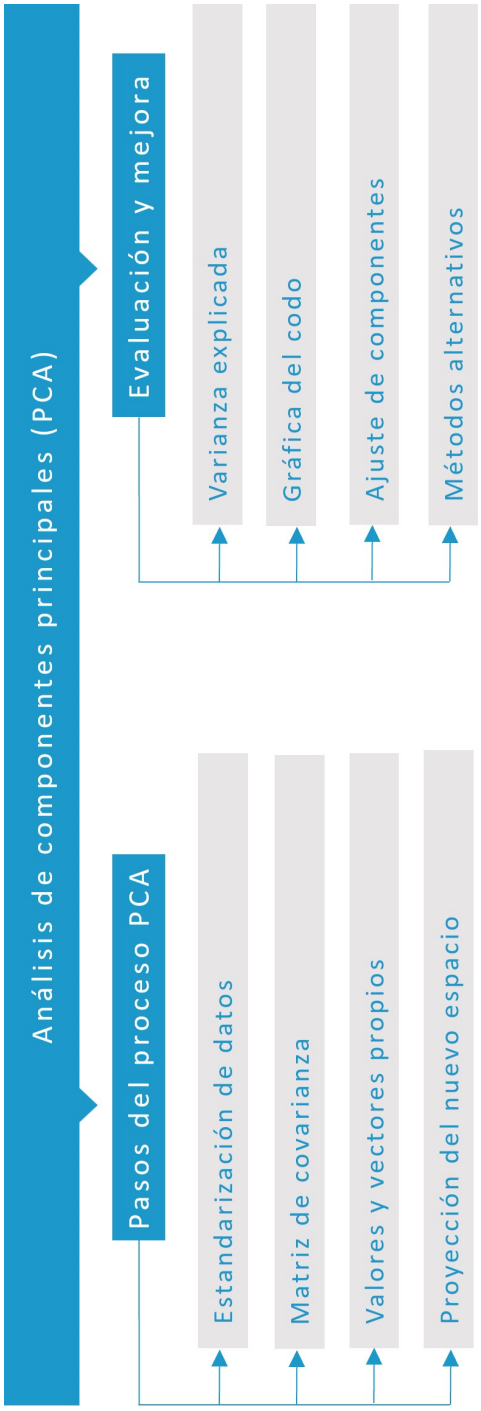
Aprendizaje Automático No Supervisado

Tema 6. Técnicas lineales. Análisis de componentes principales (PCA)

Índice

Esquema



6.1. Introducción y objetivos

En el análisis de datos, simplificar la información es una tarea compleja, pero a la vez clave para obtener información útil. El **análisis de componentes principales** (Principal Component Analysis - PCA) es una herramienta eficaz que ayuda a descubrir estructuras ocultas en grandes conjuntos de datos. No es un simple algoritmo matemático, es una técnica estadística utilizada en aprendizaje automático no supervisado que permite reducir la dimensionalidad, ver patrones y relaciones en los datos que de otra forma no se podría observar.

Esta técnica es esencial para simplificar la información y mejorar la eficiencia en diversas aplicaciones, desde la compresión de imágenes hasta la bioinformática y el análisis financiero (Ma y Dai, 2011; Boutsidis, Mahoney y Drineas, 2008; Xie y Yu, 2021).

Como **objetivos** de este tema nos planteamos:

- Explicar detalladamente los conceptos y mecanismos detrás del algoritmo PCA, incluyendo la estandarización de datos, la matriz de covarianza, y el cálculo de valores y vectores propios.
- Demostrar cómo la aplicación de PCA puede simplificar los datos y mejorar la efectividad de los algoritmos de *clustering,* a través de ejemplos prácticos y ejercicios en Python.

6.2. ¿Qué es PCA?

Vamos a imaginarnos que queremos tomar una foto a la Sagrada Familia de Barcelona. Queremos capturar su esencia, su diseño único y su construcción prolongada, pero después de muchos intentos, vemos que es imposible. Lo que podemos hacer es buscar un buen ángulo que muestre sus mejores características. Principal Component Analysis (PCA) o análisis de componentes principales examina los datos complejos y multidimensionales para encontrar las perspectivas que nos den la mayor cantidad de información sobre los datos. Resalta los patrones más importantes y reduce aquellos menos importantes.

El objetivo de PCA es reducir la dimensionalidad de los datos; ayuda a **simplificar los datos** en un conjunto más pequeño denominado «de componentes principales». Los componentes principales son variables que se construyen como combinación lineal de las variables originales, intentando capturar la máxima variación o información del conjunto de datos.

La **varianza de los datos** es como la diversidad de información. Una varianza alta es un indicador de que tiene una amplia gama de valores, lo que posiblemente indica una fuente rica en información. Entonces, PCA lo que hace es buscar direcciones donde esa variación sea máxima. Para elegir el primer componente principal, se intenta representar en una variable la mayor varianza posible. El segundo componente es independiente del primero; captura al igual que el primero la mayor varianza posible. Y así se crea cada uno de los componentes, reduciendo la dimensionalidad.

Estructura matemática de PCA

Comenzamos con un conjunto de datos generalmente grande; representado como una matriz donde las filas representan las observaciones y las columnas corresponden a los atributos o características. El algoritmo sigue estos pasos:

Estandarización de datos

Recordemos que lo que hace la estandarización es centrar los datos. Se resta la media y se divide por la desviación estándar cada uno de los datos. Así garantizamos que todas las características tengan la misma importancia.

Matriz de covarianza

Es una matriz cuadrada que describe la variabilidad conjunta de dos variables. Para un conjunto de datos con n características, la matriz de covarianza será de tamaño nxn . La matriz de covarianza es el corazón del algoritmo PCA. Podemos pensar en la matriz de covarianza como una tabla que responde a la pregunta cuando una variable aumenta, ¿qué sucede con las demás? Los valores positivos indican una relación positiva (es decir, aumentan juntos), los valores negativos indican una relación negativa y cero indica que no hay relación.

Cálculo de valores y vectores propios

Un vector propio o vector característico es un vector que cuando se transforma mediante una matriz solo se escala por un factor, conocido como el valor propio que representa la cantidad de varianza explicada por cada vector propio, a esto le llamaremos componente principal.

- **Valores propios *(eigenvalues)*:** indica cuánto estira o encoge un vector cuando se aplica una transformación lineal representada por una matriz.
- **Vectores propios *(eigenvectors)*:** son vectores que cuando se aplica una transformación lineal solo cambian en magnitud, no en dirección. Es donde la magia de PCA comienza a desarrollarse. Cada vector propio representa un componente. Lo interesante de estos vectores es que están orientados hacia donde los datos varían más.

