

# PHÂN HIỆU TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TẠI TP. HỒ CHÍ MINH

## BỘ MÔN CÔNG NGHỆ THÔNG TIN

---□□□---



### KHAI PHÁ DỮ LIỆU

## BÁO CÁO THỰC HÀNH BUỔI 3

<Họ tên sinh viên> - <MSSV>

<Lớp>

GVHD: Ths. Nguyễn Thiện Dương

*TP. Hồ Chí Minh - 2025*

**Lưu ý:**

1. Số trang trong tài liệu thực hành tính theo số trang ứng dụng đọc PDF đếm
2. SV bố trí mỗi câu ở dưới sẽ là 1 file mã nguồn riêng (file.ipynb)
3. Những câu nào có yêu cầu lập trình viết mã nguồn thì SV phải in ra họ tên – MSSV (lệnh print) trong câu đó
4. SV đặt tên cho file mã nguồn mỗi câu như sau: <MSSV>\_LAB3\_Bai<X>
5. Tắt cả các câu sau đó sẽ tổng hợp lại và push lên 1 repository duy nhất trên Github
6. Đặt tên cho repository theo cú pháp: <MSSV>\_Lab23

## BÁO CÁO THỰC HÀNH

[https://github.com/Giamy1901/-6351071046-\\_LAB3\\_N2](https://github.com/Giamy1901/-6351071046-_LAB3_N2)

### Bài 1 (Trang 1)



```
# Câu a/
from sklearn.datasets import load_iris

iris = load_iris()
data = iris.data
target = iris.target
print("Nguyễn Lê Gia Mỹ - 6351071046")

...
Nguyễn Lê Gia Mỹ - 6351071046
```

```
import pandas as pd

iris_data = pd.DataFrame(data, columns=iris.feature_names)
target_labels = target

print(f"Thuộc tính của dữ liệu: {iris.feature_names}")
print(f"Danh sách lớp mục tiêu: {iris.target_names}")

print("\nKích thước của dữ liệu:", data.shape)
print("Kích thước của nhãn mục tiêu:", target.shape)

print("\n5 mẫu đầu tiên:")
print(data[:5])
print("Nhãn của các mẫu đầu tiên:", target[:5])

print("\nThống kê mô tả của dữ liệu:")
print(iris_data.describe())

print("Nguyễn Lê Gia Mỹ - 6351071046")
```

Thuộc tính của dữ liệu: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']  
 ... Danh sách lớp mục tiêu: ['setosa' 'versicolor' 'virginica']

Kích thước của dữ liệu: (150, 4)  
 Kích thước của nhãn mục tiêu: (150,)

5 mẫu đầu tiên:  
 [[5.1 3.5 1.4 0.2]  
 [4.9 3. 1.4 0.2]  
 [4.7 3.2 1.3 0.2]  
 [4.6 3.1 1.5 0.2]  
 [5. 3.6 1.4 0.2]]  
 Nhãn của các mẫu đầu tiên: [0 0 0 0 0]

Thống kê mô tả của dữ liệu:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	0.828066
std	0.828066	0.435866	1.765298	0.828066
min	4.300000	2.000000	1.000000	4.300000
25%	5.100000	2.800000	1.600000	5.100000
50%	5.800000	3.000000	4.350000	5.800000
75%	6.400000	3.300000	5.100000	6.400000
max	7.900000	4.400000	6.900000	7.900000

	petal width (cm)
count	150.000000
mean	1.199333
std	0.762238
min	0.100000
25%	0.300000
50%	1.300000
75%	1.800000
max	2.500000

Nguyễn Lê Gia Mỹ - 6351071046

```
▶ import numpy as np

X_data = data.copy().astype(float)
Y_target = target

# Chuẩn hóa theo từng cột
X_data = X_data / X_data.max(axis=0)
X_data[:5]
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

... Nguyễn Lê Gia Mỹ - 6351071046

```
from sklearn.model_selection import train_test_split

X_train_set, X_test_set, y_train_set, y_test_set = train_test_split(
    X_data, Y_target, test_size=0.2, random_state=42, shuffle=True
)
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

Nguyễn Lê Gia Mỹ - 6351071046

```
▶ from sklearn.tree import DecisionTreeClassifier
  from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
  import seaborn as sns
  import matplotlib.pyplot as plt

  decision_tree = DecisionTreeClassifier()
  decision_tree.fit(X_train_set, y_train_set)

  y_pred_dt = decision_tree.predict(X_test_set)

  print("Accuracy Decision Tree:", accuracy_score(y_test_set, y_pred_dt))
  print("\nClassification Report:\n", classification_report(y_test_set, y_pred_dt))

  cm = confusion_matrix(y_test_set, y_pred_dt)
  sns.heatmap(cm, annot=True, cmap="YlGnBu", fmt="g")
  plt.title("Decision Tree - Confusion Matrix")
  plt.show()

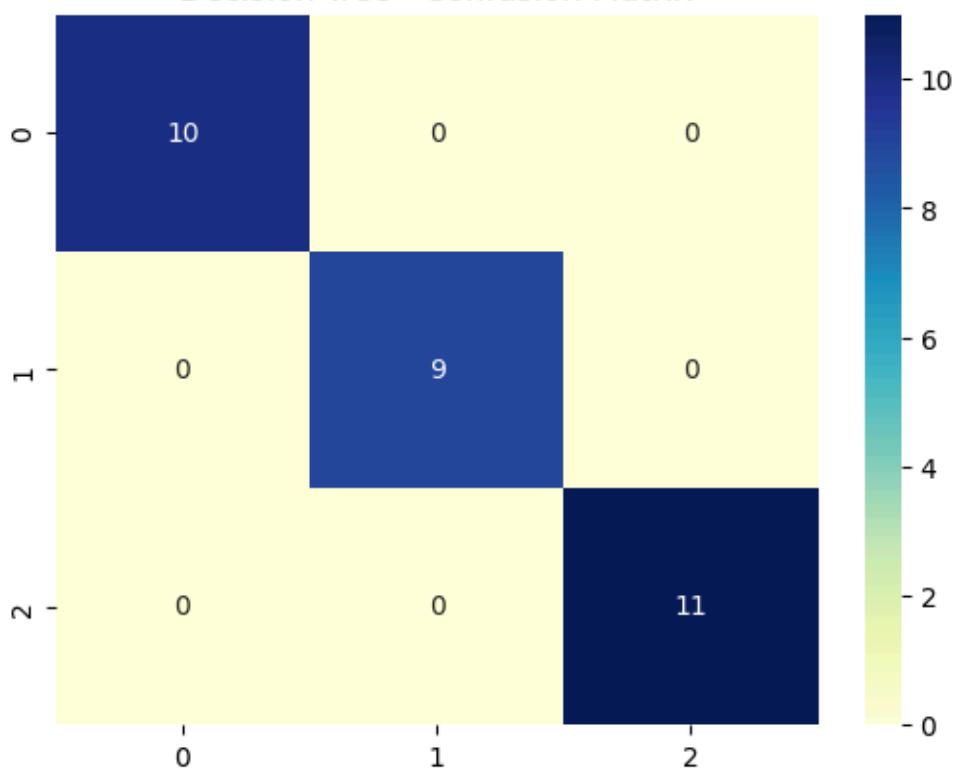
  print("Nguyễn Lê Gia Mỹ - 6351071046")
```

