

Chương 2:

Học máy và Hệ thống thông minh



Nội dung

- ❖ Học máy cổ điển
- ❖ Học sâu
- ❖ Đánh giá mô hình
- ❖ Hệ thống dựa trên học máy
- ❖ Notebook và thư viện Scikit-learn



Hệ thống dựa trên học máy

- ❖ Hệ thống dựa trên học máy
 - Các giải thuật học máy “quan sát” các ví dụ về người dùng đã làm: các mẫu trong dữ liệu **train**.
 - Học máy giúp cân bằng dữ liệu khác nhau ở đầu vào và tối ưu hóa cho nhiều loại yêu cầu khác nhau ở đầu ra.
 - Học máy có thể cá nhân hóa các quy tắc cho từng người dùng.
 - Học máy cần dữ liệu được ghi lại các tình huống mà người dùng đã trải qua và kết quả họ đã nhận được.

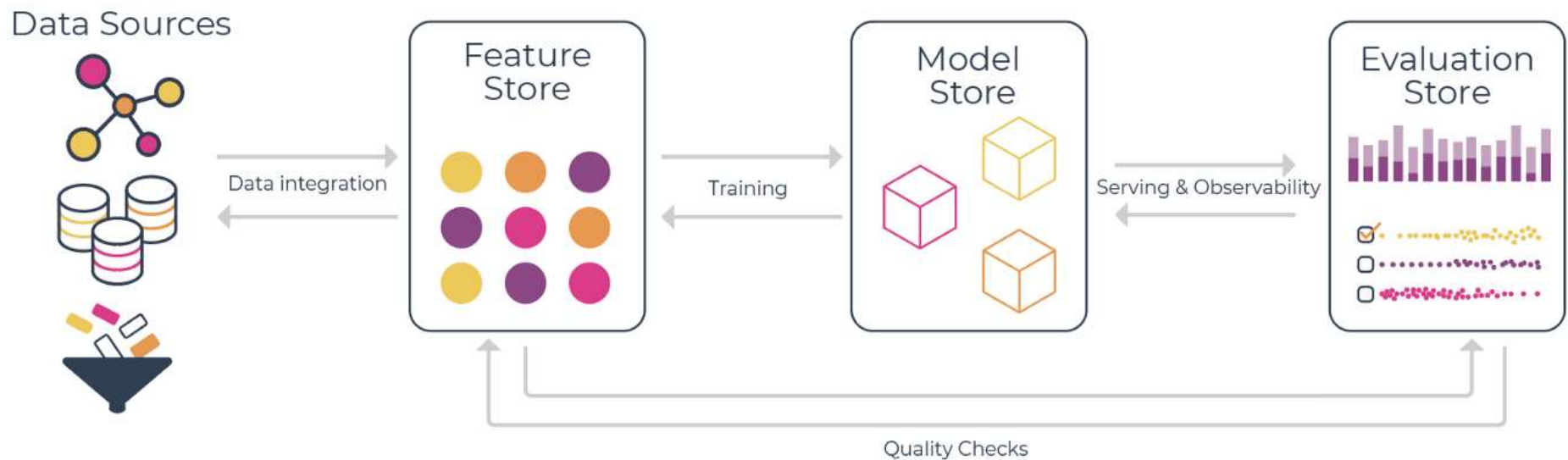


Hệ thống dựa trên học máy

- ❖ Hệ thống dựa trên học máy
 - Các kết quả được ghi nhận bao gồm cả kết quả tốt và kết quả xấu.
 - Dữ liệu ghi nhận có thể mới hoặc cũ, tùy theo mong muốn của người dùng.
 - Học máy có thể được dùng cho một chức năng hoặc tất cả các chức năng của hệ thống.
 - Kết quả thu được từ học máy có thể có sai lầm.

Hệ thống dựa trên học máy

❖ Hệ thống dựa trên học máy



<https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fthe-only-3-ml-tools-you-need-1aa750778d33&psig=AOvVaw2eKyds64bcSikUIFSh2i7H&ust=1665111484696000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCLD585XOyvoCFQAAAAAdAAAAABAD>



Hệ thống dựa trên học máy

❖ Thư viện `scikit-learn`

- Scikit-learn là một thư viện mã nguồn mở Machine Learning viết bằng Python.
- Scikit-learn hiện thực nhiều thuật toán máy học từ các thuật toán cơ bản cho đến các thuật toán phức tạp như DecisionTree, Naive Bayes, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Artificial Neural Network (ANN) ...
- <http://scikit-learn.org/>



Hệ thống dựa trên học máy

❖ Thư viện **scikit-learn**

- **Numpy**: Gói thư viện xử lý dãy số và ma trận nhiều chiều.
- **SciPy**: Gói các hàm tính toán logic khoa học.
- **Matplotlib**: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều.
- **IPython**: Notebook dùng để tương tác trực quan với Python.
- **SymPy**: Gói thư viện các kí tự toán học.
- **Pandas**: Xử lý, phân tích dữ liệu dưới dạng bảng



Thư viện scikit-learn

❖ Giới thiệu:

- Scikit-learn là một thư viện mã nguồn mở Machine Learning viết bằng Python.
- Scikit-learn hiện thực nhiều thuật toán học máy, từ các thuật toán cơ bản cho đến các thuật toán phức tạp như DecisionTree, Naive Bayes, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Artificial Neural Network (ANN) ...
- <http://scikit-learn.org/>



Thư viện scikit-learn

❖ Numpy

- Cài đặt: `pip install numpy`
- `numpy.random.randn`, `numpy.random.randint`
- Dùng gói này để tạo dữ liệu số

✓ Hồi quy (regression):

```
import numpy as np
```

```
rng = np.random.RandomState(0)  
n_samples, n_features = 5, 5  
X = rng.randn(n_samples, n_features)  
y = rng.randn(n_samples)
```

```
print(X) [[ 1.76405235  0.40015721  0.97873798  2.2408932  1.86755799]  
         [-0.97727788  0.95008842 -0.15135721 -0.10321885  0.4105985 ]  
         [ 0.14404357  1.45427351  0.76103773  0.12167502  0.44386323]  
         [ 0.33367433  1.49407907 -0.20515826  0.3130677  -0.85409574]  
         [-2.55298982  0.6536186   0.8644362  -0.74216502  2.26975462]]  
  
print(y) [-1.45436567  0.04575852 -0.18718385  1.53277921  1.46935877]
```



Thư viện scikit-learn

❖ Numpy

- Cài đặt: `pip install numpy`
- `numpy.random.randn`, `numpy.random.randint`
- Dùng gói này để tạo dữ liệu số
- ✓ Phân loại (classification):

```
import numpy as np
```

```
rng = np.random.RandomState(0)  
n_samples, n_features = 5, 5  
X = rng.randn(n_samples, n_features)  
y = rng.randint(0, 2, n_samples)
```

```
print(X) [[ 1.76405235  0.40015721  0.97873798  2.2408932  1.86755799]  
 [-0.97727788  0.95008842 -0.15135721 -0.10321885  0.4105985 ]  
 [ 0.14404357  1.45427351  0.76103773  0.12167502  0.44386323]  
 [ 0.33367433  1.49407907 -0.20515826  0.3130677  -0.85409574]  
 [-2.55298982  0.6536186   0.8644362  -0.74216502  2.26975462]]
```

```
print(y) [0 0 1 1 0]
```



Thư viện scikit-learn

❖ Scipy

- SciPy là thư viện mã nguồn mở với các công cụ khoa học cho Python.
- Cài đặt: `pip install scipy`
- SciPy phụ thuộc vào thư viện NumPy và nó tập hợp nhiều cấp cao lại với nhau thành một gói duy nhất. SciPy cung cấp các mô-đun cho:
 - ✓ Nhập/xuất file (file input/output)
 - ✓ Tối ưu hóa (optimization)
 - ✓ Đại số tuyến tính (linear algebra)
 - ✓ Biến đổi Fouries (Fourier transforms)
 - ✓ Xử lý tín hiệu (signal processing)
 - ✓ ...



