
**CENTRO DE CIENCIAS BÁSICAS**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**APRENDIZAJE INTELIGENTE**

**6° "A"**

# SEGUNDA EVALUACIÓN PARCIAL

**Profesor: Francisco Javier Luna Rosas**

**Alumnos:**

**Espinoza Sánchez Joel Alejandro**

**Gómez Garza Dariana**

**González Arenas Fernando Francisco**

**Fecha de Entrega:** Aguascalientes, Ags.**, 11** de abril de 2021

# Segunda Evaluación Parcial

# Evidencias del Examen

## 1. Introducción

a) ¿Qué es PCA?

Principal Component Analysis (PCA) es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información. Supóngase que existe una muestra con $n$ individuos cada uno con $p$ variables $X_1, X_2, \dots, X_P$ , es decir, el espacio muestral tiene $p$ dimensiones. PCA permite encontrar un número de factores subyacentes $(z < p)$ que explican aproximadamente lo mismo que las $p$ variables originales. Donde antes se necesitaban $p$ valores para caracterizar a cada individuo, ahora bastan $z$ valores. Cada una de estas $z$ nuevas variables reciben el nombre de componente principal.

Principal Component Analysis pertenece a la familia de técnicas conocida como aprendizaje no supervisado. Los métodos de aprendizaje no supervisado descritos tienen el objetivo de predecir una variable respuesta a partir de una serie de predictores. Para ello, se dispone de $p$ características $X_1, X_2, \dots, X_P$ y de la variable respuesta $Y$ medidas en n observaciones. En el caso del aprendizaje no supervisado, la variable respuesta $Y$ no se tiene en cuenta ya que el objetivo no es predecir $Y$ sino extraer información empleando los predictores, por ejemplo, para identificar subgrupos. El principal problema al que se enfrentan los métodos de aprendizaje no supervisado es la dificultad para validar los resultados dado que no se dispone de una variable respuesta que permita contrastarlos.

El método de PCA permite por lo tanto “condensar” la información aportada por múltiples variables en solo unas pocas componentes. Esto lo convierte en un método muy útil de aplicar previa utilización de otras técnicas estadísticas tales como regresión, clustering… Aun así, no hay que olvidar que sigue siendo necesario disponer del valor de las variables originales para calcular las componentes.

Si los triángulos representan datos, la dirección que representa la mayor varianza es la línea sobre la cual la dispersión de los datos es mayor. Si tomamos una línea vertical y proyectamos los datos sobre ella, vemos que los datos no están tan “separados” como cuando hacemos la misma operación sobre una línea horizontal. Por tanto, en este ejemplo concreto, el componente principal es la línea horizontal, que corresponde a una mayor varianza.

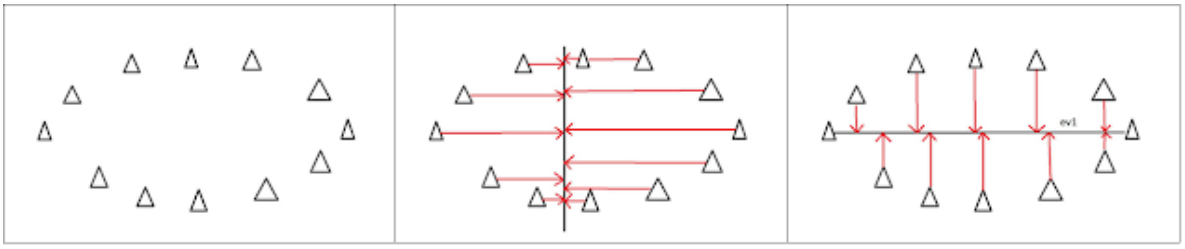

*Figura 1: Distribución de datos en la que se aprecia que la dirección de mayor dispersión es la horizontal.*

Afortunadamente, no hace falta dibujar los datos para encontrar la dirección de mayor varianza, sino que se pueden identificar por medio de métodos matemáticos.

Igual que los cocineros de moda “deconstruyen la tortilla de patatas”, con resultados, según quien sea, cuestionables, podemos deconstruir un conjunto de datos en autovectores y autovalores (*eigenvectors* y *eigenvalues*). Un autovector es una dirección, y un autovalor es un número que representa el valor de la varianza en esa dirección. Por tanto, el componente principal será el autovector con mayor autovalor (en el ejemplo, la línea horizontal).

En un conjunto de datos hay tantas parejas autovector/autovalor como dimensiones. Los autovectores no modifican los datos, sino que permiten verlos desde un punto de vista diferente, más relacionado con la estructura interna de los datos y, por tanto, ofrecen una visión mucho más intuitiva de éstos.

