Type *Markdown* and LaTeX: α^2

Actividad Semanal -- 5 Repaso Transformación y reducción de dimensiones

CIENCIA Y ANALÍTICA DE DATOS

Profesor: Jobish Vallikavungal Devassia

ALBERTO NIEVES CISNEROS

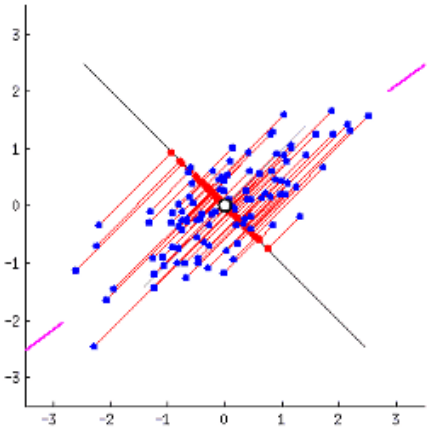
Bienvenido al notebook ¶

#Repaso de Reducción de dimensiones El objetivo es que entendamos de una manera visual, que es lo que pasa cuando nosotros seleccionamos cierto número de componentes principales o % de variabilidad de una base de datos.

Primero entenderemos, que pasa adentro de PCA que se basa en lo siguiente a grandes razgos:

Análisis de Componentes Principales

El análisis de datos multivariados involucra determinar transformaciones lineales que ayuden a entender las relaciones entre las características importantes de los datos. La idea central del Análisis de Componentes Principales (PCA) es reducir las dimensiones de un conjunto de datos que presenta variaciones correlacionadas, reteniendo una buena proporción de la variación presente en dicho conjunto. Esto se logra obteniendo la transformación a un nuevo conjunto de variables: los componentes principales (PC). Cada PC es una combinación lineal con máxima varianza en dirección ortogonal a los demás PC.



Para entender un poco más de PCA y SVD, visita el siguiente link: *Truco: Prueba entrar con tu cuenta del tec :)*

https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8 (https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8)

Basicamente, vamos a seguir los siguientes pasos:

1. Obtener la covarianza. OJO: X tiene sus datos centrados :)

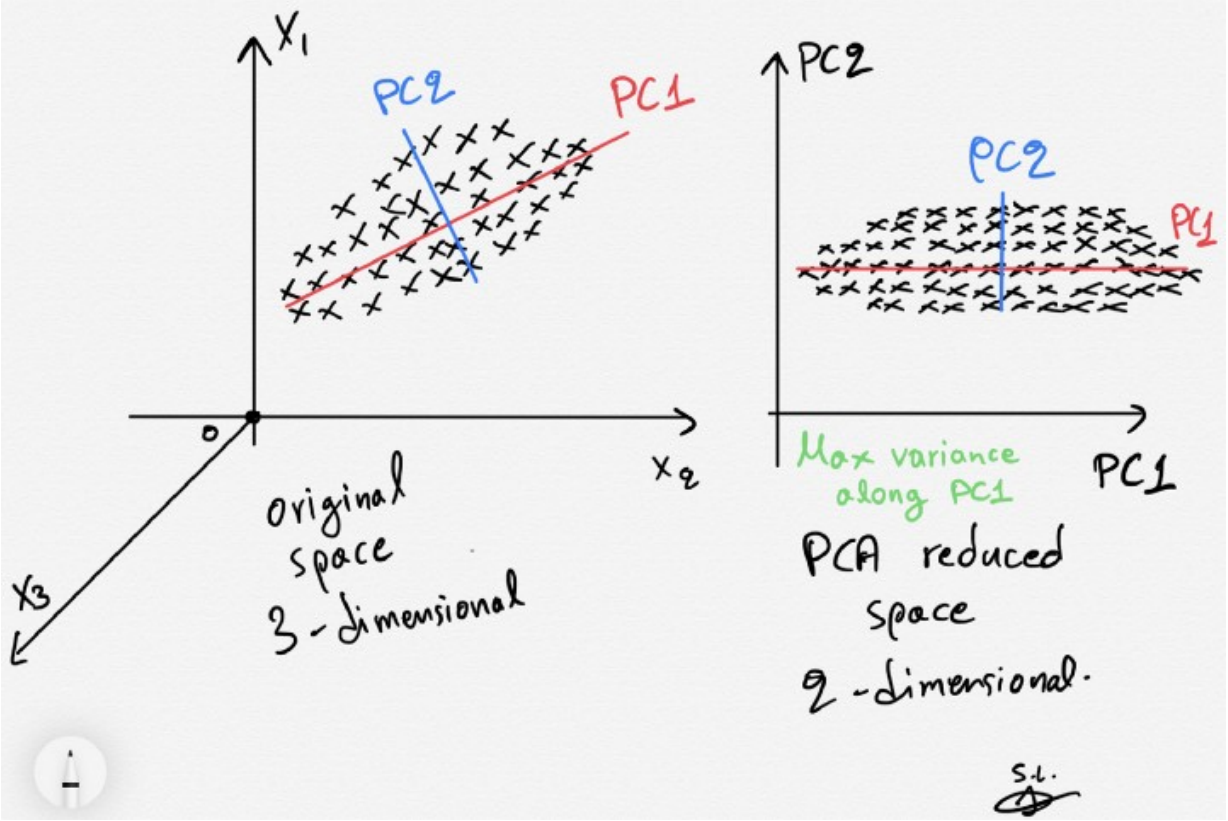
$$\mathbf{C} = \frac{\mathbf{X}^\top \mathbf{X}}{n-1}$$

