Bài 05.
MỘT SỐ THƯ VIỆN MÃ NGUỒN MỞ QUAN TRỌNG TRONG KHOA HỌC DỮ LIỆU

ThS. Lê Nhật Tùng

1. NumPy

- NumPy (Numerical Python) là một thư viện mã nguồn mở cho Python, cung cấp các công cụ mạnh mẽ để xử lý các mảng đa chiều (ndarray) và thực hiện các phép toán số học tốc độ cao.

Chức năng chính:
•Hỗ trợ mảng n-dimensional (ndarray).
•Cung cấp các hàm toán học như:
- **Phép tính mảng**: cộng, trừ, nhân, chia.
- **Đại số tuyến tính**: ma trận, ma trận nghịch đảo, định thức.
- **Thống kê**: giá trị trung bình, phương sai, độ lệch chuẩn.

•Tích hợp với C, C++ và Fortran để tăng tốc xử lý.

Lợi ích:
•Tốc độ xử lý nhanh hơn so với danh sách (list) của Python.
•Hiệu quả trong việc thao tác và phân tích dữ liệu số lượng lớn.
•Nền tảng cho các thư viện khoa học dữ liệu khác như Pandas, SciPy, Scikit-learn.

•**Ứng dụng:** Phân tích dữ liệu khoa học, học máy, mô phỏng, và xử lý tín hiệu.

```python
import numpy as np

# Tạo các mảng
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

# Các phép toán
result1 = arr1 + arr2  # Sẽ báo lỗi vì kích thước không tương thích
result2 = arr1 * 2
result3 = np.mean(arr2, axis=1)  # Trung bình theo hàng

# Indexing và slicing
element = arr1[2]
subarray = arr2[:, 1]

# Thao tác với mảng
arr3 = arr1.reshape(5, 1)
sorted_arr = np.sort(arr2, axis=0)

# Hàm toán học
sin_values = np.sin(arr1)
dot_product = np.dot(arr1, arr1)

# In kết quả
print("arr1:", arr1)
print("result2:", result2)
print("result3:", result3)
print("element:", element)
print("subarray:", subarray)
print("arr3:", arr3)
print("sorted_arr:", sorted_arr)
print("sin_values:", sin_values)
print("dot_product:", dot_product)
```

2. Pandas

- Pandas là một thư viện mã nguồn mở cho Python, cung cấp các công cụ mạnh mẽ để thao tác và phân tích dữ liệu có cấu trúc và bán cấu trúc. Đặc biệt hiệu quả cho dữ liệu dạng bảng và chuỗi thời gian.

Chức năng chính:
•**DataFrame**: Cấu trúc dữ liệu 2D giống bảng tính, với các cột có thể có kiểu dữ liệu khác nhau.
•**Series**: Cấu trúc dữ liệu 1D tương tự như danh sách (list) hoặc mảng (array).
•Xử lý dữ liệu bị thiếu, ghép, hợp nhất, và nhóm dữ liệu.
•Dễ dàng nhập/xuất dữ liệu từ/đến nhiều định dạng: CSV, Excel, SQL, JSON.

Lợi ích:
•Xử lý dữ liệu lớn nhanh chóng và hiệu quả.
•Các hàm tích hợp giúp dễ dàng thao tác, sắp xếp, lọc, và nhóm dữ liệu.
•Khả năng kết hợp với NumPy, Matplotlib để tạo ra các pipeline phân tích dữ liệu mạnh mẽ.

Ứng dụng:
•Thao tác và phân tích dữ liệu trong các dự án khoa học dữ liệu.
•Chuẩn bị và tiền xử lý dữ liệu cho học máy, phân tích tài chính, và các dự án thống kê.

```python
import pandas as pd

# Đọc dữ liệu từ file CSV
df = pd.read_csv("diem_hoc_sinh.csv")

# Hiển thị 5 dòng đầu tiên
print(df.head())

# Thông tin về DataFrame
print(df.info())

# Tính điểm trung bình các môn
df['Điểm trung bình'] = df[['Toán', 'Lý', 'Hóa']].mean(axis=1)

# Lọc học sinh có điểm Toán trên 8
hoc_sinh_gioi_toan = df[df['Toán'] > 8]

# Nhóm học sinh theo giới tính (giả sử có cột "Giới tính")
# và tính điểm trung bình mỗi nhóm
df.groupby('Giới tính')['Điểm trung bình'].mean()

# Lưu kết quả vào file Excel
df.to_excel("ket_qua.xlsx", index=False)
```

3. Matplotlib

- Matplotlib là thư viện mã nguồn mở cho Python, cung cấp các công cụ mạnh mẽ để tạo các biểu đồ và hình ảnh trực quan. Đây là một trong những thư viện phổ biến nhất để trực quan hóa dữ liệu trong khoa học dữ liệu.