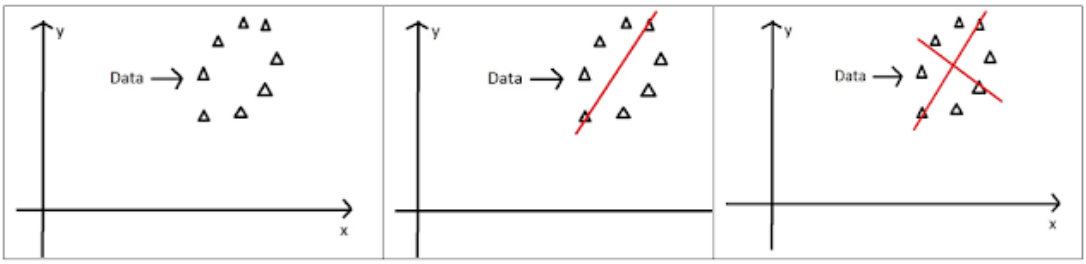


*Figura 2: Proceso de identificación de un nuevo sistema de ejes basado en los vectores propios o autovectores.*

En las figuras anteriores vemos cómo, a partir de un conjunto de datos inicial, en este caso de 2D representado en un sistema de ejes cartesianos XY, identificamos como autovector (o componente principal), la dirección de mayor varianza en los datos (línea roja en la segunda gráfica). Dado que es un sistema de 2 dimensiones, existirá otro autovector más. Si recordamos la definición anterior que hemos dado del algoritmo PCA, decíamos que era una *"transformación ortogonal".* Eso significa que el segundo autovector es perpendicular al primero (la segunda línea roja en la última gráfica). Por tanto, hemos definido un nuevo sistema de ejes sobre el que proyectaremos los datos para verlos desde otra perspectiva que facilita su comprensión, ya que los nuevos ejes son las direcciones en las que el aporte de información es mayor.

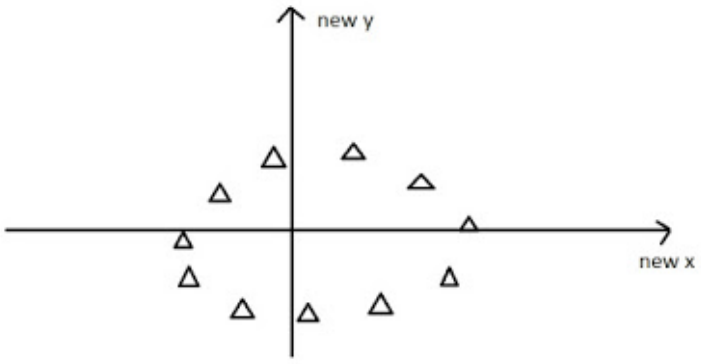


*Figura 3: Los datos se proyectan sobre un nuevo sistema de ejes.*

**¿Para qué se usa el Análisis de Componentes Principales o PCA?**

Al identificar claramente cuáles son las variables o características que aportan más información, se pueden descartar aquellas menos relevantes, reduciendo así la dimensionalidad del conjunto de datos de trabajo. De esta forma, se simplifica el problema y se agiliza todo el proceso de modelado. Por ello, es uno de los primeros pasos en este proceso.

La reducción de dimensionalidad es una de sus aplicaciones principales. Por ejemplo, en un análisis de calificaciones escolares que consideraba las notas de los alumnos en 8 asignaturas diferentes (*lengua, matemáticas, física, inglés, historia, química y gimnasia*), se determinó que 2 de los componentes principales explicaban juntos el 81% de la varianza. El primer componente estaba fuertemente *correlacionado con las asignaturas de humanidades*, mientras que el segundo, lo estaba con las de *ciencias*. Ambos conjuntos eran estadísticamente independientes (un alumno puede tener buenas notas en sólo uno de ellos, en los dos o en ninguno). De esta forma, se pudo simplificar el análisis de este conjunto de datos, de trabajar con un problema de 8 dimensiones, a otro de 2.

b) ¿Qué diferencia existe entre Clustering Jerárquico (CJ)?

El PCA Es un método matemático que se utiliza para reducir el número de variables de forma que pasemos a tener el mínimo número de nuevas variables y que representen a todas las antiguas variables de la forma más representativa posible. Es decir, si se reduce el número de variables a dos o tres nuevas, se pueden representar los datos originales en el plano o en un gráfico de 3-dimensiones y, así, se visualiza de forma gráfica un resumen de nuestros datos. El simple hecho de poder tener los datos de manera visible simplifica mucho el entender qué puede estar pasando y ayuda a tomar decisiones.

Mientras que el clustering jerárquico tiene como finalidad agrupar o segmentar una colección de objetos en subconjuntos o clusters, de manera que dentro de cada

grupo compartan características estrechamente similares que con los que están en diferente grupo

c) ¿Qué diferencia existe con K-Means?