2. Los componentes principales se van a obtener de la eigen descomposicion de la matriz de covarianza.

$$\mathbf{C} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^{-1}$$

3. Para la reducción de dimensiones vamos a seleccionar k vectores de W y proyectaremos nuestros datos.

$$\mathbf{X}_k = \mathbf{X}\mathbf{W}_k$$



Ejercicio 1, Descomposición y composición

Descomposición

Encuentra los eigenvalores y eigenvectores de las siguientes matrices

$$A = \begin{pmatrix} 3,0,2 \\ 3,0,-2 \\ 0,1,1 \end{pmatrix} A2 = \begin{pmatrix} 1,3,8 \\ 2,0,0 \\ 0,0,1 \end{pmatrix} A3 = \begin{pmatrix} 5,4,0 \\ 1,0,1 \\ 10,7,1 \end{pmatrix}$$

y reconstruye la matriz original a traves de las matrices WDW^{-1} (OJO. Esto es lo mismo de la ecuación del paso 2 solo le cambiamos la variable a la matriz diagonal)

Eigenvalores y eigenvectores

In [ ]:

```python
###-----------------EJEMPLO DE EIGENVALORES
import numpy as np
from numpy import array
from numpy.linalg import eig
# define la matriz
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("-------Matriz original-------")
print(A)
print("-----------------------------")
# calcula la eigendescomposición
values, vectors = eig(A)
print(values) #D
print(vectors) #W

#Ejemplo de reconstrucción


values, vectors = np.linalg.eig(A)

W = vectors
Winv = np.linalg.inv(W)
D = np.diag(values)
#la matriz B tiene que dar igual a A
#reconstruye la matriz
print("-------Matriz reconstruida-------")
# Realiza la reconstruccion de B=W*D*Winv, te da lo mismo de A?
#ojo, estas multiplicando matrices, no escalares ;)
#TU CODIGO AQUI-------------------------------------
#B=
B = W@D@Winv
print(B)
print("-----------------------------")
```

```
-------Matriz original-------
[[1 2 3]
 [4 5 6]
 [7 8 9]]
-----------------------------
[ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
-------Matriz reconstruida-------
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
-----------------------------
```

In [ ]:

```
A = array([[3, 0, 2], [3, 0, -2], [0, 1, 1]])
values, vectors = eig(A)
print(values) #D
print(vectors) #W
```

```
[3.54451153+0.j         0.22774424+1.82582815j 0.22774424-1.82582815j]
[[-0.80217543+0.j         -0.04746658+0.2575443j  -0.04746658-0.2575443j ]
 [-0.55571339+0.j          0.86167879+0.j          0.86167879-0.j        ]
 [-0.21839689+0.j         -0.16932106-0.40032224j -0.16932106+0.40032224j]]
```