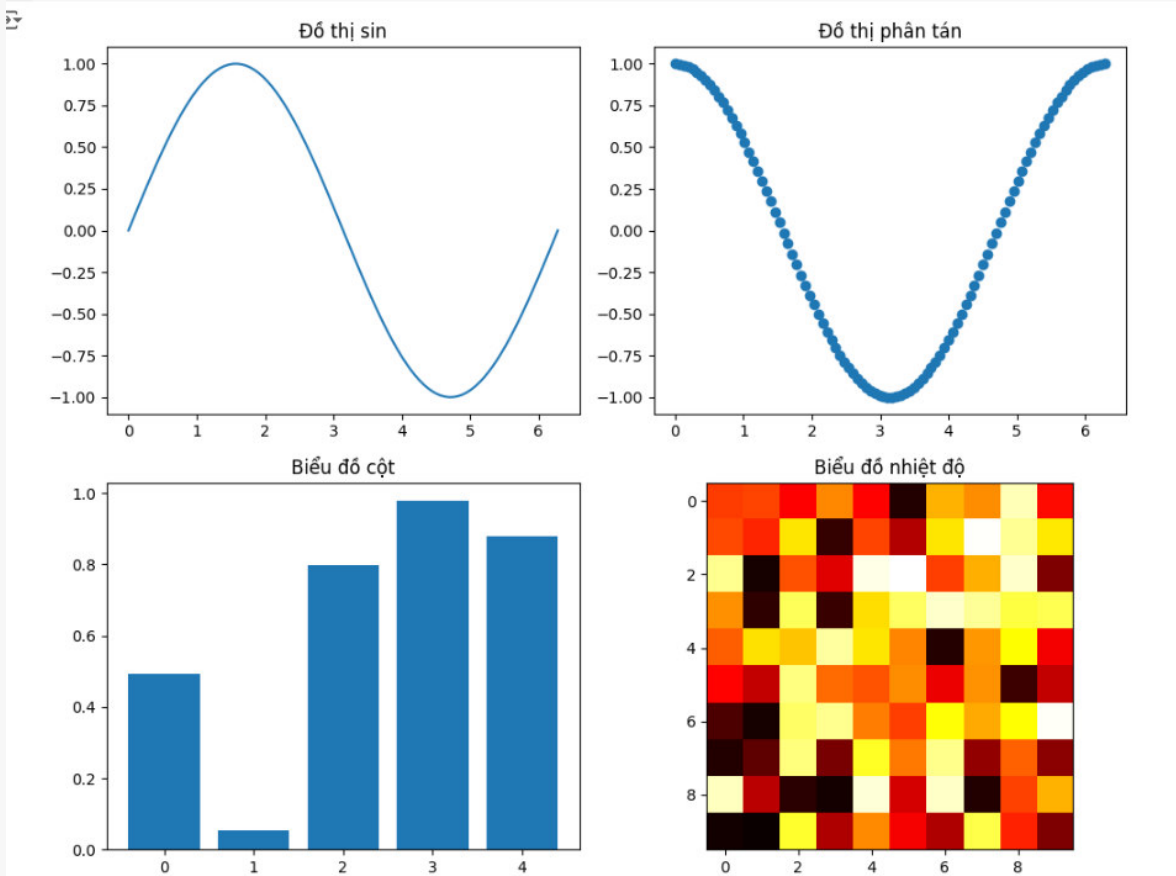
Chức năng chính:
•Hỗ trợ nhiều loại biểu đồ:
- **Biểu đồ đường**, **biểu đồ cột**, **biểu đồ phân tán**, **biểu đồ tròn**, **biểu đồ hộp**.

•Tùy chỉnh các thuộc tính biểu đồ: tiêu đề, nhãn trục, màu sắc, lưới, chú thích.
•Khả năng tích hợp với NumPy và Pandas để trực quan hóa dữ liệu trực tiếp.
•Hỗ trợ vẽ biểu đồ 2D tĩnh và có thể tạo biểu đồ tương tác thông qua các công cụ mở rộng.

Lợi ích:
•Dễ sử dụng và tùy chỉnh, phù hợp cho cả người mới và chuyên gia.
•Cung cấp các công cụ mạnh mẽ cho báo cáo trực quan và phân tích dữ liệu.
•Là nền tảng cho nhiều thư viện trực quan hóa khác như Seaborn.