La relación matemática entre una matriz A , un vector propio v y su valor propio

λ se expresa a través de la siguiente ecuación:

$$Av = \lambda v$$

Aplicamos la matriz A al vector v y el resultado es el vector v escalado por λ.

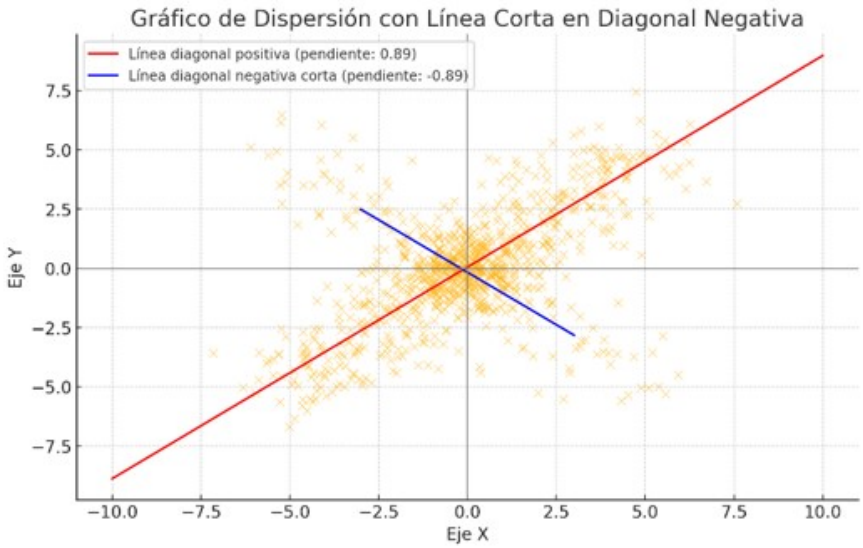


Figura 1. Vectores propios dibujados sobre un gráfico de dispersión. Fuente: elaboración propia.

Ordenar valores y vectores propios

Para poder identificar los componentes principales, se deben ordenar los valores propios de forma descendente, de tal manera que mayor valor sea el primero de la lista.

Seleccionar los componentes principales

Se seleccionan los vectores propios con mayor valor propio. El primer componente principal corresponde al vector propio con el valor propio más grande, explicando la mayor varianza existente en el conjunto de datos. El segundo componente principal corresponde al vector propio con el segundo valor propio más grande, y así sucesivamente. Los componentes principales son ortogonales, lo que significa que

no están correlacionados. El resultado es una matriz que proyecta los datos originales en un espacio de menor dimensión, reduciendo el conjunto de datos.

El análisis de componentes principales (PCA) no solo destaca por su elegancia matemática, sino que también se presenta como una herramienta esencial y eficaz para solucionar problemas reales.

6.3. ¿Por qué PCA simplifica los datos para el clustering?

Cuando tenemos un conjunto de datos de alta dimensionalidad, sufrimos la maldición de la **dimensionalidad.** A medida que aumenta el número de dimensiones (características o variables) en un conjunto de datos, pueden ocurrir varios fenómenos complejos:

- **Escasez de datos:** los puntos de datos tienden a estar más dispersos. Esto significa que para mantener la misma densidad de datos que en espacios de menor dimensión, se necesitarían más datos, lo cual rara vez es posible.
- **Grandes distancias entre datos:** la distancia entre puntos de datos tiende a aumentar y las diferencias entre las distancias se vuelven menos pronunciadas. Esto puede dificultar la identificación de patrones y la agrupación de datos.
- **Relevancia de las variables:** es probable que muchas dimensiones o atributos sean ruidosas o irrelevantes, complicando la aplicación de modelos o reduciendo su rendimiento.
- **Problemas computacionales:** el procesamiento y análisis de datos en alta dimensión suele ser costoso en términos de tiempo y recursos computacionales. Los algoritmos que funcionan bien en baja dimensión pueden volverse ineficientes o ineficaces en alta dimensión.

PCA reduce la dimensionalidad al proyectar los datos en nuevos espacios de dimensiones, facilitando la visualización, la eliminación de ruido y las redundancias. Al trabajar con un conjunto de datos de menor dimensionalidad, la complejidad computacional se reduce, se tienen procesamientos más rápidos y menor exigencia en recursos computacionales. Los datos a menudo vienen con ruido, variaciones que no son de interés pero que pueden oscurecer otros patrones. Los datos proyectados

en los componentes principales son menos susceptibles al ruido y a las variaciones aleatorias de las características originales, mejorando la separabilidad de los posibles grupos existentes.

> Ejemplo de cómo PCA nos ayuda cuando queremos realizar *clustering.*
>
> Imaginemos que estamos trabajando con un conjunto de datos de clientes con 50 características, como edad, ingresos, hábitos de compra, etc. Si aplicamos PCA, podemos reducir de 50 características a un conjunto de 2 o 3 componentes principales que capturan la mayor parte de la variabilidad de los datos. Al realizar *clustering* sobre las componentes principales, es probable que obtengamos clústeres más claros y significativos, lo que facilita la segmentación de clientes y la identificación de patrones de comportamiento.

6.4. Ventajas y desventajas de PCA

Algunas de las **ventajas** que podemos resaltar de PCA son las siguientes:

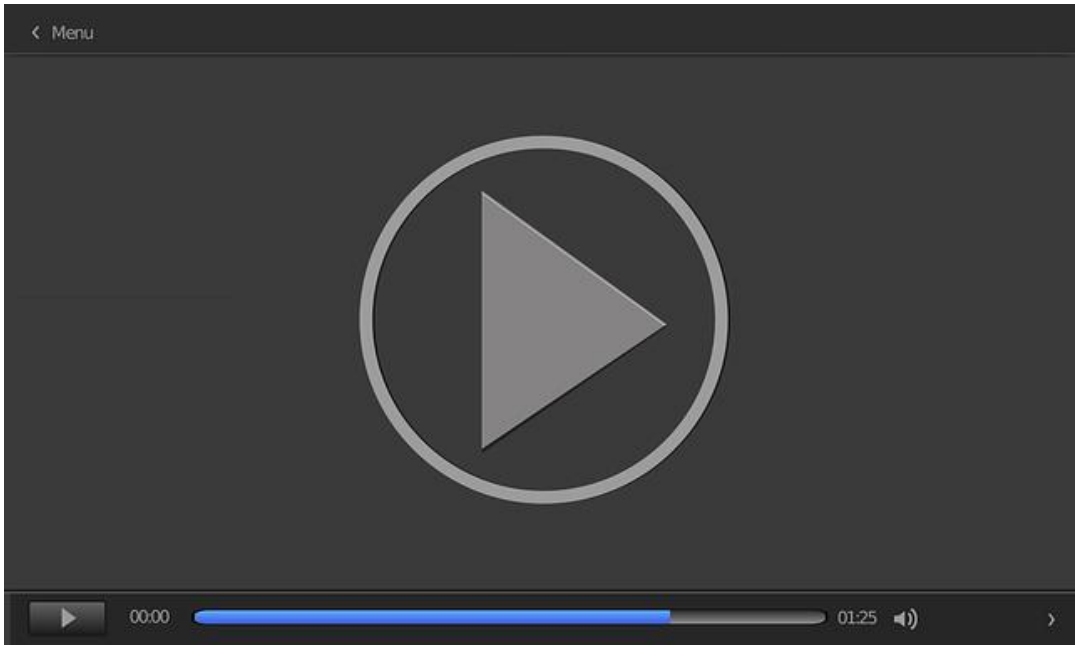
- **Reducción de dimensionalidad:** al reducir el número de variables, los modelos se vuelven más simples. Mejora el rendimiento computacional de los modelos que se aplican después de aplicar PCA y, por ende, el tiempo de entrenamiento es menor.
- **Eliminación de redundancia:** ayuda a eliminar la multicolinealidad al transformar las variables originales en un nuevo conjunto de variables independientes.
- **Visualización en 2D y 3D:** facilita la visualización de datos complejos en dos o tres dimensiones, ayudando sobre todo en el análisis exploratorio de los datos.
- **Reducción de ruido:** al enfocarse en las componentes principales que captura la mayor varianza, deja de lado los componentes que representan ruido en el conjunto de datos.
- **Mejora de la interpretabilidad:** facilita la identificación de patrones y relaciones subyacentes en los datos que no son evidentes en el espacio original.

PCA se aplica en diferentes dominios, tales como (Dey, 2023):

- **Compresión de imágenes:** al reducir la dimensionalidad de las imágenes y preservar su esencia, se pueden almacenar y transferir las imágenes.
- **Bioinformática:** las expresiones genéticas suelen ser conjuntos de datos de alta dimensionalidad. Con PCA se pueden identificar patrones y reducir el ruido.
- **Reconocimiento facial:** permite la extracción de rasgos faciales esenciales para tareas de reconocimiento.
- **Sistemas de recomendación:** usar PCA suele obtener recomendaciones muy eficientes en datos de interacción entre un usuario y elementos en particular,

generalmente elementos web.

A continuación, veremos el vídeo de ***PCA utilizando Python.***



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=00d8d339-9b7c-4478-acb7-b1be0170d19f

6.5. ¿Cómo evaluar y mejorar PCA?

Existen algunos métodos para determinar el valor óptimo para el parámetro $n_{componentes}$. Se puede realizar a través de dos visualizaciones diferentes.

La **primera visualización** es **graficar el porcentaje de varianza** explicada de los componentes individuales y el porcentaje de varianza total que capturan todos los componentes principales. Veamos el código que hace posible dicha visualización (Kavlakoglu, 2024):

```
# Calcular la matriz de covarianza, los vectores y valores propios
cov_mat = np.cov(X_train_scaled.T)
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)

exp_var = []

# Ordenar los valores propios en orden descendente
eigen_vals = np.sort(eigen_vals)[::-1]

for i in eigen_vals:
  var = (i / np.sum(eigen_vals)) * 100
  exp_var.append(var)

bar = plt.bar(range(1, 14), exp_var, align='center',
        label='Individual explained variance')

# Agregar etiquetas a las barras más altas.
for i, bar in enumerate(bar):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
f'{exp_var[i]:.1f}%',
             ha='center', va='bottom')

plt.ylabel('Percentage of variance explained (%)')
plt.xlabel('Principal component index')
plt.xticks(ticks=list(range(1, 14)))
plt.legend(loc='best')
plt.tight_layout()
```

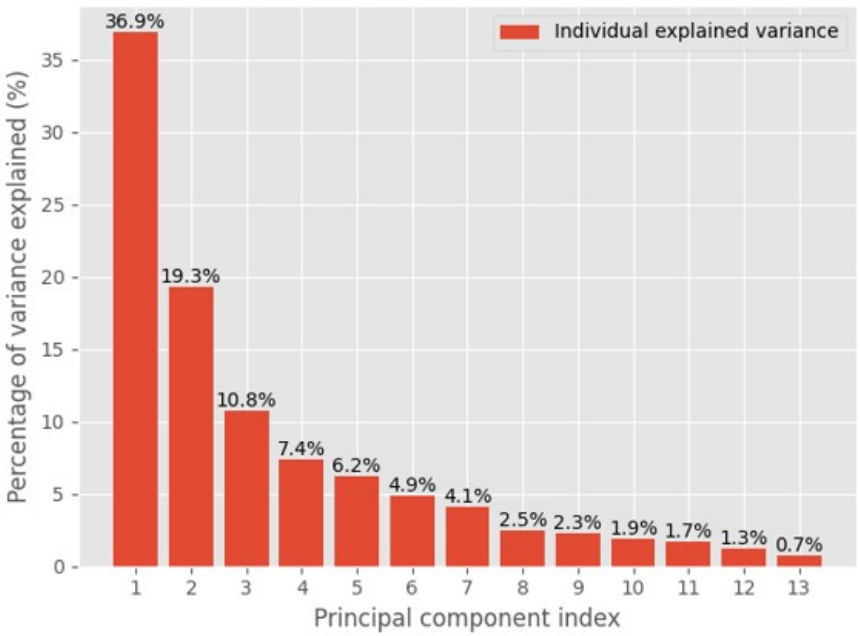


Figura 2. Varianza explicada vs. índice de componente principal. Fuente: Kavlakoglu, 2024.