In [ ]:

```
#Matriz 1
A1 = array([[3,0,2],[3,0,-2],[0,1,1]])
print("-------Matriz original-------")
print(A1)
print("------------------------------")
# calcula la eigendescomposición
values1, vectors1 = eig(A1)
print(values1) #D
print(vectors1) #W

#Reconstrucción


values1,vectors1 = np.linalg.eig(A1)

W1 = vectors1
Winv1 = np.linalg.inv(W1)
D1 = np.diag(values1)
#la matriz B tiene que dar igual a A
#reconstruye la matriz
print("-------Matriz reconstruida-------")
# Realiza la reconstruccion de B=W*D*Winv, te da lo mismo de A?
#ojo, estas multiplicando matrices, no escalares ;)
#TU CODIGO AQUI-------------------------------------
#B=
B1 = np.round(np.real(W1@D1@Winv1),10)
print(B1)
print("------------------------------")

```

```
-------Matriz original-------
[[ 3  0  2]
 [ 3  0 -2]
 [ 0  1  1]]
------------------------------
[3.54451153+0.j         0.22774424+1.82582815j 0.22774424-1.82582815j]
[[-0.80217543+0.j         -0.04746658+0.2575443j  -0.04746658-0.2575443j ]
 [-0.55571339+0.j          0.86167879+0.j          0.86167879-0.j        ]
 [-0.21839689+0.j         -0.16932106-0.40032224j -0.16932106+0.40032224j]]
-------Matriz reconstruida-------
[[ 3.  0.  2.]
 [ 3.  0. -2.]
 [ 0.  1.  1.]]
------------------------------
```

In [ ]:

```
#Matriz 2

A2 = array([[1, 3, 8], [2, 0, 0], [0,0, 1]])
print("-------Matriz original-------")
print(A2)
print("------------------------------")
# calcula la eigendescomposición
values2, vectors2 = eig(A2)
print(values2) #D
print(vectors2) #W

#Reconstrucción


values2, vectors2 = np.linalg.eig(A2)

W2 = vectors2
Winv2 = np.linalg.inv(W2)
D2 = np.diag(values2)
#la matriz B tiene que dar igual a A
#reconstruye la matriz
print("-------Matriz reconstruida-------")
# Realiza la reconstruccion de B=W*D*Winv, te da lo mismo de A?
#ojo, estas multiplicando matrices, no escalares ;)
#TU CODIGO AQUI-------------------------------------
#B=
B2 = W2@D2@Winv2
print(np.round(B2,10))
print("------------------------------")

```

```
-------Matriz original-------
[[1 3 8]
 [2 0 0]
 [0 0 1]]
------------------------------
[ 3. -2.  1.]
[[ 0.83205029 -0.70710678 -0.42399915]
 [ 0.5547002   0.70710678 -0.8479983 ]
 [ 0.          0.          0.31799936]]
-------Matriz reconstruida-------
[[1. 3. 8.]
 [2. 0. 0.]
 [0. 0. 1.]]
------------------------------
```

In [ ]:

```
#Matriz 3
A3 = array([[5,4,0], [1, 0, 1], [10, 7,1]])
print("-------Matriz original-------")
print(A3)
print("------------------------------")
# calcula la eigendescomposición
values3, vectors3 = eig(A3)
print(values3) #D
print(vectors3) #W

#Reconstrucción


values3, vectors3 = np.linalg.eig(A3)

W3 = vectors3
Winv3 = np.linalg.inv(W3)
D3 = np.diag(values3)
#la matriz B tiene que dar igual a A
#reconstruye la matriz
print("-------Matriz reconstruida-------")
# Realiza la reconstruccion de B=W*D*Winv, te da lo mismo de A?
#ojo, estas multiplicando matrices, no escalares ;)
#TU CODIGO AQUI-------------------------------------
#B=
B3 = W3@D3@Winv3

print(np.round(B3,10))
print("------------------------------")

```

```
-------Matriz original-------
[[ 5  4  0]
 [ 1  0  1]
 [10  7  1]]
------------------------------
[ 6.89167094 -0.214175   -0.67749594]
[[ 0.3975395   0.55738222  0.57580768]
 [ 0.18800348 -0.72657211 -0.81728644]
 [ 0.89811861 -0.40176864 -0.02209943]]
-------Matriz reconstruida-------
[[ 5.  4. -0.]
 [ 1. -0.  1.]
 [10.  7.  1.]]
------------------------------
```

¿Qué significa reducir dimensiones?

Esto será cuando proyectemos a ese espacio de los componentes principales pero no los seleccionemos todos, solo los más importantes y viajemos de regreso a nuestras unidades a través de una proyección.