Thư viện scikit-learn

❖ Scipy - Linear algebra

- SciPy được tối ưu hóa cho khả năng tính toán đại số tuyến tính nhanh.
- Lớp ma trận được khởi tạo bằng lệnh SciPy **mat** là cách viết tắt thuận tiện cho ma trận.

```
A = matrix('1.0 2.0; 3.0 4.0')
print(A)

/////

[[ 1.  2.]
 [ 3.  4.]
```



Thư viện scikit-learn

❖ Scipy - Linear algebra

➤ Nghịch đảo ma trận

```
from numpy import mat
from scipy.sparse.sputils import matrix

A = mat('[1 3 5; 2 5 1; 2 3 8]')
print(A)
```

```
[[1 3 5]
 [2 5 1]
 [2 3 8]]
```

```
print(A.I)
```

```
[[-1.48  0.36  0.88]
 [ 0.56  0.08 -0.36]
 [ 0.16 -0.12  0.04]]
```

Thư viện scikit-learn

❖ Scipy - Linear algebra

➤ Giải hệ phương trình

$$\begin{aligned}x + 3y + 5z &= 0 \\ 2x + 5y + z &= -7 \\ 2x + 3y + 8z &= 4\end{aligned}$$

➤ $S = A^{-1} B$, với $S = [x \ y \ z]$ và $B = [0 \ -7 \ 4]$

```
A = mat('[1 3 5; 2 5 1; 2 3 8]')  
b = mat('[0;-7;4]')  
print(A.I*b)
```

```
[[ 1.]  
 [-2.]  
 [ 1.]]
```

Thư viện scikit-learn

❖ Scipy - Linear algebra

➤ Tính Determinant (det) của ma trận

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}$$

$$\begin{aligned} |\mathbf{A}| &= 1 \begin{vmatrix} 5 & 1 \\ 3 & 8 \end{vmatrix} - 3 \begin{vmatrix} 2 & 1 \\ 2 & 8 \end{vmatrix} + 5 \begin{vmatrix} 2 & 5 \\ 2 & 3 \end{vmatrix} \\ &= 1(5 \cdot 8 - 3 \cdot 1) - 3(2 \cdot 8 - 2 \cdot 1) + 5(2 \cdot 3 - 2 \cdot 5) = -25. \end{aligned}$$

```
A = mat('[1 3 5; 2 5 1; 2 3 8]')  
print(linalg.det(A))
```

-25.000000000000000004

Thư viện scikit-learn

❖ Scipy - Linear algebra

➤ Hồi quy tuyến tính

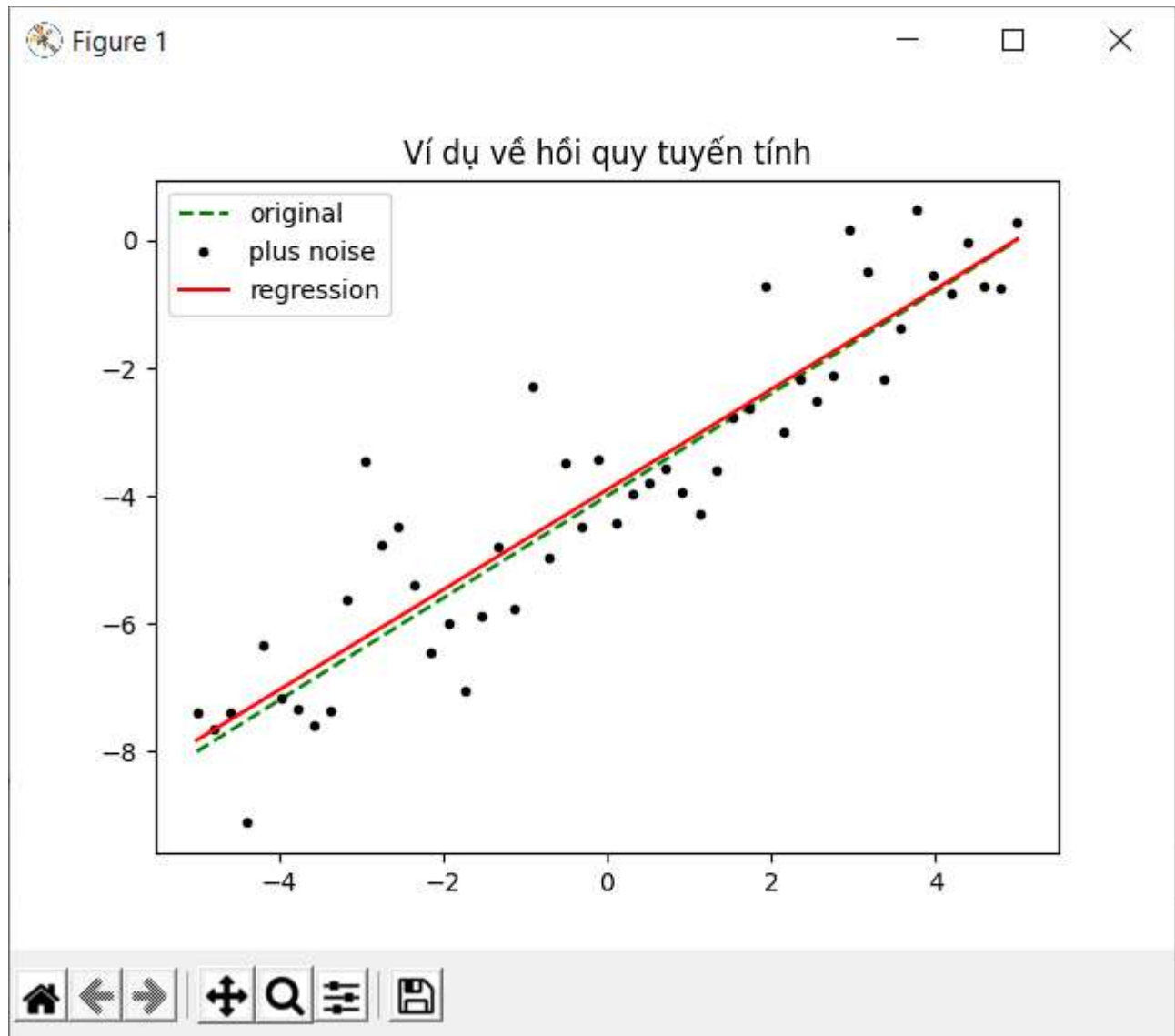
```
import numpy as np
import pylab as plt

n=50      # number of points
x=np.linspace(-5,5,n)      # create x axis data
a, b=0.8, -4
y=np.polyval([a,b],x)
yn=y+np.random.randn(n)      # add some noise
(ar,br)=np.polyfit(x,yn,1)
yr=np.polyval([ar,br],x)
err=np.sqrt(sum((yr-yn)**2)/n) # compute the mean square error
print('Hồi quy tuyến tính bằng polyfit')
print('Tham số ban đầu: a=%.2f b=%.2f' % (a,b))
print('Hồi quy: a=%.2f b=%.2f, ms error= %.3f' % (ar,br,err))
plt.title('Ví dụ về hồi quy tuyến tính')
plt.plot(x,y,'g--')
plt.plot(x,yn,'k.')
plt.plot(x,yr,'r-')
plt.legend(['original', 'plus noise', 'regression'])
plt.show()
```


