SCIPY

¿Qué es SciPy?

SciPy (Scientific Python) es una librería basada en NumPy que ofrece herramientas para:

- Álgebra lineal
- Optimización
- Cálculo numérico e integral
- Estadística y probabilidades
- Procesamiento de señales e imágenes

Funciones principales

Módulo	Función principal	Aplicación en ingeniería
scipy.linalg	Álgebra lineal (resolver sistemas, descomposición LU, QR)	Modelado de circuitos, cálculos estructurales
scipy.optimize	Optimización (mínimos, máximos, raíces)	Ajuste de parámetros, diseño óptimo
scipy.integrate	Integración y ODEs	Resolver ecuaciones diferenciales, dinámica de sistemas
scipy.stats	Estadística y distribuciones	Pruebas de hipótesis, análisis de datos experimentales
scipy.signal	Procesamiento de señales	Filtrado de audio, análisis de vibraciones
scipy.ndimage	Procesamiento de imágenes	Visión por computadora, análisis biomédico

Aplicaciones de SciPy

Área	Aplicación real
Ingeniería eléctrica	Análisis de circuitos y simulación de señales (uso de scipy.signal).
Mecatrónica/Robótica	Control de sistemas dinámicos resolviendo ecuaciones diferenciales (scipy.integrate).
Ingeniería civil	Optimización estructural en diseño de puentes y edificios (scipy.optimize).
Procesamiento de imágenes	Filtros, detección de bordes y análisis biomédico en imágenes médicas (scipy.ndimage).
Ciencia de datos	Análisis estadístico de grandes volúmenes de datos (scipy.stats).
Aeronáutica	Modelado y simulación de trayectorias de vuelo con ecuaciones diferenciales.

pip install scipy

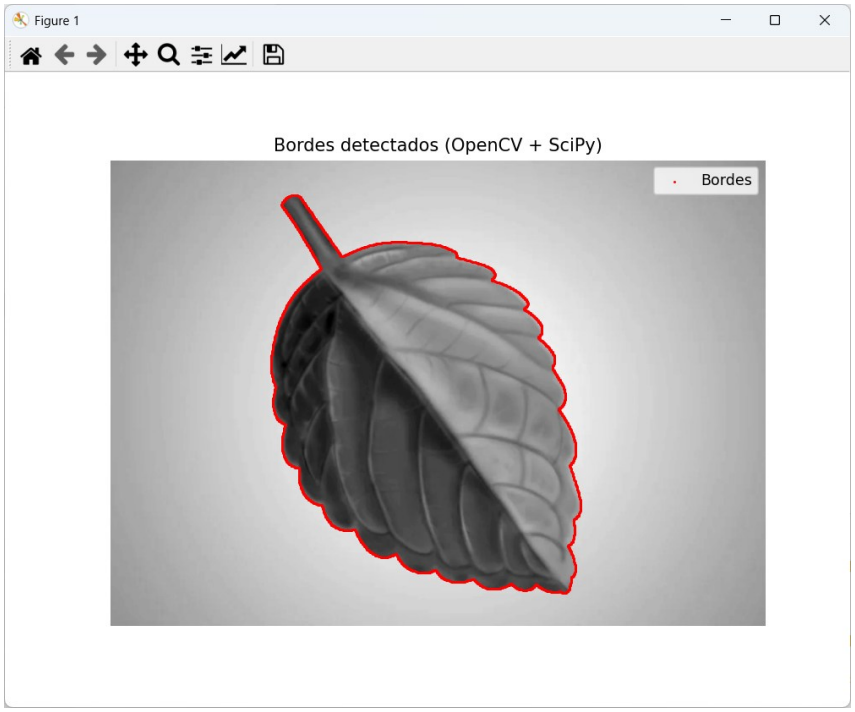
EJERCICIOS

Ejercicio 1

Implementa el siguiente algoritmo.

3_SciPy_Nltk > A001_Trazar_Bordes.py > ...

```python
import cv2
import matplotlib.pyplot as plt
from scipy import ndimage

ruta = r"C:\Users\DELL\Desktop\python\senati\3_SciPy_Nltk\hoja.png"
# 1. Cargar la imagen en escala de grises
img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError("No se pudo cargar la imagen 'hoja.png'.")

# 2. Detectar bordes con Canny
edges = cv2.Canny(img, 100, 200)

# 3. Obtener coordenadas de píxeles con valor distinto de 0 usando SciPy
ys, xs = ndimage.measurements.find_objects(edges > 0)[0]
ys_indices, xs_indices = (edges > 0)[ys, xs].nonzero()
ys_coords = ys.start + ys_indices
xs_coords = xs.start + xs_indices

# 4. Graficar los bordes
plt.figure(figsize=(8, 6))
plt.imshow(img, cmap='gray')
plt.scatter(xs_coords, ys_coords, s=0.5, color='red', label='Bordes')
plt.title("Bordes detectados (OpenCV + SciPy)")
plt.legend()
plt.gca().invert_yaxis()
plt.axis('off')
plt.show()
```



Ejercicio 2

Implementa el siguiente algoritmo.