La diferencia que existe con el algoritmo K-Means es únicamente que en el PCA se reducen las dimensiones de los datos originales, lo que causa que la información se condense en menos dimensiones. Al contrario que en el algoritmo de K-Means, donde se grafica la nube de puntos con los datos originales y se clasifican los datos originales en clústers sin importar la cantidad o dimensiones de estos. Sin embargo, con los 2 algoritmos se puede trabajar con volúmenes grandes de información, al contrario del clustering jerárquico, que se recomienda trabajar con pocos datos para un mejor resultado.

d) Aplicaciones de PCA

Esta técnica es de gran interés, porque se aplica a una gran variedad de casos. Por ejemplo, en seguridad informática se han desarrollado métodos de detección de intrusos mediante redes neuronales, basados en reducción de características. También se utiliza en problemas de clasificación de píxeles en base a su color, que son la base de muchas aplicaciones de análisis de imágenes para sistemas biométricos de reconocimiento facial, detección de defectos de fabricación en procesos industriales, etiquetado de tejidos en imágenes médicas, localización de regiones en imágenes por satélite etc. Incluso se pueden encontrar aplicaciones de PCA a investigación genética o predicción de ganancias en el mercado de valores.

## 2. Explicación y análisis

A. Transformación de la tabla de datos en componentes.

La transformación de datos es el proceso de conversión de datos de un formato a otro, normalmente del formato de un sistema fuente al exigido por el sistema de destino. La transformación de datos es un componente de la mayoría de tareas de integración y gestión de datos, como por ejemplo el data wrangling o el almacenamiento de datos.

La transformación de datos, que constituye un paso del proceso de ELT/ETL, puede describirse como “simple” o “compleja”, según el tipo de cambios que exijan los datos antes de entregarse a su destino final. El proceso de transformación de datos puede automatizarse, administrarse manualmente o completarse combinando ambas opciones.

El objetivo del proceso de transformación pasa por extraer datos de una fuente, convertirlos a un formato explotable y hacerlos llegar a un destino. Todo este proceso se conoce por la sigla ETL (extraer, transformar, cargar). Durante la fase de extracción, los datos se identifican y extraen de ubicaciones o fuentes muy distintas y se introducen en un único repositorio.

B. Reducción de la dimensión de los individuos (plano de similitud de individuos en dos dimensiones).

Las principales técnicas de reducción de la dimensión tienen por objetivo final condensar la información de un conjunto de variables en un nuevo conjunto de variables (de menor número que el anterior), con el menor coste de información posible.

El análisis de correspondencias permite representar la información por medio de tablas de contingencia, en ellas se recogen las frecuencias absolutas de las variables. Sería un análisis equivalente al método de componentes principales, pero para atributos o variables cualitativas. Por tanto, su aplicación está muy generalizada debido a que, cada vez con mayor frecuencia, se recurre al estudio de variable de tipo cualitativo, lo que supone que esta técnica asuma una mayor importancia dentro de las técnicas de reducción de la dimensión.

C. Reducción de la dimensión de las variables (circulo de correlación).

Muestra una proyección de las variables iniciales en el espacio factorial. Cuando dos variables están lejos del centro, tenemos varias posibilidades: si están próximas una a la otra, están positivamente correlacionadas; si son ortogonales, no están correlacionadas; si están en lados opuestos con respecto al centro están negativamente correlacionadas.

Cuando las variables están próximas al centro, alguna información es transportada a otros ejes, y cualquier interpretación podría resultar peligrosa. Por ejemplo, podríamos estar tentados a interpretar que existe correlación entre Migración doméstica neta y Migración internacional neta aunque, de hecho, no existe.

D. Calidad de la representación de individuos y variables (cosenos cuadrados).

Una vez llegado a este punto, es momento de evaluar la representación que se ha elegido, para ello se utilizará el cálculo por medio del coseno cuadrado del ángulo de cada punto.

La evaluación se realiza condicionando los resultados a ser cercanos al valor de 1, pues si esto ocurre, la representación del individuo es buena, pues esto significa que están representadas correctamente aquellas variables que están ubicadas cerca de la frontera o borde del círculo de correlaciones.

E. Interpretación de los clusters utilizando el principio de dualidad.

Para interpretar el círculo de relaciones que se genera se debe acudir a las correlaciones entre las variables, pues la tendencia de los datos deberá arrojar relaciones en las que algunos individuos muestren alto apego a ciertas variables y que esto parezca una equivalencia de debilidad en otras variables de modo que aquellos individuos que muestren mayor fuerza en estas últimas, sean todo lo contrario en las primeras variables mencionadas.

Asimismo, la interpretación no nos permite hallar alguna correlación acerca de comportamientos que se ven ortogonales, es decir, independientes en las variables en las que se relacionan.