La **segunda visualización** es utilizar la **técnica del codo** utilizando el porcentaje de varianza explicada frente al número de componentes principales. Para poder decidir cuál es el número de componentes principales ideal. Veamos el código que hace posible dicha visualización (Kavlakoglu, 2024):

```
# generate scree plot
pca = PCA()
X_train = pca.fit_transform(X_train_scaled)
explained_variance = pca.explained_variance_ratio_
plt.plot(pca.explained_variance_, marker='o')
plt.xlabel("Eigenvalue number")
plt.ylabel("Eigenvalue size")
plt.title("Scree Plot")
```

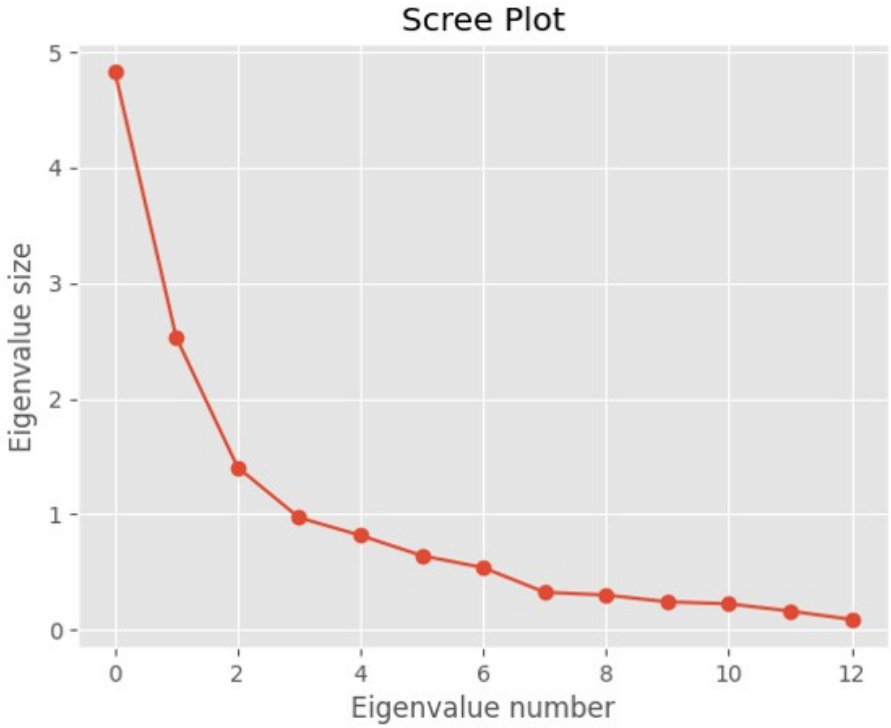


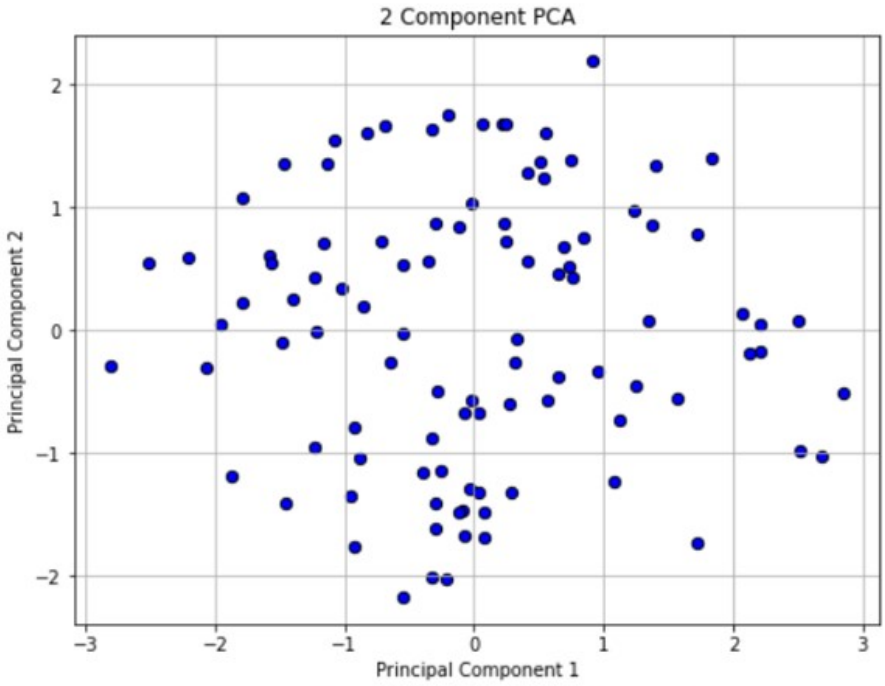
Figura 3. Valor y número de valores propios, gráfica del codo. Fuente: (Kavlakoglu, 2024).

6.6. Implementación en Python

En este apartado veremos dos ejemplos. Empecemos con un ejemplo sencillo de implementación en Python de PCA utilizando la biblioteca de Scikit-learn.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Generar datos aleatorios de ejemplo
np.random.seed(42)
data = np.random.rand(100, 5) # 100 muestras, 5 características
df = pd.DataFrame(data, columns=[f'Feature_{i+1}' for i in range(5)])

# 1. Estandarización de los datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# 2. Aplicación de PCA
# Reducimos de 5 características a 2 componentes principales

pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_data)
pca_df = pd.DataFrame(data=principal_components, columns=['Principal
Component 1', 'Principal Component 2'])

# 3. Explicación de la varianza
explained_variance = pca.explained_variance_ratio_

# 4. Visualización de los resultados
#Creamos un gráfico de dispersión para visualizar los datos proyectados en
las dos componentes principales.

plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Principal Component 1'], pca_df['Principal Component
2'], c='blue', edgecolor='k', s=40)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2 Component PCA')
plt.grid()
plt.show()
```

```
# Mostrar la varianza explicada por cada componente
# Se imprime la proporción de varianza explicada por cada componente
principal para entender cuánta información se retiene en las nuevas
componentes.

print(f'Varianza explicada por la componente principal 1:
{explained_variance[0]:.2f}')
print(f'Varianza explicada por la componente principal 2:
{explained_variance[1]:.2f}')
```




```
Varianza explicada por el componente principal 1: 0.29
Varianza explicada por el componente principal 2: 0.23
```

Figura 4. Resultado final de aplicar PCA a un conjunto de datos de cinco características. Fuente: elaboración propia.

Continuemos con un ejemplo más complejo, utilizaremos el conjunto de datos que se puede encontrar en: https://github.com/jgscott/STA380/blob/master/data/protein.csv. Contiene la cantidad de alimentos consumidos por país: leche, huevos, pescado, cereales, carnes rojas, carnes blancas, nueces y verduras. Inspirados en el artículo de Lee (2024) desarrollaremos el siguiente ejercicio:

- Leemos el conjunto de datos.

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

url = "https://raw.githubusercontent.com/fenago/datasets/main/protein.csv"
food_data = pd.read_csv(url)
```

- Estandarizamos los datos para que tengan una media de 0 y una desviación de 1.

```
scaler = StandardScaler()
food_scaled = scaler.fit_transform(food_data.iloc[:, 1:])
```

- Aplicamos PCA sobre los datos estandarizados. No vamos a especificar cuántos componentes deseamos, después hacemos un análisis sobre esto.

```
pca = PCA()
principal_components = pca.fit_transform(food_scaled)
```