_SciPy_Nltk > A002_Desenfoque.py > ...

```python
import cv2
import matplotlib.pyplot as plt
from scipy import ndimage

ruta = r"C:\Users\DELL\Desktop\python\senati\3_SciPy_Nltk\paisaje.jpg"

# 1. Cargar la imagen con OpenCV (escala de grises)
img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError("No se pudo cargar la imagen 'hoja.png'.")

# 2. Aplicar filtro Gaussiano con SciPy (sigma controla la intensidad del desenfoque)
img_scipy = ndimage.gaussian_filter(img, sigma=10)

# 3. Para comparar: aplicar filtro Gaussiano con OpenCV
img_cv2 = cv2.GaussianBlur(img, (0, 0), sigmaX=3)

# 4. Mostrar los resultados
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.title('Imagen Original')
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Filtro Gaussiano (SciPy)')
plt.imshow(img_scipy, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Filtro Gaussiano (OpenCV)')
plt.imshow(img_cv2, cmap='gray')
plt.axis('off')

plt.suptitle('Comparación: Desenfoque Gaussiano - SciPy vs OpenCV')
plt.show()
```

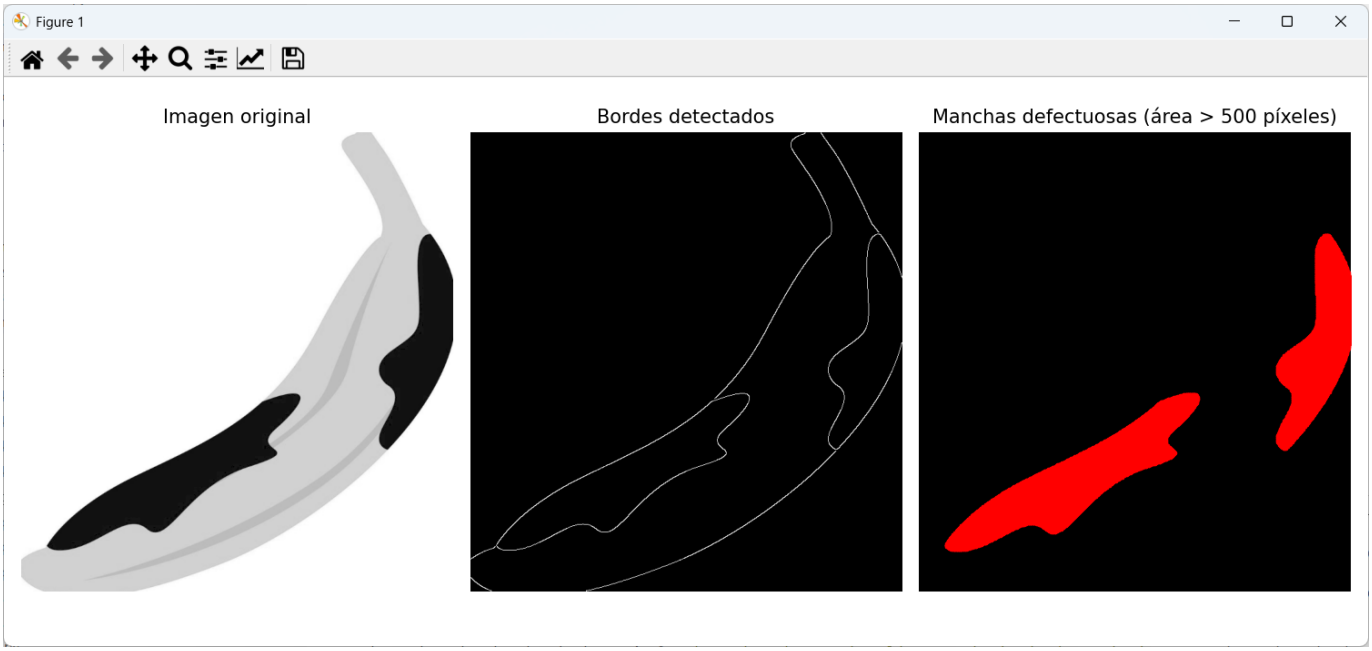
Ejercicio 3

Implementa el siguiente algoritmo.

