



**UNIVERSIDAD
CATÓLICA**
SEDES SAPIENTIAE

Inteligencia Artificial

Ing. Juancarlos Santana Huamán
jsantana@ucss.edu.pe



Análisis de componentes principales (Principal Component Analysis PCA)

- El PCA (Análisis de Componentes Principales) es una técnica de reducción de dimensionalidad utilizada en el análisis de datos y el aprendizaje automático. Permite reducir el número de características en un conjunto de datos, conservando la información más importante. Transforma las características originales en nuevas características; estas nuevas características no se superponen entre sí y las primeras conservan la mayoría de las diferencias importantes encontradas en los datos originales.
- El PCA se utiliza comúnmente para el preprocesamiento de datos con algoritmos de aprendizaje automático. Ayuda a eliminar la redundancia, mejorar la eficiencia computacional y facilitar la visualización y el análisis de datos, especialmente al trabajar con datos de alta dimensión.



Cómo funciona el análisis de componentes principales

- El análisis de componentes principales (PCA) utiliza álgebra lineal para transformar los datos en nuevas características llamadas componentes principales. Estos se obtienen calculando vectores propios (direcciones) y valores propios (importancia) a partir de la matriz de covarianza. El PCA selecciona los componentes principales con los valores propios más altos y proyecta los datos sobre ellos para simplificar el conjunto de datos.
Nota: Prioriza las direcciones donde los datos varían más porque mayor variación = más información útil.
- Imagina que observas una nube desordenada de puntos de datos, como estrellas en el cielo, y quieres simplificarla. El análisis de componentes principales (PCA) te ayuda a encontrar los ángulos más importantes para observar esta nube y así no perderte los patrones principales.



Cómo funciona el análisis de componentes principales

Paso 1: Estandarizar los datos

- Diferentes características pueden tener unidades y escalas diferentes, como el salario frente a la edad. Para compararlas de forma justa, el PCA primero estandariza los datos, haciendo que cada característica tenga:

- Una media de 0
- Una desviación estándar de 1

$$Z = \frac{\text{incógnita} - \text{micras}}{\sigma}$$

- Dónde:
 - micras es la media de características independientes:
 $\text{micras} = \{\text{micras}_1, \text{micras}_2, \dots, \text{micras}_{\text{metro}}\}$
 - σ es la desviación estándar de las características independientes:
 $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{\text{metro}}\}$



Cómo funciona el análisis de componentes principales

Paso 2: Calcular la matriz de covarianza

- A continuación, el PCA calcula la matriz de covarianza para ver cómo se relacionan las características entre sí, independientemente de si aumentan o disminuyen juntas. La covarianza entre dos características...*incógnita*₁ y *incógnita*₂ es:

$$\text{cov}(x_1, x_2) = \frac{\sum_{i=1}^{\text{norte}} (x1_i - \overline{x1})(x2_i - \overline{x2})}{n - 1}$$

- Dónde:
 - $\overline{\text{incognita}_1}$ y $\overline{\text{incognita}_2}$ son los valores medios de las características *incongnita*₁ y *incongnita*₂.
 - *norte* es el número de puntos de datos
- El valor de la covarianza puede ser positivo, negativo o cero.



Cómo funciona el análisis de componentes principales

Paso 3: Encuentra los componentes principales

- PCA identifica **nuevos ejes** donde los datos se dispersan más:
 - **Primer componente principal (CP1)**: La dirección de máxima varianza (mayor dispersión).
 - **2do componente principal (PC2)**: La siguiente mejor dirección, *perpendicular a PC1* y así sucesivamente.
- Estas direcciones provienen de los **vectores propios** de la matriz de covarianza y su importancia se mide mediante **los valores propios**. Para una matriz cuadrada A , un **vector propio** X (un vector distinto de cero) y su **valor propio** correspondiente λ satisfacen:

$$A X = \lambda X$$

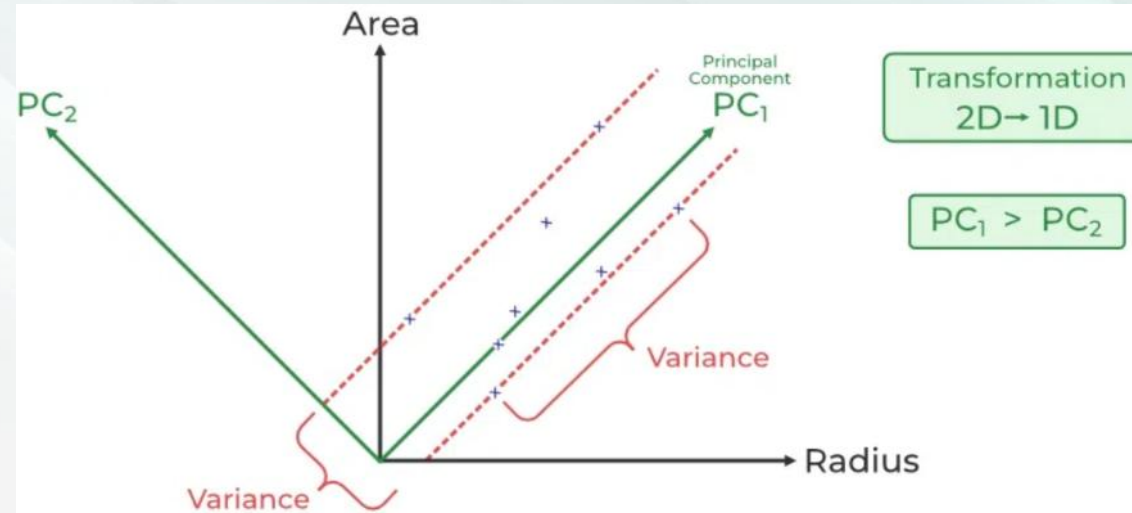
- Esto significa:
 - Cuando A actúa sobre X , solo estira o encoge X según el escalar λ .
 - La dirección de X permanece sin cambios, por lo tanto, los vectores propios definen "direcciones estables" de A .
- Los valores propios ayudan a clasificar estas direcciones por importancia.



Cómo funciona el análisis de componentes principales

Paso 4: Seleccione las direcciones principales y transforme los datos

- Tras calcular los autovalores y vectores propios, el análisis de componentes principales (PCA) los clasifica según la cantidad de información que capturan. A continuación:
- **Seleccione los k componentes principales** que capturan la mayor parte de la variación, como el 95 %.
- **Transforme el conjunto de datos original** proyectándolo sobre estos componentes superiores.
- Esto significa que reducimos el número de características (dimensiones) mientras mantenemos los patrones importantes en los datos.



Cómo funciona el análisis de componentes principales

- En la imagen superior, el conjunto de datos original presenta dos características, "Radio" y "Área", representadas por los ejes negros. El análisis de componentes principales (PCA) identifica dos nuevas direcciones: **PC₁** y **PC₂**, que son los **componentes principales**.
- Estos nuevos ejes son versiones rotadas de los originales. **PC₁** captura la máxima varianza de los datos, lo que significa que contiene la mayor cantidad de información, mientras que **PC₂** captura la varianza restante y es perpendicular a PC₁.
- La dispersión de datos es mucho mayor a lo largo de PC₁ que a lo largo de PC₂. Por ello, se elige PC₁ para la reducción de dimensionalidad. Al proyectar los puntos de datos (cruces azules) en PC₁, **transformamos eficazmente los datos 2D en 1D** y conservamos la mayor parte de la estructura y los patrones importantes.



Implementación del análisis de componentes principales en Python

- Por lo tanto, el PCA utiliza una transformación lineal que se basa en preservar la máxima varianza en los datos utilizando el menor número de dimensiones. Implica los siguientes pasos:

Paso 1: Importar las bibliotecas necesarias

- Importamos las bibliotecas necesarias como pandas , numpy , scikit learn , seaborn y matplotlib para visualizar los resultados.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```



Implementación del análisis de componentes principales en Python

Paso 2: Creación de un conjunto de datos de muestra

- Creamos un pequeño conjunto de datos con tres características: Altura, Peso, Edad y Género.

```
data = {  
    'Height': [170, 165, 180, 175, 160, 172, 168, 177, 162, 158],  
    'Weight': [65, 59, 75, 68, 55, 70, 62, 74, 58, 54],  
    'Age': [30, 25, 35, 28, 22, 32, 27, 33, 24, 21],  
    'Gender': [1, 0, 1, 1, 0, 1, 0, 1, 0, 0] # 1 = Male, 0 = Female  
}  
df = pd.DataFrame(data)  
print(df)
```

	Height	Weight	Age	Gender
0	170	65	30	1
1	165	59	25	0
2	180	75	35	1
3	175	68	28	1
4	160	55	22	0
5	172	70	32	1
6	168	62	27	0
7	177	74	33	1
8	162	58	24	0
9	158	54	21	0



Implementación del análisis de componentes principales en Python

Paso 3: Estandarización de los datos

- Dado que las características tienen diferentes escalas (altura y edad), estandarizamos los datos. Esto hace que todas las características tengan una media de 0 y una desviación estándar de 1, de modo que ninguna característica predomine simplemente por sus unidades.

```
X = df.drop('Gender', axis=1)
y = df['Gender']
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)
```