Thư viện scikit-learn

❖ Scipy - Linear algebra

➤ Hồi quy tuyến tính





Thư viện scikit-learn

➤ SymPy

➤ Gói toán học dạng tượng trưng (symbolic)

➤ Cài đặt: `pip install sympy`

- `x,y = symbols('x y') # khai báo`

- `a = 2*x + y`

- `a + y`

- `expand(a**3) # khai triển`

- `factor(x**3+3*x**2+3*x+1) # phân tích thành nhân tử`

- `simplify(x**2-y**2-(x+y)*(x-y)) # đơn giản hóa`



Thư viện scikit-learn

- ❖ **Simpy** - Gói toán học dạng tượng trưng (symbolic)
 - Phép thay thế (substitution):

```
x,y = symbols('x y')  
expr = cos(x) + 1  
expr = expr.subs(x,y)  
print(expr)
```

$\cos(y) + 1$

```
x,y = symbols('x y')  
expr = cos(x) + 1  
expr = expr.subs(x,0)  
print(expr)
```

2



Thư viện scikit-learn

❖ Simpy

- Gói toán học dạng tượng trưng (symbolic)
- ✓ Phép thay thế (substitution):

```
expr = x**y  
expr = expr.subs(y, x**y)  
print(expr)
```

```
x**(x**y)
```



Thư viện scikit-learn

❖ SymPy

- Gói toán học dạng tượng trưng (symbolic)
 - ✓ Lượng giác (trig)

```
x, y = symbols('x y')  
trigx = sin(2*x) + cos(2*x)  
print(expand_trig(trigx))
```

$$2*\sin(x)*\cos(x) + 2*\cos(x)**2 - 1$$

```
x, y = symbols('x y')  
print(trigsimp(cos(x)**2 + sin(x)**2))
```

$$1$$



Thư viện scikit-learn

❖ Pandas

- Cài đặt:
- ✓ `pip install pandas`
- Chuyển các mảng numpy thành dataframe
- Có thể thực hiện toàn bộ quy trình phân tích dữ liệu
- Sử dụng:
- ✓ `import pandas as pd`

Hoặc

- ✓ `from pandas import DataFrame, Series`

Thư viện scikit-learn

❖ Pandas - Series:

- Là đối tượng giống như mảng một chiều chứa một mảng dữ liệu (của bất kỳ kiểu dữ liệu NumPy nào) và một mảng nhãn dữ liệu được liên kết, được gọi là chỉ mục (index).

```
obj = pd.Series([4, 7, -5, 3])  
print(obj)
```

0	4
1	7
2	-5
3	3
dtype: int64	

```
obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])  
print(obj2)
```

d	4
b	7
a	-5
c	3

Thư viện scikit-learn

❖ Pandas - Series:

- Truy xuất riêng phần giá trị bằng lệnh: *.values*

- ✓ `obj[2]` # -6

- ✓ `obj[[0, 1, 3]]`

0	4
1	7
3	3

- Truy xuất riêng phần chỉ số bằng lệnh: *.index*

- ✓ `obj.index`

RangeIndex(start=0, stop=4, step=1)

- Tính toán trên toàn *series*

- ✓ `obj * 3`

0	12
1	21
2	-15
3	9

Thư viện scikit-learn

❖ Pandas - DataFrame:

- Cấu trúc dữ liệu dạng bảng, giống như bảng tính, chứa một tập các cột được sắp xếp.
- có thể chứa nhiều kiểu dữ liệu khác nhau (số, chuỗi, v.v.)
- Có cả chỉ mục hàng và cột.
- Có thể được coi là một 'dict of Series'

```
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],  
        'year': [2000, 2001, 2002, 2001, 2002],  
        'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}  
frame = pd.DataFrame(data)  
print(frame)
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9



Thư viện scikit-learn

❖ Pandas - DataFrame:

- Một cột trong **DataFrame** có thể được truy xuất theo tên gọi hoặc như một thuộc tính thuộc tính

- `print(frame['state'])`

0	Ohio
1	Ohio
2	Ohio
3	Nevada
4	Nevada

- `print(frame.state)`

0	Ohio
1	Ohio
2	Ohio
3	Nevada
4	Nevada



Thư viện scikit-learn

❖ Pandas - DataFrame:

➤ Một hàng trong DataFrame có thể được truy xuất bằng index

✓ `print(DataFrame(frame, index = [2]))`

	state	year	pop
2	Ohio	2002	3.6

➤ Truy xuất từng phần tử trong DataFrame: dựa theo cột/chỉ số

✓ `print(frame['state'][3])`

Nevada

Thư viện scikit-learn

❖ Pandas - DataFrame:

➤ Thêm một cột dữ liệu

```
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],  
        'year': [2000, 2001, 2002, 2001, 2002],  
        'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}  
  
frame = pd.DataFrame(data)  
frame['salary'] = 2000  
print(frame)
```

	state	year	pop	salary
0	Ohio	2000	1.5	2000
1	Ohio	2001	1.7	2000
2	Ohio	2002	3.6	2000
3	Nevada	2001	2.4	2000
4	Nevada	2002	2.9	2000

Thư viện scikit-learn

❖ Pandas - DataFrame:

➤ Chuyển đổi dòng – cột (ma trận chuyển vị)

```
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],  
        'year': [2000, 2001, 2002, 2001, 2002],  
        'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}  
  
frame = pd.DataFrame(data)  
print(frame)  
print(frame.T)
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9



	0	1	2	3	4
state	Ohio	Ohio	Ohio	Nevada	Nevada
year	2000	2001	2002	2001	2002
pop	1.5	1.7	3.6	2.4	2.9



❖ Pandas - DataFrame:

➤ Đọc dữ liệu từ các nguồn khác và tạo DataFrame

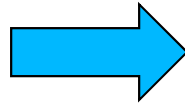
- ✓ `pd.read_csv(...)`
- ✓ `pd.read_excel(...)`
- ✓ `pd.read_html(...)`
- ✓ `pd.read_table(...)`
- ✓ `pd.read_clipboard()`

Thư viện scikit-learn

❖ Pandas - DataFrame:

```
Ex = pd.read_excel('D:\Jobs\HV BCVT\Python - AI\Thuc hanh\Test.xlsx', 'Sheet1')  
print(Ex)
```

Stt	Att1	Att2
1	a	A
2	b	B
3	c	C
4	d	D
5	e	E
6	f	F
7	g	G
8	h	H
9	i	I
10	j	J



	Stt	Att1	Att2
0	1	a	A
1	2	b	B
2	3	c	C
3	4	d	D
4	5	e	E
5	6	f	F
6	7	g	G
7	8	h	H
8	9	i	I
9	10	j	J
10	11	k	K

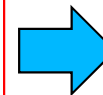
Thư viện scikit-learn

❖ Pandas - DataFrame:

➤ Lọc dữ liệu

```
Ex = pd.read_excel('D:\Jobs\HV BCVT\Python - AI\Thuc hanh\Test.xlsx', 'Sheet1')
print(Ex)
Ex_filter = Ex[(Ex['Stt'] < 4) |
               (Ex['Stt'] > 7)]
print(Ex_filter)
```

	Stt	Att1	Att2
0	1	a	A
1	2	b	B
2	3	c	C
3	4	d	D
4	5	e	E
5	6	f	F
6	7	g	G
7	8	h	H
8	9	i	I
9	10	j	J
10	11	k	K



	Stt	Att1	Att2
0	1	a	A
1	2	b	B
2	3	c	C
7	8	h	H
8	9	i	I
9	10	j	J
10	11	k	K

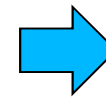
Thư viện scikit-learn

❖ Pandas - DataFrame:

➤ Tạo câu truy vấn để lọc dữ liệu

```
Ex = pd.read_excel('D:\Jobs\HV BCVT\Python - AI\Thuc hanh\Test.xlsx', 'Sheet1')
print(Ex)
sql_str = "Stt < 4 or Stt > 7 and Att1 < 'i'"
Ex_filter = Ex.query(sql_str)
print(Ex_filter)
```

	Stt	Att1	Att2
0	1	a	A
1	2	b	B
2	3	c	C
3	4	d	D
4	5	e	E
5	6	f	F
6	7	g	G
7	8	h	H
8	9	i	I
9	10	j	J
10	11	k	K



	Stt	Att1	Att2
0	1	a	A
1	2	b	B
2	3	c	C
7	8	h	H



Thư viện scikit-learn

- ❖ Sử dụng thư viện **scikit-learn**
 - Ví dụ về dùng scikit-learn nhận dạng ký số 0-9 viết tay

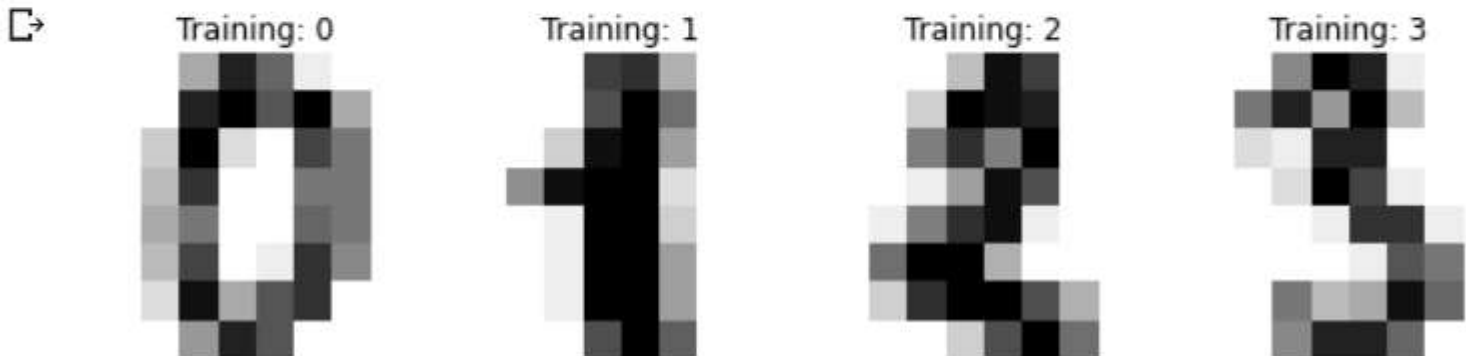
```
✓ [3] # Author: Gael Varoquaux <gael dot varoquaux at normalesup dot org>  
0s   # License: BSD 3 clause  
  
# Standard scientific Python imports  
import matplotlib.pyplot as plt  
  
# Import datasets, classifiers and performance metrics  
from sklearn import datasets, svm, metrics  
from sklearn.model_selection import train_test_split
```

Thư viện scikit-learn

- ❖ Sử dụng thư viện **scikit-learn**
 - Ví dụ về dùng scikit-learn nhận dạng ký số 0-9 viết tay

```
✓ 0s ▶ digits = datasets.load_digits()

_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" % label)
```





Thư viện scikit-learn

❖ Sử dụng thư viện **scikit-learn**

- Các bước thực hiện:
- ✓ Làm phẳng các hình ảnh, chuyển từng mảng 2-D các giá trị chỉ độ xám từ ma trận (8, 8) thành ma trận (64,).
- ✓ Xử lý tương tự cho toàn bộ tập dữ liệu: `n_samples` là số lượng hình ảnh, `n_features` là tổng số pixel trong mỗi hình ảnh.
- ✓ Chia dữ liệu thành tập huấn luyện và tập test cho giải thuật SVM.
- ✓ Chạy giải thuật và dự đoán.

```
✓ 0s # flatten the images
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
clf = svm.SVC(gamma=0.001)

# Split data into 50% train and 50% test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False
)

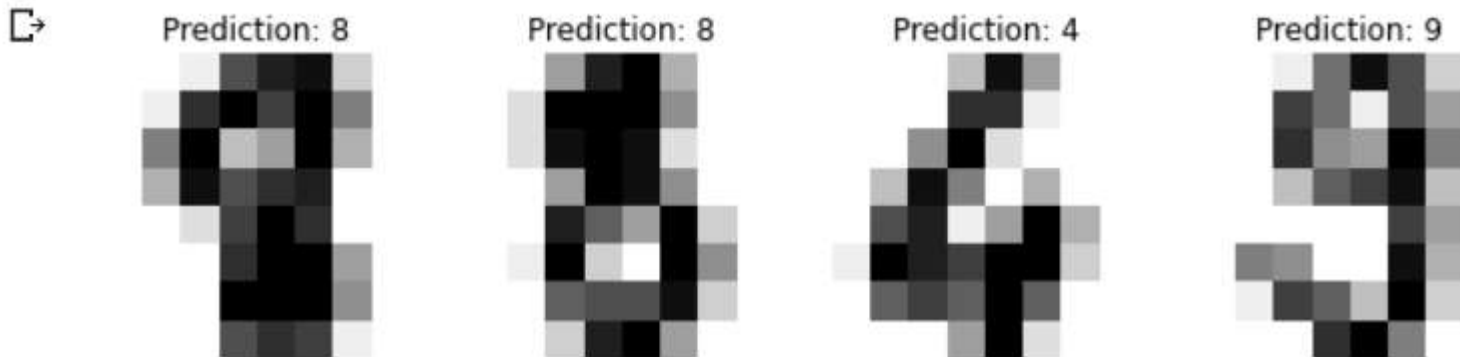
# Learn the digits on the train subset
clf.fit(X_train, y_train)

# Predict the value of the digit on the test subset
predicted = clf.predict(X_test)
```