3_SciPy_Nltk > A003_Contar_manchas.py > ...

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import label, median_filter

ruta = r"C:\Users\DELL\Desktop\python\senati\3_SciPy_Nltk\platano3.png"

# 1. Cargar imagen (ejemplo: una fruta)
img = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError("No se encontró la imagen 'platano'.")

# 2. Suavizar la imagen para reducir ruido (filtro mediano con SciPy)
img_smooth = median_filter(img, size=5)

# 3. Detectar bordes con Canny (OpenCV)
edges = cv2.Canny(img_smooth, 50, 150)

# 4. Binarizar para detectar manchas (invertir para que manchas sean '1')
_, thresh = cv2.threshold(img_smooth, 100, 255, cv2.THRESH_BINARY_INV)

# 5. Etiquetar regiones (manchas) con SciPy
labeled_array, num_features = label(thresh)

print(f"Cantidad de posibles manchas detectadas: {num_features}")

# 6. Analizar cada mancha: área mínima para considerar defecto
areas = []
for i in range(1, num_features + 1):
    area = np.sum(labeled_array == i)
    areas.append(area)

areas = np.array(areas)
manchas_grandes = areas > 10
# umbral área para considerar defecto / bajamos el valor para mayor deteccion
```

```
36
37  print(f"Manchas consideradas defectos: {np.sum(manchas_grandes)}")
38
39  # 7. Visualizar resultados
40  plt.figure(figsize=(12, 5))
41
42  plt.subplot(1, 3, 1)
43  plt.title('Imagen original')
44  plt.imshow(img, cmap='gray')
45  plt.axis('off')
46
47  plt.subplot(1, 3, 2)
48  plt.title('Bordes detectados')
49  plt.imshow(edges, cmap='gray')
50  plt.axis('off')
51
52  plt.subplot(1, 3, 3)
53  plt.title('Manchas defectuosas (área > 500 píxeles)')
54  plt.imshow(thresh, cmap='gray')
55  # Resaltar manchas defectuosas
56  mask = np.isin(labeled_array, np.where(manchas_grandes)[0] + 1)
57  plt.scatter(*np.nonzero(mask)[::-1], s=1, c='red')
58  plt.axis('off')
59
60  plt.tight_layout()
61  plt.show()
```



NLTK

¿Qué es NLTK?

NLTK (Natural Language Toolkit) es una biblioteca de Python de código abierto diseñada para el procesamiento de lenguaje natural (NLP). Proporciona una colección de herramientas y recursos para trabajar con texto y realizar tareas como tokenización, etiquetado de partes del discurso, análisis de sentimientos, análisis sintáctico y más.

Características

- **Amplia Funcionalidad:** Ofrece una amplia gama de algoritmos y herramientas para el procesamiento de texto y el análisis lingüístico.
- **Facilidad de Uso:** Proporciona una interfaz fácil de usar y documentación detallada para facilitar su uso.
- **Recursos Lingüísticos:** Incluye una variedad de recursos lingüísticos, como corpus de texto etiquetado y herramientas para el análisis de datos lingüísticos.
- **Comunidad Activa:** Cuenta con una comunidad activa de desarrolladores y usuarios que contribuyen con nuevos recursos y funcionalidades.

Aplicaciones

NLTK se utiliza en una variedad de aplicaciones de procesamiento de lenguaje natural, incluyendo la construcción de sistemas de chatbots, análisis de sentimientos en redes sociales, extracción de información, y más.

En resumen, SciPy es una biblioteca para cálculos científicos y técnicos, mientras que NLTK es una biblioteca especializada en el procesamiento de lenguaje natural, ambas ampliamente utilizadas en el campo de la ciencia de datos y el análisis de texto en Python

Funciones principales

Módulo	Función principal	Aplicación en ingeniería
nltk.tokenize	Tokenización (dividir texto en palabras o frases)	Preprocesamiento de datos textuales
nltk.corpus.stopwords	Eliminación de stopwords (palabras irrelevantes)	Limpieza de texto, reducción de ruido
nltk.stem	Stemming (raíz de palabras) y Lematización (forma base)	Normalización de datos para IA
nltk.pos_tag	Etiquetado gramatical (sustantivos, verbos, etc.)	Análisis sintáctico, extracción de información
nltk.FreqDist	Análisis de frecuencia de palabras	Minería de texto, nubes de palabras
nltk.classify	Clasificación de textos	Análisis de sentimientos, detección de spam

pip install nltk

EJERCICIOS

Ejercicio 1

Implementa el siguiente algoritmo.

3_SciPy_Nltk > N001_Nltk_1.py > ...