Implementación del análisis de componentes principales en Python

Paso 4: Aplicación del algoritmo PCA

- Reducimos los datos de tres características a dos nuevas características llamadas componentes principales. Estos componentes capturan la mayor parte de la información original, pero en menos dimensiones.
- Dividimos los datos en un 70% de conjuntos de entrenamiento y un 30% de conjuntos de prueba.
- Entrenamos un modelo de regresión logística en los datos de entrenamiento reducidos y predecimos etiquetas de género en el conjunto de prueba.

```
pca = PCA(n_components=2)  
X_pca = pca.fit_transform(X_scaled)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)
```

```
model = LogisticRegression()  
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```



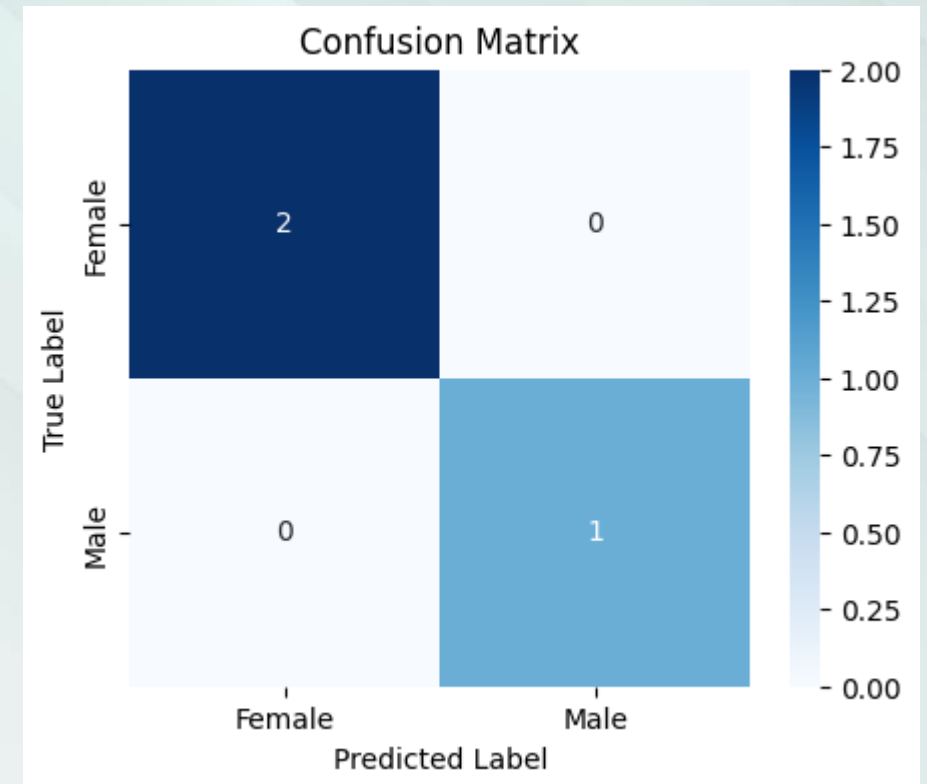
Implementación del análisis de componentes principales en Python

Paso 5: Evaluación con la Matriz de Confusión

- La matriz de confusión compara las etiquetas reales con las predichas. Esto facilita identificar dónde las predicciones fueron correctas o incorrectas.

```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Female', 'Male'], yticklabels=['Female', 'Male'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Implementación del análisis de componentes principales en Python

Paso 6: Visualización del resultado del PCA

```
y_numeric = pd.factorize(y)[0]

plt.figure(figsize=(12, 5))

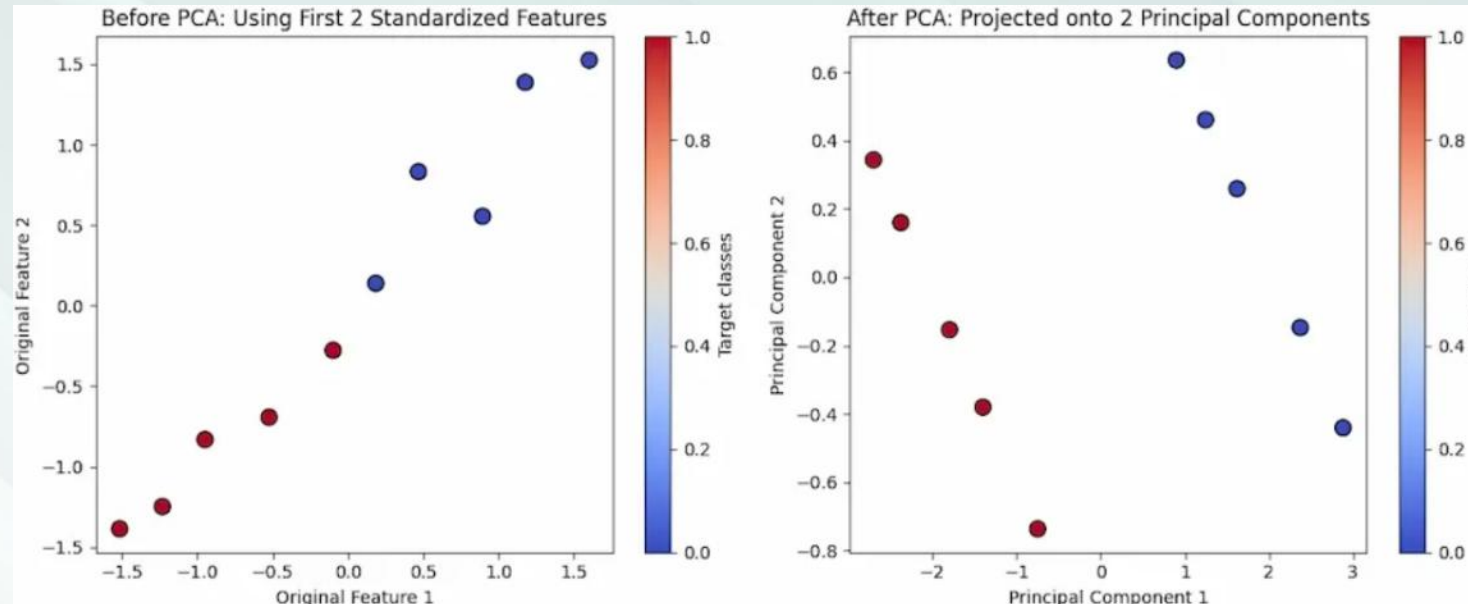
plt.subplot(1, 2, 1)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_numeric, cmap='coolwarm', edgecolor='k', s=80)
plt.xlabel('Original Feature 1')
plt.ylabel('Original Feature 2')
plt.title('Before PCA: Using First 2 Standardized Features')
plt.colorbar(label='Target classes')

plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_numeric, cmap='coolwarm', edgecolor='k', s=80)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('After PCA: Projected onto 2 Principal Components')
plt.colorbar(label='Target classes')

plt.tight_layout()
plt.show()
```



Implementación del análisis de componentes principales en Python



- **Gráfico izquierdo antes del PCA:** Muestra los datos estandarizados originales, graficados utilizando las dos primeras características. No se garantiza una separación clara entre clases, ya que se trata de dimensiones de entrada sin procesar.
- **Gráfico derecho después del PCA:** Muestra los datos transformados utilizando los dos componentes principales. Estos nuevos componentes capturan la varianza máxima, mostrando a menudo una mejor separación y estructura de clases, lo que facilita el análisis o modelado.

Implementación del análisis de componentes principales en Python

Ventajas del análisis de componentes principales

1. **Manejo de multicolinealidad:** crea nuevas variables no correlacionadas para abordar problemas cuando las características originales están altamente correlacionadas.
2. **Reducción de ruido:** elimina componentes con baja variación y mejora la claridad de los datos.
3. **Compresión de datos:** Representa datos con menos componentes que reducen las necesidades de almacenamiento y aceleran el procesamiento.
4. **Detección de valores atípicos:** identifica puntos de datos inusuales mostrando cuáles se desvían significativamente en el espacio reducido.



Implementación del análisis de componentes principales en Python

Desventajas del análisis de componentes principales

1. **Desafíos de interpretación:** Los nuevos componentes son combinaciones de variables originales que pueden ser difíciles de explicar.
2. **Sensibilidad de escalamiento de datos:** requiere un escalamiento adecuado de los datos antes de la aplicación o los resultados pueden ser engañosos.
3. **Pérdida de información:** al reducir las dimensiones se puede perder información importante si se mantienen muy pocos componentes.
4. **Supuesto de linealidad:** funciona mejor cuando las relaciones entre las variables son lineales y puede presentar dificultades con datos no lineales.
5. **Complejidad computacional:** puede ser lento y consumir muchos recursos en conjuntos de datos muy grandes.
6. **Riesgo de sobreajuste:** utilizar demasiados componentes o trabajar con un conjunto de datos pequeño puede generar modelos que no se generalicen bien.





Análisis de componentes independientes (ICA)

- El Análisis de Componentes Independientes (ICA) es una técnica potente y versátil para el análisis de datos, que ofrece una perspectiva única para la exploración y extracción de patrones ocultos en conjuntos de datos complejos. En esencia, el ICA es un método de procesamiento de señales que busca separar un conjunto de señales mixtas en componentes estadísticamente independientes, lo que proporciona información valiosa y aplicaciones en diversos dominios.