\*\*\* Accuracy Decision Tree: 1.0

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Decision Tree - Confusion Matrix



Nguyễn Lê Gia Mỹ - 6351071046

---

```
# Câu f/
from sklearn.naive_bayes import GaussianNB

naive_bayes = GaussianNB()
naive_bayes.fit(X_train_set, y_train_set)

y_pred_nb = naive_bayes.predict(X_test_set)

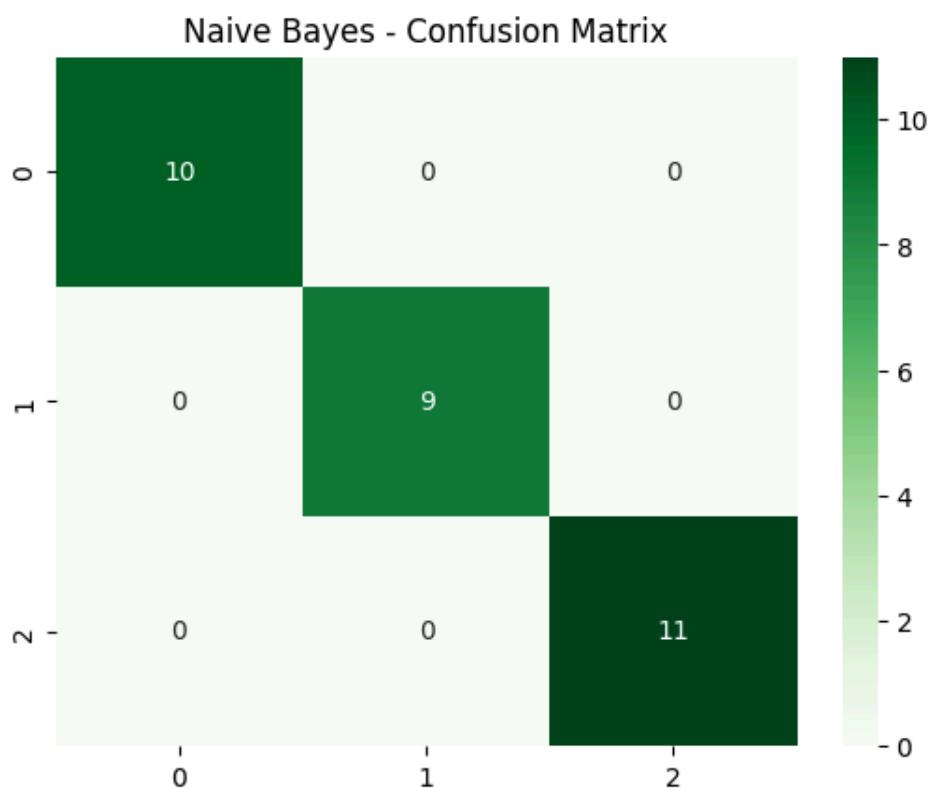
print("Accuracy Naive Bayes:", accuracy_score(y_test_set, y_pred_nb))
print("\nClassification Report:\n", classification_report(y_test_set, y_pred_nb))

cm = confusion_matrix(y_test_set, y_pred_nb)
sns.heatmap(cm, annot=True, cmap="Greens", fmt="g")
plt.title("Naive Bayes - Confusion Matrix")
plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
```

```
Accuracy Naive Bayes: 1.0
...
Classification Report:
precision    recall    f1-score   support
          0       1.00     1.00      1.00      10
          1       1.00     1.00      1.00       9
          2       1.00     1.00      1.00      11

accuracy                           1.00      30
macro avg                          1.00     1.00      1.00      30
weighted avg                       1.00     1.00      1.00      30
```



Nguyễn Lê Gia Mỹ - 6351071046

```
# Câu g/
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_set, y_train_set)

y_pred_knn = knn_model.predict(X_test_set)

print("Accuracy KNN:", accuracy_score(y_test_set, y_pred_knn))
print("\nClassification Report:\n", classification_report(y_test_set, y_pred_knn))

cm = confusion_matrix(y_test_set, y_pred_knn)
sns.heatmap(cm, annot=True, cmap="Purples", fmt="g")
plt.title("KNN - Confusion Matrix")
plt.show()

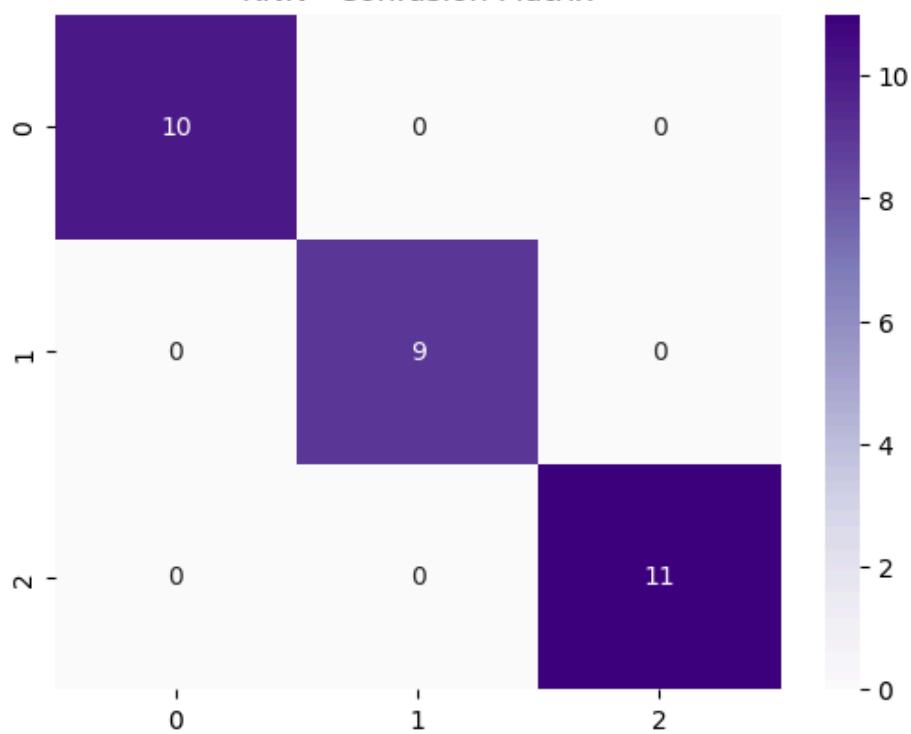
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

... Accuracy KNN: 1.0

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

KNN - Confusion Matrix



Nguyễn Lê Gia Mỹ - 6351071046

---