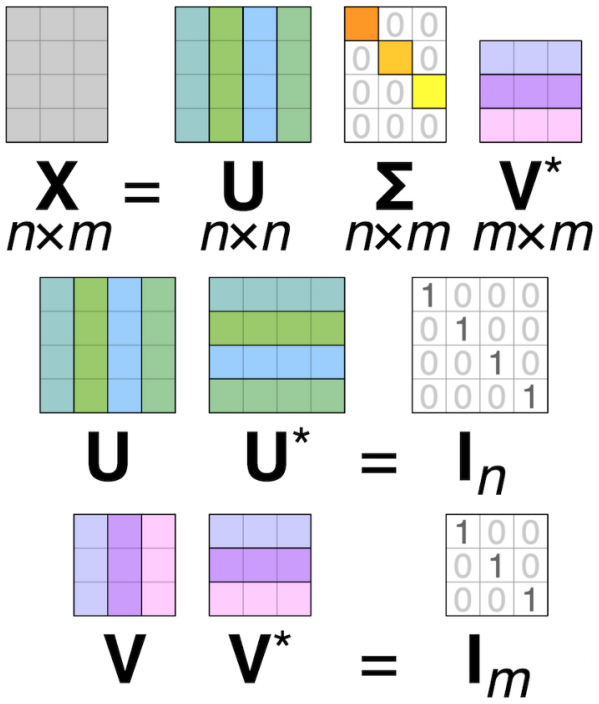
Es decir: Unidades-PC PC-Unidades

Veamoslo gráficamente, ¿qué pasa con esa selección de los PCs y su efecto?.

Para ello usaremos Singular Value Descomposition (SVD).

Singular Value Descomposition(SVD)

Es otra descomposición que tambien nos ayudara a reducir dimensiones.



#Ejercicio 2 Juega con Lucy, una cisne, ayudala a encontrar cuantos valores singulares necesita para no perder calidad a través de SVD. Posteriormente usa 3 imágenes de tu preferencia y realiza la misma acción :D

A esto se le llama **compresión de imagenes** :o

Type *Markdown* and LaTeX: α^2

In [ ]:

```
from six.moves import urllib
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('classic')
img = Image.open(urllib.request.urlopen('https://biblioteca.acropolis.org/wp
#img = Image.open('lucy.jpg')
imggray = img.convert('LA')
imgmat = np.array(list(imggray.getdata(band=0)),float)

print(imgmat)

imgmat.shape = (imggray.size[1],imggray.size[0])

plt.figure(figsize=(9,6))
plt.imshow(imgmat,cmap='gray')
plt.show()
print(img)
```

```
[72. 73. 74. ... 48. 47. 47.]
```

```
<PIL.Image.Image image mode=LA size=1024x660 at 0x7F72A186E450>
```