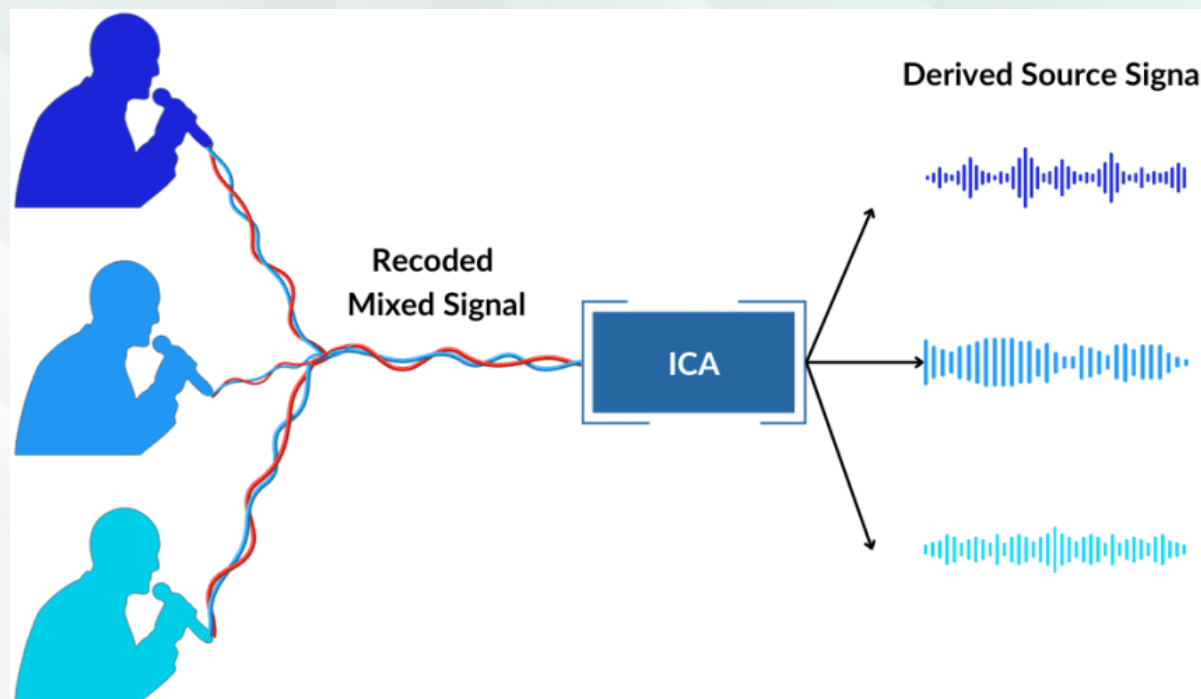
La importancia del análisis de componentes independientes (ICA)

- En un mundo inundado de datos, la capacidad de descubrir información significativa oculta en un mar de ruido es fundamental. El ICA desempeña un papel fundamental en este esfuerzo, permitiéndonos identificar las fuentes subyacentes de los datos observados, incluso cuando no se comprende completamente el proceso de mezcla. Ya sea en el procesamiento de imágenes, el reconocimiento de voz, las finanzas o la investigación médica, el ICA es una herramienta fundamental para desentrañar estructuras de datos complejas.



El problema de la separación ciega de fuentes

- Una característica distintiva del ICA es su enfoque en la separación ciega de fuentes. Este problema surge cuando solo tenemos acceso a las señales mixtas y necesitamos aplicar ingeniería inversa a las fuentes sin conocer previamente sus propiedades. Esta desafiante tarea es similar a resolver un rompecabezas con piezas que podrían haber sido reorganizadas o transformadas, lo que convierte al ICA en una esperanza para investigadores y analistas que se enfrentan a estos escenarios.



Comprensión del análisis de componentes independientes (ICA)

Asunción de la Independencia

- El Análisis de Componentes Independientes se basa en la suposición de que las señales observadas son una combinación lineal de señales fuente estadísticamente independientes. Esta suposición es crucial, ya que permite al ICA aprovechar las dependencias estadísticas presentes en los datos para recuperar las fuentes.

Modelo de mezcla lineal

- El ICA se basa en el modelo de mezcla lineal, que se puede representar como $X = AS$ Donde:
 - X son los datos observados (señales mixtas).
 - A es la matriz de mezcla, que describe cómo se combinan las fuentes.
 - S es la matriz fuente, que representa los componentes independientes que se pretenden extraer. El objetivo del ICA es estimar la matriz de mezcla A y recuperar la matriz fuente S .



ICA vs. PCA: Diferencias clave

- Ortogonalidad vs. Independencia. El Análisis de Componentes Principales (ACP) y el Análisis de Componentes Independientes (ACI) se comparan a menudo, pero sus principios fundamentales difieren significativamente. El ACP busca componentes ortogonales (no correlacionados) en los datos, mientras que el ACI busca componentes estadísticamente independientes. Esta distinción hace que el ACI sea ideal para descubrir fuentes ocultas en datos mixtos.

Separación ciega de fuentes

- El ICA se utiliza principalmente para la separación ciega de fuentes, lo que significa que puede gestionar situaciones en las que el número de fuentes y sus propiedades estadísticas se desconocen a priori. Por el contrario, el PCA se utiliza para la reducción de dimensionalidad y la decorrelación sin centrarse en la separación de fuentes.





Aplicaciones del análisis de componentes independientes (ICA)

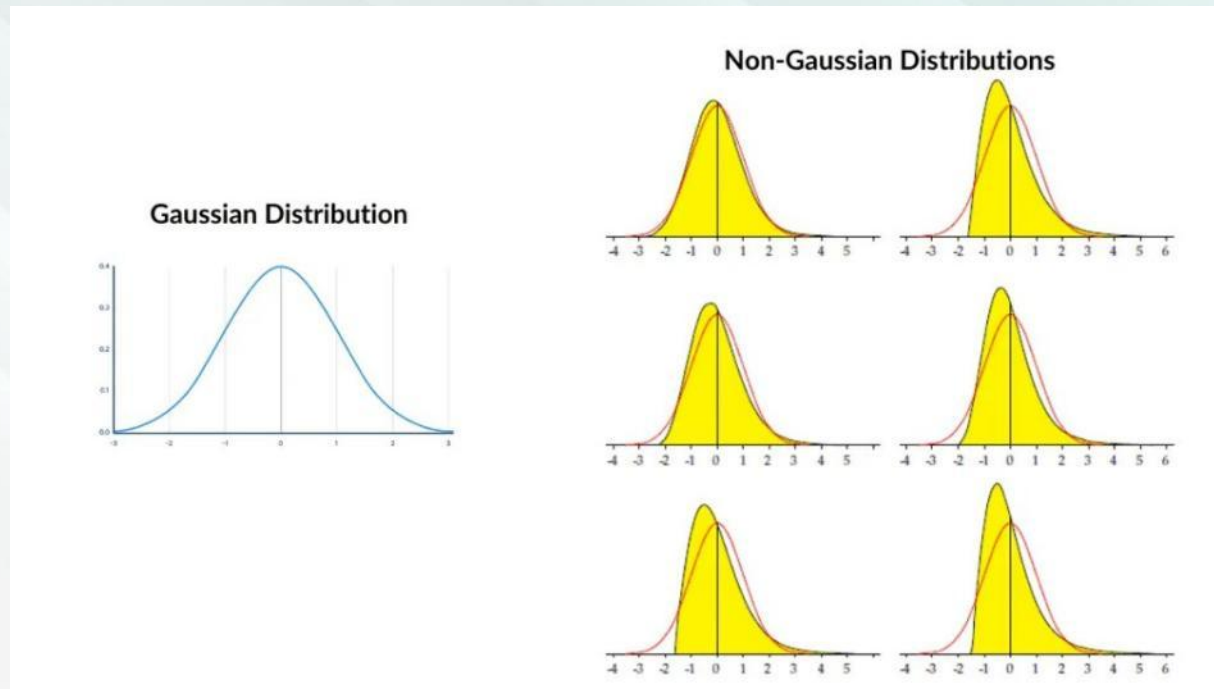
- **Procesamiento de imágenes:** en el procesamiento de imágenes, se puede emplear ICA para separar una imagen en sus componentes subyacentes, como texturas, patrones o incluso las fuentes de ruido, lo que permite aplicaciones como mejora de imágenes, eliminación de ruido y reconocimiento de objetos.
- **Separación de Señales:** El ICA se utiliza ampliamente en el procesamiento de señales de audio para separar fuentes de audio mixtas en situaciones como el problema de las fiestas de cóctel. Permite recuperar voces o instrumentos individuales de una mezcla de señales de sonido.
- **Imágenes médicas:** La ICA es crucial en las imágenes médicas, en particular en la resonancia magnética funcional (fMRI). Ayuda a separar y analizar los patrones de activación cerebral, lo que facilita el estudio de los procesos cognitivos y los trastornos neurológicos.
- **Análisis de datos financieros:** en finanzas, el ICA se utiliza para identificar factores latentes que influyen en los movimientos de los precios de las acciones, lo que ayuda a los comerciantes y analistas a descubrir anomalías y tendencias ocultas del mercado.
- Comprender estos conceptos básicos y las diferencias entre el ICA y el PCA es esencial para comprender los fundamentos del Análisis de Componentes Independientes. En las siguientes secciones, exploraremos los fundamentos matemáticos del ICA y las técnicas utilizadas para extraer componentes independientes de datos mixtos.



Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

Distribuciones gaussianas y no gaussianas

- El Análisis de Componentes Independientes asume que las señales fuente son estadísticamente independientes. Para facilitar esta independencia, el ICA funciona mejor cuando las fuentes siguen distribuciones de probabilidad no gaussianas. La no gaussianidad es crucial porque implica falta de linealidad y permite al ICA aprovechar la independencia estadística.



Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

Estimación de máxima verosimilitud

- El objetivo de la ICA es encontrar la matriz de mezcla óptima A y la matriz de fuentes S que maximizan la probabilidad de los datos observados X , dadas las fuentes estimadas S . Esto se puede expresar como:
- $P(X|A,S)$
- El ICA busca encontrar A y S que maximicen esta función de verosimilitud. Esto se logra mediante diversos algoritmos y técnicas que refinan iterativamente las estimaciones de A y S para aumentar la verosimilitud de los datos observados.



Fundamentos matemáticos del análisis de componentes independientes (ICA) - Distribuciones de probabilidad

Encontrar componentes independientes

- **Blanqueo.** Uno de los pasos iniciales del ICA consiste en blanquear los datos observados. Este blanqueamiento transforma los datos en un espacio donde los componentes no están correlacionados y presentan varianzas unitarias. Esto simplifica el problema del ICA y lo hace más fácil de separar.
- **Algoritmo de deflación.** El algoritmo de deflación es un componente crucial del ICA. Extrae los componentes independientes uno a uno. Cada iteración estima y elimina un componente de los datos, lo que facilita la estimación de los componentes subsiguientes. El proceso continúa hasta que se extraen todos los componentes independientes.



Los 3 principales algoritmos y técnicas de análisis de componentes independientes (ICA)

FastICA. Es un algoritmo ICA popular y eficiente que busca encontrar componentes independientes maximizando la no gaussianidad. Se basa en un esquema de iteración de punto fijo y puede trabajar con mezclas lineales y no lineales.

- Características principales
 - FastICA maximiza la negentropía, una medida de no gaussianidad, para separar componentes independientes.
 - Emplea un algoritmo de deflación para extraer componentes uno a uno.
 - FastICA es computacionalmente eficiente y se utiliza ampliamente en diversas aplicaciones, incluido el procesamiento de imágenes y voz.

Infomax. Es un algoritmo ICA inspirado en la teoría de la información. Busca maximizar la información mutua entre los componentes independientes estimados. Se utiliza frecuentemente en aplicaciones como la separación ciega de fuentes y el entrenamiento de redes neuronales.

- Características principales
 - Infomax maximiza la información mutua, que mide la independencia estadística de los componentes.
 - Se basa en arquitecturas de redes neuronales y reglas de aprendizaje para ajustar iterativamente la matriz de mezcla y estimar componentes independientes.



Los 3 principales algoritmos y técnicas de análisis de componentes independientes (ICA)

JADE (Diagonalización aproximada conjunta de automatrices). Es un algoritmo ICA diseñado para funcionar con fuentes no gaussianas. Funciona diagonalizando conjuntamente las matrices cumulantes de cuarto orden de los datos observados. Este enfoque resulta conveniente en escenarios donde las fuentes presentan un comportamiento supergaussiano.

- Características principales
 - JADE es adecuado para separar fuentes con propiedades estadísticas de orden superior.
 - Proporciona una alternativa a los enfoques basados en curtosis y es robusto a la no gaussianidad.





Comparación de diferentes algoritmos

- Los algoritmos de ICA varían en sus enfoques, ventajas y limitaciones. La elección del algoritmo adecuado depende de las características específicas de los datos y de los objetivos del análisis. Al seleccionar un algoritmo, se deben considerar aspectos como la distribución de la fuente, la dimensionalidad de los datos y la eficiencia computacional.
- En la práctica, es habitual experimentar con diferentes algoritmos de ICA y comparar su rendimiento en el conjunto de datos dado. La elección del algoritmo puede influir significativamente en la calidad de los componentes independientes extraídos y en el éxito del análisis.
- Comprender estos algoritmos y técnicas de ICA es esencial para quienes aplican el ICA al análisis de datos reales. La selección del algoritmo más adecuado debe basarse en las características específicas de los datos y los objetivos del análisis. En la siguiente sección, profundizaremos en la implementación práctica del ICA, incluyendo el preprocesamiento de datos, la interpretación de componentes y los errores comunes que se deben evitar.



Preprocesamiento de datos

- **Centrado de datos:** Antes de aplicar el ICA, es fundamental centrar los datos restando la media de cada variable. Esto garantiza que la matriz de mezcla capture las relaciones entre las señales sin sesgos.
- **Blanqueamiento de datos:** El blanqueamiento de datos es un paso crucial que los transforma en un espacio donde los componentes no están correlacionados y presentan varianzas unitarias. El blanqueamiento simplifica el problema del ICA y lo hace más fácil de separar.
- **Reducción de dimensionalidad (opcional):** en conjuntos de datos de alta dimensión, las técnicas de reducción de dimensionalidad como el análisis de componentes principales (PCA) pueden reducir la complejidad computacional y al mismo tiempo preservar la información más relevante.





Software y bibliotecas para el análisis de componentes independientes (ICA)

- Existen diversos paquetes de software y bibliotecas para ICA, lo que lo hace accesible a una amplia gama de usuarios. Algunas herramientas populares incluyen MATLAB, scikit-learn de Python y bibliotecas independientes específicas para ICA, como FastICA. Estos recursos proporcionan implementaciones listas para usar de algoritmos de ICA, lo que reduce la necesidad de que los usuarios desarrollen su código desde cero.
- La implementación práctica del ICA implica un preprocesamiento cuidadoso de los datos, la selección del número adecuado de componentes, la interpretación de los resultados y la solución de problemas comunes. Es fundamental adaptar la implementación a las características específicas de los datos y a los objetivos del análisis. En la siguiente sección, exploraremos aplicaciones reales del ICA, demostrando su versatilidad en diversos ámbitos.





Implementar el análisis de componentes independientes (ICA) en Python

- Para realizar el Análisis de Componentes Independientes (ICA) en Python, se pueden usar diversas bibliotecas y paquetes que ofrecen implementaciones de ICA. Una de las bibliotecas más utilizadas para ICA en Python es `scikit-learn`, que ofrece una forma sencilla y práctica de realizarlo. Aquí tienes una guía paso a paso sobre cómo usar `scikit-learn` para aplicar ICA en Python:

`pip install scikit-learn`

```
from sklearn.decomposition import FastICA
import numpy as np
import matplotlib.pyplot as plt
```

```
# Generación de señales aleatoriamente
np.random.seed(0)
n_samples = 200
time = np.linspace(0, 8, n_samples)
s1 = np.sin(2 * time) # Señal 1
s2 = np.sign(np.sin(3 * time)) # Señal 2
s3 = np.random.randn(n_samples) # Señal 3
S = np.c_[s1, s2, s3] # Matriz combinada
A = np.array([[1, 1, 1], [0.5, 2, 1], [1.5, 1, 2]])
X = np.dot(S, A.T) # Señales combinadas
```

```
# Aplicación ICA
ica = FastICA(n_components=3)
independent_components = ica.fit_transform(X)
```

```
# Visualización de los componentes individuales
plt.figure(figsize=(12, 6))
```

```
plt.subplot(4, 1, 1)
plt.title("Señales Originales")
plt.plot(S)
```

```
plt.subplot(4, 1, 2)
plt.title("Señales Combinadas")
plt.plot(X)
```



Implementar el análisis de componentes independientes (ICA) en Python

```
plt.subplot(4, 1, 3)
plt.title("Componentes ICA")
plt.plot(independent_components)

plt.subplot(4, 1, 4)
plt.title("Señales Originales (Antes ICA)")
reconstructed_signals = np.dot(independent_components, A)
plt.plot(reconstructed_signals)

plt.tight_layout()
plt.show()
```





Análisis de componentes independientes (ICA) en el aprendizaje automático

- **Extracción de características:** El ICA puede utilizarse para extraer características independientes y significativas de datos de alta dimensión. Al transformar los datos en un nuevo espacio de características, el ICA puede descubrir patrones subyacentes y reducir la dimensionalidad de los datos, haciéndolos más manejables para los algoritmos de aprendizaje automático. Esto puede ser especialmente útil en tareas como el reconocimiento de imágenes, donde ayuda a identificar características clave.
- **Separación de fuentes ciegas:** El ICA se utiliza para la separación ciega de fuentes, cuyo objetivo es separar señales mixtas en sus fuentes independientes. En el aprendizaje automático, esto puede ser beneficioso para el procesamiento de audio y voz. Por ejemplo, el ICA puede separar las voces de varios hablantes en una grabación de audio, lo que facilita el análisis y la transcripción del habla.
- **Preprocesamiento de datos:** El ICA puede emplearse como paso de preprocesamiento para mejorar la calidad de los datos utilizados en el aprendizaje automático. Ayuda a eliminar ruido y artefactos, mejorando la relación señal-ruido y el rendimiento de los algoritmos de aprendizaje automático posteriores. Esto es especialmente útil en tareas como el análisis de imágenes médicas, donde la eliminación de artefactos es crucial para un diagnóstico preciso.
- **Detección de anomalías:** El ICA puede utilizarse para la detección de anomalías mediante la identificación de desviaciones de los patrones esperados. Se puede aplicar para detectar eventos inusuales en los datos, como la detección de fraudes en transacciones financieras o la detección de fallos en los procesos de fabricación. La capacidad del ICA para identificar componentes independientes lo hace ideal para detectar anomalías que podrían no ajustarse a los patrones típicos.