## 3. Implementación

El código realizado se encuentra tanto anexo al documento como al entregable en forma de archivo ejecutable para Python.

El enfoque del algoritmo fue formular el problema creando objetos con atributos que el PCA posee, en ese sentido, el programa está pensado para un enfoque de

objetos con una única clase denominada PCA y se crea la variable inmediatamente después de que se extraen los datos del archivo.

```
#%% APARTADO DEL PROCESO DEL PCA
#   Se comienza el procedimiento PCA

#%% Se extraen los datos del archivo y se crea el objeto PCA
os.chdir("C:/Users/Dell/Desktop/datos") #Dirección del archivo
datos_est = pd.read_csv('EjemploEstudiantes.csv', delimiter = ';', decimal = ',', index_col = 0)
datos = ACP(datos_est)
print(datos.DF)
```

Únicamente para probar que se extraen correctamente los datos, se observa en ejecución lo siguiente:

| | Matematicas | Ciencias | Espanol | Historia | EdFisica |
|---|---|---|---|---|---|
| Lucia | 7.0 | 6.5 | 9.2 | 8.6 | 8.0 |
| Pedro | 7.5 | 9.4 | 7.3 | 7.0 | 7.0 |
| Ines | 7.6 | 9.2 | 8.0 | 8.0 | 7.5 |
| Luis | 5.0 | 6.5 | 6.5 | 7.0 | 9.0 |
| Andres | 6.0 | 6.0 | 7.8 | 8.9 | 7.3 |
| Ana | 7.8 | 9.6 | 7.7 | 8.0 | 6.5 |
| Carlos | 6.3 | 6.4 | 8.2 | 9.0 | 7.2 |
| Jose | 7.9 | 9.7 | 7.5 | 8.0 | 6.0 |
| Sonia | 6.0 | 6.0 | 6.5 | 5.5 | 8.7 |
| Maria | 6.8 | 7.2 | 8.7 | 9.0 | 7.0 |

Esta clase posee algunos métodos que son distintivos del PCA, en ellos están los necesarios para hacer el procedimiento, siendo el primero el cálculo del círculo de correlaciones por medio del Centrado Reducido.

```python
#%% APARTADO DE LA CREACIÓN DE LA CLASE
#   Se crea la clase de PCA
class ACP():

    #%% Se inicializa la clase
    def __init__(self, DF = pd.DataFrame()):
        self.filas = DF.shape[0]
        self.columnas = DF.shape[1]
        self.DF = DF

        def filas(self):
            return self.filas

        def columnas(self):
            return self.columnas

        def DF(self):
            return self.DF

    #%% Se define el método para hacer el cálculo del Centrado Reducido
    def CentradoReducido(self):
        MatrizLista = np.empty([self.filas, self.columnas])
        for i in range(self.filas):
            for j in range(self.columnas):
                a = self.DF.iloc[i,j]
                media = np.mean(self.DF.iloc[:,j])
                desviacion = np.std(self.DF.iloc[:,j])
                r = (a - media) / desviacion
                MatrizLista[i,j] = r
        return MatrizLista
```

Y reflejado en el código es lo siguiente:

```python
#%% Procedimiento para centrar y reducir datos de matriz
matrizX = datos.CentradoReducido()
centrado_reducido = pd.DataFrame(matrizX)
print(centrado_reducido)
```

Y al ejecutarse en el código se aprecia el centrado reducido de la siguiente forma:

```
          0         1         2         3         4
0  0.232631 -0.752986  1.788485  0.657923  0.658581
1  0.786514  1.145849 -0.538996 -0.845901 -0.476903
2  0.897290  1.014894  0.318497  0.093989  0.090839
3 -1.982900 -0.752986 -1.518987 -0.845901  1.794065
4 -0.875135 -1.080371  0.073499  0.939889 -0.136258
5  1.118843  1.276803 -0.049000  0.093989 -1.044645
6 -0.542805 -0.818463  0.563495  1.033878 -0.249807
7  1.229620  1.342280 -0.293998  0.093989 -1.612388
8 -0.875135 -1.080371 -1.518987 -2.255735  1.453420
9  0.011078 -0.294647  1.175990  1.033878 -0.476903
```

El siguiente método de la clase es el cálculo de la matriz de individuos, que dentro de la clase se aprecia de la siguiente forma:

```python
#%% Se calcula la Matriz de Individuos
def MatrizRIndividuos(self, matriz):
    A = np.matrix(matriz)
    B = np.matrix(matriz)
    T = np.transpose(A)
    TB = T * B
    R = TB / self.filas
    return R
```

Donde en el código se tiene lo siguiente:

```python
#%% Se crea la matriz de correlaciones
MatrizRind = datos.MatrizRIndividuos(matrizX)
MatrizR = pd.DataFrame(MatrizRind)
print(MatrizR)
```