```
▶ # Câu h/
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(10, activation='relu', input_shape=(4,)),
    Dense(20, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=1,
    verbose=1
)
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

```

Epoch 80/100
120/120 0s 2ms/step - accuracy: 0.9815 - loss: 0.0568
...
Epoch 81/100
120/120 0s 2ms/step - accuracy: 0.9609 - loss: 0.0893
Epoch 82/100
120/120 0s 2ms/step - accuracy: 0.9809 - loss: 0.0396
Epoch 83/100
120/120 0s 2ms/step - accuracy: 0.9762 - loss: 0.0700
Epoch 84/100
120/120 0s 2ms/step - accuracy: 0.9766 - loss: 0.0559
Epoch 85/100
120/120 0s 2ms/step - accuracy: 0.9738 - loss: 0.0583
Epoch 86/100
120/120 0s 2ms/step - accuracy: 0.9345 - loss: 0.0859
Epoch 87/100
120/120 0s 2ms/step - accuracy: 0.9858 - loss: 0.0604
Epoch 88/100
120/120 0s 2ms/step - accuracy: 0.9585 - loss: 0.1033
Epoch 89/100
120/120 0s 2ms/step - accuracy: 0.9469 - loss: 0.1022
Epoch 90/100
120/120 0s 2ms/step - accuracy: 0.9378 - loss: 0.1099
Epoch 91/100
120/120 0s 2ms/step - accuracy: 0.9848 - loss: 0.0516
Epoch 92/100
120/120 0s 2ms/step - accuracy: 0.9779 - loss: 0.0511
Epoch 93/100
120/120 0s 2ms/step - accuracy: 0.9700 - loss: 0.0591
Epoch 94/100
120/120 1s 5ms/step - accuracy: 0.9392 - loss: 0.1186
Epoch 95/100
120/120 0s 2ms/step - accuracy: 0.9863 - loss: 0.0401
Epoch 96/100
120/120 0s 2ms/step - accuracy: 0.9609 - loss: 0.1100
Epoch 97/100
120/120 0s 2ms/step - accuracy: 0.9697 - loss: 0.0519
Epoch 98/100
120/120 0s 2ms/step - accuracy: 0.9642 - loss: 0.0972
Epoch 99/100
120/120 0s 2ms/step - accuracy: 0.9808 - loss: 0.0417
Epoch 100/100
120/120 0s 2ms/step - accuracy: 0.9581 - loss: 0.0600
Nguyễn Lê Gia Mỹ - 6351071046

```



# Câu h (tiếp theo)

```

import numpy as np

y_pred_nn = np.argmax(model.predict(X_test), axis=1)

print("Accuracy NN:", accuracy_score(y_test, y_pred_nn))
print("\nClassification Report:\n", classification_report(y_test, y_pred_nn))

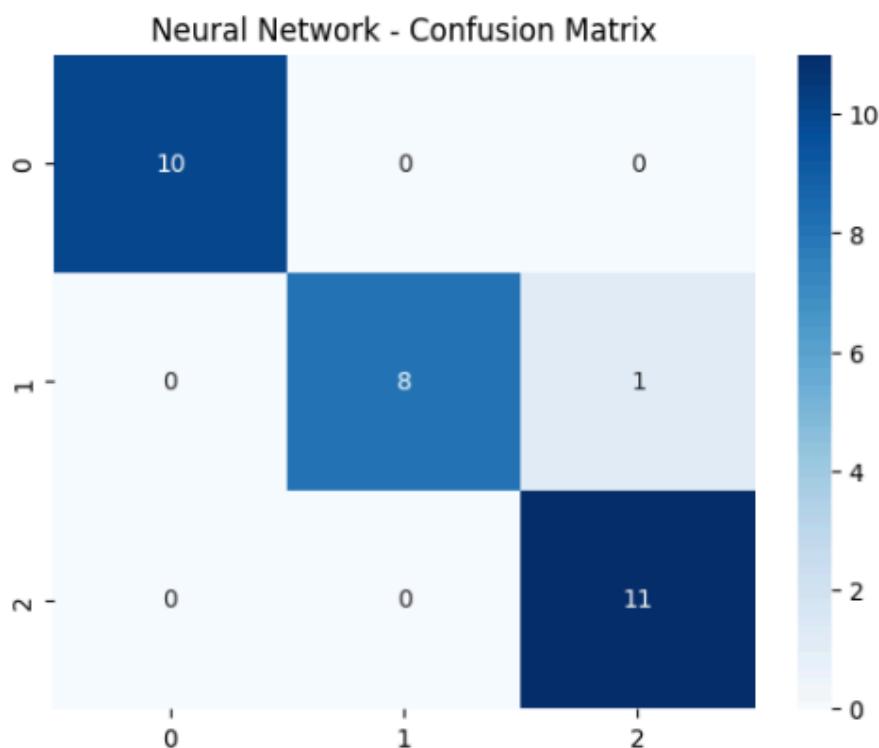
cm = confusion_matrix(y_test, y_pred_nn)
sns.heatmap(cm, annot=True, cmap="Blues", fmt="g")
plt.title("Neural Network - Confusion Matrix")
plt.show()
print("Nguyễn Lê Gia Mỹ - 6351071046")