Ứng dụng:
•Tạo biểu đồ trong các báo cáo khoa học, phân tích tài chính.
•Trực quan hóa dữ liệu trong học máy, mô hình thống kê và chuỗi thời gian.

```python
import matplotlib.pyplot as plt
import numpy as np

# Tạo dữ liệu mẫu
x = np.linspace(0, 2 * np.pi, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Tạo figure và các subplot
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# Vẽ biểu đồ đường trên subplot đầu tiên
axes[0, 0].plot(x, y1)
axes[0, 0].set_title('Đồ thị sin')

# Vẽ biểu đồ phân tán trên subplot thứ hai
axes[0, 1].scatter(x, y2)
axes[0, 1].set_title('Đồ thị phân tán')

# Vẽ biểu đồ cột trên subplot thứ ba
axes[1, 0].bar(np.arange(5), np.random.rand(5))
axes[1, 0].set_title('Biểu đồ cột')

# Vẽ biểu đồ nhiệt độ trên subplot thứ tư
axes[1, 1].imshow(np.random.rand(10, 10), cmap='hot')
axes[1, 1].set_title('Biểu đồ nhiệt độ')

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



4. Seaborn

- Seaborn là một thư viện mã nguồn mở xây dựng trên Matplotlib, được thiết kế để tạo ra các biểu đồ trực quan phức tạp và đẹp mắt một cách dễ dàng. Seaborn cung cấp các công cụ trực quan hóa dữ liệu theo thống kê và các mối quan hệ giữa các biến.

Chức năng chính:

•Cung cấp các kiểu biểu đồ nâng cao:

- **Biểu đồ phân phối** (distplot), **biểu đồ hồi quy** (regplot), **biểu đồ hộp** (boxplot), **biểu đồ điểm** (pointplot).

•Hỗ trợ tự động định dạng biểu đồ với phong cách trực quan hiện đại và thẩm mỹ.

•Dễ dàng tích hợp với Pandas để vẽ biểu đồ từ DataFrame.

•Hỗ trợ hiển thị mối quan hệ giữa nhiều biến và trực quan hóa ma trận tương quan.

Lợi ích:

•Giao diện thân thiện và dễ sử dụng để trực quan hóa dữ liệu phức tạp.

•Được tối ưu hóa để làm việc với các dữ liệu có cấu trúc như bảng (DataFrame).

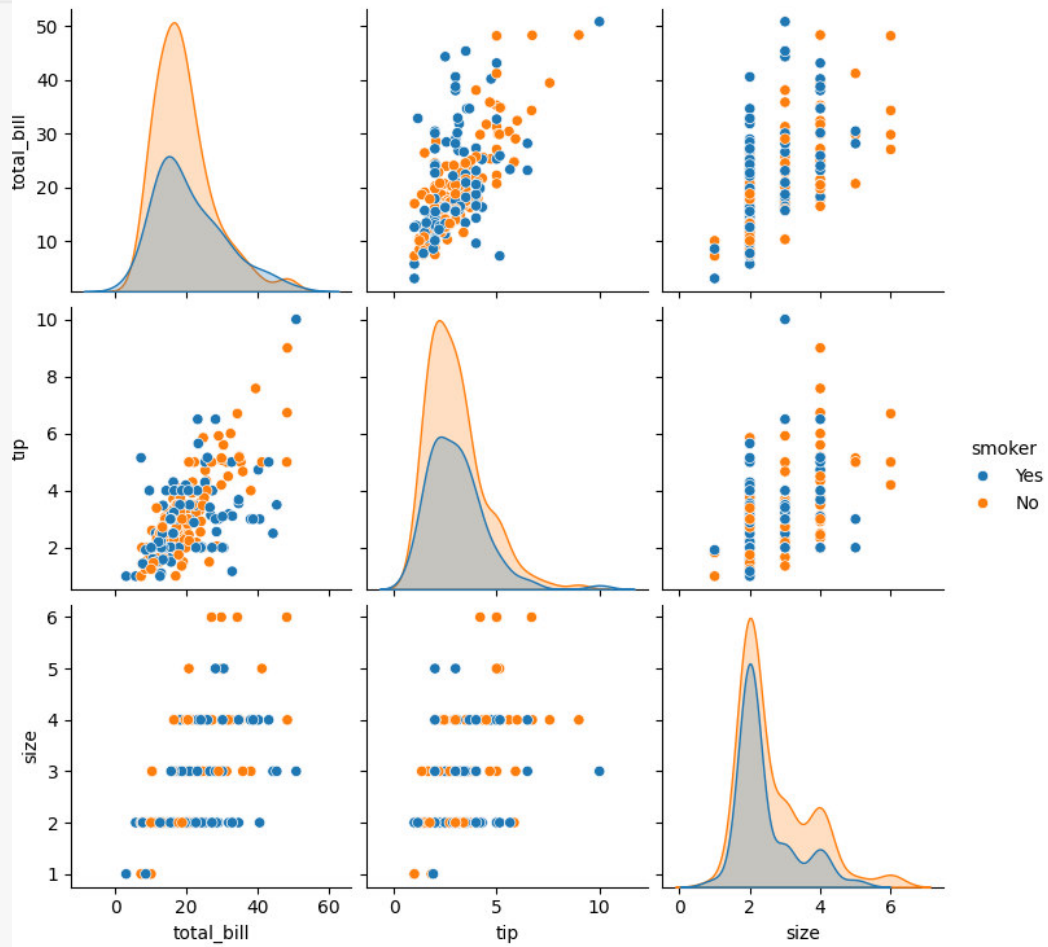
•Tạo các biểu đồ với định dạng đẹp mắt và chuyên nghiệp mà không cần tùy chỉnh nhiều.

Ứng dụng:

•Phân tích dữ liệu thống kê, trực quan hóa mối quan hệ giữa các biến.

•Được sử dụng rộng rãi trong khoa học dữ liệu, tài chính, và học máy để tạo ra các biểu đồ phân tích đẹp và dễ hiểu.

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Load dữ liệu (giả sử dữ liệu đã được lưu trong file CSV)
tips = sns.load_dataset("tips")

# 1. Biểu đồ phân tán (scatter plot)
sns.scatterplot(x="total_bill", y="tip", hue="smoker", data=tips)
plt.title("Mối quan hệ giữa tổng hóa đơn và tiền boa")
plt.show()

# 2. Biểu đồ hộp (box plot)
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("Phân bố tổng hóa đơn theo ngày")
plt.show()

# 3. Biểu đồ phân phối (histogram)
sns.histplot(data=tips, x="total_bill", hue="sex", multiple="stack")
plt.title("Phân phối tổng hóa đơn theo giới tính")
plt.show()

# 4. Ma trận tương quan (pairplot)
sns.pairplot(tips, hue="smoker")
plt.show()
```