Y al ejecutarse en el código, las instrucciones son las siguientes:

```
          0         1         2         3         4
0  1.000000  0.854079  0.384574  0.207194 -0.787163
1  0.854079  1.000000 -0.020052 -0.021539 -0.687721
2  0.384574 -0.020052  1.000000  0.820916 -0.365543
3  0.207194 -0.021539  0.820916  1.000000 -0.508001
4 -0.787163 -0.687721 -0.365543 -0.508001  1.000000
```

Posteriormente, se deben realizar los cálculos de los componentes principales y puede verse el método en la clase:

```python
#%% Se hacen los cálculos de las matrices para los Componentes Principales
def Componentes(self, X, V):
    ComponentesPrincipales = X * V
    return ComponentesPrincipales
```

Y la instrucción de código es la siguiente:

```python
#%% Se crea la matriz de componentes principales
ComponentesPrincipales = datos.Componentes(matrizX, MatrizRind)
print(ComponentesPrincipales)
mi_df = pd.DataFrame(ComponentesPrincipales)
print(mi_df)
```

Con un resultado en la ejecución de código siguiente:

```
[[-1.04765891e-01 -1.05725509e+00  2.19240751e+00  1.85597784e+00
   5.31210419e-03]
 [ 1.75800945e+00  2.17459756e+00 -7.79584035e-01 -9.07822999e-01
  -1.25729621e+00]
 [ 1.83454486e+00  1.71036814e+00  6.87172716e-01  4.73355507e-01
  -1.47760942e+00]
 [-4.79766005e+00 -3.63167543e+00 -3.61668293e+00 -3.39887600e+00
   4.85774870e+00]
 [-1.46759416e+00 -1.75581652e+00  5.79987659e-01  9.11393299e-01
   7.91276697e-01]
 [ 3.03226917e+00  2.94976513e+00  8.14696310e-01  7.88761949e-01
  -2.83327555e+00]
 [-6.14279358e-01 -1.14383294e+00  1.31120114e+00  1.52852605e+00
   9.14978506e-03]
 [ 3.55165378e+00  3.50521476e+00  8.18521570e-01  8.97594806e-01
  -3.44368935e+00]
 [-3.99347368e+00 -2.74830595e+00 -4.21693516e+00 -4.39908764e+00
   4.58645900e+00]
 [ 8.01295882e-01 -3.05964722e-03  2.20921522e+00  2.25017719e+00
  -1.23807576e+00]]
```

El siguiente paso es hacer el cálculo individual del coseno para cada elemento:

```python
#%% Se hacen los cálculos para los cosenos de cada individuo
def CosenosIndividuos(self, comp, matrizX):
    matriz_cosenos = np.empty([self.filas, self.columnas])
    cuadradoX = np.power(matrizX,2)
    sumarenglones = cuadradoX.sum(axis = 1)
    for i in range(self.filas):
        for j in range(self.columnas):
            componente_cuadrado = np.power(comp, 2)
            matriz_cosenos[i,j] = componente_cuadrado[i,j] / sumarenglones[i]
    return matriz_cosenos
```

Que la instrucción en código es la siguiente:

```python
#%% Se crea la matriz de cosenos cuadrados
cosenos = datos.CosenosIndividuos(ComponentesPrincipales, matrizX)
mi_df_cosenos = pd.DataFrame(cosenos)
print(mi_df_cosenos)
```

Y en ejecución se tiene:

```
          0         1         2         3         4
0 -0.104766 -1.057255  2.192408  1.855978  0.005312
1  1.758009  2.174598 -0.779584 -0.907823 -1.257296
2  1.834545  1.710368  0.687173  0.473356 -1.477609
3 -4.797660 -3.631675 -3.616683 -3.398876  4.857749
4 -1.467594 -1.755817  0.579988  0.911393  0.791277
5  3.032269  2.949765  0.814696  0.788762 -2.833276
6 -0.614279 -1.143833  1.311201  1.528526  0.009150
7  3.551654  3.505215  0.818522  0.897595 -3.443689
8 -3.993474 -2.748306 -4.216935 -4.399088  4.586459
9  0.801296 -0.003060  2.209215  2.250177 -1.238076
```

Nuevamente se da el uso del centrado reducido para el siguiente paso, el cual es para crear el círculo de correlaciones de las variables:

```
#%% Se crea el círculo de correlaciones
matrizX = datos.CentradoReducido()
print("Matriz X de variables")
print(matrizX)
```