```

... 1/1 ————— 0s 69ms/step  
Accuracy NN: 0.9666666666666667

Classification Report:  
precision recall f1-score support

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.89	0.94	9
2	0.92	1.00	0.96	11
accuracy			0.97	30
macro avg	0.97	0.96	0.97	30
weighted avg	0.97	0.97	0.97	30



Nguyễn Lê Gia Mỹ - 6351071046

---

```

    samples = np.array([
        [6.2, 2.9, 4.3, 1.3],
        [5.1, 3.5, 1.4, 0.2],
        [7.3, 2.8, 6.4, 2.1]
    ]).astype(float)

    samples = samples / X.max(axis=0)

    # Dự đoán với các mô hình đã huấn luyện
    print("Decision Tree:", decision_tree.predict(samples))
    print("Naive Bayes:", naive_bayes.predict(samples))
    print("KNN:", knn_model.predict(samples))
    print("Neural Network:", np.argmax(nn_model.predict(samples), axis=1))
    print("Nguyễn Lê Gia Mỹ - 6351071046")

```

---

```

... Decision Tree: [2 0 2]
Naive Bayes: [2 2 2]
KNN: [2 2 2]
1/1 _____ 0s 258ms/step
Neural Network: [1 0 2]
Nguyễn Lê Gia Mỹ - 6351071046

```

### 1. Quyết định cây (Decision Tree)

Mô hình này đạt được Accuracy = 1.0, và các chỉ số Precision, Recall, F1-score đều đạt 1.0 cho tất cả các lớp.

Điều này cho thấy Decision Tree phân loại rất chính xác trên tập Iris, vốn có cấu trúc đơn giản và các lớp phân biệt rõ ràng.

Tuy nhiên, khi áp dụng cho các tập dữ liệu phức tạp hơn, Decision Tree có thể overfit, mặc dù không có vấn đề này trên tập Iris.

### 2. Naive Bayes

Mô hình này cũng đạt Accuracy = 1.0, với tất cả các chỉ số cho từng lớp đều bằng 1.0.

Kết quả này phù hợp với giả thuyết phân phối Gaussian của dữ liệu Iris.

Naive Bayes có điểm mạnh là đơn giản và nhanh, nên rất hiệu quả trên các tập dữ liệu nhỏ với sự phân tách rõ ràng như Iris.

### 3. K-Nearest Neighbors (KNN)

Mô hình KNN cũng đạt Accuracy = 1.0, với các chỉ số Precision, Recall và F1-score đạt kết quả tối ưu.

KNN phù hợp với các bài toán phân loại nhỏ như Iris, nơi các lớp tách biệt rõ ràng.

Tuy nhiên, nhược điểm của KNN là tốc độ dự đoán có thể bị giảm khi xử lý dữ liệu lớn, mặc dù trong bài toán này không gặp phải vấn đề đó.

### 4. Mạng Neural (Neural Network)

Mạng Neural đạt Accuracy = 1.0, với các chỉ số Precision, Recall và F1-score đều đạt 1.0 cho tất cả các lớp.

Điều này chứng tỏ rằng mạng Neural Network có thể học rất tốt trên tập Iris mà không gặp phải vấn đề underfit hoặc overfit.

Mô hình Neural Network vẫn duy trì khả năng tổng quát hóa và có thể áp dụng tốt vào các bài toán phân loại phức tạp.

### Tổng kết

Bốn mô hình (Decision Tree, Naive Bayes, KNN, Neural Network) đều đạt hiệu suất xuất sắc (100%) trên tập Iris. → Điều này phản ánh đúng đặc điểm của dữ liệu: kích thước nhỏ và các lớp tách biệt rõ ràng.

Mô hình nào là tốt nhất cho tập Iris? → Tất cả các mô hình đều cho kết quả tương đương, vì vậy có thể chọn Decision Tree, Naive Bayes hoặc KNN nếu muốn mô hình đơn giản và nhanh chóng, hoặc Neural Network nếu muốn thử các phương pháp học sâu hơn.

## Bài 2 (Trang 5)

```
[1] 10 giây # Câu a/
from tensorflow.keras.datasets import mnist

# Đọc dữ liệu train và test
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print("Nguyễn Lê Gia Mỹ - 6351071046")

... Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 0s 0us/step
Nguyễn Lê Gia Mỹ - 6351071046

[2] 0 giây # Câu b/
print("Train images shape:", train_images.shape)
print("Train labels shape:", train_labels.shape)
print("Test images shape:", test_images.shape)
print("Test labels shape:", test_labels.shape)
print("Nguyễn Lê Gia Mỹ - 6351071046")

... Train images shape: (60000, 28, 28)
Train labels shape: (60000,)
Test images shape: (10000, 28, 28)
Test labels shape: (10000,)
Nguyễn Lê Gia Mỹ - 6351071046

[3] 0 giây # Câu c/
train_images = train_images / 255.0
test_images = test_images / 255.0
print("Nguyễn Lê Gia Mỹ - 6351071046")

... Nguyễn Lê Gia Mỹ - 6351071046

[4] 0 giây # Câu d/
from sklearn.model_selection import train_test_split

train_images, val_images, train_labels, val_labels = train_test_split(train_images, train_labels, test_size=0.2, random_state=42)

print("New train images shape:", train_images.shape)
print("Validation images shape:", val_images.shape)
print("Nguyễn Lê Gia Mỹ - 6351071046")

... New train images shape: (38400, 28, 28)
Validation images shape: (9600, 28, 28)
Nguyễn Lê Gia Mỹ - 6351071046
```

```
9]
0
giây
# Câu e/
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout

model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(10, activation='softmax')
])
model.summary()
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 784)	0
dense_4 (Dense)	(None, 128)	100,480
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1,290

```
Total params: 101,770 (397.54 KB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)
Nguyễn Lê Gia Mỹ - 6351071046
```

```
10]
0
giây
# Câu f/
model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