- Analizamos los componentes principales y determinamos cuántas componentes debemos tener.

```
loadings = pd.DataFrame(pca.components_.T, columns=[f'PC{i}' for i in
range(1, len(food_data.columns))], index=food_data.columns[1:])
print(loadings)
```

- ¿Cuáles son las características más importantes de cada componente principal?

```
import pandas as pd

# Establecer un umbral de 0.3 para saber con qué características trabajar.
threshold = 0.3

# Encontrar características con cargas por encima del umbral para cada
componente principal.

important_features = {}
for column in loadings.columns:
    important_features[column] = loadings.index[loadings[column].abs() >
threshold].tolist()

# Ahora el diccionario 'important_features' contiene las características
```

```
importantes
for pc, features in important_features.items():
    print(f"{pc}: {', '.join(features)}")
```

- Veamos el mapa de calor de todos los componentes principales.

```
import seaborn as sns

# Creamos un mapa de calor

plt.figure(figsize=(12, 8))
sns.heatmap(loadings, annot=True, cmap='coolwarm')
plt.title('PCA Loadings Heatmap')
plt.show()
```

Figura 5. Mapa de calor de las componentes principales. Fuente: elaboración propia.

- ¿Con cuántos componentes me quedo?

```
import numpy as np
# Crear un diagrama del codo
plt.figure(figsize=(8,5))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
pca.explained_variance_ratio_, marker='o', linestyle='--')
plt.title('Scree Plot')
```

```
plt.xlabel('Number of components')
plt.ylabel('Explained variance ratio')
plt.show()

# Calcular la varianza acumulada
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)
# Crear un gráfico para explicar la varianza acumulada
plt.figure(figsize=(8,5))
plt.plot(range(1, len(cumulative_explained_variance) + 1),
cumulative_explained_variance, marker='o', linestyle='--')
plt.title('Cumulative Explained Variance')
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
plt.axhline(y=0.9, color='r', linestyle='-') # 90% variance line
plt.text(0.5, 0.85, '90% cut-off threshold', color = 'red', fontsize=16)
plt.show()
```

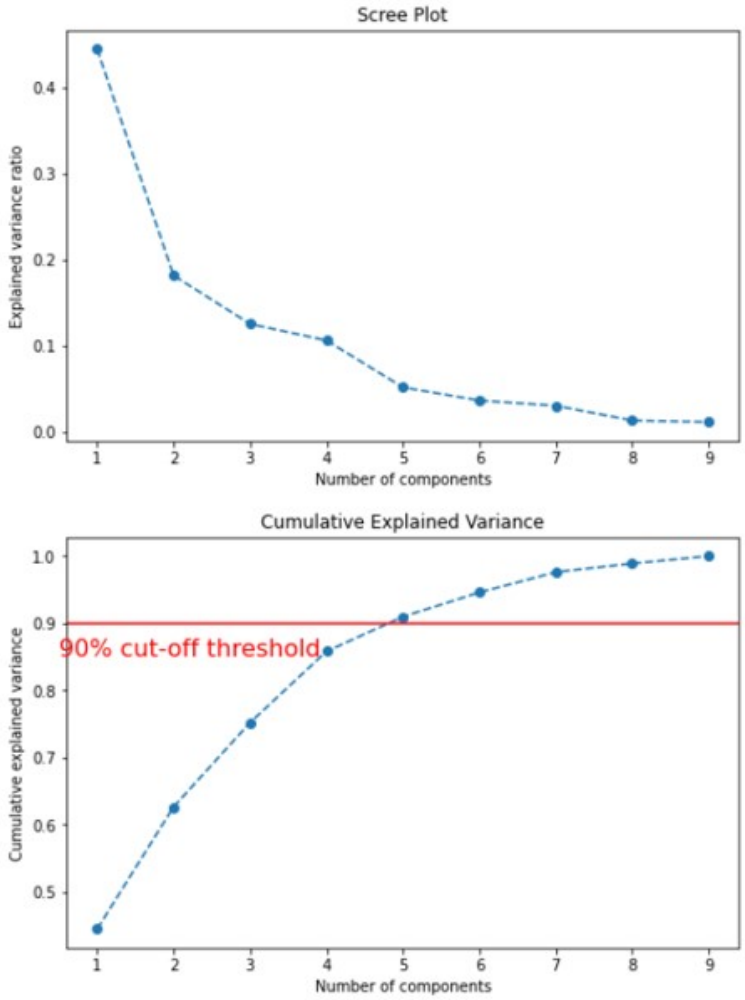


Figura 6. Gráfico del codo: número de componentes vs. varianza acumulada explicada. Fuente:

elaboración propia.

Según el resultado gráfico, el umbral que supera el 90 % está entre 4 y 5 componentes. Si elegimos el 4 estamos justo por debajo del umbral y si agregamos los cinco componentes podemos superarlo. Entonces nos quedaremos con los componentes PC1, PC2, PC3, PC4 y PC5. Si se eligen cuatro componentes es totalmente válido, sería un modelo más simple y con menos varianza acumulada explicada.

Existe otro método para decidir qué componentes principales son las mejores. Se llama el **método del criterio de Kaiser:**

```
eigenvalues = pca.explained_variance_

# Aplicando el criterio Kaiser
kaiser_criterion = eigenvalues > 1

# Obtener el valor que satisfaga el criterio Kaiser.
n_components_kaiser = sum(kaiser_criterion)

# Imprimir las components que satisfacen el criterio Kaiser
print(f"Number of components with eigenvalue > 1: {n_components_kaiser}")
important_components = np.arange(1, n_components_kaiser + 1)
print(f"Important components according to Kaiser Criterion:
{important_components}")

for i, (ev, satisfy) in enumerate(zip(eigenvalues, kaiser_criterion),
start=1):
    if satisfy:
        print(f"Component {i} with eigenvalue: {ev}")

Number of components with eigenvalue > 1: 3
Important components according to Kaiser Criterion: [1 2 3]
Component 1 with eigenvalue: 4.173372472563685
Component 2 with eigenvalue: 1.7031244242017378
Component 3 with eigenvalue: 1.1749161502198004
```

Dataframe final:

```
import pandas as pd
```

```
# Convertir componentes a un DataFrame
principal_components_df = pd.DataFrame(principal_components, columns=
[f'PC{i}' for i in range(1, len(food_data.columns))])

# Concatenar con la columna país

final_df = pd.concat([food_data['Country'], principal_components_df],
axis=1)

# Desplegar las 5 primeras filas del DataFrame
final_df.head()
```

	Country	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
0	Albania	3.557238	-1.664102	1.797551	0.234394	0.023733	-1.055592	-0.481470	-0.777255	0.105382
1	Austria	-1.452006	-1.062702	-1.365390	0.171564	-0.952695	0.222932	-0.184890	0.256178	0.221930
2	Belgium	-1.655480	0.162784	-0.221000	0.531464	0.770661	-0.295781	-0.199630	0.207504	0.033855
3	Bulgaria	3.198708	-1.327895	-0.154409	0.218606	-0.494741	-0.709921	0.474367	0.824911	0.306046
4	Czechoslovakia	-0.378104	-0.615096	-1.220603	-0.473550	0.262120	-0.840063	0.321443	-0.012552	0.152530

Figura 7. Datos. Fuente: elaboración propia.