Thư viện scikit-learn

- ❖ Sử dụng thư viện **scikit-learn**
 - Biểu diễn 4 mẫu thử đầu tiên và hiển thị giá trị chữ số dự đoán của chúng trong tiêu đề.

```
✓ 0s ▶ _, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
    for ax, image, prediction in zip(axes, X_test, predicted):
        ax.set_axis_off()
        image = image.reshape(8, 8)
        ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
        ax.set_title(f"Prediction: {prediction}")
```



Thư viện scikit-learn

❖ Sử dụng thư viện **scikit-learn**

function:~sklearn
.metrics.classification_report:
xây dựng một báo cáo văn bản hiển thị các chỉ số phân loại chính.

✓
0s



```
print(  
    f"Classification report for classifier {clf}:\n"  
    f"{metrics.classification_report(y_test, predicted)}\n"  
)
```



Classification report for classifier SVC(gamma=0.001):

	precision	recall	f1-score	support
0	1.00	0.99	0.99	88
1	0.99	0.97	0.98	91
2	0.99	0.99	0.99	86
3	0.98	0.87	0.92	91
4	0.99	0.96	0.97	92
5	0.95	0.97	0.96	91
6	0.99	0.99	0.99	91
7	0.96	0.99	0.97	89
8	0.94	1.00	0.97	88
9	0.93	0.98	0.95	92
accuracy			0.97	899
macro avg	0.97	0.97	0.97	899
weighted avg	0.97	0.97	0.97	899

}

Thư viện scikit-learn

❖ Sử dụng thư viện **scikit-learn**

- Vẽ ma trận nhầm lẫn <**confusion_matrix**> của các giá trị chữ số thực và giá trị chữ số dự đoán.

✓
1s

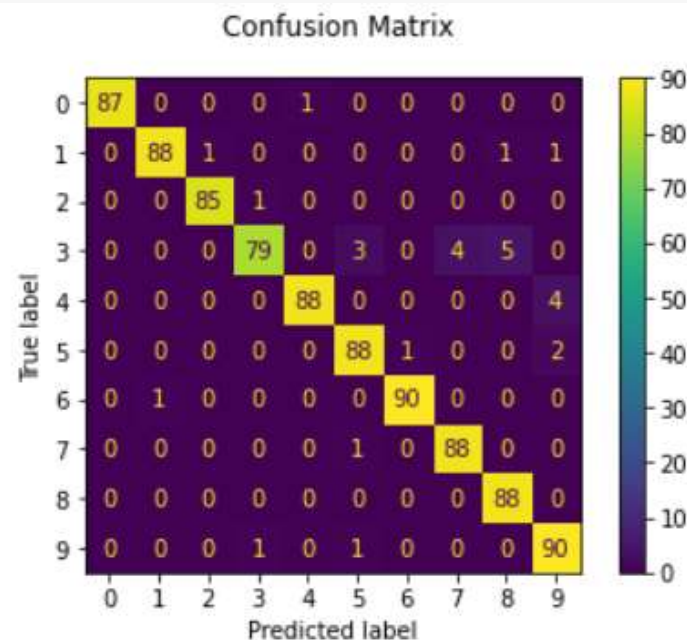


```
disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")

plt.show()
```

Confusion matrix:

```
[[87  0  0  0  1  0  0  0  0  0]
 [ 0 88  1  0  0  0  0  0  1  1]
 [ 0  0 85  1  0  0  0  0  0  0]
 [ 0  0  0 79  0  3  0  4  5  0]
 [ 0  0  0  0 88  0  0  0  0  4]
 [ 0  0  0  0  0 88  1  0  0  2]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  0  0  0  0  1  0 88  0  0]
 [ 0  0  0  0  0  0  0  0 88  0]
 [ 0  0  0  1  0  1  0  0  0 90]]
```





Thư viện scikit-learn

- ❖ Sử dụng thư viện **Scikit-learn**
 - ❖ Phân loại văn bản bằng SVM
 - Tính đặc trưng văn bản dựa trên tần suất từ
 - Các từ xuất hiện nhiều lần trong văn bản thì sẽ quan trọng hơn các từ xuất hiện ít lần
 - Các từ xuất trong ít văn bản thì có tính phân biệt văn bản cao hơn từ xuất hiện nhiều.



Thư viện scikit-learn

- ❖ Tính đặc trưng văn bản dựa trên tần suất từ
 - Tần suất từ (*term frequency*)
 - ✓ Tần suất của một từ w trong văn bản d , ký hiệu $TF(w, d)$, là số lần xuất hiện của từ w trong văn bản d .
 - Tần suất văn bản (*document frequency*)
 - ✓ Tần suất văn bản của một từ w , ký hiệu $DF(w)$, là số lượng văn bản mà từ w có xuất hiện.
 - ✓ Nghịch đảo của tần suất văn bản (*inverse document frequency*) của một từ w , ký hiệu $IDF(w)$ được cho bởi công thức:

$$IDF(w) = 1 + \log(|D| / DF(w))$$



Thư viện scikit-learn

- ❖ Tính đặc trưng văn bản dựa trên tần suất từ
 - Tần suất TF-IDF (*term document frequency*)
 - ✓ Kết hợp hai loại tần suất thực thể và tần suất văn bản:

$$TF-IDF(w, d) = TF(w, d) * IDF(w)$$



Thư viện scikit-learn

- ❖ Tính đặc trưng văn bản dựa trên tần suất từ
 - ✓ Chỉ số **$TF(w)$** của một từ w cao khi từ đó xuất hiện nhiều lần trong văn bản. Tức là, nội dung của nó trong văn bản có giá trị cao.
 - ✓ Chỉ số **$IDF(w)$** của một từ w cao nếu từ đó xuất hiện trong một số ít văn bản. Tức là từ đó có giá trị phân biệt văn bản cao.
 - ✓ Các từ có giá trị **$TF-IDF(w,d)$** cao sẽ đặc trưng cho một văn bản.



