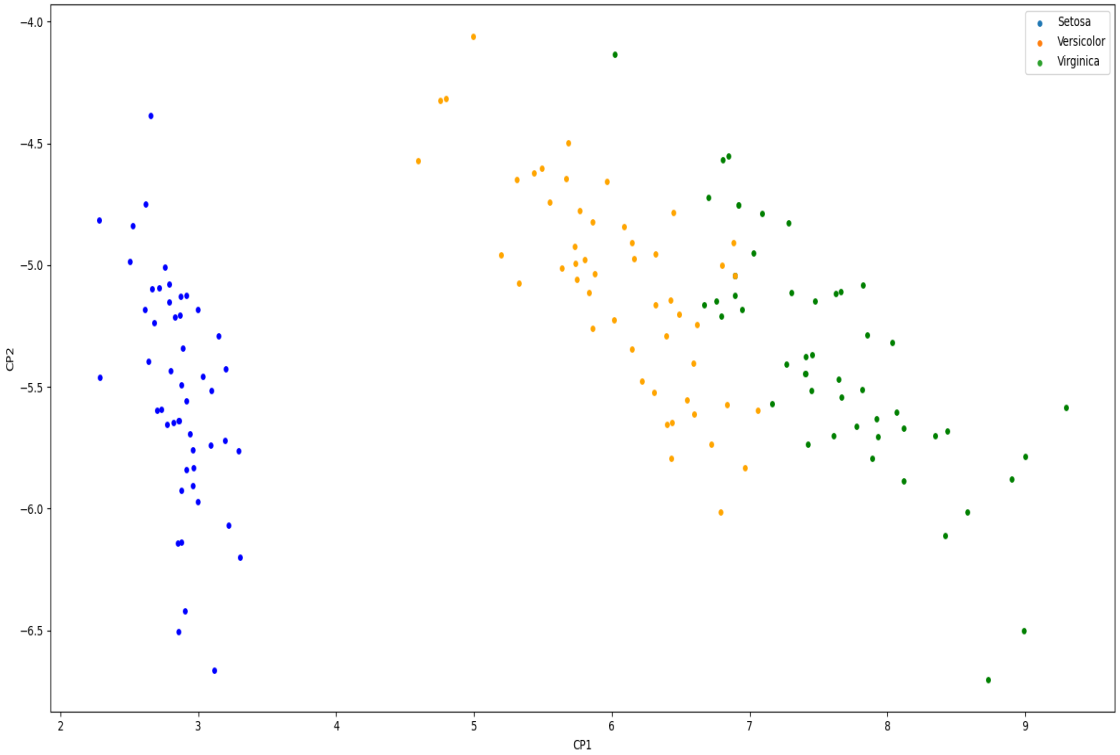


Figura 1. Gráfica de valores no estandarizados.

```
Matriz de covarianza:
 [[ 0.68112222 -0.04215111  1.26582     0.51282889]
 [-0.04215111  0.18871289 -0.32745867 -0.12082844]
 [ 1.26582    -0.32745867  3.09550267  1.286972  ]
 [ 0.51282889 -0.12082844  1.286972    0.57713289]]
Valores propios:
 [4.20005343 0.24105294 0.0776881  0.02367619]
Vectores propios:
 [[ 0.36138659 -0.65658877 -0.58202985  0.31548719]
 [-0.08452251 -0.73016143  0.59791083 -0.3197231 ]
 [ 0.85667061  0.17337266  0.07623608 -0.47983899]
 [ 0.3582892   0.07548102  0.54583143  0.75365743]]
[ 0.36138659 -0.08452251  0.85667061  0.3582892 ]
[-0.65658877 -0.73016143  0.17337266  0.07548102]
```

Figura 2. Resultados de valores no estandarizados.