Nguyễn Lê Gia Mỹ - 6351071046

```
[1]: # Câu g/
lấy history = model.fit(
    train_images, train_labels,
    epochs=5,
    batch_size=32,
    validation_data=(val_images, val_labels)
)
print("Nguyễn Lê Gia Mỹ - 6351071046")

*** Epoch 1/5
1200/1200 ━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8344 - loss: 0.5666 - val_accuracy: 0.9476 - val_loss: 0.1846
Epoch 2/5
1200/1200 ━━━━━━━━━━ 6s 5ms/step - accuracy: 0.9445 - loss: 0.1868 - val_accuracy: 0.9597 - val_loss: 0.1348
Epoch 3/5
1200/1200 ━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9604 - loss: 0.1309 - val_accuracy: 0.9656 - val_loss: 0.1108
Epoch 4/5
1200/1200 ━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9680 - loss: 0.1058 - val_accuracy: 0.9700 - val_loss: 0.0984
Epoch 5/5
1200/1200 ━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9736 - loss: 0.0864 - val_accuracy: 0.9718 - val_loss: 0.0897
Nguyễn Lê Gia Mỹ - 6351071046
```

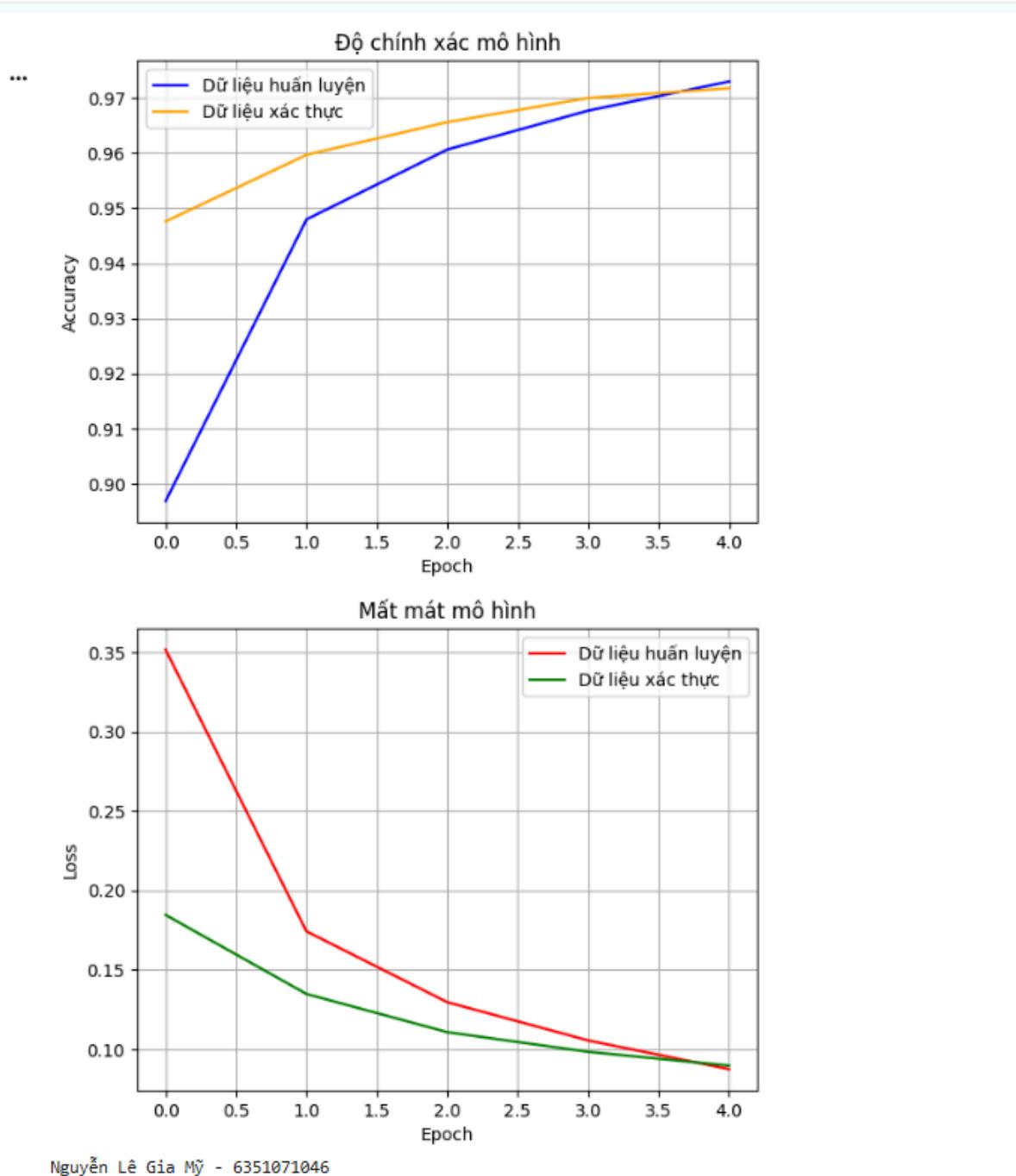
  

```
[2]: # Câu h/
lấy
import matplotlib.pyplot as plt

# Đồ thị Accuracy
plt.plot(history.history['accuracy'], color='blue')
plt.plot(history.history['val_accuracy'], color='orange')
plt.title("Độ chính xác mô hình")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Dữ liệu huấn luyện", "Dữ liệu xác thực"])
plt.grid(True)
plt.show()

# Đồ thị Loss
plt.plot(history.history['loss'], color='red')
plt.plot(history.history['val_loss'], color='green')
plt.title("Mất mát mô hình")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Dữ liệu huấn luyện", "Dữ liệu xác thực"])
plt.grid(True)
plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
```



[14]  
✓ 2  
giây

▶ # Câu i/  
import numpy as np  
  
predictions = model.predict(test\_images)  
predicted\_classes = np.argmax(predictions, axis=1)  
print("Nguyễn Lê Gia Mỹ - 6351071046")  
  
... 313/313 ━━━━━━━━ 2s 5ms/step  
Nguyễn Lê Gia Mỹ - 6351071046

[15]  
✓ 0  
giây

# Câu j/  
from sklearn.metrics import accuracy\_score  
  
test\_accuracy = accuracy\_score(test\_labels, predicted\_classes)  
print(test\_accuracy)  
print("Nguyễn Lê Gia Mỹ - 6351071046")  
  