Thư viện scikit-learn

❖ Dữ liệu

- Chẩn đoán bệnh theo các triệu chứng.
- Số bệnh được chẩn đoán (nhãn văn bản):
 - ✓ Bệnh hạ huyết áp: 149 mẫu
 - ✓ Bệnh viêm đường ruột: 125 mẫu
 - ✓ Chưa xác định: 589 mẫu

Thư viện scikit-learn

❖ Dữ liệu

id	text	polarity
19	buồn nôn , choáng váng, mờ mắt, ngất, run tay, đổ mồ hôi, mệt mỏi, đau đầu, gõ vang	bệnh hạ huyết áp
20	sốt, chán ăn, đi tiêu ra máu, sóng y xuống lờm sâu	chưa xác định
21	sốt, chán ăn, đi tiêu ra máu, tiêu chảy, sụt cân, đau nhức cơ, buồn nôn , dấu hiệu hutchinson	bệnh viêm đường ruột
22	sốt, chán ăn, đi tiêu ra máu, tiêu chảy, tiếng tim tách đôi: tách đôi sinh lý	bệnh viêm đường ruột
23	sốt, chóng mặt, bệnh võng mạc do tăng huyết áp	chưa xác định
24	sốt, co kéo cơ hô hấp phụ	chưa xác định
25	sốt, đau bụng, chán ăn, đi tiêu ra máu, tiếng t2: mờ	bệnh viêm đường ruột
26	sốt, đau bụng, chán ăn, jvp: dạng sóng bình thường	chưa xác định
27	sốt, đau bụng, tiếng cọ màng phổi	chưa xác định
28	sốt, đau nhức cơ, buồn nôn , chóng mặt, dấu hiệu leser- trélat	chưa xác định
29	sốt, đau nhức cơ, buồn nôn , phù ngoại vi	chưa xác định
30	sốt, đau nhức cơ, xanh tím trung tâm	chưa xác định
31	sốt, đi tiêu ra máu, tiêu chảy, âm thổi - tâm thu: hẹp van ĐM phổi	chưa xác định
32	sốt, nôn, co kéo khoang gian sườn	bệnh viêm đường ruột
33	sốt, nôn, đau bụng, bệnh võng mạc do tăng huyết áp: bắt chéo động - tĩnh mạch	chưa xác định
34	sốt, nôn, đau bụng, chán ăn, đi tiêu ra máu, tiêu chảy, sụt cân, đau nhức cơ, giảm trương lực	bệnh viêm đường ruột



Thư viện scikit-learn

❖ Khai báo thư viện sử dụng

```
import pickle
import pandas as pd
from sklearn.metrics import accuracy_score
# Tách bộ dữ liệu thành train và test
from sklearn.model_selection import train_test_split
# chuyển chuỗi ký tự thành vector số
from sklearn.feature_extraction.text import CountVectorizer
# dùng TF IDF chuyển dữ liệu chuỗi thành vector số
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.svm import SVC # Support Vector Machine
from sklearn.pipeline import Pipeline # dùng pipeline
# from gensim import parsing # To stem data
```

Thư viện scikit-learn

❖ Đọc dữ liệu từ file Excel

```
train_data = "D:\\Jobs\\HV BCVT\\Python - AI\\Thực hành\\corpus.xls"
# Đọc dữ liệu từ file Excel vào dataframe
df = pd.read_excel(train_data, 'Sheet1')
'''
# Dùng cho tiếng Anh trong việc xử lý các từ như 'trying' thành "try",
# dùng thư viện gensim
def parse(s):
    parsing.stem_text(s)
    return s

# Sử dụng parse.
for i in range(0, len(df)):
    df.iloc[i, 2] = parse(df.iloc[i, 2])
'''
```




Thư viện scikit-learn

- ❖ Xử lý dữ liệu và thiết lập tham số cho giải thuật

```
# Tách phần nội dung và phần nhãn lớp của dữ liệu
X, y = df['text'].tolist(), df['polarity'].tolist()

# Tách dữ liệu vào các phần train và test.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = Pipeline([('vect', CountVectorizer()), ('tfidf',
                                                    TfidfTransformer()),
                  ('clf', SVC(kernel='rbf', C=1.0))])
```




Thư viện scikit-learn

- ❖ Thực hiện thử nghiệm trên dữ liệu test và kết quả
 - Thử nghiệm

```
# Dự đoán lớp trên dữ liệu test
predicted = model.predict(X_test)
print("predicted Accuracy Score -> ", accuracy_score(predicted, y_test)*100)
```

- Kết quả

```
C:\Users\Study\anaconda3\python.exe C:/Projects/Python4AI/SVM.py
predicted Accuracy Score -> 87.86127167630057

Process finished with exit code 0
```



Thư viện scikit-learn

- ❖ Thực hiện quá trình huấn luyện và lưu lại mô hình
 - Mô hình lưu lại để sử dụng cho ứng dụng mà không cần huấn luyện lại

```
# train model
model.fit(X_train, y_train)

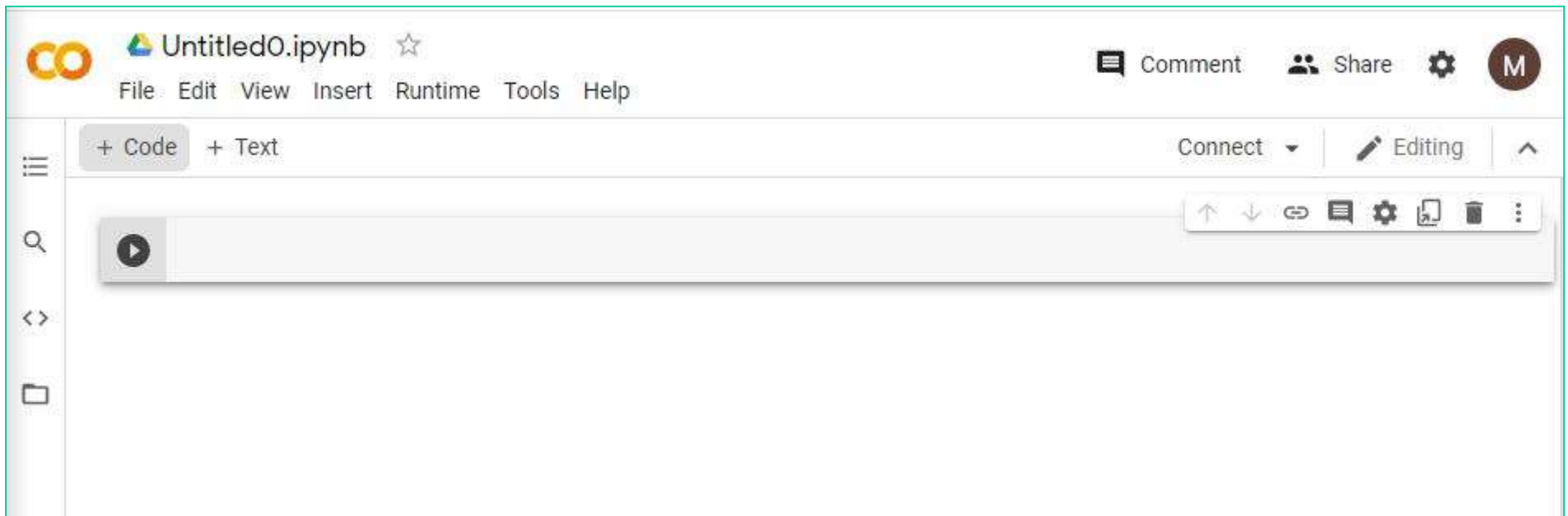
# Lưu lại model vào file
file_name = 'D:\\Jobs\\HV BCVT\\Python - AI\\Thực hành\\chan_doan.sav'
pickle.dump(model, open(file_name, 'wb'))
```