Cómo interpretarlo:

- **PC1:** carnes rojas, carnes blancas, huevos, leche, cereales y nueces. PC1 representa los elementos fundamentales de una dieta basada en proteínas junto con alimentos básicos como cereales y nueces. Se utilizaría esta componente para comprender patrones generales de consumo de proteínas y calorías.

- **PC2:** pescado, almidón, frutas y vegetales. PC2 representa patrones dietéticos que se inclinan hacia una dieta basada en pescado como fuente principal de proteína y gran cantidad de frutas y vegetales.

- **PC3:** carne blanca, leche, pescado, frutas y vegetales. PC3 contiene elementos que contrastan entre la carne blanca y el pescado, además la presencia de leche. Podría explicar una dieta donde el elemento común es la leche y se podría explorar con este componente la interacción entre el consumo de aves, mariscos y lácteos.

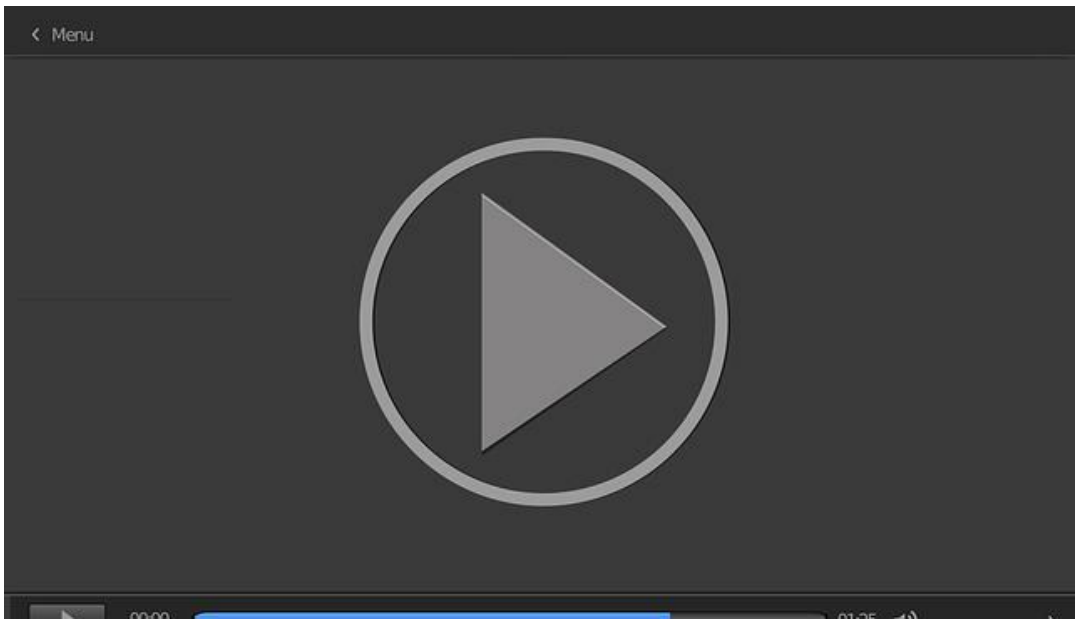
- **PC4:** carnes rojas, huevos, almidón, frutos secos, alimentos frescos y vegetales.

PC4 representa una dieta mixta, que combina proteína animal, huevos y una variedad de alimentos de origen vegetal. No se inclina por una única fuente de proteínas.

- **PC5:** carne roja, carne blanca y almidón. En PC5 la carne roja, blanca y almidón son importantes. Parece ser una dieta que busca el equilibrio entre consumo de carne y carbohidratos.
- **PC6:** carnes rojas, huevos, leche y nueces.
- **PC7:** huevos, leche, cereales, nueces, frutas y vegetales.
- **PC8:** huevos, pescado y cereales.
- **PC9:** carne blanca, huevos, pescado y frutos secos.

Cada componente resume un aspecto diferente de los hábitos alimenticios, y estos conocimientos pueden ser muy valiosos para estudios relacionados con la nutrición, la salud pública o la investigación de mercados de productos.

A continuación, veremos el vídeo ***PCA: aplicaciones avanzadas y optimización en Python.***



Ideas clave

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=9ed75fc9-515e-43d6-9697-b1be017a858c

6.7. Cuaderno de ejercicios

Ejercicio 1. Conceptos básicos

Explica con tus propias palabras qué es el análisis de componentes principales (PCA) y cuál es su objetivo principal. Luego aplica PCA a un conjunto de datos que desees y muestra los dos primeros componentes principales.

Solución

PCA es una técnica de reducción de dimensionalidad que transforma un conjunto de datos de alta dimensión en un espacio de menor dimensión, capturando la mayor varianza posible en las nuevas componentes principales. Su objetivo es simplificar los datos sin perder información relevante. Para aplicar PCA en Python, primero estandarizamos los datos y luego utilizamos la biblioteca Scikit-learn para calcular los componentes principales:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Generar datos aleatorios
np.random.seed(42)
data = np.random.rand(100, 5)
df = pd.DataFrame(data, columns=[f'Feature_{i+1}' for i in range(5)])

# Estandarizar los datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Aplicar PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_data)
print(principal_components[:5]) # Mostrar las dos primeras componentes
principales
```

Ejercicio 2. Detalle del paso a paso en PCA

Describe los pasos básicos para implementar PCA en Python utilizando un conjunto de datos de cinco características. Luego muestra un gráfico de dispersión de los dos primeros componentes principales.

Solución

Para implementar PCA se deben seguir estos pasos: estandarizar los datos, calcular las componentes principales y visualizar los resultados. El siguiente código muestra cómo hacerlo:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Generar datos aleatorios
np.random.seed(42)
data = np.random.rand(100, 5)
df = pd.DataFrame(data, columns=[f'Feature_{i+1}' for i in range(5)])

# 1. Estandarizar los datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# 2. Aplicar PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_data)
pca_df = pd.DataFrame(data=principal_components, columns=['Principal
Component 1', 'Principal Component 2'])

# 3. Visualizar los resultados
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Principal Component 1'], pca_df['Principal Component
2'], c='blue', edgecolor='k', s=40)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2 Component PCA')
plt.grid()
plt.show()
```

Ejercicio 3. Identificación de componentes principales e interpretabilidad del primer componente

Se tiene un conjunto de datos de calidad del aire con las siguientes columnas: ciudad, PM2.5, PM10, NO2, CO, SO2, O3, temperatura, humedad y velocidad del viento.