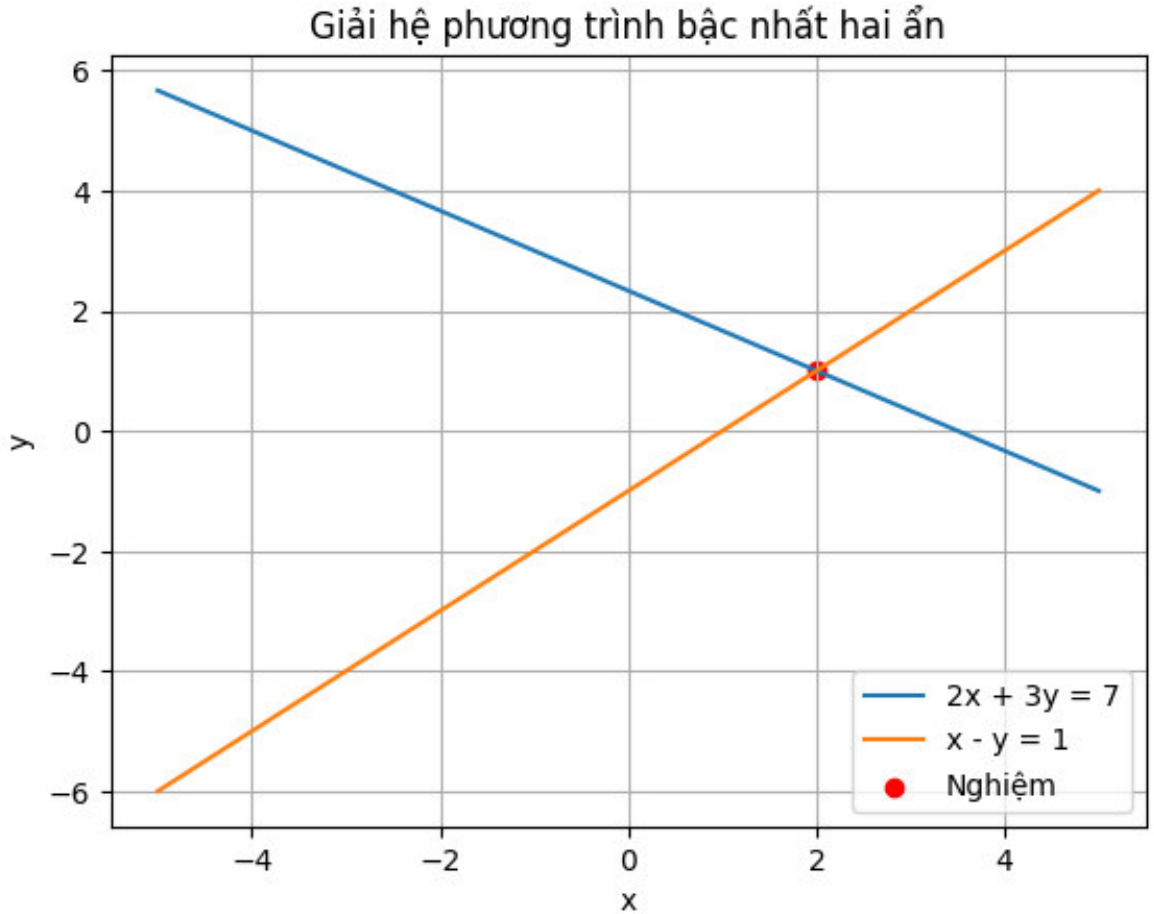
5. SciPy

- SciPy là thư viện mã nguồn mở cho Python, được xây dựng trên nền tảng NumPy, cung cấp các công cụ và thuật toán cho tính toán khoa học và kỹ thuật phức tạp, bao gồm tối ưu hóa, tích phân, nội suy và xử lý tín hiệu.

Chức năng chính:
•**Tối ưu hóa**: Các thuật toán tối ưu phi tuyến và tối ưu ràng buộc.
•**Tích phân**: Các hàm tính tích phân, vi phân, và giải phương trình vi phân.
•**Xử lý tín hiệu**: Các công cụ lọc, biến đổi Fourier, và phân tích tín hiệu.
•**Đại số tuyến tính**: Các phép toán ma trận, phân tích ma trận, phân tích đặc trưng.
•**Thống kê**: Các hàm phân phối xác suất, kiểm định giả thuyết, hồi quy thống kê.

Lợi ích:
•Cung cấp các công cụ tính toán tiên tiến cho các bài toán khoa học, kỹ thuật và thống kê.
•Được tối ưu hóa để xử lý hiệu quả các bài toán phức tạp và dữ liệu lớn.
•**Ứng dụng**:
- Giải quyết các bài toán tối ưu hóa và mô phỏng trong khoa học và kỹ thuật.
- Phân tích và xử lý tín hiệu, hình ảnh.
- Phân tích dữ liệu thống kê, kiểm định giả thuyết và phân tích số liệu thực nghiệm.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import solve