❖ Notebook:

- Google colab là một tiện ích của google, là một dịch vụ trên cloud, miễn phí.
- Colab không yêu cầu thiết lập phức tạp. Các notebook mà ta tạo có thể được các thành viên trong nhóm chỉnh sửa đồng thời - theo cách tương tự như ta chỉnh sửa tài liệu trong Google docs.
- Ưu điểm lớn nhất là Colab hỗ trợ hầu hết các thư viện học máy phổ biến và có thể dễ dàng khi cài đặt một thư viện mới.

Notebook

❖ Notebook:



- ❖ Notebook:
- ❖ Kết nối Colab với Google Drive

```
[ ] from google.colab import drive  
    drive.mount('/content/gdrive')
```

➞ Go to this URL in a browser: <https://accounts.google.com/o/>

Enter your authorization code:

.....

Mounted at /content/gdrive

❖ Notebook:

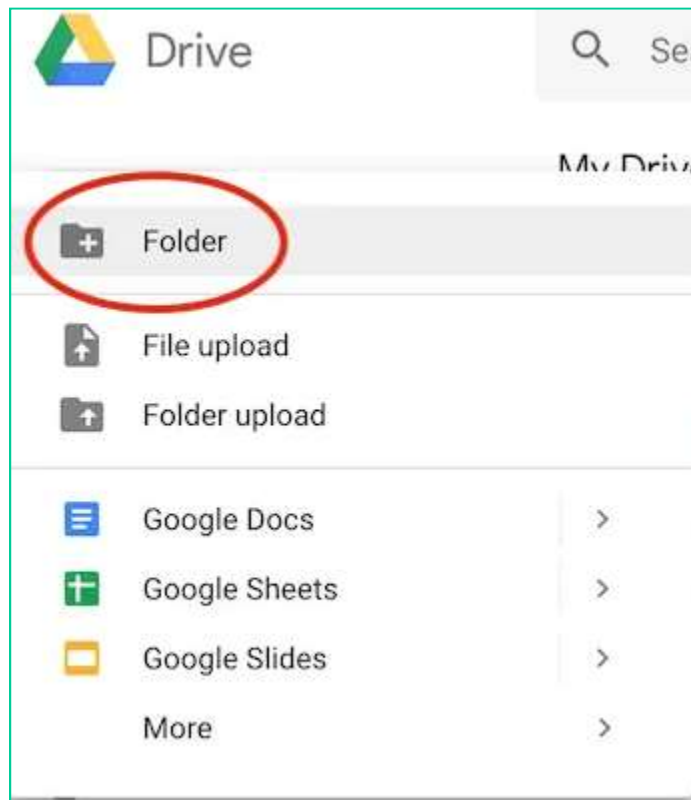
- Viết và thực thi mã bằng Python
- Tạo / Tải lên / Chia sẻ các file notebook
- Import/ save notebook vào Google Drive
- Import các notebooks từ GitHub
- Tích hợp sẵn [PyTorch](#), [TensorFlow](#), [Keras](#), [OpenCV](#)
- Dịch vụ cloud miễn phí với GPU miễn phí, có thể nâng cấp GPU bằng cách sử dụng GPU trả phí.

❖ Notebook:

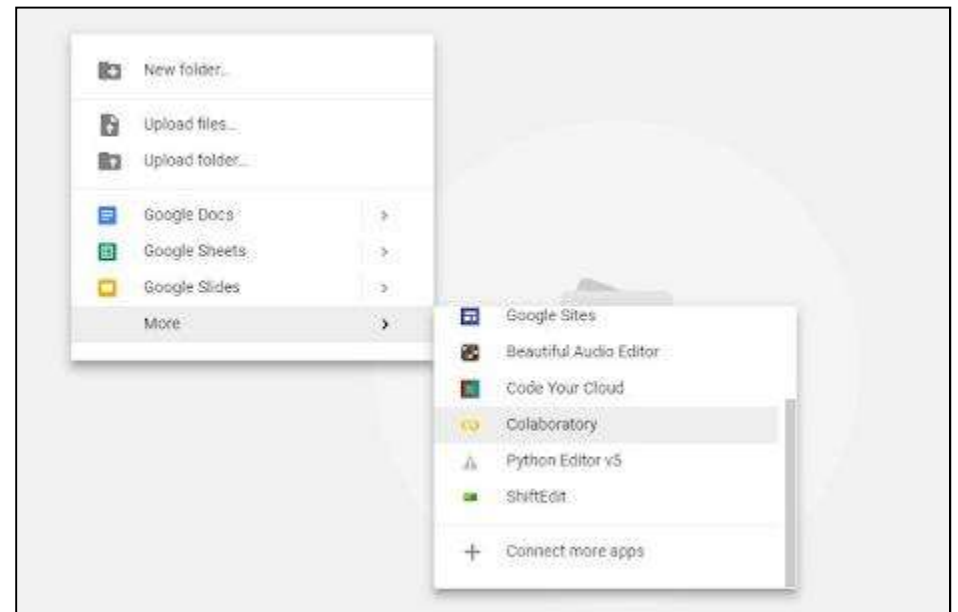
- Để sử dụng [Colaboratory](#), ta phải có tài khoản Google và sau đó truy cập [Colaboratory](#) bằng tài khoản đó. Nếu không, hầu hết các tính năng của [Colaboratory](#) sẽ không hoạt động.
- Để tạo một file google colab đầu tiên ta truy cập vào folder trên google driver nơi ta muốn tạo file colab sau đó ta nhấn chuột phải sau đó chọn vào tùy chọn “[Colaboratory](#)”:

Notebook

❖ Notebook:



Tạo thư mục trên Google Drive



Tạo Notebook trên Colab

Notebook

❖ Notebook:

- Chọn dung GPU:
- ✓ Chọn “runtime”
- ✓ Chọn “change runtime type”
- ✓ Chọn “GPU” từ mục “hardware accelerator”

Notebook settings

Hardware accelerator
GPU ⌵ ?