In [ ]:
```
1 U,D,V = np.linalg.svd(imgmat)
2 imgmat.shape
3
```

Out[7]:
```
(660, 1024)
```

In [ ]:
```
1 U.shape
```

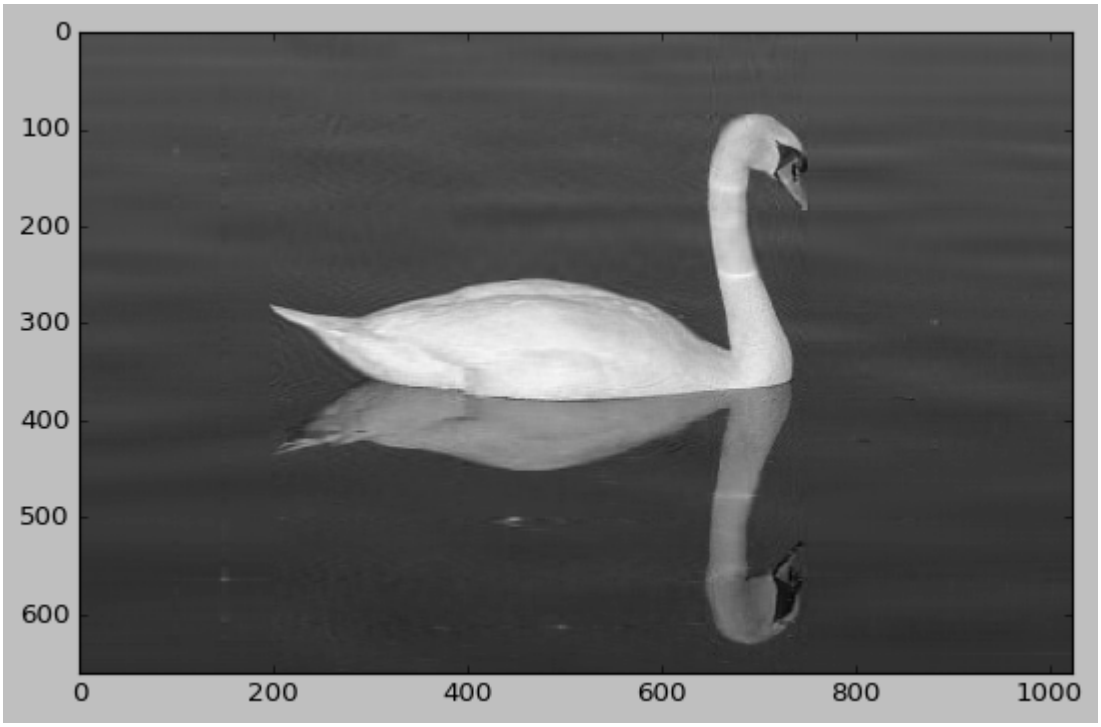
Out[8]:
```
(660, 660)
```

In [ ]:
```
1 V.shape
```

Out[9]:
```
(1024, 1024)
```

In [ ]:

```
#Cuantos valores crees que son necesarios?
#A=U*D*V
#aqui los elegiremos--------------------------
# por las dimensiones de este caso en particular
#iremos de 0-660, siendo 660 como normalmente están los datos
#con 50 podemos observar que Lucy se ve casi igual, es decir conservamos aqu
# realidad estaba aportando a la imagen en este caso :D por medio de la vari
#juega con el valor nvalue y ve que pasa con otros valores
nvalue = 50
#------------------------------
reconstimg = np.matrix(U[:,:nvalue])*np.diag(D[:nvalue])*np.matrix(V[:nvalue
#ve las dimensiones de la imagen y su descomposicion
#660x1024= U(660X660)D(660X1024)V(1024x1024)
        #=U(660Xnvalues)D(nvaluesXnvalue)V(nvaluesx1024)

        #=U(660X50)(50X50)(50X1024)
plt.imshow(reconstimg,cmap='gray')
plt.show()
print("Felicidades la imagen está comprimida")
```




```
Felicidades la imagen está comprimida
```

¡Ahora es tu turno!, comprime 3 imagenes

In [ ]:

```
#imagen 1

plt.style.use('classic')
img = Image.open(urllib.request.urlopen('https://biblioteca.acropolis.org/wp
imggray = img.convert('LA')
imgmat = np.array(list(imggray.getdata(band=0)),float)

print(imgmat)

imgmat.shape = (imggray.size[1],imggray.size[0])

plt.figure(figsize=(9,6))
plt.imshow(imgmat,cmap='gray')
plt.show()
print(img)
```

```
[ 16.  17.  17. ... 151. 148. 147.]
```


```
<PIL.Image.Image image mode=LA size=1200x800 at 0x7F729DC24CD0>
```

In [ ]:
```
U,D,V = np.linalg.svd(imgmat)
imgmat.shape
```

Out[12]: (800, 1200)

In [ ]:
```
U.shape
```

Out[13]: (800, 800)

In [ ]:
```
V.shape
```

Out[14]: (1200, 1200)

In [ ]:
```
#A=U*D*V
nvalue = 70
#-----------------------------
reconstimg = np.matrix(U[:,:nvalue])*np.diag(D[:nvalue])*np.matrix(V[:nvalue
#ve las dimensiones de la imagen y su descomposicion
#800x1200= U(800,800)D(800X1200)V(1200,1200)
        #=U(800Xnvalues)D(nvaluesXnvalue)V(nvaluesx1200)

        #=U(800X70)(70X70)(70X1200)
plt.imshow(reconstimg,cmap='gray')
plt.show()
print("Felicidades la imagen está comprimida")
```


```
Felicidades la imagen está comprimida
```