# Ma trận hệ số
A = np.array([[2, 3],
              [1, -1]])

# Ma trận cột các số hạng tự do
b = np.array([7, 1])

# Giải hệ phương trình
x = solve(A, b)
print("Nghiệm của hệ phương trình là:", x)

# Vẽ đồ thị
x_values = np.linspace(-5, 5, 100)
y1 = (7 - 2 * x_values) / 3
y2 = x_values - 1

plt.plot(x_values, y1, label='2x + 3y = 7')
plt.plot(x_values, y2, label='x - y = 1')
plt.scatter(x[0], x[1], color='red', label='Nghiệm')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Giải hệ phương trình bậc nhất hai ẩn')
plt.legend()
plt.grid(True)
plt.show()
```



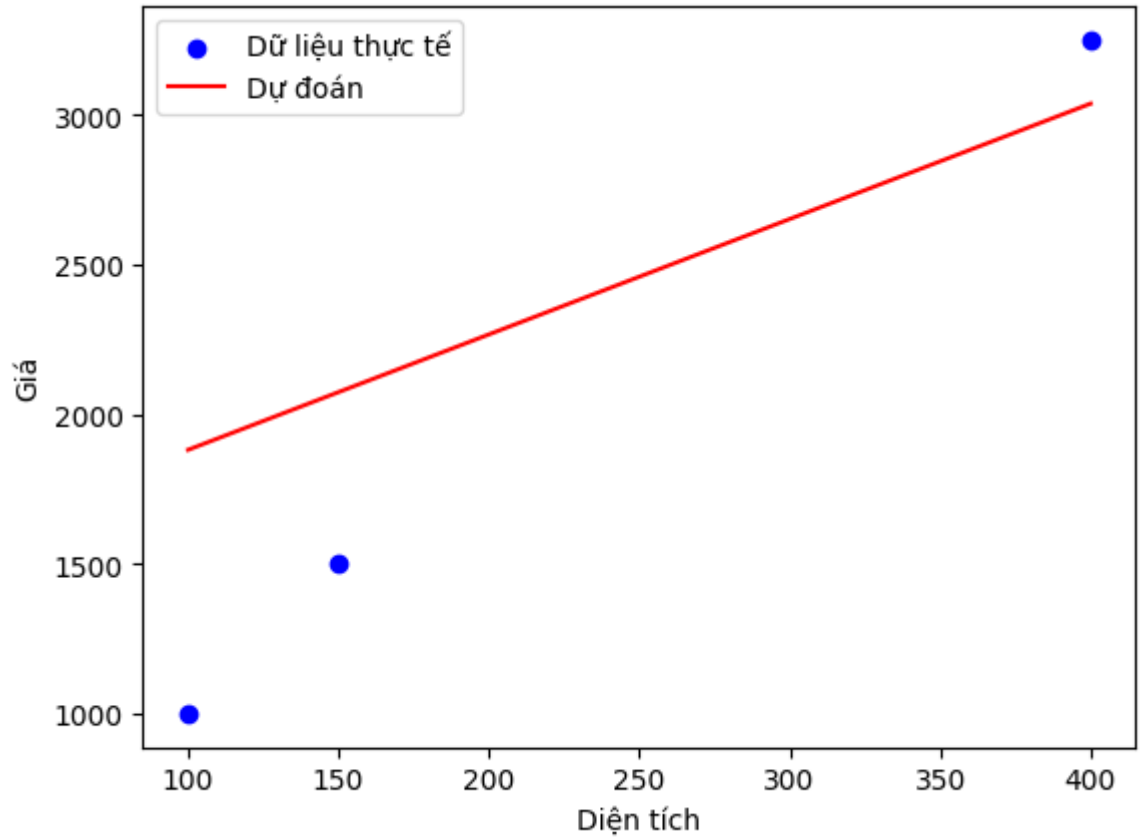
6. Scikit-learn

- Scikit-learn là một thư viện mã nguồn mở mạnh mẽ cho Python, cung cấp các công cụ và thuật toán phổ biến cho học máy (machine learning). Nó hỗ trợ cả các thuật toán học giám sát (supervised) và không giám sát (unsupervised).

Chức năng chính:
•**Thuật toán phân loại**: Hỗ trợ các thuật toán như KNN, SVM, Naive Bayes, Decision Trees.
•**Thuật toán hồi quy**: Hồi quy tuyến tính, hồi quy logistic.
•**Thuật toán cụm**: K-means, Hierarchical Clustering, DBSCAN.
•**Giảm kích thước**: Principal Component Analysis (PCA), t-SNE.
•**Đánh giá mô hình**: Cross-validation, confusion matrix, precision, recall.

Lợi ích:
Dễ sử dụng và thân thiện với người mới bắt đầu.
Cung cấp nhiều công cụ tiện ích như chuẩn hóa dữ liệu, chia tách dữ liệu (train-test split).
Tích hợp tốt với các thư viện như NumPy và Pandas để xây dựng pipeline học máy toàn diện.
Được tối ưu hóa để xử lý hiệu quả các thuật toán phức tạp với tốc độ nhanh.
Ứng dụng:
Phân loại và hồi quy trong học máy: dự đoán, phân loại hình ảnh, phân tích văn bản.
Phân tích cụm và giảm kích thước dữ liệu để khám phá các mẫu ẩn trong dữ liệu lớn.
Đánh giá và tối ưu hóa mô hình học máy để cải thiện độ chính xác và hiệu suất.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Tạo dữ liệu mẫu (bạn có thể thay thế bằng dữ liệu thực tế)
data = {'dientich': [100, 150, 200, 250, 300, 400, 500],
        'gia': [1000, 1500, 2000, 2500, 3000, 3250, 3300]}

# Tạo DataFrame từ dữ liệu
df = pd.DataFrame(data)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X = df[['dientich']]  # Tính năng (diện tích)
y = df['gia']  # Nhãn (giá)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Tạo mô hình hồi quy tuyến tính
model = LinearRegression()

# Huấn luyện mô hình
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Vẽ đồ thị
plt.scatter(X_test, y_test, color='blue', label='Dữ liệu thực tế')
plt.plot(X_test, y_pred, color='red', label='Dự đoán')
plt.xlabel('Diện tích')
plt.ylabel('Giá')
plt.legend()
plt.show()
```



7. TensorFlow

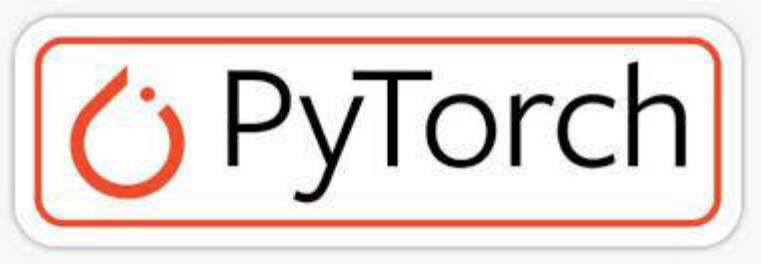
- TensorFlow là một thư viện mã nguồn mở mạnh mẽ được phát triển bởi Google, hỗ trợ xây dựng và huấn luyện các mô hình học sâu (deep learning) và học máy (machine learning) trên nhiều quy mô khác nhau, từ máy tính cá nhân đến các hệ thống phân tán.

