Análisis de Componenetes Principales

Autores: Johao Hernandez, Jeiner Cantillo, Jader Gonzalez

IES INFOTEP CIENAGA,

Abstract

Este informe esta basado en el taller de analisis de componentes principales o (PCA). Este se utiliza para reducir la dimensionalidad de los conjuntos de datos mientras se conserva la información esencial permitiendo estandarizar datos y calcular valores propios.

palabras clabes: Aprendizaje no supervisado, vectores, integridad de datos, varianza

1 Introduction

El PCA nos ayuda a reducir la cantidad de variables en un conjunto de datos sin perder información importante. Básicamente, encuentra nuevas combinaciones de las variables originales para representar los datos de una manera más simple y rapida. Esto no solo facilita la visualización y el entendimiento de los datos, sino que también mejora el rendimiento de los modelos al eliminar información redundante.

2 Objetivos del Informe

Comprender y aplicar el Análisis de Componentes Principales permitiendo transformar conjuntos de datos complejos en representaciones más simples sin perder información esencial, con el fin de optimizar del aprendizaje automático.

3 Descripcion de la actividad

En esta actividad se nos pidio resolver los siguientes puntos:

3.1 Analisis bivariado:

Nos pide examinar la relación entre variables mediante correlaciones, gráficos de dispersión para observar la relación entre las variables y interpretar los resultados.

para realizar este analisis lo que se hiso fue elimiar las columnas no numericas para que filtre solo las columnas que contienen datos numéricos para despues graficarla

```
numeric_cols = blue_jays.select_dtypes(include=np.number).columns
blue_jays_numeric = blue_jays[numeric_cols]
```

Figure 1: aqui vemos el paso de como selecciona los datos numericos.

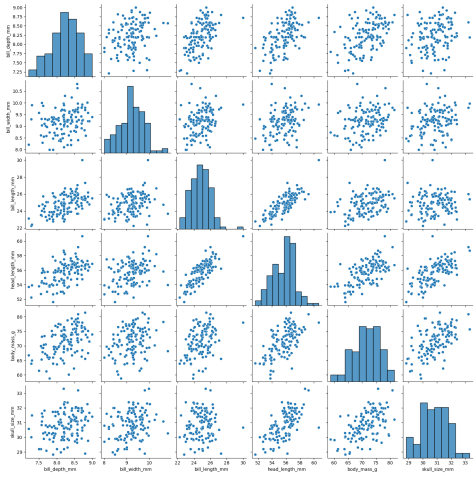

Figure 2: este fue el resultado del analisis bivariado todos contra todos.

3.2 Cálculo de la matriz de covarianza:

Determinar la relación entre las diferentes variables del conjunto de datos permitiendo identificar qué variables están correlacionadas y determinar qué información es importante.

```
cov_matrix = blue_jays_numeric.cov()
print("Matriz de Covarianza:\n", cov_matrix)
```

```
Matriz de Covarianza:
                bill_depth_mm  bill_width_mm  bill_length_mm  head_length_mm  \
bill_depth_mm        0.152248       0.056229        0.253296        0.341562
bill_width_mm        0.056229       0.284618        0.185944        0.302935
bill_length_mm       0.253296       0.185944        1.382099        1.503559
head_length_mm       0.341562       0.302935        1.503559        2.463638
body_mass_g          0.879464       0.736828        2.296619        4.709377
skull_size_mm        0.088184       0.116841        0.121306        0.960268

                body_mass_g  skull_size_mm
bill_depth_mm      0.879464       0.088184
bill_width_mm      0.736828       0.116841
bill_length_mm     2.296619       0.121306
head_length_mm     4.709377       0.960268
body_mass_g       22.721393       2.414333
skull_size_mm      2.414333       0.839314
```

Figure 3: aqui podemos observar como se extraen los daros de las columnas de el archivo "blue jays" para calcular la matriz covarianza.

3.3 Cálculo de valores y vectores propios:

obtener los valores y vectores propios de la matriz de covarianza para identificar las direcciones de mayor variabilidad en los datos.

```
valoresp, vectoresp = np.linalg.eig(cov_matrix)

print("\nValores propios:\n", valoresp)
print("\nVectores propios:\n", vectoresp)
```

```
Valores propios:
 [2.43991220e+01 2.29236306e+00 8.15169027e-01 2.44139906e-01
 9.25132232e-02 2.79516378e-06]

Vectores propios:
 [[ 3.96155230e-02 -8.98955208e-02  6.76378245e-02 -5.77088162e-02
   -9.91183080e-01  5.70543991e-04]
 [ 3.36238425e-02 -8.14750304e-02 -1.95565347e-02 -9.93775593e-01
   6.52587322e-02  4.08710414e-04]
 [ 1.11595883e-01 -6.11662426e-01  5.18544435e-01  5.00460941e-02
   9.27388770e-02  5.77427047e-01]
 [ 2.19929745e-01 -7.28830670e-01 -2.80503677e-01  7.58264347e-02
   5.10029831e-02 -5.77411029e-01]
 [ 9.61627350e-01  2.57413727e-01  9.19828457e-02  1.09828047e-02
   2.07254470e-02 -5.58377455e-05]
 [ 1.08398127e-01 -1.17020166e-01 -7.99381748e-01  2.65496024e-02
  -4.08173092e-02  5.77212278e-01]]
```

Figure 4: aqui calculamos los valores y vectores propios para indicar cuánta varianza explica cada vector propio.

3.4 Determinar cuantos y cuales componentes son necesarios para describir el 90% de la varianza de los datos:

este paso de hiso de la siguente manera:

El primer paso fue calcular la varianza explicada por cada componente principal. Para ello, utilizamos los valores propios.

luego para identificar cuánta varianza se acumula se calcula la suma acumulada de la varianza explicada y para despues buscar el mínimo número de componentes necesarios para alcanzar el 90% de la varianza acumulada.

```
varianza_explicada = valoresp / np.sum(valoresp)
varianza_acumulada = np.cumsum(varianza_explicada)

n_components_90 = np.argmax(varianza_acumulada >= 0.9) + 1

print(f"\ncomponentes para explicar el 90% de la varianza: {n_components_90}")
print(f"Componentes: {list(blue_jays_numeric.columns[:n_components_90])}")

print(f"\nProporción de Varianza Explicada por componente: {varianza_explicada}")
print(f"\nProporción de Varianza Explicada Acumulada: {varianza_acumulada}")
```

Figure 5: 90% de la varianza.

4 Conclusión

En este taller vimos cómo el PCA nos ayuda a reducir la cantidad de variables en un conjunto de datos, resaltando lo más importante y eliminando lo que no aporta mucho. Tambien nos permite tomar decisiones más acertadas al trabajar con grandes volúmenes de datos.