In [ ]:

```
#imagen 2

plt.style.use('classic')
img = Image.open(urllib.request.urlopen('https://biblioteca.acropolis.org/wp
imggray = img.convert('LA')
imgmat = np.array(list(imggray.getdata(band=0)),float)

print(imgmat)

imgmat.shape = (imggray.size[1],imggray.size[0])

plt.figure(figsize=(9,6))
plt.imshow(imgmat,cmap='gray')
plt.show()
print(img)
```

```
[34. 34. 34. ... 88. 92. 95.]
```


```
<PIL.Image.Image image mode=LA size=1200x800 at 0x7F729DC24B50>
```

In [ ]:

```
U,D,V = np.linalg.svd(imgmat)
imgmat.shape
```

Out[17]:

```
(800, 1200)
```

In [ ]:
```
1 U.shape
```

Out[18]: (800, 800)

In [ ]:
```
1 V.shape
```

Out[19]: (1200, 1200)

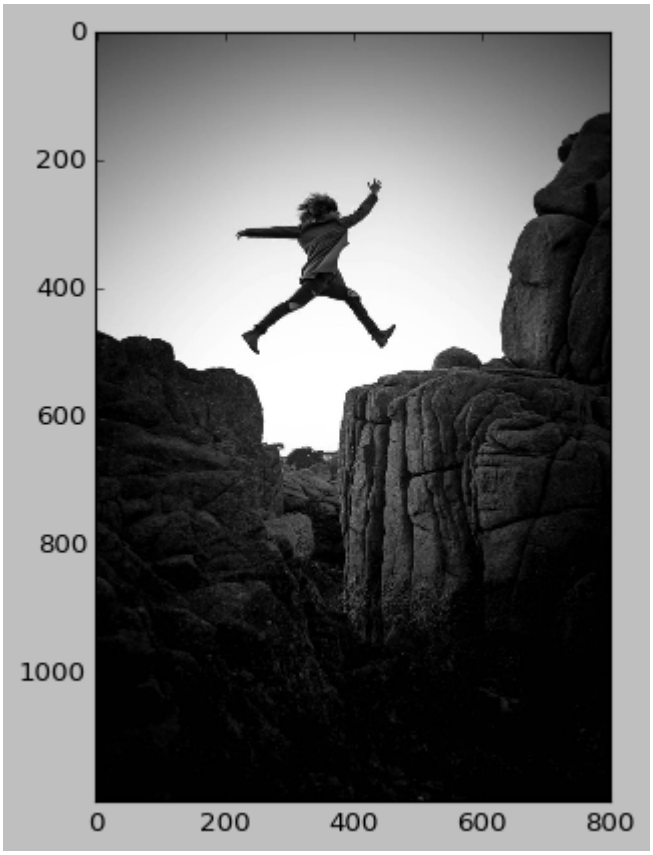
In [ ]:
```
1  nvalue = 40
2  #------------------------------
3  reconstimg = np.matrix(U[:,:nvalue])*np.diag(D[:nvalue])*np.matrix(V[:nvalue
4  #ve las dimensiones de la imagen y su descomposicion
5  #800x1200= U(800,800)D(800X1200)V(1200,1200)
6          #=U(660Xnvalues)D(nvaluesXnvalue)V(nvaluesx1024)
7
8          #=U(800X40)(40X40)(40X1200)
9  plt.imshow(reconstimg,cmap='gray')
10 plt.show()
11 print("Felicidades la imagen está comprimida")
```


```
Felicidades la imagen está comprimida
```

In [ ]:

```
#imagen 3
plt.style.use('classic')
img = Image.open(urllib.request.urlopen('https://biblioteca.acropolis.org/wp
#img = Image.open('lucy.jpg')
imggray = img.convert('LA')
imgmat = np.array(list(imggray.getdata(band=0)),float)

print(imgmat)

imgmat.shape = (imggray.size[1],imggray.size[0])

plt.figure(figsize=(9,6))
plt.imshow(imgmat,cmap='gray')
plt.show()
print(img)


```

```
[71. 71. 71. ...  1.  1.  1.]
```



```
<PIL.Image.Image image mode=LA size=801x1200 at 0x7F729F160950>
```

In [ ]:
```
1 U,D,V = np.linalg.svd(imgmat)
2 imgmat.shape
```

Out[27]: (1200, 801)