Contiene los siguientes datos:

```
data = { 'Ciudad': ['CiudadA', 'CiudadB', 'CiudadC', 'CiudadD'], 'PM2.5':
[30, 70, 40, 50], 'PM10': [50, 90, 60, 70], 'NO2': [20, 60, 30, 40], 'CO':
[0.5, 1.0, 0.7, 0.8], 'SO2': [10, 30, 15, 20], 'O3': [25, 35, 30, 28],
'Temperatura': [22, 25, 20, 23], 'Humedad': [55, 65, 60, 58], 'Velocidad
del Viento': [3.5, 4.0, 3.8, 4.2] }
```

- Interpreta los componentes principales obtenidos mediante PCA.
- Utilizando el conjunto de datos de calidad del aire en diferentes ciudades, identifica las características más importantes en el primer componente principal.

Solución

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Conjunto de datos de calidad del aire
data = {
    'Ciudad': ['CiudadA', 'CiudadB', 'CiudadC', 'CiudadD'],
    'PM2.5': [30, 70, 40, 50],
    'PM10': [50, 90, 60, 70],
    'NO2': [20, 60, 30, 40],
    'CO': [0.5, 1.0, 0.7, 0.8],
    'SO2': [10, 30, 15, 20],
    'O3': [25, 35, 30, 28],
    'Temperatura': [22, 25, 20, 23],
    'Humedad': [55, 65, 60, 58],
    'Velocidad del Viento': [3.5, 4.0, 3.8, 4.2]
}

# Crear DataFrame
df = pd.DataFrame(data)
df.set_index('Ciudad', inplace=True)
```

```
# Estandarización de los datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Aplicar PCA
pca = PCA(n_components=2) # Reducimos a 2 componentes para simplificar la
visualización
pca_result = pca.fit_transform(scaled_data)

# Interpretación de las componentes principales
pc1 = pca.components_[0]
explained_variance_ratio = pca.explained_variance_ratio_

# Crear un DataFrame para visualizar las cargas de las variables en la PC1
loadings = pd.DataFrame(pc1, index=df.columns, columns=['PC1'])

# Mostrar el DataFrame de cargas de las variables en la PC1
print("Cargas de las variables en la PC1:\n", loadings)

# Varianza explicada por cada componente principal
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = explained_variance_ratio.cumsum()

# Crear el gráfico de varianza explicada acumulada
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio) + 1),
cumulative_explained_variance, marker='o', linestyle='--', color='b')
plt.title('Gráfico de Varianza Explicada Acumulada')
plt.xlabel('Número de Componentes Principales')
plt.ylabel('Varianza Explicada Acumulada')
plt.grid(True)
plt.show()

# Mostrar varianza explicada por cada componente principal
for i, var_exp in enumerate(explained_variance_ratio, start=1):
    print(f"Componente Principal {i}: Varianza Explicada = {var_exp:.4f}")

# Mostrar varianza explicada acumulada
print("Varianza Explicada Acumulada:\n", cumulative_explained_variance)
```

Interpretación:

Cargas de las variables en la PC1:

	PC1
PM2.5	0.35862z
PM10	0.358624
NO2	0.358624
CO	0.355984
SO2	0.358624
O3	0.329616
Temperatura	0.275062
Humedad	0.329616
Velocidad del viento	0.256668

Tabla 1. Cargas de las variables en la PC1. Fuente: elaboración propia.

En este ejemplo, las variables PM2.5, PM10 y NO2 son las que más contribuyen al primer componente principal. Esto sugiere que estos contaminantes son las características más importantes para explicar la variabilidad en la calidad del aire en estas ciudades.

Ejercicio 4. Interpretación de todos los componentes principales

Explica cómo se puede evaluar la cantidad de varianza explicada por cada componente principal en PCA. Luego genera un gráfico de varianza explicada acumulada utilizando el conjunto de datos del ejercicio anterior.

Solución

La varianza explicada por cada componente principal se puede evaluar observando

los valores propios *(eigenvalues)*. Un gráfico de varianza explicada acumulada muestra cuánta varianza total se captura al agregar más componentes. Para generar este gráfico:

```
import matplotlib.pyplot as plt

pca = PCA()
principal_components = pca.fit_transform(scaled_data)
explained_variance = np.cumsum(pca.explained_variance_ratio_)

# Graficar la varianza explicada acumulada
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variance) + 1), explained_variance,
marker='o', linestyle='--')
plt.xlabel('Número de componentes')
plt.ylabel('Varianza acumulada explicada')
plt.title('Gráfico de varianza explicada acumulada')
plt.axhline(y=0.9, color='r', linestyle='-')
plt.text(0.5, 0.85, 'Umbral del 90%', color='red', fontsize=16)
plt.show()
```

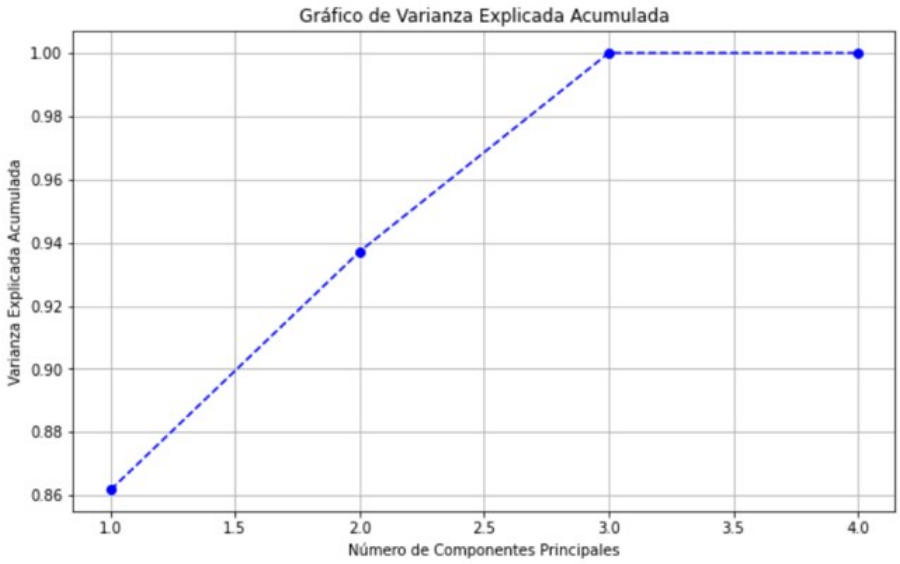


Figura 8. Gráfico de varianza explicada acumulada. Fuente: elaboración propia.