Que en ejecución se tiene lo siguiente:

```
Matriz X de variables
[[ 0.23263076 -0.7529862   1.78848525  0.65792263  0.65858084]
 [ 0.78651352  1.14584856 -0.53899555 -0.84590053 -0.47690337]
 [ 0.89729007  1.01489444  0.31849737  0.09398895  0.09083874]
 [-1.98290027 -0.7529862  -1.51898747 -0.84590053  1.79406505]
 [-0.87513476 -1.0803715   0.07349939  0.93988948 -0.13625811]
 [ 1.11884317  1.27680268 -0.0489996   0.09398895 -1.04464547]
 [-0.5428051  -0.81846326  0.56349535  1.03387842 -0.24980653]
 [ 1.22961972  1.34227974 -0.29399757  0.09398895 -1.61238758]
 [-0.87513476 -1.0803715  -1.51898747 -2.25573474  1.45341979]
 [ 0.01107766 -0.29464677  1.1759903   1.03387842 -0.47690337]]
```

Como penúltimo paso se deberá calcular la matriz real de variables con un método de la clase que se aprecia a continuación:

```python
#%% Se calcula la Matriz de Variables
def matrizRVariables(self, matriz):
    A = np.matrix(matriz)
    B = np.matrix(matriz)
    T = np.transpose(B)
    producto = A * T
    R = producto / self.filas
    return R
```

Que en código se observa lo siguiente:

```python
#%% Se crea la matriz de correlaciones
print("Matriz real de variables")
matrizRvar = datos.matrizRVariables(matrizX)
print(MatrizR)
```

Y al ejecutarse se observa lo siguiente:

```
Matriz real de variables
          0         1         2         3         4
0  1.000000  0.854079  0.384574  0.207194 -0.787163
1  0.854079  1.000000 -0.020052 -0.021539 -0.687721
2  0.384574 -0.020052  1.000000  0.820916 -0.365543
3  0.207194 -0.021539  0.820916  1.000000 -0.508001
4 -0.787163 -0.687721 -0.365543 -0.508001  1.000000
```

Finalmente se observará el ajuste de los vectores y valores propios y se tiene como última actividad de este algoritmo el cual se hacen las siguientes instrucciones en la clase:

```
#%% Se realizan los cálculos de Álgebra Lineal para los eigenvalores
def Eigenvalores(self, matriz):
    valores_propios, vectores_propios = la.eig(matriz)

    valores_propios = valores_propios.real
    vectores_propios = vectores_propios.real
    indice_fila, indice_columna = vectores_propios.shape

    cuadrado_vectores = np.power(vectores_propios, 2)

    print("Lambda1")
    lambda1 = valores_propios[0]
    print(lambda1.real)

    print("Vectores propios al cuadrado")
    print(cuadrado_vectores)
    cuadrado = pd.DataFrame(cuadrado_vectores)
    print(cuadrado)

    v1 = cuadrado_vectores[:,0] * valores_propios[0]
    df_v1 = pd.DataFrame(v1)
    print("v1 * lambda")
    print(df_v1)
```

Con la instrucción de código siguiente:

```
#%% Se calculan los valores propios y se imprimen los datos finales del procedimiento PCA
datos.Eigenvalores(matrizRvar)
```