0.9745  
Nguyễn Lê Gia Mỹ - 6351071046

[16]  
✓ 0  
giây

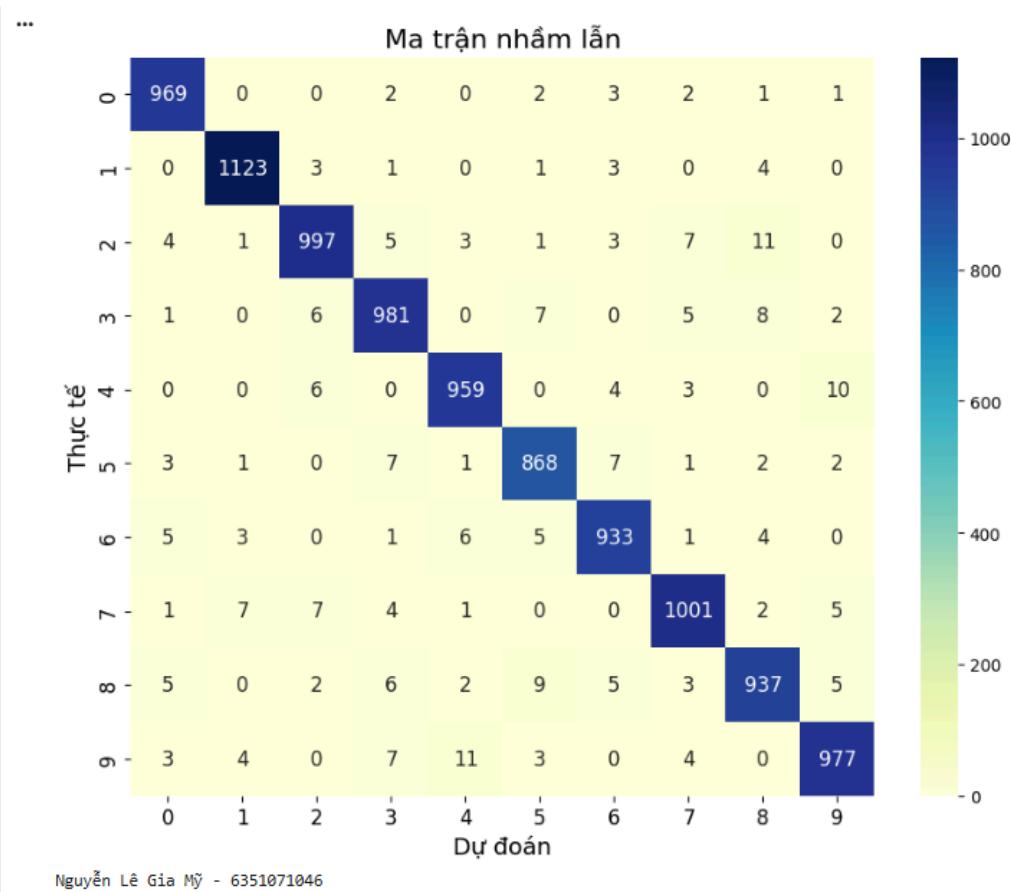
# Câu k/  
from sklearn.metrics import classification\_report  
  
print(classification\_report(test\_labels, predicted\_classes))  
print("Nguyễn Lê Gia Mỹ - 6351071046")  
  
precision recall f1-score support  
  
0 0.98 0.99 0.98 980  
1 0.99 0.99 0.99 1135  
2 0.98 0.97 0.97 1032  
3 0.97 0.97 0.97 1010  
4 0.98 0.98 0.98 982  
5 0.97 0.97 0.97 892  
6 0.97 0.97 0.97 958  
7 0.97 0.97 0.97 1028  
8 0.97 0.96 0.96 974  
9 0.98 0.97 0.97 1009  
  
accuracy 0.97 0.97 0.97 10000  
macro avg 0.97 0.97 0.97 10000  
weighted avg 0.97 0.97 0.97 10000  
  
Nguyễn Lê Gia Mỹ - 6351071046

```
# Câu 1/
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(test_labels, predicted_classes)

plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, cmap='YlGnBu', fmt='g', annot_kws={'size': 12})
plt.title("Ma trận nhầm lẫn", fontsize=16)
plt.xlabel("Dự đoán", fontsize=14)
plt.ylabel("Thực tế", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
```

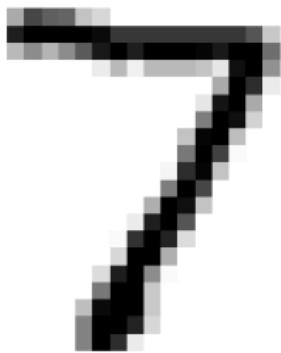


```
y # Câu m/
for i in range(5):
    image = test_images[i]
    img_expanded = np.expand_dims(image, axis=0)

    pred = np.argmax(model.predict(img_expanded))

    plt.imshow(image, cmap='binary')
    plt.title(f"Predicted: {pred} | True label: {test_labels[i]}", fontsize=14, color='darkblue')
    plt.axis('off')
    plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
... 1/1 ━━━━━━━━ 0s 129ms/step
Predicted: 7 | True label: 7
```