```
Componentes Principales:
       PM2.5      PM10       NO2        CO       SO2        O3  Temperatura  \
0  0.358624  0.358624  0.358624  0.355984  0.358624  0.329616     0.275062  \
1  0.023835  0.023835  0.023835  0.019162  0.023835 -0.471610     0.391678
2 -0.061146 -0.061146 -0.061146  0.172786 -0.061146  0.106382    -0.739362
3 -0.069481 -0.122013  0.833874  0.115397  0.091376 -0.251425    -0.294336

    Humedad  Velocidad del Viento
0  0.329616              0.256668
1 -0.471610              0.631757
2  0.106382              0.621208
3 -0.251425             -0.224079
Componente Principal 1: Varianza Explicada = 0.8618
Componente Principal 2: Varianza Explicada = 0.0754
Componente Principal 3: Varianza Explicada = 0.0628
Componente Principal 4: Varianza Explicada = 0.0000
Varianza Explicada Acumulada:
 [0.86177129 0.93717805 1.         1.        ]
```

El estudiante analizará las características que más pesan en cada componente.

6.8. Referencias bibliográficas

Boutsidis, C., Mahoney, M. W. y Drineas, P. (2008). *Unsupervised feature selection for principal components analysis.* Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.

Dey, R. (2023, octubre 6). *Understanding Principal Component Analysis (PCA).* Medium. https://medium.com/@roshmitadey/understanding-principal-component-analysis-pca-d4bb40e12d33

Kavlakoglu, E. (2024, marzo 6). *Reducing dimensionality with principal component analysis with Python.* IBM Developer. https://developer.ibm.com/tutorials/awb-reducing-dimensionality-with-principal-component-analysis/

Lee, E. (2024, abril 9). *Secrets of PCA: A Comprehensive Guide to Principal Component Analysis with Python and Colab.* Medium. https://drlee.io/secrets-of-pca-a-comprehensive-guide-to-principal-component-analysis-with-python-and-colab-6f7f3142e721

Ma, S. y Dai, Y. (2011). Principal component analysis based methods in bioinformatics studies. *Briefings in bioinformatics, 12*(6), 714-722.

Xie, L. y Yu, S. (2021). Unsupervised feature extraction with convolutional autoencoder with application to daily stock market prediction. *Concurrency and Computation: Practice and Experience, 33(*16), e6282.

Vídeo explicativo sobre PCA

Stanford Online. (2023, agosto 14). *Stanford CS229 Machine Learning I Factor Analysis/PCA I 2022 I Lecture 14* [Vídeo]. YouTube. https://www.youtube.com/watch?v=o2KzJdbOwMc

En la lección 14 de Stanford CS229, se explica el análisis de componentes principales (PCA) y el análisis factorial. La clase cubre cómo estas técnicas de reducción de dimensionalidad ayudan a identificar y eliminar la redundancia en los datos, mejorando la eficiencia de los algoritmos de aprendizaje automático. Además, se discuten ejemplos prácticos y aplicaciones en diversos campos.

Diferencia entre análisis factorial (FA) y análisis de componentes principales (PCA)

Thesis Helper. (2023, octubre 9). *Differences Between Factor Analysis and Principal Component Analysis* [Vídeo]. YouTube. https://www.youtube.com/watch?v=8vH0WH7p39M

Explica las diferencias clave entre el análisis factorial (FA) y el análisis de componentes principales (PCA). FA se enfoca en modelar la estructura de las correlaciones entre variables observadas para identificar factores subyacentes no observados, mientras que PCA es una técnica de reducción de dimensionalidad que transforma las variables originales en un nuevo conjunto de variables no correlacionadas, llamadas componentes principales.

Test

1. ¿Qué significa PCA?

   A. Principal Component Analysis.

   B. Pattern Classification Algorithm.

   C. Probability Calculation Approach.

   D. Partial Correlation Adjustment.

2. ¿Cuál es el principal objetivo de PCA?

   A. Aumentar el número de variables en un conjunto de datos.

   B. Reducir la dimensionalidad manteniendo la mayor cantidad de variabilidad.

   C. Clasificar datos categóricos.

   D. Eliminar datos redundantes sin considerar la variabilidad.

3. ¿Qué se estandariza en PCA?

   A. Las filas de la matriz de datos.

   B. Los valores y vectores propios.

   C. Las columnas de la matriz de datos.

   D. Los clústeres resultantes.

4. ¿Qué representa un valor propio en PCA?

   A. La cantidad de varianza explicada por cada componente principal.

   B. El valor de la covarianza entre dos variables.

   C. La media de los datos.

   D. La distancia entre clústeres.

5. ¿Qué se utiliza para calcular la matriz de covarianza en PCA?

A. La matriz transpuesta de los datos originales.

B. Los datos estandarizados.

C. Los datos en su forma original.

D. Los valores propios de los datos.

6. ¿Por qué es importante reducir la dimensionalidad de los datos?

A. Para aumentar la complejidad computacional.

B. Para facilitar la visualización y el análisis.

C. Para eliminar toda la variabilidad.

D. Para duplicar los datos disponibles.

7. ¿Qué es un vector propio en PCA?

A. Un vector que cambia de dirección durante la transformación.

B. Un vector que se utiliza para estandarizar datos.

C. Un vector que solo cambia en magnitud, no en dirección.

D. Un vector que se elimina tras la transformación.

8. ¿Qué técnica de visualización se utiliza para decidir el número de componentes en PCA?

A. Matriz de confusión.

B. Gráfica de codo.

C. Histograma.

D. Análisis de regresión.

9. ¿Cuál es una ventaja de PCA en *clustering*?

A. Incrementa la dimensionalidad de los datos.

B. Facilita la identificación de clústeres en datos de alta dimensionalidad.

C. Elimina completamente el ruido de los datos.

D. Siempre produce mejores resultados que otros algoritmos.

10. ¿Cómo se seleccionan las componentes principales en PCA?

A. Por orden alfabético de los nombres de las variables originales.

B. Por el orden de aparición en el conjunto de datos original.

C. Ordenando los valores propios de mayor a menor y seleccionando los vectores propios correspondientes.

D. Aleatoriamente.