Valores estandarizados

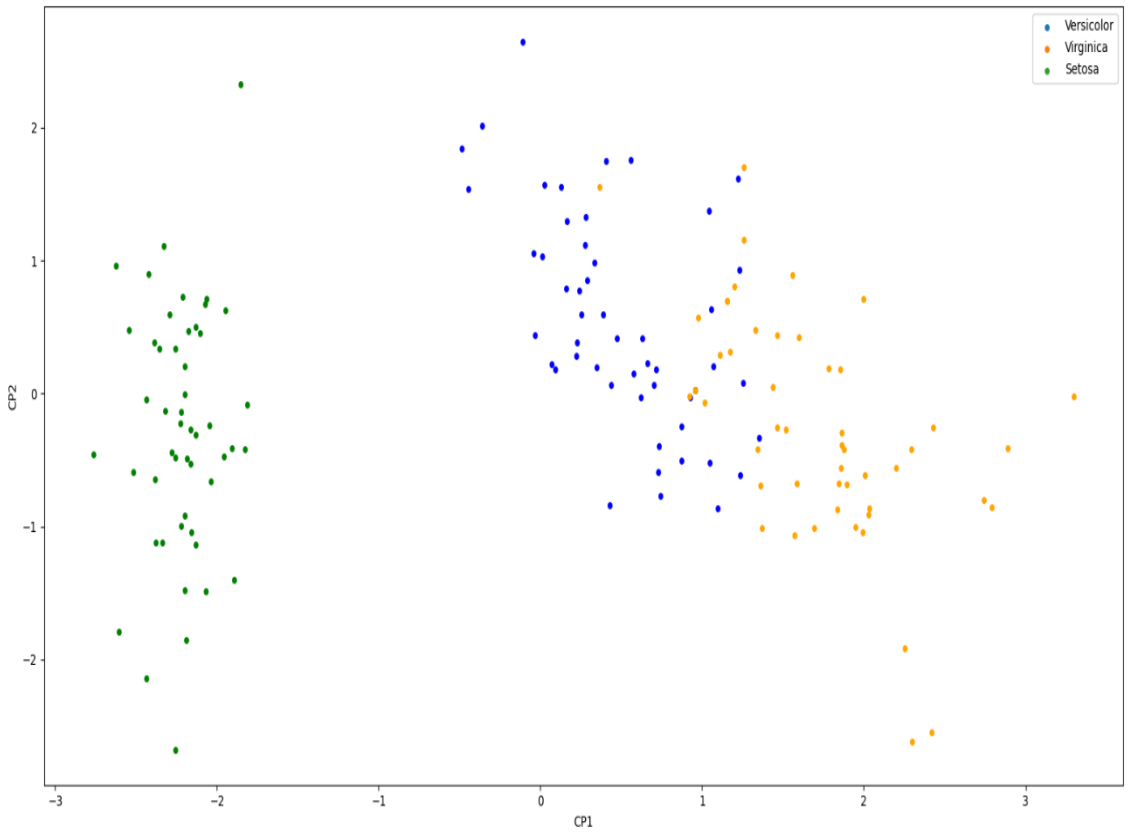


Figura 3. Gráfica de valores estandarizados.

```
Matriz de covarianza:
 [[ 0.99333333 -0.11678599  0.86594208  0.81248819]
 [-0.11678599  0.99333333 -0.42558384 -0.36368509]
 [ 0.86594208 -0.42558384  0.99333333  0.95644633]
 [ 0.81248819 -0.36368509  0.95644633  0.99333333]]
Valores propios:
 [2.89904116 0.90793693 0.1457785  0.02057674]
Vectores propios:
 [[ 0.52106591 -0.37741762 -0.71956635  0.26128628]
 [-0.26934744 -0.92329566  0.24438178 -0.12350962]
 [ 0.5804131  -0.02449161  0.14212637 -0.80144925]
 [ 0.56485654 -0.06694199  0.63427274  0.52359713]]
[ 0.52106591 -0.26934744  0.5804131   0.56485654]
[-0.37741762 -0.92329566 -0.02449161 -0.06694199]
```

Figura 4. Resultados de valores estandarizados.

Conclusiones

Mediante el desarrollo de esta práctica pudimos darnos cuenta de lo importante que es el análisis de componentes principales cuando se trabaja con un banco de datos muy grande, pues gracias a ese método podemos reducir el número de atributos (tal y como lo trabajamos en esta práctica) y con ello lograr visualizarlos de mejor manera con dimensiones que se puedan graficar, además, permite simplificar el análisis de los datos pues al trabajar con una menor cantidad de ellos es posible identificar más rápido aquellos atributos que resultan importantes y así notar las variaciones que hay entre ellos de forma más eficiente. Todo eso mejora si se realiza con valores estandarizados, pues así se garantiza una menor variación entre ellos.

Referencias

CAPÍTULO 4 ANÁLISIS DE COMPONENTES PRINCIPALES. (s. f.). En *INGENIERÍA UNAM* (p. 44). UNAM. http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/139/7/A7.pdf

Análisis de Componentes Principales (ACP). (s. f.). XLSTAT, Your data analysis solution. https://www.xlstat.com/es/soluciones/funciones/analisis-de-componentes-principales-acp