```python

#pip install nltk==3.8.1
from nltk.tokenize import TreebankWordTokenizer

tokenizer = TreebankWordTokenizer()

texto = "¡Hola! ¿Cómo estás hoy?"
tokens = tokenizer.tokenize(texto)

print("Tokens:", tokens)
```

Ejercicio 2

Implementa el siguiente algoritmo.

3_SciPy_Nltk > N002_Nltk_2.py > ...

```python
from nltk.tokenize import TreebankWordTokenizer
from nltk.corpus import stopwords
from collections import Counter
import nltk

# Asegúrate de usar la ruta correcta a tu nltk_data
nltk.data.path.append(r"C:\Users\DELL\Desktop\python\senati\3_SciPy_Nltk\nltk_data")

texto = "Este texto es un ejemplo para demostrar cómo encontrar las palabras más comunes en un texto cualquiera."

tokenizer = TreebankWordTokenizer()
tokens = tokenizer.tokenize(texto.lower())

# Cargar stopwords en español
stop_words = set(stopwords.words('spanish'))

# Eliminar stopwords y contar frecuencia
tokens_filtrados = [t for t in tokens if t.isalpha() and t not in stop_words]
conteo = Counter(tokens_filtrados)

print("Palabras más comunes:")
for palabra, frecuencia in conteo.most_common(5):
    print(f"{palabra}: {frecuencia}")
```

Ejercicio 3

Implementa el siguiente algoritmo.

3_SciPy_Nltk > N003_Nltk_3.py > ...

```python
from nltk.tokenize import TreebankWordTokenizer

# Listas simples (se pueden ampliar)
positivas = {"bueno", "excelente", "positivo", "feliz", "genial"}
negativas = {"malo", "terrible", "horrible", "negativo", "triste"}

texto = "La película fue excelente, aunque algunas escenas fueron malo y triste, en parte se muestra una escena horrib

tokenizer = TreebankWordTokenizer()
tokens = tokenizer.tokenize(texto.lower())

# Contar sentimientos
positivo = sum(1 for t in tokens if t in positivas)
negativo = sum(1 for t in tokens if t in negativas)

print(f"Palabras positivas: {positivo}")
print(f"Palabras negativas: {negativo}")

if positivo > negativo:
    print("Sentimiento general: POSITIVO")
elif negativo > positivo:
    print("Sentimiento general: NEGATIVO")
else:
    print("Sentimiento general: NEUTRO")
```

Ejercicio 4

Implementa el siguiente algoritmo.

3_SciPy_Nltk > N004_Nltk_4.py > ...

```python
from nltk.tokenize import TreebankWordTokenizer, sent_tokenize
from nltk.corpus import stopwords
from collections import Counter
import nltk

# Descargar recursos necesarios
nltk.download("punkt")
nltk.download("punkt_tab")
nltk.download("stopwords")

# ========= 1. Texto de ejemplo =========
texto = """
Ayer, Edgar, María y Pedro visitaron Madrid.
Después se encontraron con Juan Pérez en la Universidad de Barcelona.
"""

# ========= 2. Tokenización en oraciones y palabras =========
tokenizer = TreebankWordTokenizer()
stop_words = set(stopwords.words("spanish"))

candidatos = []

for oracion in sent_tokenize(texto, language="spanish"):
    tokens = tokenizer.tokenize(oracion)
    for i, palabra in enumerate(tokens):
        # Regla: empieza con mayúscula y no es stopword
        if palabra[0].isupper() and palabra.lower() not in stop_words:
            # Evitar la primera palabra de la oración si no parece nombre propio
            if i != 0 or palabra in ["María","Pedro","Madrid","Juan","Pérez","Barcelona"]:
                candidatos.append(palabra)
```

```
31
32  # ========= 3. Contar ocurrencias =========
33  conteo = Counter(candidatos)
34
35  # ========= 4. Mostrar resultados =========
36  print("Texto original:", texto)
37  print("Nombres propios detectados:", list(conteo.keys()))
38  print("Frecuencia:", conteo)
```