Análisis de componentes independientes (ICA) en el aprendizaje automático

- **Reducción de la multicolinealidad:** En el análisis de regresión y los modelos de aprendizaje automático, la multicolinealidad (alta correlación entre variables predictoras) puede afectar el rendimiento y la interpretabilidad del modelo. El ICA puede ayudar a reducir la multicolinealidad extrayendo componentes independientes que capturan información distinta de las características originales.
- **Mejorar la interpretabilidad de los datos:** El ICA puede facilitar la interpretación de los datos al extraer componentes más fáciles de comprender o analizar. Por ejemplo, en neurociencia, el ICA puede revelar fuentes cerebrales independientes asociadas con funciones cognitivas específicas, lo que facilita a los investigadores la interpretación de los resultados.
- **Clasificación y agrupamiento:** En ciertas aplicaciones, los datos transformados mediante ICA pueden incorporarse a algoritmos de aprendizaje automático para su clasificación o agrupamiento. Al reducir la dimensionalidad y centrarse en componentes independientes, ICA puede mejorar el rendimiento de estas tareas.

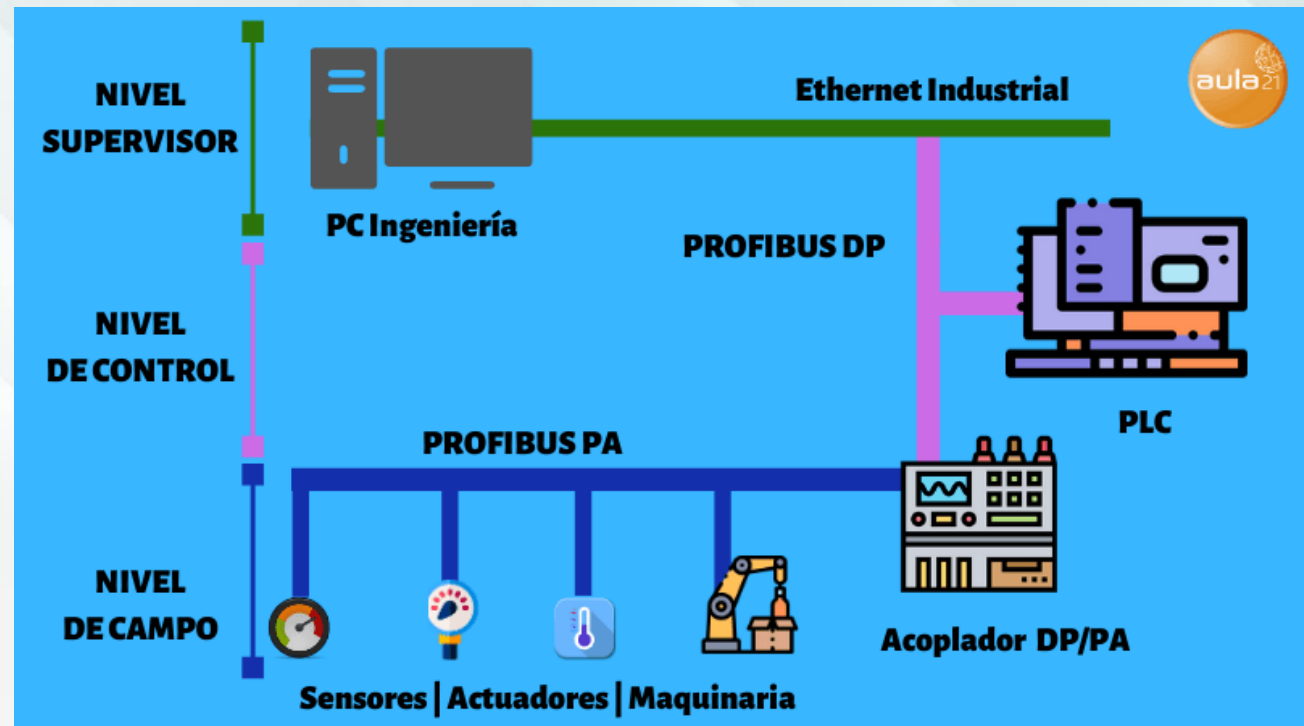
Es importante tener en cuenta que la decisión de usar ICA en el aprendizaje automático depende de la naturaleza de los datos y del problema específico en cuestión. Si bien ICA puede ser una herramienta potente para la extracción de características y el preprocesamiento de datos, no es una solución universal, y su idoneidad debe evaluarse en el contexto de la tarea de aprendizaje automático. Además, la interpretación de los componentes independientes suele ser específica de la aplicación y requiere conocimiento del dominio.





Ejemplo: Señales de Sensores

- Imaginemos que tenemos 3 sensores que registran señales en una fábrica. Cada sensor recibe una mezcla de:
 - Una señal de vibración de una máquina (fuente 1)
 - Una señal de interferencia eléctrica (fuente 2)
 - Ruido aleatorio (fuente 3)
- Nuestro objetivo es separar estas señales originales.





Usemos Python para Analizar

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA, FastICA

# Configuración
np.random.seed(42)
n_muestras = 1000
tiempo = np.linspace(0, 8, n_muestras)

# Señales originales (fuentes independientes)
s1 = np.sin(2 * tiempo) # Señal de máquina
s2 = np.sign(np.sin(3 * tiempo)) # Interferencia (señal cuadrada)
s3 = np.random.normal(0, 1, n_muestras) # Ruido aleatorio
```

```
# Matriz de fuentes (cada fila es una fuente)
S = np.c_[s1, s2, s3]

# Mezclamos las señales (lo que captan los sensores)
A = np.array([[0.5, 0.3, 0.2], # Matriz de mezcla
              [0.4, 0.6, 0.1],
              [0.2, 0.3, 0.8]])

# Datos observados (lo que miden los sensores)
X = np.dot(S, A.T)
```



Visualización de las Señales

```
plt.figure(figsize=(15, 10))

# Señales originales
plt.subplot(3, 1, 1)
for i in range(3):
    plt.plot(tiempo, S[:, i] + i*3, label=f'Fuente {i+1}')
plt.title('Señales Originales (Fuentes)')
plt.legend()

# Señales mezcladas (lo que ven los sensores)
plt.subplot(3, 1, 2)
for i in range(3):
    plt.plot(tiempo, X[:, i] + i*3, label=f'Sensor {i+1}')
plt.title('Señales Mezcladas (Observaciones)')
plt.legend()

plt.tight_layout()
plt.show()
```

Aplicación de PCA

```
# Aplicamos PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)

print("Componentes principales (PCA):")
print(pca.components_)
print(f"Varianza explicada: {pca.explained_variance_ratio_}")
```

Aplicación de ICA

```
# Aplicamos ICA
ica = FastICA(n_components=3, random_state=42)
X_ica = ica.fit_transform(X)

print("\nMatriz de separación (ICA):")
print(ica.components_)
```



```
plt.figure(figsize=(15, 12))

# Señales recuperadas por PCA
plt.subplot(3, 1, 1)
for i in range(3):
    plt.plot(tiempo, X_pca[:, i] + i*3, label=f'Componente PCA {i+1}')
plt.title('Señales Recuperadas por PCA')
plt.legend()

# Señales recuperadas por ICA
plt.subplot(3, 1, 2)
for i in range(3):
    plt.plot(tiempo, X_ica[:, i] + i*3, label=f'Componente ICA {i+1}')
plt.title('Señales Recuperadas por ICA')
plt.legend()

# Comparación con originales (normalizadas)
plt.subplot(3, 1, 3)
for i in range(3):
    # Normalizamos para comparar
    señal_norm = S[:, i] / np.std(S[:, i])
    plt.plot(tiempo, señal_norm + i*3, '--', alpha=0.7, label=f'Original {i+1}')
plt.title('Señales Originales (Referencia)')
plt.legend()

plt.tight_layout()
plt.show()
```

Comparación de Resultados



Métricas de evaluación