Y en su ejecución se tiene el siguiente resultado:

```
Lambda1
2.8932496734179463
Vectores propios al cuadrado
[[3.60734373e-03 1.92910834e-01 4.14639236e-01 2.46849744e-03
  1.48474832e-04 1.31446162e-02 7.60885426e-05 7.60885426e-05
  6.34673996e-02 1.20974054e-02]
 [1.53049754e-02 1.64881591e-01 6.10605551e-03 4.33870617e-04
  1.71198779e-01 4.76660210e-01 3.29308184e-03 3.29308184e-03
  1.34904597e-02 4.00617239e-03]
 [3.47395038e-02 1.63287785e-02 1.14109685e-01 2.17525933e-01
  2.29587197e-01 1.96862969e-02 3.50141974e-03 3.50141974e-03
  2.02449603e-01 1.44573845e-01]
 [3.47781444e-01 4.23997570e-03 4.20932799e-02 3.74661385e-01
  4.39379307e-02 4.65406118e-02 8.66160282e-03 8.66160282e-03
  8.95889256e-04 7.90638031e-04]
 [8.26032727e-03 1.14470414e-01 2.01277156e-01 1.97950024e-02
  1.71170915e-01 1.67880811e-01 4.51261829e-06 4.51261829e-06
  7.81308893e-02 1.09571895e-01]
 [1.00904790e-01 6.40942816e-02 4.65919361e-03 3.64289473e-03
  7.19414930e-03 8.69538075e-02 4.94792666e-01 4.94792666e-01
  1.11987707e-02 1.34399994e-01]
 [1.57879096e-04 1.31300601e-01 7.39418080e-02 1.13423412e-02
  1.93714136e-03 1.07673023e-04 6.55459644e-02 6.55459644e-02
  1.27987243e-01 4.87524941e-01]
 [1.39896713e-01 9.99496493e-02 8.48038057e-02 3.19050613e-02
  3.41847744e-03 1.60536131e-01 5.18783076e-02 5.18783076e-02
  3.21209693e-03 7.33788837e-02]
 [3.19846171e-01 9.66889843e-02 5.81215853e-02 3.34059370e-01
  1.61421395e-02 1.48960256e-02 4.90300764e-02 4.90300764e-02
  6.39417738e-02 2.71143278e-02]
 [2.95008527e-02 1.15134890e-01 2.48195348e-04 4.16564357e-03
  3.55264796e-01 1.35938170e-02 4.00749816e-02 4.00749816e-02
  4.35225874e-01 6.54189806e-03]]
```

```
          0         1         2  ...         7         8         9
0  0.003607  0.192911  0.414639  ...  0.000076  0.063467  0.012097
1  0.015305  0.164882  0.006106  ...  0.003293  0.013490  0.004006
2  0.034740  0.016329  0.114110  ...  0.003501  0.202450  0.144574
3  0.347781  0.004240  0.042093  ...  0.008662  0.000896  0.000791
4  0.008260  0.114470  0.201277  ...  0.000005  0.078131  0.109572
5  0.100905  0.064094  0.004659  ...  0.494793  0.011199  0.134400
6  0.000158  0.131301  0.073942  ...  0.065546  0.127987  0.487525
7  0.139897  0.099950  0.084804  ...  0.051878  0.003212  0.073379
8  0.319846  0.096689  0.058122  ...  0.049030  0.063942  0.027114
9  0.029501  0.115135  0.000248  ...  0.040075  0.435226  0.006542

[10 rows x 10 columns]
v1 * lambda
          0
0  0.010437
1  0.044281
2  0.100510
3  1.006219
4  0.023899
5  0.291943
6  0.000457
7  0.404756
8  0.925395
9  0.085353
```

# Conclusiones

**Espinoza Sánchez Joel Alejandro:** Opino que con este examen se nos permite ver las múltiples formas de hacer análisis de conglomerados con otras técnicas que hemos probado e incluso hacer el comparativo en esta evaluación, así como la implementación del algoritmo faltante junto con su interpretación al analizarlo y creo que su importancia recae en qué tipo de algoritmo se usará, ya sea K-Means, Clustering Jerárquico o el recién implementado PCA.

**Gómez Garza Dariana:** Al resolver el examen pudimos reforzar muchos puntos tanto teóricos como prácticos (en este caso en Python) del PCA y su importancia en la inteligencia artificial. Gracias a las clases que tenemos cotidianamente y a los ejemplos expuestos por el profesor pudimos tener una mejor base para arrancar y

desarrollar el examen, al igual que el ejemplo programado en R. Además que pudimos ver la diferencia entre el clustering jerárquico y el PCA.

**González Arenas Fernando Francisco:** Esta técnica de reducción de dimensiones para clusterización es muy útil para trabajar con un número menor de dimensiones y así poder representar los datos de una manera más sencilla generalmente con solo 2 componentes. Este algoritmo tiene una gran variedad de aplicaciones prácticas como, por ejemplo: en redes neuronales para el reconocimiento facial, en análisis genéticos, en inversiones en la bolsa de valores, etc. Este es un algoritmo muy importante e interesante, el cual combinado con otras técnicas como el clustering jerárquico podría ser aún más eficiente.

# Referencias


1. Wikipedia. (2016). *Análisis de componentes principales.* Abril 10, 2021, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales
2. XLSTAT. (2018). *Análisis de Componentes Principales (ACP).* Abril 9, 2021, de XLSTAT Sitio web: https://www.xlstat.com/es/soluciones/funciones/analisis-de-componentes-principales-acp#:~:text=El%20an%C3%A1lisis%20de%20componentes%20principales%20(ACP)%20consiste%20en%20expresar%20un,la%20variabilidad%20de%20los%20datos.
3. Zacarías, J. (2017). *Análisis de Componentes Principales.* Puebla: Editorial Benemérita Universidad Autónoma de Puebla.

