**EJERCICIOS DE PROGRAMACIÓN USANDO PCA**

A continuación, se plantean una serie de ejercicios. Por favor, responde a las preguntas que plantean. A parte de ello, no olvides adjuntar el código empleado para producir las respuestas a cada ejercicio.

**1 (2.5 puntos).** Carga el archivo de datos “Ingredient\_data.xlsx”, el cual contiene información en 5 variables diferentes representadas en sus 5 columnas (aluminato tricálcico, silicato tricálcico, aluminoferrito tetracálcico, silicato beta-dicálcico, calor durante endurecimiento a lo largo de 180 días) para 12 ingredientes diferentes. Aunque el conjunto de datos está caracterizado por 5 variables, representa los observaciones en un scatterplot de 3 dimensiones caracterizado por las 2 primeras variables y la última: aluminato tricálcico (columna 1), silicato tricálcico (columna 2) y calor durante endurecimiento a lo largo de 180 días (columna 5). Haz que en el gráfico los diferentes ejes tengan el nombre correspondiente: “aluminato tricálcico”, “silicato tricálcico” y “calor”.

Para poder realizar un gráfico de 3 dimensiones importamos las siguientes librerías:

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

Cargamos los datos y echamos un vistazo a su aspecto:

ingredient\_data=pd.read\_excel("Ingredient\_data.xls", header=None)

ingredient\_data.shape

nombres=['aluminato tricálcico', 'silicato tricálcico', 'aluminoferrito tetracálcico', 'silicato beta-dicálcico', 'calor 180 días']

ingredient\_data.columns=nombres

ingredient\_data.head()

Realizamos el gráfico de 3 dimensiones:

fig = plt.figure(figsize=(**10**,**8**))

ax = fig.add\_subplot(**111**, projection='3d')

x = ingredient\_data['aluminato tricálcico']

y = ingredient\_data['silicato tricálcico']

z = ingredient\_data['calor 180 días']

ax.scatter(x, y, z, c='b', marker='o')

ax.set\_xlabel('aluminato tricálcico',fontsize=**12**)

ax.set\_ylabel('silicato tricálcico',fontsize=**12**)

ax.set\_zlabel('calor 180 días',fontsize=**12**)

ax.view\_init(elev=**20**,azim=**10**)

plt.title("Scatterplot 3D de Ingredientes",fontsize=**14**)

plt.show()

Output:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

**2 (2.5 puntos).** Normaliza los datos de tal manera que su media sea 0 y su desviación típica 1. Indica los valores de la media y la desviación típica de los datos resultantes de la normalización.

Seguidamente, realiza un análisis de componentes principales en el que extraigas los 3 principales componentes mediante los cuales podrías llevar a cabo una reducción de la dimensionalidad. Indicar los autovectores correspondientes a cada uno de estos 3 componentes e indica también el autovalor de cada uno.

En primer lugar, normalizamos los datos y mostramos su media y su desviación típica:

#Normalizacion

**from** **sklearn.preprocessing** **import** StandardScaler

x=ingredient\_data.loc[:,nombres].values

x=StandardScaler().fit\_transform(x)

np.mean(x),np.std(x)

Output:

(2.049642507000289e-17, 0.9999999999999999)

Después realizamos un análisis de componentes principales reducciendo sus dimensiones a 3 y mostramos por pantalla los autovectores correspondientes y los autovalores de cada dimensión:

feat\_cols=['nombres'+str(i) **for** i **in** range(x.shape[**1**])]

normalised\_ingredient=pd.DataFrame(x,columns=feat\_cols)

normalised\_ingredient.tail()

#PCA

**from** **sklearn.decomposition** **import** PCA

pca\_ingredient=PCA(n\_components=**3**)

principalComponents\_ingredient=pca\_ingredient.fit\_transform(x)

#Autovectores

np.transpose(pca\_ingredient.components\_) #en vertical para que filas=variables y columnas=componentes principales

Output (Autovectores):

array([[-0.40115034, -0.51247785, 0.58088361],

[-0.46827113, 0.40959776, -0.38985593],

[ 0.31889626, 0.60802569, 0.6746767 ],

[ 0.460288 , -0.4470898 , -0.10385812],

[-0.55339017, 0.00340611, 0.21121418]])

#Autovalores

pca\_ingredient.explained\_variance\_ #sorprendentemente esto nos indica que podriamos reducir a 2 componentes

Output (Autovalores):

array([3.47926995, 1.70745323, 0.21554644])

**3 (2.5 puntos).** Haz un PCA en el que extraigas los 2 componentes principales. Seguidamente, haz un DataFrame en el que para cada uno de los ingredientes se especifique su puntuación en el componente principal 1 y en el componente principal 2. Llama a las columnas de este DataFrame “componente principal 1” y “componente principal 2” respectivamente, y muestra los valores numéricos de dicho DataFrame (para ello, puedes simplemente generar el DataFrame e imprimirlo en la ventana de comandos con la función “print”. Seguidamente, basta con que pegues una captura de pantalla del output en tu respuesta).

En primer lugar, realizamos un análisis de componentes principales indicando que extraiga 2 dimensiones:

pca\_ingredient\_2=PCA(n\_components=**2**)

principalComponents\_ingredient2=pca\_ingredient\_2.fit\_transform(x)

Después, mostramos por pantalla un dataframe con las puntuaciones de cada ingrediente en cada componente principal:

principal\_ingredient2\_df=pd.DataFrame(data=principalComponents\_ingredient2,

columns=['componente principal 1', 'componente principal 2'])

principal\_ingredient2\_df

Output:

Tabla

Descripción generada automáticamente

¿Qué porcentaje de la varianza total de los datos explican respectivamente el componente principal 1 y el componente principal 2?

pca\_ingredient\_2.explained\_variance\_ratio\_f

Output:

array([0.64232676, 0.31522213])

El componente principal 1 aglutina el 64.23% de la varianza total y el componente principal 2 el 31.52%.

**4 (2.5 puntos).** Representa gráficamente los datos en función de los 2 componentes principales identificados. En la representación gráfica, llama a los ejes x e y “componente principal 1” y “componente principal 2” respectivamente.

plt.figure(figsize=(**8**, **6**))

plt.scatter(

principal\_ingredient2\_df['componente principal 1'],

principal\_ingredient2\_df['componente principal 2'],

alpha=**0.7**, edgecolor='k'

)

# Etiquetar los ejes

plt.xlabel('Componente principal 1', fontsize=**14**)

plt.ylabel('Componente principal 2', fontsize=**14**)

plt.title('Representación de los datos en los dos componentes principales', fontsize=**16**)

# Mostrar la gráfica

plt.grid(True)

plt.show()

# Proporciones de varianza explicada

explained\_variance\_ratio = pca\_ingredient\_2.explained\_variance\_ratio\_

**print**("Proporción de varianza explicada por cada componente:")

**print**(f"Componente principal 1: {explained\_variance\_ratio[0]:.2f}")

**print**(f"Componente principal 2: {explained\_variance\_ratio[1]:.2f}")

Output:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Proporción de varianza explicada por cada componente:

Componente principal 1: 0.64

Componente principal 2: 0.32