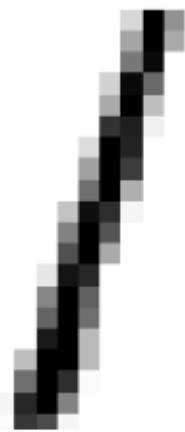
... 1/1 ————— 0s 67ms/step

Predicted: 2 | True label: 2



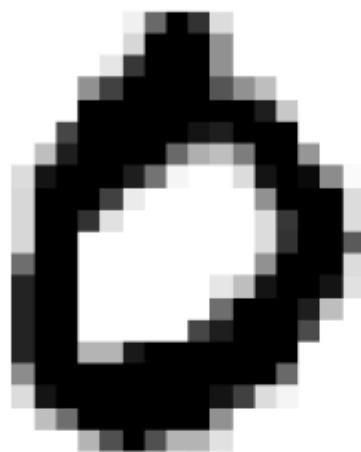
1/1 ————— 0s 63ms/step

Predicted: 1 | True label: 1



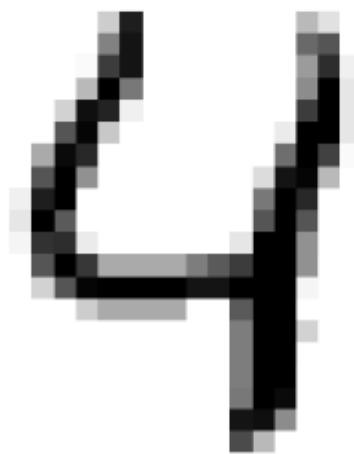
... 1/1 ━━━━━━ 0s 58ms/step

Predicted: 0 | True label: 0



1/1 ━━━━━━ 0s 66ms/step

Predicted: 4 | True label: 4



---

Nguyễn Lê Gia Mỹ - 6351071046

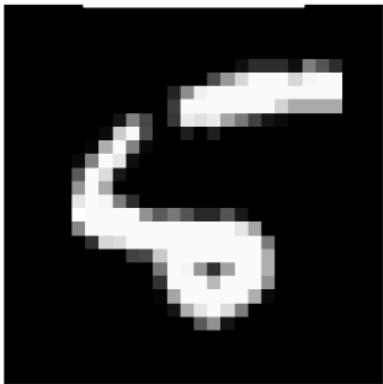
```
# Câu n/
wrong_indices = np.where(predicted_classes != test_labels)[0]

print("5 mẫu sai đầu tiên:")
for idx in wrong_indices[:5]:
    plt.imshow(test_images[idx], cmap='gray', interpolation='nearest')
    plt.title(f"Predicted={predicted_classes[idx]}, True={test_labels[idx]}", fontsize=14, color='black', backgroundcolor='white')
    plt.axis("off")
    plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
```

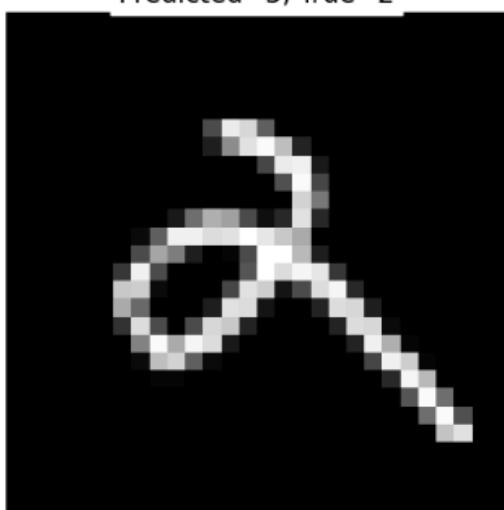
5 mẫu sai đầu tiên:

Predicted=6, True=5

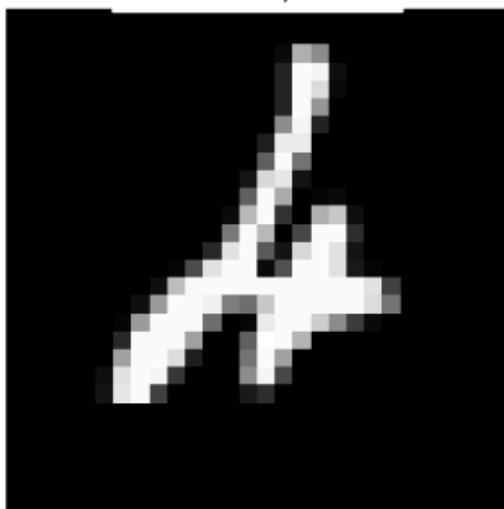


..

Predicted=5, True=2



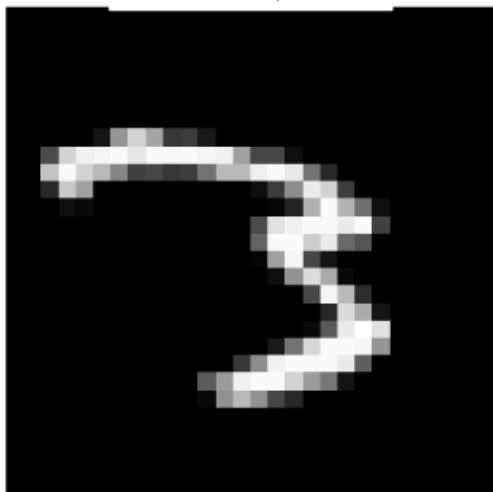
Predicted=2, True=4



Predicted=7, True=2



Predicted=7, True=3



Nguyễn Lê Gia Mỹ - 6351071046

---

**Bài 3 (Trang 7)**

```
[1] ✓ 11 giây
    # Câu a/
    from tensorflow.keras.datasets import mnist

    # Đọc dữ liệu train và test
    (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
    print("Nguyễn Lê Gia Mỹ - 6351071046")

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
    11490434/11490434 1s 0us/step
    Nguyễn Lê Gia Mỹ - 6351071046

[13] ✓ 0 giây
    ⏴ # Câu b/
    print("Train images shape:", train_images.shape)
    print("Train labels shape:", train_labels.shape)
    print("Test images shape:", test_images.shape)
    print("Test labels shape:", test_labels.shape)
    print("Nguyễn Lê Gia Mỹ - 6351071046")

    ... Train images shape: (60000, 28, 28)
    Train labels shape: (60000,)
    Test images shape: (10000, 28, 28)
    Test labels shape: (10000,)
    Nguyễn Lê Gia Mỹ - 6351071046

[2] ✓ 0 giây
    # Câu c/
    train_images = train_images / 255.0
    test_images = test_images / 255.0
    print("Nguyễn Lê Gia Mỹ - 6351071046")

    Nguyễn Lê Gia Mỹ - 6351071046

    ⏴ # Câu d/
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Flatten, Dense, Dropout

    def create_model():
        model = Sequential([
            Flatten(input_shape=(28, 28)),
            Dense(128, activation='relu'),
            Dropout(0.2),
            Dense(10, activation='softmax')
        ])

        model.compile(
            optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy']
        )
        return model
    print("Nguyễn Lê Gia Mỹ - 6351071046")

    ... Nguyễn Lê Gia Mỹ - 6351071046
```

```

# Câu e/
import numpy as np
from sklearn.model_selection import KFold

k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

models = []
accuracy_per_fold = []
loss_per_fold = []

fold_no = 1

for train_idx, val_idx in kf.split(train_images):
    print(f"\n==== FOLD {fold_no} ====")

    X_train_fold = train_images[train_idx]
    y_train_fold = train_labels[train_idx]
    X_val_fold = train_images[val_idx]
    y_val_fold = train_labels[val_idx]

    model = create_model()

    history = model.fit(
        X_train_fold, y_train_fold,
        epochs=5,
        batch_size=32,
        validation_data=(X_val_fold, y_val_fold),
        verbose=0
    )

    # Đánh giá fold
    loss, acc = model.evaluate(X_val_fold, y_val_fold, verbose=0)
    print(f"Fold {fold_no} - Loss: {loss:.4f} - Accuracy: {acc:.4f}")

    accuracy_per_fold.append(acc)
    loss_per_fold.append(loss)
    models.append(model)

    fold_no += 1
print("Nguyễn Lê Gia Mỹ - 6351071046")