Chức năng chính:
•**Mạng nơ-ron nhân tạo**: Xây dựng và huấn luyện các mô hình học sâu phức tạp như CNN, RNN, và Transformer.
•**Hỗ trợ GPU/TPU**: Tối ưu hóa tính toán trên các hệ thống phần cứng tăng tốc như GPU và TPU.
•**TensorFlow Lite**: Triển khai mô hình trên các thiết bị di động và IoT.
•**TensorFlow Extended (TFX)**: Xây dựng pipeline cho toàn bộ quy trình học máy từ tiền xử lý đến triển khai.
•**TensorBoard**: Công cụ trực quan hóa quá trình huấn luyện và đánh giá mô hình.

Ứng dụng:
•Nhận diện hình ảnh, phân loại văn bản, xử lý ngôn ngữ tự nhiên (NLP), dịch máy.
•Phân tích chuỗi thời gian, hệ thống đề xuất, xe tự hành và nhiều ứng dụng AI phức tạp khác.
•Triển khai mô hình trên các thiết bị di động và nhúng nhờ TensorFlow Lite.

```python
import tensorflow as tf
import numpy as np

# Tạo dữ liệu mẫu
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 4, 5, 4, 6])

# Xây dựng mô hình
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

# Biên dịch mô hình
model.compile(optimizer='sgd', loss='mean_squared_error')

# Huấn luyện mô hình
model.fit(X, y, epochs=500)

# Dự đoán
print(model.predict([10.0]))
```

8. PyTorch



- PyTorch là một thư viện mã nguồn mở cho học máy và học sâu, nổi bật với tính dễ sử dụng và hỗ trợ tính toán động.
- Nó được ứng dụng rộng rãi trong nghiên cứu AI, xử lý ngôn ngữ tự nhiên, nhận diện hình ảnh, và hệ thống đề xuất.
- PyTorch hỗ trợ huấn luyện các mạng nơ-ron trên GPU, giúp tăng tốc phát triển và triển khai mô hình AI.

```python
import torch
import torch.nn as nn
import torch.optim as optim

# Tạo dữ liệu mẫu
X = torch.tensor([[1.0], [2.0], [3.0], [4.0], [5.0]], dtype=torch.float32)
y = torch.tensor([[2.0], [4.1], [5.9], [7.2], [8.5]], dtype=torch.float32)

# Xây dựng mô hình
model = nn.Linear(1, 1)  # Một lớp linear với 1 input và 1 output

# Định nghĩa hàm loss và optimizer
criterion = nn.MSELoss()  # Mean Squared Error
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Huấn luyện mô hình
for epoch in range(1000):
    # Tính toán output
    outputs = model(X)

    # Tính toán loss
    loss = criterion(outputs, y)

    # Backpropagation và cập nhật trọng số
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 100 == 0:
        print(f'Epoch [{epoch+1}/{1000}], Loss: {loss.item():.4f}')

# Dự đoán
new_data = torch.tensor([[7.0]])
predicted = model(new_data)
print(predicted)
```

```
Epoch [100/1000], Loss: 0.0912
Epoch [200/1000], Loss: 0.0789
Epoch [300/1000], Loss: 0.0726
Epoch [400/1000], Loss: 0.0695
Epoch [500/1000], Loss: 0.0679
Epoch [600/1000], Loss: 0.0670
Epoch [700/1000], Loss: 0.0666
Epoch [800/1000], Loss: 0.0664
Epoch [900/1000], Loss: 0.0663
Epoch [1000/1000], Loss: 0.0663
tensor([[11.9964]], grad_fn=<AddmmBackward0>)
```

9. Keras

- Keras là một thư viện mã nguồn mở cho học sâu, được phát triển để đơn giản hóa việc xây dựng và huấn luyện các mô hình mạng nơ-ron.
- Nó cung cấp giao diện thân thiện và dễ sử dụng, cho phép người dùng nhanh chóng triển khai các mô hình phức tạp trong các lĩnh vực như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và học máy.
- Keras có thể chạy trên TensorFlow, Theano hoặc Microsoft CNTK, giúp tối ưu hóa hiệu suất và tính linh hoạt trong việc phát triển ứng dụng AI.

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np

# Tạo dữ liệu mẫu
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 4, 5, 4, 6])

# Xây dựng mô hình
model = Sequential([
    Dense(units=1, input_shape=[1])
])

# Biên dịch mô hình
model.compile(optimizer='sgd', loss='mean_squared_error')

# Huấn luyện mô hình
model.fit(X, y, epochs=500)

# Dự đoán
print(model.predict([10.0]))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.5947
```

11. NLTK (Natural Language Toolkit)

python™ NLTK

- NLTK (Natural Language Toolkit) là một thư viện mã nguồn mở cho Python, cung cấp các công cụ và tài nguyên để xử lý ngôn ngữ tự nhiên.
- Nó cho phép người dùng thực hiện các tác vụ như phân tích cú pháp, gán nhãn từ loại, phân tích cảm xúc và tách từ. NLTK được ứng dụng rộng rãi trong các nghiên cứu về ngôn ngữ, phát triển ứng dụng chatbot, và xây dựng các hệ thống thông minh xử lý văn bản.
- Thư viện này cung cấp một bộ sưu tập lớn các dữ liệu và tài nguyên giúp hỗ trợ việc học và nghiên cứu trong lĩnh vực ngôn ngữ tự nhiên.