# Anexos

## Anexo 1: Código en Python 3 de la red neuronal.

```
#%% Presentación
'''
   Universidad Autónoma de Aguascalientes

         Centro de Ciencias Básicas
 Departamento de Ciencias de la Computación
          Aprendizaje Inteligente
                   6° "A"

                 Segunda Evaluación Parcial

   Profesor: Francisco Javier Luna Rosas
  Alumnos:
      Espinoza Sánchez Joel Alejandro
      Gómez Garza Dariana
      González Arenas Fernando Francisco

  Fecha de Entrega: 11 de abril del 2021

Descripción: Código correspondiente a la segunda evaluación parcial de la
implementación del PCA
'''

#%% Importamos las librerías
import os
import pandas as pd
import numpy as np
import scipy.linalg as la

#%% APARTADO DE LA CREACIÓN DE LA CLASE
#   Se crea la clase de PCA
class ACP():

    #%% Se inicializa la clase
    def __init__(self, DF = pd.DataFrame()):
        self.filas = DF.shape[0]
        self.columnas = DF.shape[1]
        self.DF = DF

        def filas(self):
            return self.filas

        def columnas(self):
            return self.columnas

        def DF(self):
```

```
            return self.DF

    #%% Se define el método para hacer el cálculo del Centrado Reducido
    def CentradoReducido(self):
        MatrizLista = np.empty([self.filas, self.columnas])
        for i in range(self.filas):
            for j in range(self.columnas):
                a = self.DF.iloc[i,j]
                media = np.mean(self.DF.iloc[:,j])
                desviacion = np.std(self.DF.iloc[:,j])
                r = (a - media) / desviacion
                MatrizLista[i,j] = r
        return MatrizLista

    #%% Se calcula la Matriz de Individuos
    def MatrizRIndividuos(self, matriz):
        A = np.matrix(matriz)
        B = np.matrix(matriz)
        T = np.transpose(A)
        TB = T * B
        R = TB / self.filas
        return R

    #%% Se hacen los cálculos de las matrices para los Componentes Principales
    def Componentes(self, X, V):
        ComponentesPrincipales = X * V
        return ComponentesPrincipales

    #%% Se hacen los cálculos
    def CosenosIndividuos(self, comp, matrizX):
        matriz_cosenos = np.empty([self.filas, self.columnas])
        cuadradoX = np.power(matrizX,2)
        sumarenglones = cuadradoX.sum(axis = 1)
        for i in range(self.filas):
            for j in range(self.columnas):
                componente_cuadrado = np.power(comp, 2)
                matriz_cosenos[i,j] = componente_cuadrado[i,j] /
sumarenglones[i]
        return matriz_cosenos

    #%% Se realizan los cálculos de Álgebra Lineal para los eigenvalores
    def valores_propios_var(self, matriz):
        valores_propios, vectores_propios = la.eig(matriz)

        valores_propios = valores_propios.real
        vectores_propios = vectores_propios.real
        indice_fila, indice_columna = vectores_propios.shape

        cuadrado_vectores = np.power(vectores_propios, 2)

        print("Lambda1")
```

```
        lambda1 = valores_propios[0]
        print(lambda1.real)

        print("Vectores propios al cuadrado")
        print(cuadrado_vectores)
        cuadrado = pd.DataFrame(cuadrado_vectores)
        print(cuadrado)

        v1 = cuadrado_vectores[:,0] * valores_propios[0]
        df_v1 = pd.DataFrame(v1)
        print("v1 * lambda")
        print(df_v1)

    #%% Se calcula la Matriz de Variables
    def matrizRVariables(self, matriz):
        A = np.matrix(matriz)
        B = np.matrix(matriz)
        T = np.transpose(B)
        producto = A * T
        R = producto / self.filas
        return R

#%% APARTADO DEL PROCESO DEL PCA
#   Se comienza el procedimiento PCA

#%% Se extraen los datos del archivo y se crea el objeto PCA
os.chdir("C:/Users/Dell/Desktop/datos") #Dirección del archivo
datos_est = pd.read_csv('EjemploEstudiantes.csv', delimiter = ';', decimal =
',', index_col = 0)
datos = ACP(datos_est)
print(datos.DF)

#%% Procedimiento para centrar y reducir datos de matriz
matrizX = datos.CentradoReducido()
centrado_reducido = pd.DataFrame(matrizX)
print(centrado_reducido)

#%% Se crea la matriz de correlaciones
MatrizRind = datos.MatrizRIndividuos(matrizX)
MatrizR = pd.DataFrame(MatrizRind)
print(MatrizR)

#%% Se crea la matriz de componentes principales
ComponentesPrincipales = datos.Componentes(matrizX, MatrizRind)
print(ComponentesPrincipales)
mi_df = pd.DataFrame(ComponentesPrincipales)
print(mi_df)

#%% Se crea la matriz de cosenos cuadrados
cosenos = datos.CosenosIndividuos(ComponentesPrincipales, matrizX)
mi_df_cosenos = pd.DataFrame(cosenos)
```

```
print(mi_df_cosenos)

#%% Se crea el círculo de correlaciones
matrizX = datos.CentradoReducido()
print("Matriz X de variables")
print(matrizX)

#%% Se crea la matriz de correlaciones
print("Matriz real de variables")
matrizRvar = datos.matrizRVariables(matrizX)
print(MatrizR)

#%% Se calculan los valores propios y se imprimen los datos finales del
procedimiento PCA
datos.valores_propios_var(matrizRvar)
```