```
from sklearn.metrics import correlation_score
```

```
print("Correlación con señales originales:")
```

```
print("PCA vs Originales:")
```

```
for i in range(3):
```

```
    corr = np.corrcoef(X_pca[:, i], S[:, i])[0, 1]
```

```
    print(f"Componente {i+1}: {corr:.3f}")
```

```
print("\nICA vs Originales:")
```

```
for i in range(3):
```

```
    corr = np.corrcoef(X_ica[:, i], S[:, i])[0, 1]
```

```
    print(f"Componente {i+1}: {corr:.3f}")
```





Análisis de Resultados










Conclusiones

PCA (Análisis de Componentes Principales)

-  Reduce dimensionalidad manteniendo la máxima varianza
-  Ordena componentes por importancia (varianza explicada)
-  No separa perfectamente las fuentes originales
-  Asume distribución gaussiana

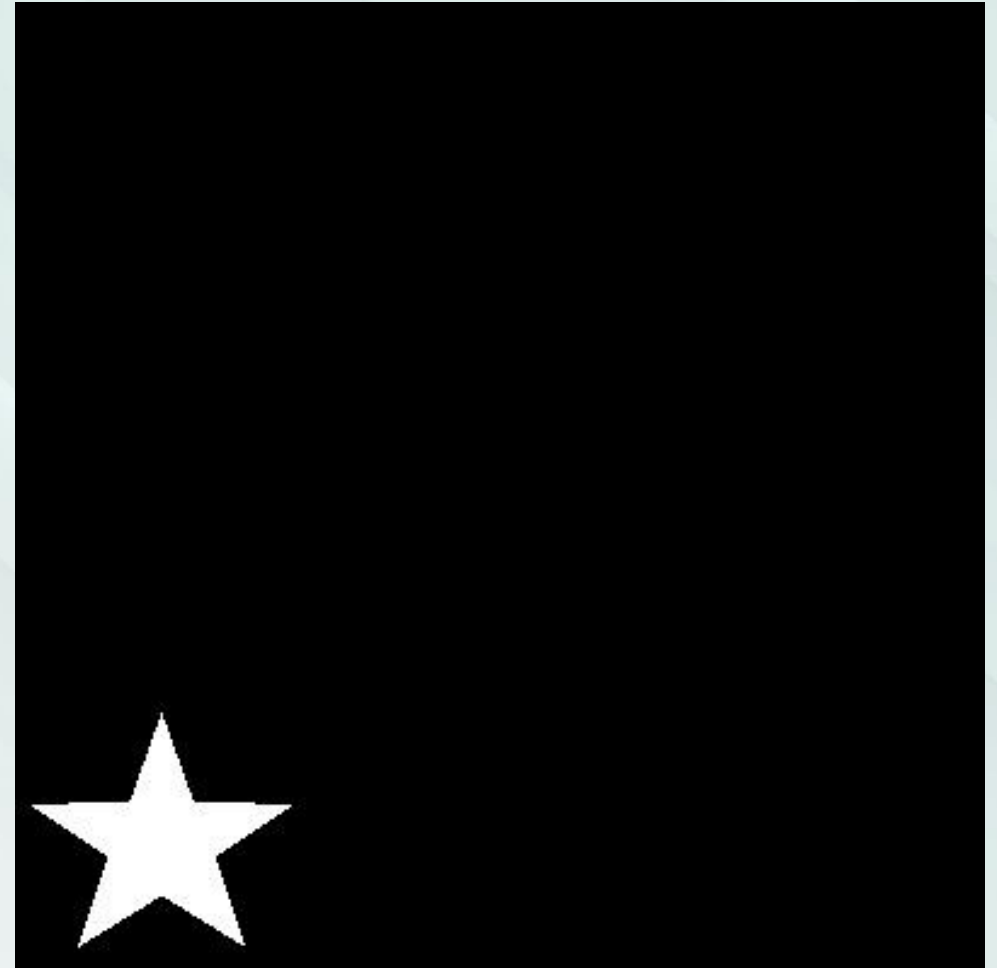
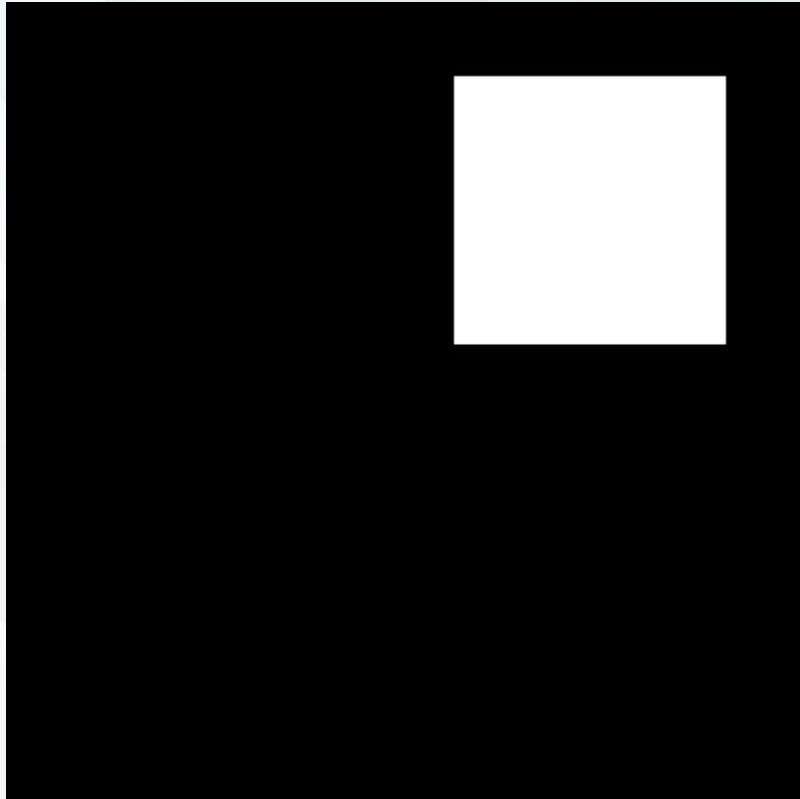
ICA (Análisis de Componentes Independientes)

-  Separa fuentes independientes incluso si están mezcladas
-  No requiere que las fuentes sean gaussianas
-  Preserva la independencia estadística
-  El orden de las componentes es arbitrario
-  Más sensible al ruido



Imágenes de Prueba:

- Condición: Ambas imágenes deben de ser del mismo tamaño.





Sumar Imágenes:

Se pueden añadir dos imágenes por la función OpenCV, `cv2.add()` o simplemente por la operación numpy. Ambas imágenes deben tener la misma profundidad y tipo, o la segunda imagen puede ser un valor escalar. $res = img1 + img2$

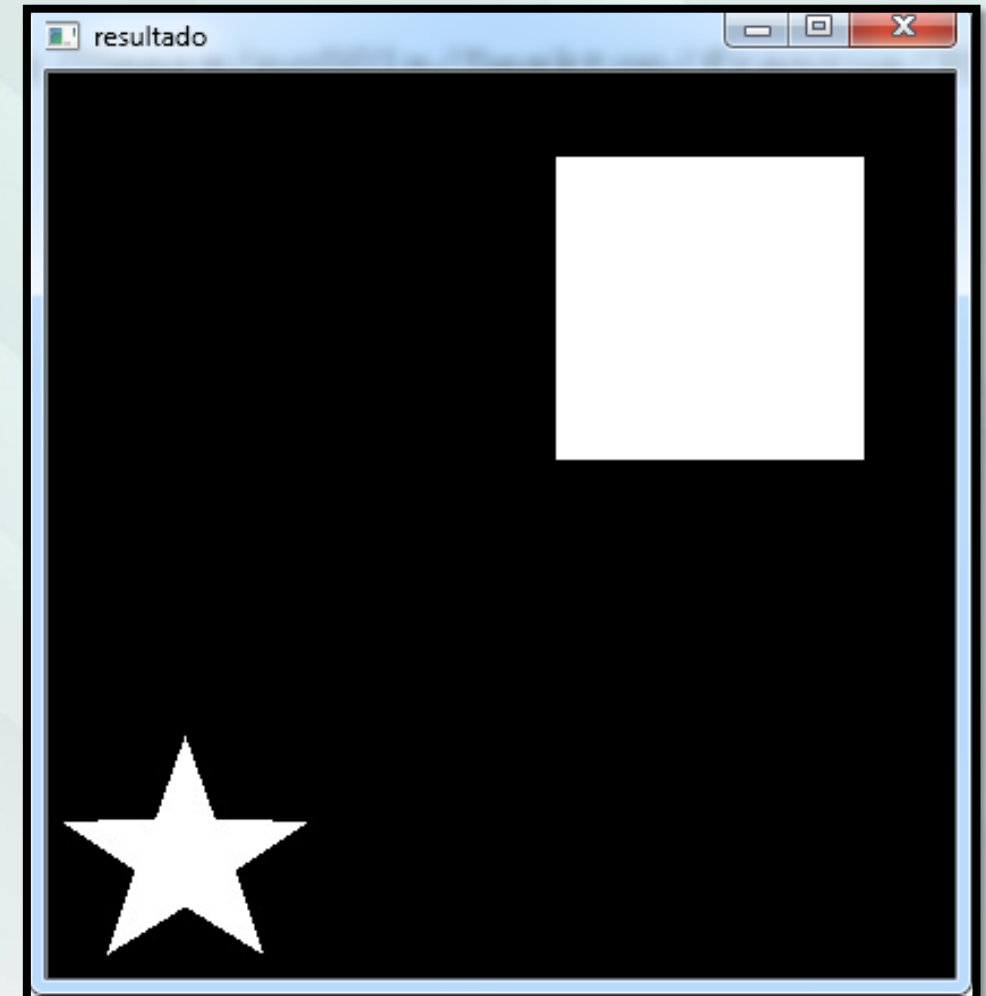
```
import numpy as np
import cv2

img1 = cv2.imread('cuadrado.jpg')
img2 = cv2.imread('estrella.jpg')

img3 = cv2.add(img1, img2)

print(img1[0,0])
print(img2[0,0])
print(img3[0,0])

cv2.imshow('resultado', img3)
```





Resultado de suma de imágenes:

```
import numpy as np
import cv2

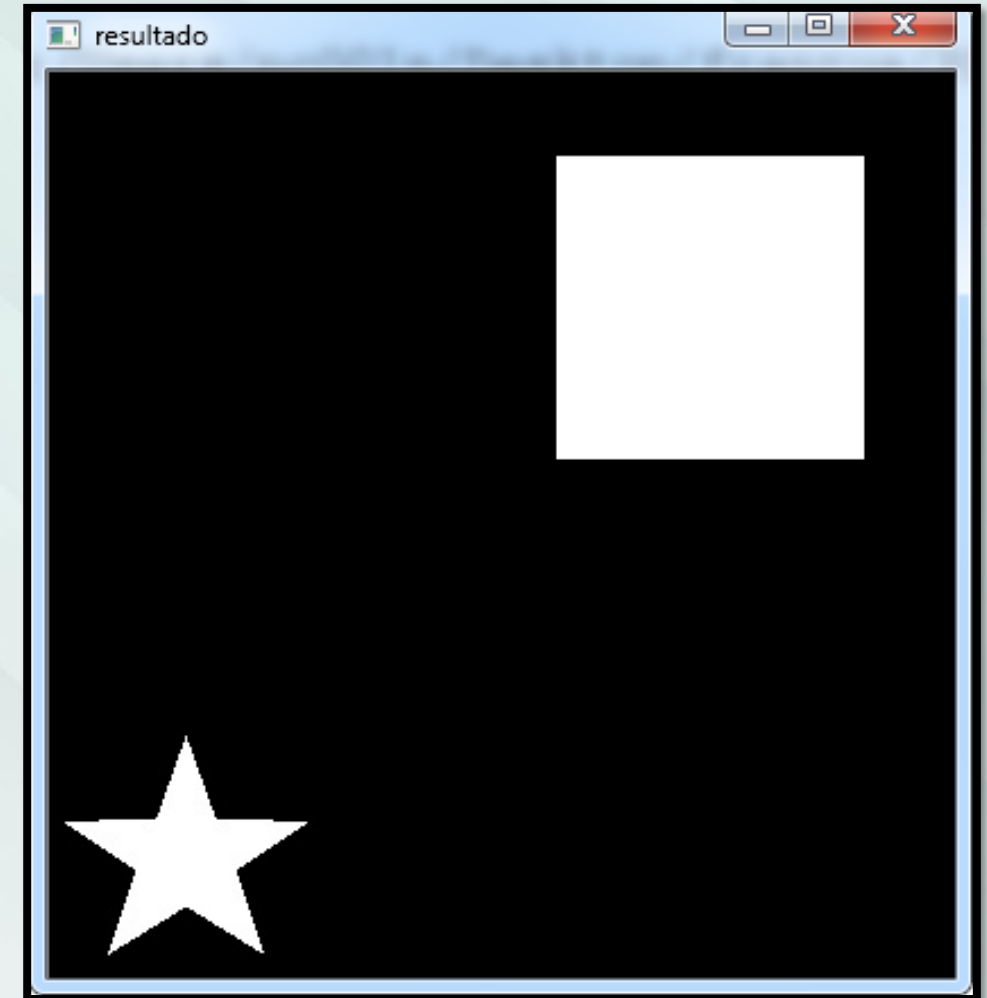
img1 = cv2.imread('cuadrado.jpg')
img2 = cv2.imread('estrella.jpg')

img3= cv2.add(img1,img2)

print(img1[0,0])
print(img2[0,0])
print(img3[0,0])

cv2.imshow('resultado',img3)
```

```
[ 29 229 181]
[ 29 229 181]
[ 58 255 255]
```





Mezclar Imágenes:

Esto también es una adición de imagen, pero se otorgan diferentes pesos a las imágenes para que parezca una mezcla o transparencia. Las imágenes se agregan según la siguiente ecuación:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

Al variar α de $0 \rightarrow 1$, puede realizar una transición genial entre una imagen a otra.

Aquí tomé dos imágenes para combinarlas. La primera imagen recibe un peso de 0.7 y la segunda imagen recibe 0.3. `cv2.addWeighted()` aplica la siguiente ecuación en la imagen.

$$dst = \alpha \cdot img1 + \beta \cdot img2 + \gamma$$





Mezclar Imágenes:

- Imágenes de prueba





Resultado de Suma de Imágenes:

```
import numpy as np
import cv2

img1 = cv2.imread('img1.jpg')
img2 = cv2.imread('img2.jpg')

img3= cv2.addWeighted(img1,0.5,img2,0.5,0)
cv2.imshow('resultado',img3)
```

$$dst = 0.5 * img1 + 0.5 * img2 + 0$$





Resta de Imágenes:



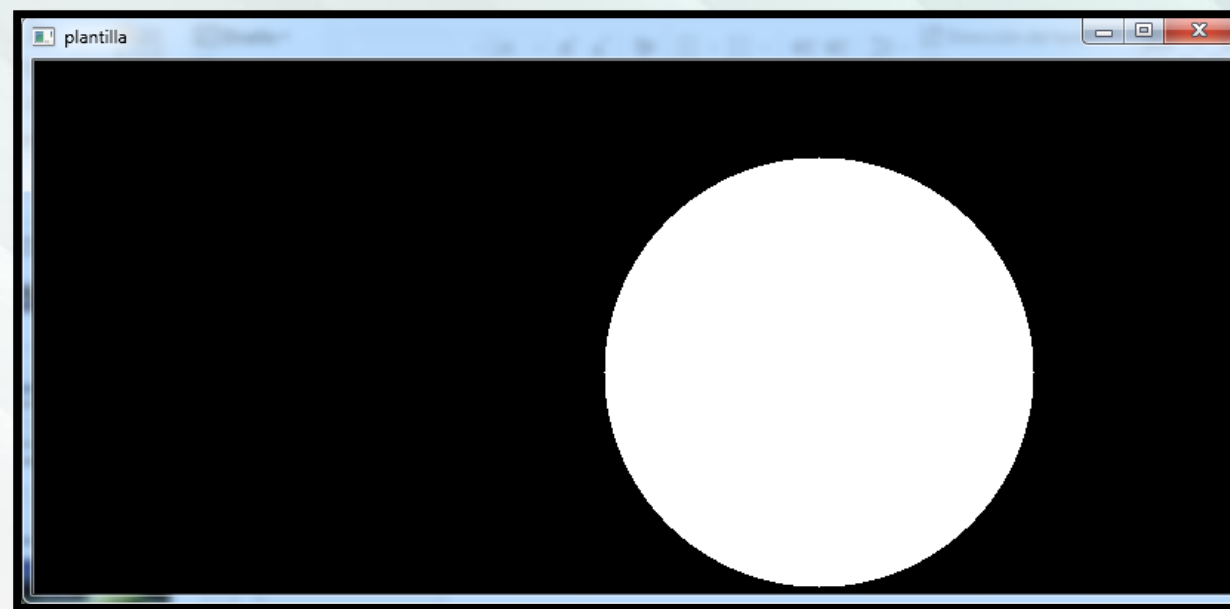
Plantilla de Imagen:



UNIVERSIDAD
CATÓLICA
SEDES SAPIENTIAE

```
import numpy as np
import cv2

img = np.zeros((360,810,3),np.uint8)
img = cv2.circle(img, (530,210),145, (255,255,255),-1)
cv2.imshow('plantilla',img)
```



Resta de Imágenes:



UNIVERSIDAD
CATÓLICA
SEDES SAPIENTIAE

```
import numpy as np
import cv2

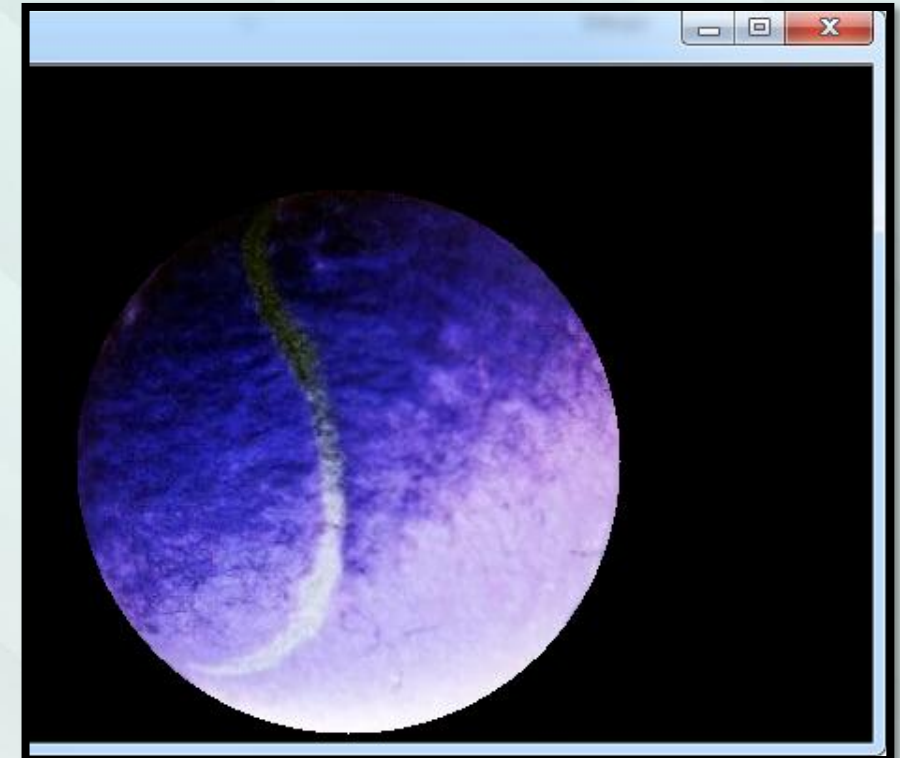
img = np.zeros((360,810,3),np.uint8)
img = cv2.circle(img, (530,210),145, (255,255,255),-1)
cv2.imshow('plantilla',img)

img1 = cv2.imread('tenis.jpg')

img3= cv2.subtract(img,img1)