```python
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

# Tải dữ liệu
nltk.download('punkt')

# Đoạn văn bản mẫu
text = "Đây là một ví dụ về cách sử dụng NLTK để token hóa văn bản. Chúng ta sẽ tách câu và tách từ."

# Tách câu
sentences = sent_tokenize(text)
print(sentences)

# Tách từ trong câu đầu tiên
words = word_tokenize(sentences[0])
print(words)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Đây là một ví dụ về cách sử dụng NLTK để token hóa văn bản.', 'Chúng ta sẽ tách câu và tách từ.']
['Đây', 'là', 'một', 'ví', 'dụ', 'về', 'cách', 'sử', 'dụng', 'NLTK', 'để', 'token', 'hóa', 'văn', 'bản', '.']
```

12. Gensim

- Gensim là một thư viện mã nguồn mở cho Python, chuyên về xử lý ngôn ngữ tự nhiên và học máy, đặc biệt trong việc phân tích văn bản và mô hình hóa chủ đề.
- Nó hỗ trợ các tác vụ như trích xuất đặc trưng từ văn bản, xây dựng mô hình nhúng từ (word embeddings) như Word2Vec và FastText, và thực hiện phân tích chủ đề với LDA (Latent Dirichlet Allocation).
- Gensim được ứng dụng rộng rãi trong việc xây dựng hệ thống gợi ý, phân tích cảm xúc, và khám phá thông tin từ các tập dữ liệu văn bản lớn. Thư viện này nổi bật với khả năng xử lý các tập dữ liệu lớn mà không cần tải toàn bộ vào bộ nhớ.

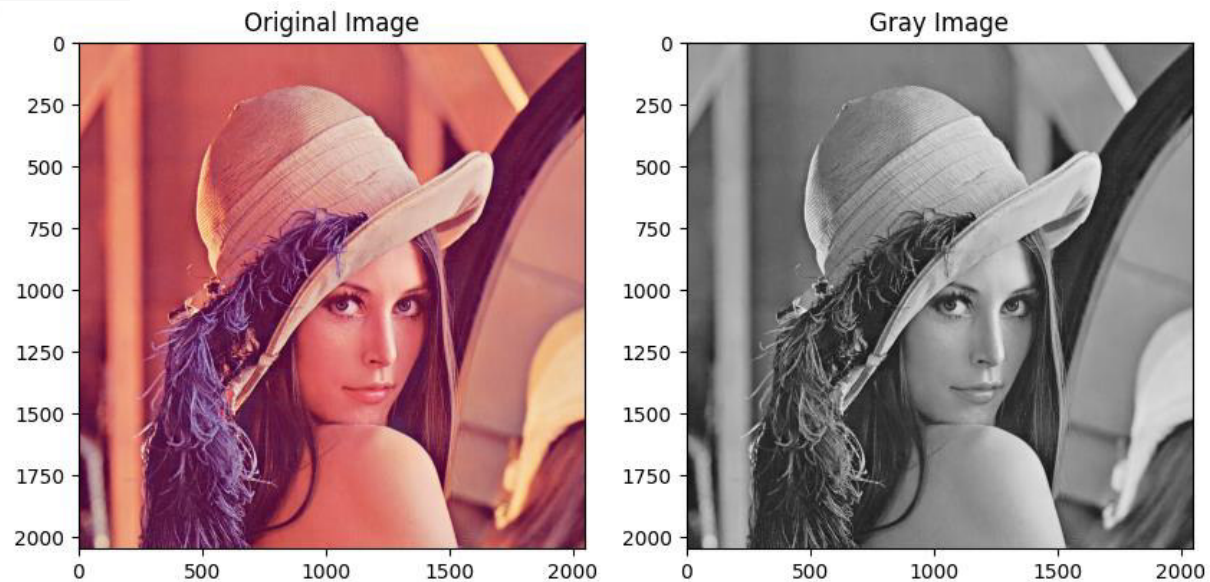
```python
import gensim
from gensim.models import Word2Vec

# Tạo một danh sách các câu
sentences = [
    ['cat', 'sat', 'on', 'the', 'mat'],
    ['the', 'dog', 'barked', 'at', 'the', 'cat'],
]

# Tạo mô hình Word2Vec
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Lưu mô hình
model.save("word2vec.model")

# Tải mô hình đã lưu
model = Word2Vec.load("word2vec.model")

# Xem vector của từ "cat"
print(model.wv['cat'])

# Tìm các từ tương tự với từ "cat"
print(model.wv.most_similar('cat'))
```

```
[-8.6196875e-03  3.6657380e-03  5.1898835e-03  5.7419385e-03
  7.4669183e-03 -6.1676754e-03  1.1056137e-03  6.0472824e-03
 -2.8400505e-03 -6.1735227e-03 -4.1022300e-04 -8.3689485e-03
 -5.6000124e-03  7.1045388e-03  3.3525396e-03  7.2256695e-03
  6.8002474e-03  7.5307419e-03 -3.7891543e-03 -5.6180597e-04
  2.3483764e-03 -4.5190323e-03  8.3887316e-03 -9.8581640e-03
  6.7646410e-03  2.9144168e-03 -4.9328315e-03  4.3981876e-03
 -1.7395747e-03  6.7113843e-03  9.9648498e-03 -4.3624435e-03
```

13. OpenCV

- OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở mạnh mẽ cho xử lý ảnh và thị giác máy tính.
- Nó cung cấp nhiều công cụ và thuật toán để thực hiện các tác vụ như nhận diện khuôn mặt, theo dõi đối tượng, xử lý video, và phân tích hình ảnh.
- OpenCV được ứng dụng rộng rãi trong các lĩnh vực như robot, nhận diện hình ảnh, thực tế ảo, và các hệ thống an ninh. Thư viện này hỗ trợ nhiều ngôn ngữ lập trình như Python, C++, và Java, giúp phát triển ứng dụng nhanh chóng và hiệu quả.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import urllib

# Tải hình ảnh từ internet
url = 'https://upload.wikimedia.org/wikipedia/ru/2/24/Lenna.png'
img_resp = urllib.request.urlopen(url)
img_arr = np.array(bytearray(img_resp.read()), dtype=np.uint8)
img = cv2.imdecode(img_arr, cv2.IMREAD_COLOR)

# Chuyển ảnh thành ảnh xám
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Hiển thị ảnh gốc và ảnh xám
plt.figure(figsize=(10,5))
plt.subplot(1,2,1),plt.imshow(img[:,:,::-1]),plt.title('Original Image')
plt.subplot(1,2,2),plt.imshow(gray, cmap='gray'),plt.title('Gray Image')
plt.show()
```