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Background execution

Want your notebook to keep running even after you close your browser? [Upgrade to Colab Pro+](#)

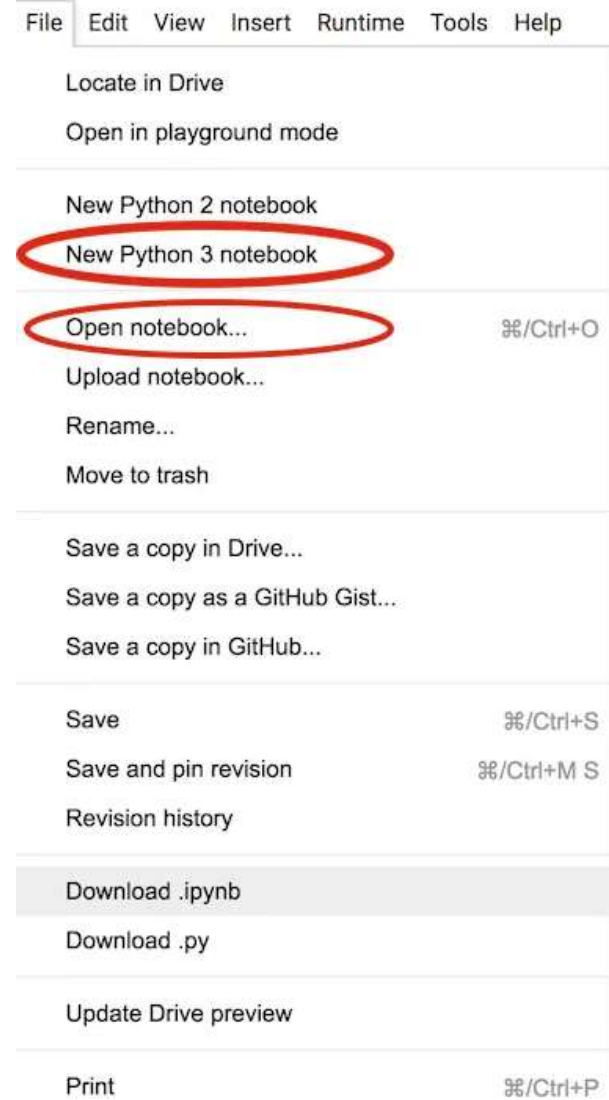
☐ Omit code cell output when saving this notebook

Cancel [Save](#)

Notebook

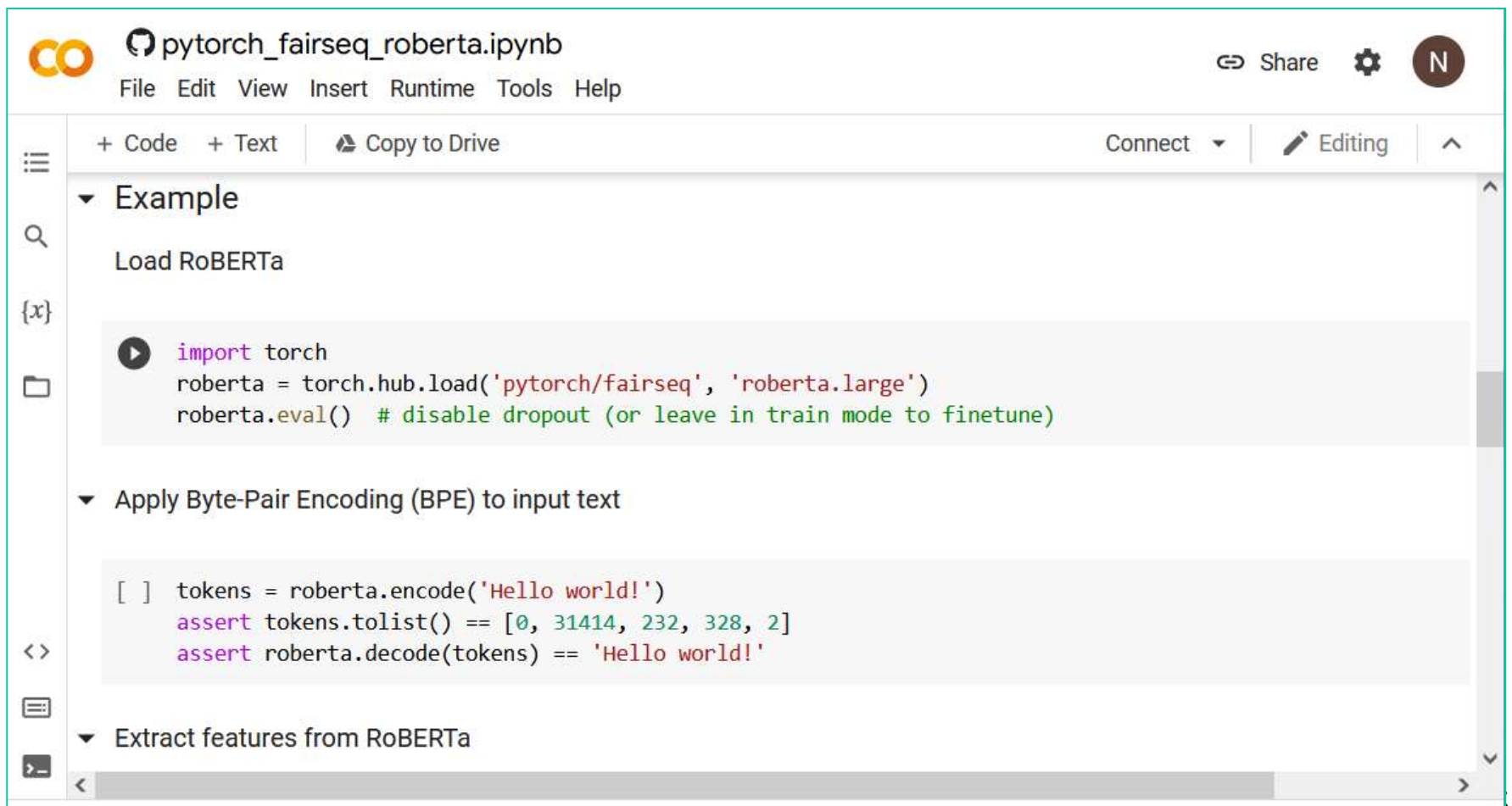
❖ Notebook:

- Mở lại hoặc tạo mới một notebook



Notebook

❖ Notebook:



The screenshot displays a Jupyter Notebook titled 'pytorch_fairseq_roberta.ipynb'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'Share', 'Settings', and a user profile icon labeled 'N'. Below the menu, a toolbar shows '+ Code', '+ Text', 'Copy to Drive', 'Connect', 'Editing', and a scroll indicator. The notebook content is organized into sections: 'Example', 'Load RoBERTa', 'Apply Byte-Pair Encoding (BPE) to input text', and 'Extract features from RoBERTa'. The 'Load RoBERTa' section contains a code cell with the following Python code:

```
import torch
roberta = torch.hub.load('pytorch/fairseq', 'roberta.large')
roberta.eval() # disable dropout (or leave in train mode to finetune)
```

The 'Apply Byte-Pair Encoding (BPE) to input text' section contains a code cell with the following Python code:

```
[ ] tokens = roberta.encode('Hello world!')
assert tokens.tolist() == [0, 31414, 232, 328, 2]
assert roberta.decode(tokens) == 'Hello world!'
```

The 'Extract features from RoBERTa' section is currently empty. The left sidebar shows a file explorer with a folder icon and a search icon.