In [ ]:
```
1 U.shape
```

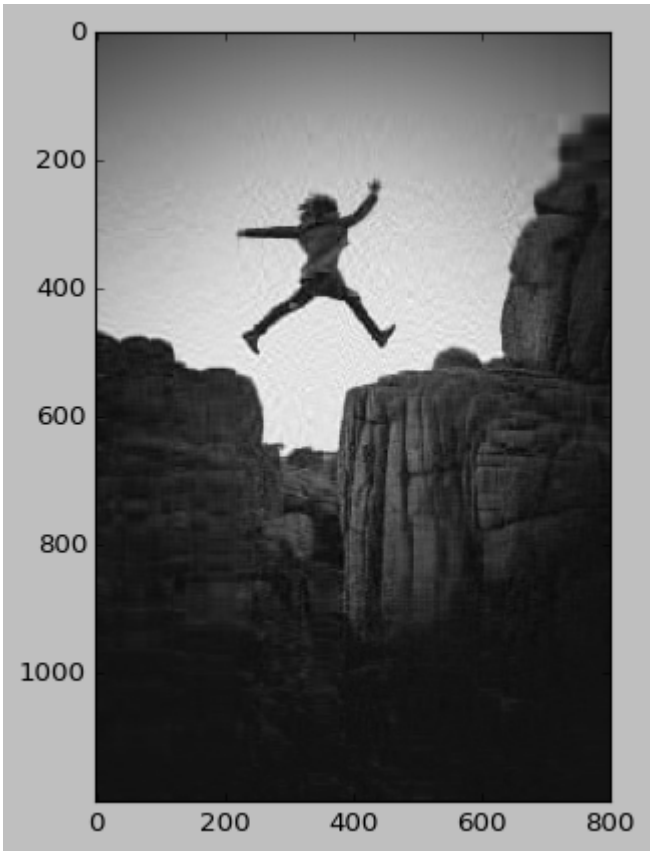
Out[28]: (1200, 1200)

In [ ]:
```
1 V.shape
```

Out[29]: (801, 801)

In [ ]:

```
1
2  nvalue = 40
3  #------------------------------
4  reconstimg = np.matrix(U[:,:nvalue])*np.diag(D[:nvalue])*np.matrix(V[:nvalue
5  #ve las dimensiones de la imagen y su descomposicion
6  #800x1200= U(800, 800)D(800X1200)V(1200, 1200)
7          #=U(800Xnvalues)D(nvaluesXnvalue)V(nvaluesx1200)
8
9          #=U(800X40)(40X40)(40X1200)
10 plt.imshow(reconstimg,cmap='gray')
11 plt.show()
12 print("Felicidades la imagen está comprimida")
```




```
Felicidades la imagen está comprimida
```