14. XGBoost

XGBoost

- XGBoost (Extreme Gradient Boosting) là một thư viện mã nguồn mở mạnh mẽ cho học máy, chuyên về thuật toán boosting để cải thiện độ chính xác của mô hình dự đoán.
- Nó nổi bật với khả năng xử lý nhanh, hiệu quả và giảm thiểu overfitting thông qua các kỹ thuật regularization.
- XGBoost được ứng dụng rộng rãi trong các bài toán phân loại và hồi quy, đặc biệt là trong các cuộc thi khoa học dữ liệu như Kaggle. Thư viện này cung cấp nhiều tùy chọn tùy chỉnh, cho phép người dùng điều chỉnh các tham số mô hình để tối ưu hóa hiệu suất trong các tập dữ liệu lớn và phức tạp.

```python
import xgboost as xgb
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Tải dữ liệu về nhà ở California
housing = fetch_california_housing()
X, y = housing.data, housing.target

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Tạo mô hình XGBoost
model = xgb.XGBRegressor()

# Huấn luyện mô hình
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình bằng Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.2225899267544737
```

15. LightGBM

- LightGBM (Light Gradient Boosting Machine) là một thư viện mã nguồn mở cho học máy, được phát triển bởi Microsoft, chuyên về thuật toán boosting với hiệu suất cao và tốc độ nhanh.
- LightGBM sử dụng phương pháp gradient boosting để xây dựng các mô hình dự đoán, và nổi bật với khả năng xử lý dữ liệu lớn nhờ vào việc sử dụng thuật toán phân tách tối ưu hóa và giảm thiểu bộ nhớ.
- Nó được ứng dụng rộng rãi trong các bài toán phân loại, hồi quy, và phân tích chuỗi thời gian. LightGBM thường được ưa chuộng trong các cuộc thi khoa học dữ liệu và các dự án yêu cầu tốc độ huấn luyện nhanh và hiệu suất cao trên các tập dữ liệu phức tạp.

```
#!pip install  dask-expr
import lightgbm as lgb
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# Load dữ liệu
data = load_breast_cancer()
X, y = data.data, data.target

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Tạo dataset cho LightGBM
train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test)

# Thiết lập các tham số
params = {
    'objective': 'binary',
    'metric': 'binary_logloss',
    'num_leaves': 31,
    'learning_rate': 0.05,
    'feature_fraction': 0.9
}

# Huấn luyện mô hình
gbm = lgb.train(params,
                train_data,
                num_boost_round=100,
                valid_sets=test_data,
                early_stopping_rounds=10)

# Dự đoán trên tập kiểm tra
y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration)
```

```
4.5.0
```

16. Statsmodels

- Statsmodels là một thư viện mã nguồn mở cho Python, cung cấp các công cụ và phương pháp thống kê để phân tích dữ liệu.
- Nó cho phép người dùng thực hiện các tác vụ như hồi quy tuyến tính, hồi quy logistic, phân tích thời gian, và kiểm định giả thuyết. Statsmodels cung cấp các mô hình thống kê phong phú và dễ sử dụng, giúp người dùng hiểu rõ hơn về mối quan hệ giữa các biến trong dữ liệu.
- Thư viện này thường được sử dụng trong nghiên cứu kinh tế, khoa học xã hội và phân tích dữ liệu để cung cấp các phương pháp phân tích chuyên sâu và trực quan hóa kết quả thống kê.

```python
import statsmodels.api as sm
import pandas as pd
import numpy as np

# Tạo dữ liệu mẫu
np.random.seed(0)
x = np.random.rand(100)
y = 2 * x + 1 + np.random.randn(100)

# Tạo DataFrame
df = pd.DataFrame({'x': x, 'y': y})

# Thêm hằng số vào biến độc lập
X = sm.add_constant(df['x'])

# Xây dựng mô hình hồi quy tuyến tính
model = sm.OLS(df['y'], X)
results = model.fit()

# In kết quả
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.231
Method:                 Least Squares   F-statistic:                     30.79
Date:                Sun, 20 Oct 2024   Prob (F-statistic):           2.45e-07
Time:                        10:14:35   Log-Likelihood:                -141.51
No. Observations:                 100   AIC:                             287.0
Df Residuals:                      98   BIC:                             292.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.2222      0.193      6.323      0.000       0.839       1.606
x              1.9369      0.349      5.549      0.000       1.244       2.630
==============================================================================
Omnibus:                       11.746   Durbin-Watson:                   2.083
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                4.097
Skew:                           0.138   Prob(JB):                        0.129
Kurtosis:                       2.047   Cond. No.                         4.30
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```