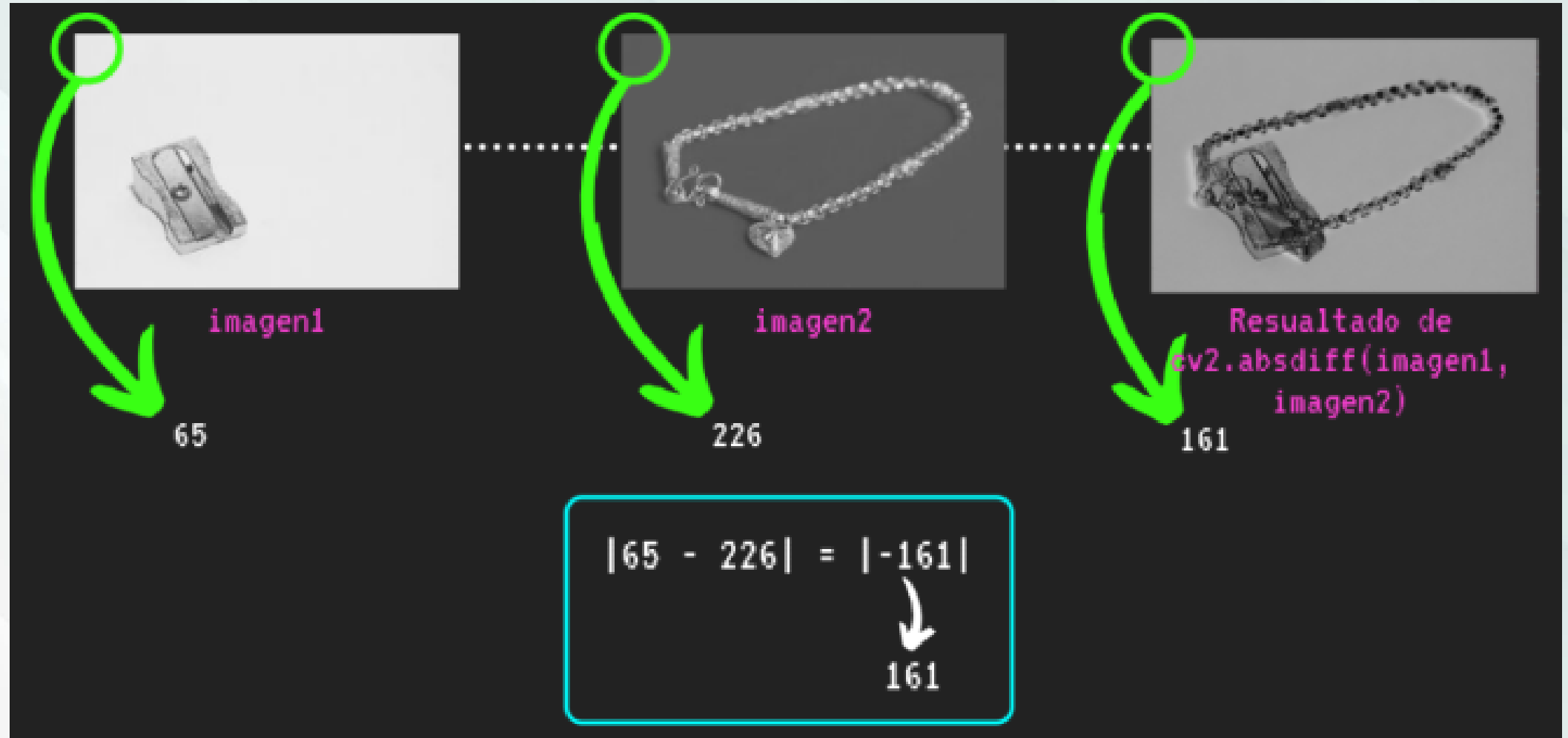
print(img1[0,0])
print(img3[0,0])

cv2.imshow('resultado',img3)
```





Resta Absoluta de Imágenes (absdiff):



Resta Absoluta de Imágenes:

- Para aplicar sustracción de imágenes OpenCV nos ofrece dos funciones: `cv2.subtract` y `cv2.absdiff`, vamos a ver como usar cada una de ellas y lo que hacen, ¡vamos por ello!.





Resta Absoluta de Imágenes:

```
import numpy as np
import cv2

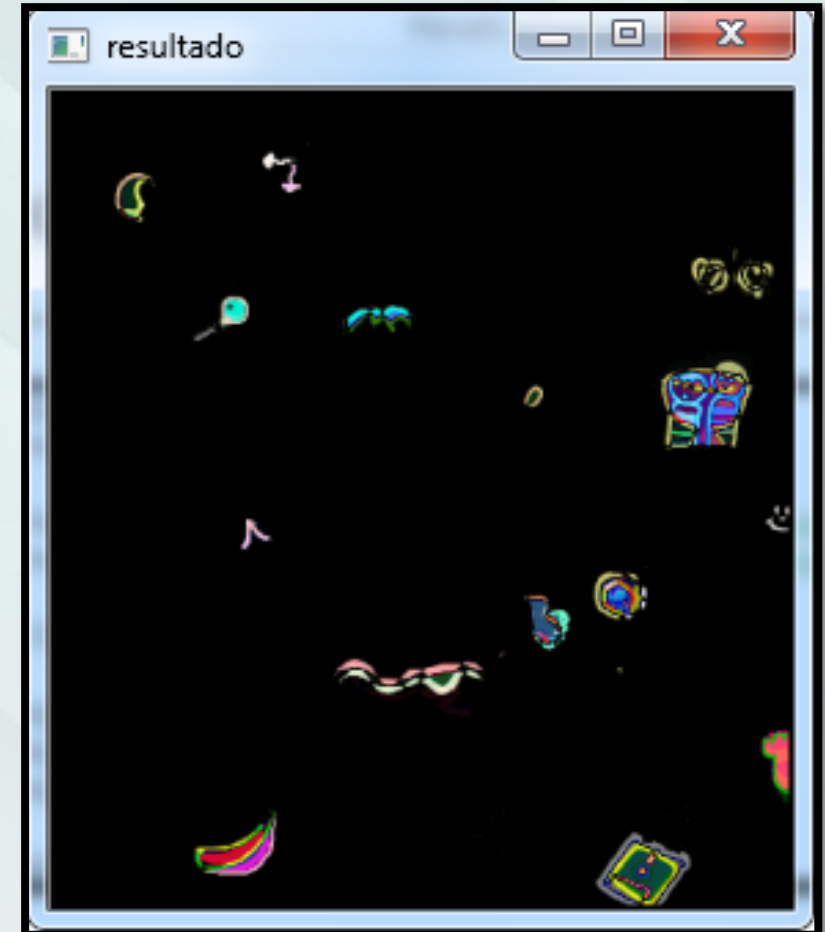
img1 = cv2.imread('imagen1.png')
img2 = cv2.imread('imagen2.png')

img3= cv2.absdiff(img1, img2)

print(img1[290, 250])
print(img2[290, 250])
print(img3[290, 250])

cv2.imshow('resultado', img3)
```

```
[120  41  0]
[120  41  0]
[0  0  0]
```





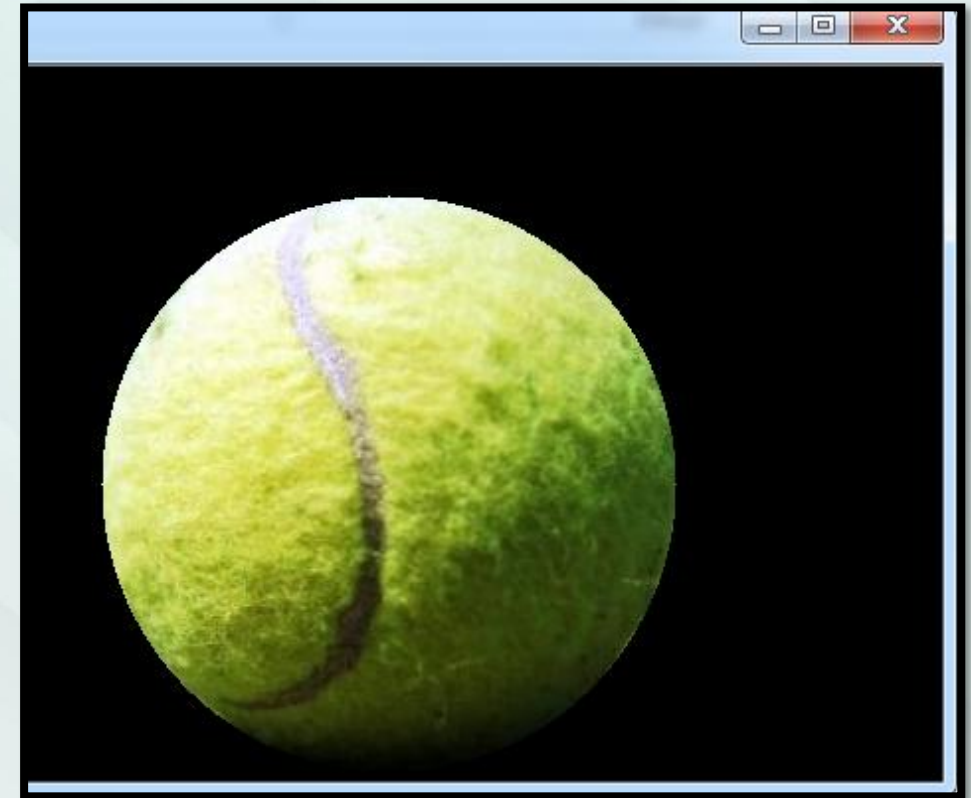
Bitwise aplicado en Imágenes:

```
import numpy as np
import cv2

mask = np.zeros((360,810,3),np.uint8)
mask= cv2.circle(mask,(530,210),145,(255,255,255),-1)
img1 = cv2.imread('tenis.jpg')

img2= cv2.bitwise_and(img1,mask)

print(img1[0,0])
print(mask[0,0])
cv2.imshow('resultado',img2)
```





**UNIVERSIDAD
CATÓLICA**
SEDES SAPIENTIAE