Ejercicio 3

Feature importances

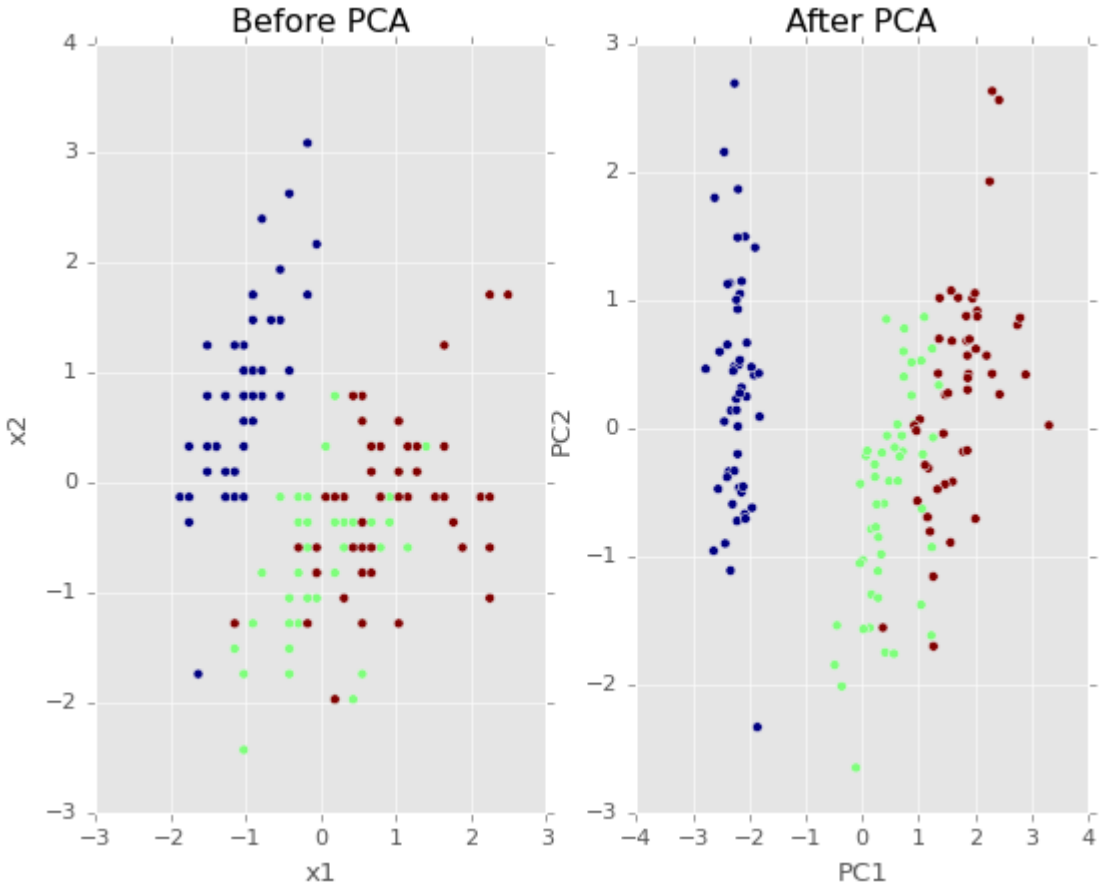
Para este ejercicio, te pediremos que sigas el tutorial de la siguiente pagina:

https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e (https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e)

In [ ]:

```python
#tu codigo aqui
#Se vió el siguiente código en el ejercicio
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler
plt.style.use('ggplot')
# Load the data
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Z-score the features
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
# The PCA model
pca = PCA(n_components=2) # estimate only 2 PCs
X_new = pca.fit_transform(X) # project the original data into the PCA space
```

In [ ]:
```
fig, axes = plt.subplots(1,2)
axes[0].scatter(X[:,0], X[:,1], c=y)
axes[0].set_xlabel('x1')
axes[0].set_ylabel('x2')
axes[0].set_title('Before PCA')
axes[1].scatter(X_new[:,0], X_new[:,1], c=y)
axes[1].set_xlabel('PC1')
axes[1].set_ylabel('PC2')
axes[1].set_title('After PCA')
plt.show()
```



In [ ]:
```
print(pca.explained_variance_ratio_)

```

```
[0.72962445 0.22850762]
```

In [ ]:
```
np.cov(X_new.T)

```

Out[39]:
```
array([[2.93808505e+00, 5.33928780e-16],
       [5.33928780e-16, 9.20164904e-01]])
```

In [ ]:
```
pca.explained_variance_

```

Out[38]:
```
array([2.93808505, 0.9201649 ])
```

In [ ]:
```
print(abs( pca.components_ ))

```

```
[[0.52106591 0.26934744 0.5804131  0.56485654]
 [0.37741762 0.92329566 0.02449161 0.06694199]]
```

¿Qué es feature importance y para que nos sirve? la importancia de cada 'feature' está reflejada por la magnitud de los valores correspondientes en los eigenvectores (una más alta magnitud implica una mayor importancia) En resumen, se buscan los valores absolutos de los eigenvectores de los componentes correspondientes a los k mayores eigenvalores. A medida que estos valores absolutos sean mayores, mayor es la contribución de un feature a un componente principal

¿Qué hallazgos fueron los más relevantes durante el análisis del ejercicio? Se realizó una explicación de qué son y que función tienen el análisis de componentes principales y el análisis de SVD

¿Dónde lo aplicarías o te sería de utilidad este conocimiento? Este análisis se puede emplear de manera no supervisada, para realizar una reducción de dimensionalidad, en caso de que se cuenten con muchas variables 'features'. Estos se puede realizar para acelerar el entrenamiento de un modelo. también para visuzalización de datos

Describe lo relevante del ejercicio y que descubriste de las variables análizadas.

Se muestra paso a paso, el procedimiento para realizar el análisis de componentes principales (PCA), de detallan sus aplicaciones para machine learning, también se explora su aplicación en python a través de la librería skilearn.