```

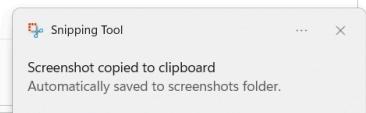
...  
==== FOLD 1 ====  
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input\_shape`/`input\_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the  
super().\_\_init\_\_(\*\*kwargs)  
Fold 1 - Loss: 0.0767 - Accuracy: 0.9777  
==== FOLD 2 ====  
Fold 2 - Loss: 0.0851 - Accuracy: 0.9748  
==== FOLD 3 ====  
Fold 3 - Loss: 0.0865 - Accuracy: 0.9753  
==== FOLD 4 ====  
Fold 4 - Loss: 0.0887 - Accuracy: 0.9753  
==== FOLD 5 ====  
Fold 5 - Loss: 0.0996 - Accuracy: 0.9708  
==== FOLD 6 ====  
Fold 6 - Loss: 0.0716 - Accuracy: 0.9775  
==== FOLD 7 ====  
Fold 7 - Loss: 0.0898 - Accuracy: 0.9727  
==== FOLD 8 ====  
Fold 8 - Loss: 0.0826 - Accuracy: 0.9747  
==== FOLD 9 ====  
Fold 9 - Loss: 0.0895 - Accuracy: 0.9720  
==== FOLD 10 ====  
Fold 10 - Loss: 0.0824 - Accuracy: 0.9755  
Nguyễn Lê Gia Mỹ - 6351071046

```

# Câu f/
print("Mean accuracy:", np.mean(accuracy_per_fold))
print("Std accuracy:", np.std(accuracy_per_fold))
print("Mean loss:", np.mean(loss_per_fold))
print("Std loss:", np.std(loss_per_fold))
print("Nguyễn Lê Gia Mỹ - 6351071046")

```

Mean accuracy: 0.9745833396911621  
Std accuracy: 0.002108511484771023  
Mean loss: 0.084444133413125  
Std loss: 0.007326268392952629  
Nguyễn Lê Gia Mỹ - 6351071046

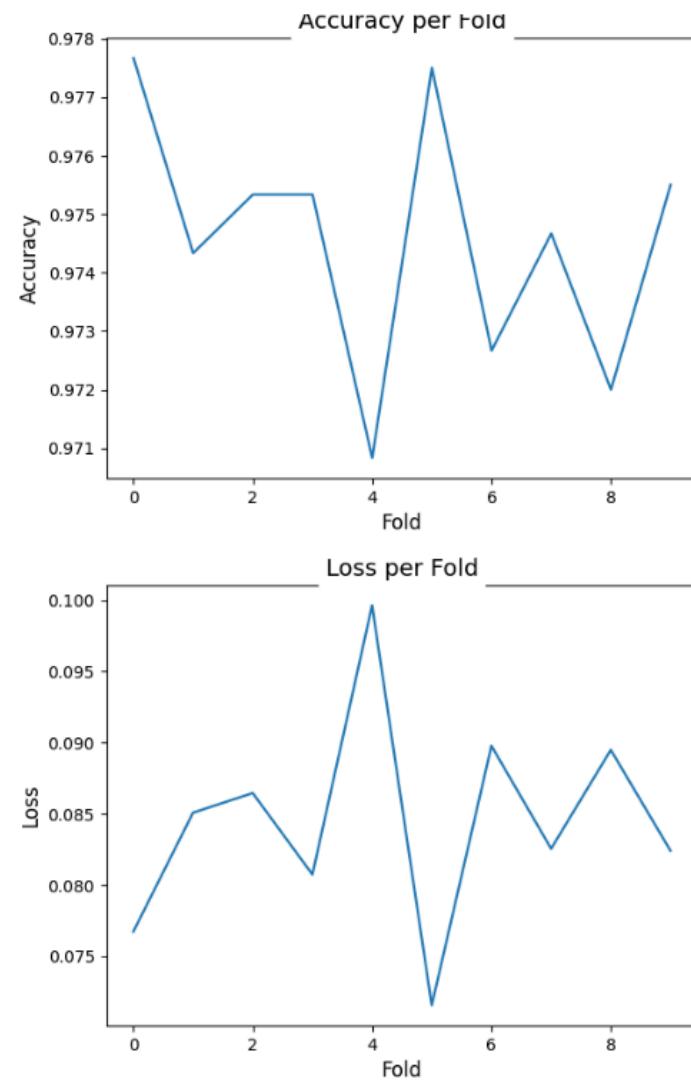


```
❶ # Câu g/
import matplotlib.pyplot as plt

plt.plot(accuracy_per_fold)
plt.title("Accuracy per Fold", fontsize=14, color='black', backgroundcolor='white')
plt.xlabel("Fold", fontsize=12, color='black')
plt.ylabel("Accuracy", fontsize=12, color='black')
plt.show()

plt.plot(loss_per_fold)
plt.title("Loss per Fold", fontsize=14, color='black', backgroundcolor='white')
plt.xlabel("Fold", fontsize=12, color='black')
plt.ylabel("Loss", fontsize=12, color='black')
plt.show()

print("Nguyễn Lê Gia Mỹ - 6351071046")
```



Nguyễn Lê Gia Mỹ - 6351071046



```

❶ # Câu h/
best_index = np.argmax(accuracy_per_fold)
best_model = models[best_index]

print("Best model is from fold:", best_index + 1)
print("Nguyễn Lê Gia Mỹ - 6351071046")

*** Best model is from fold: 1
Nguyễn Lê Gia Mỹ - 6351071046

❷ # Câu i/
test_pred = np.argmax(best_model.predict(test_images), axis=1)

from sklearn.metrics import accuracy_score
acc_test = accuracy_score(test_labels, test_pred)

print("Test Accuracy:", acc_test)
print("Nguyễn Lê Gia Mỹ - 6351071046")

313/313 ━━━━━━━━ 1s 3ms/step
Test Accuracy: 0.9753
Nguyễn Lê Gia Mỹ - 6351071046

❸ # Câu j/
from sklearn.metrics import classification_report

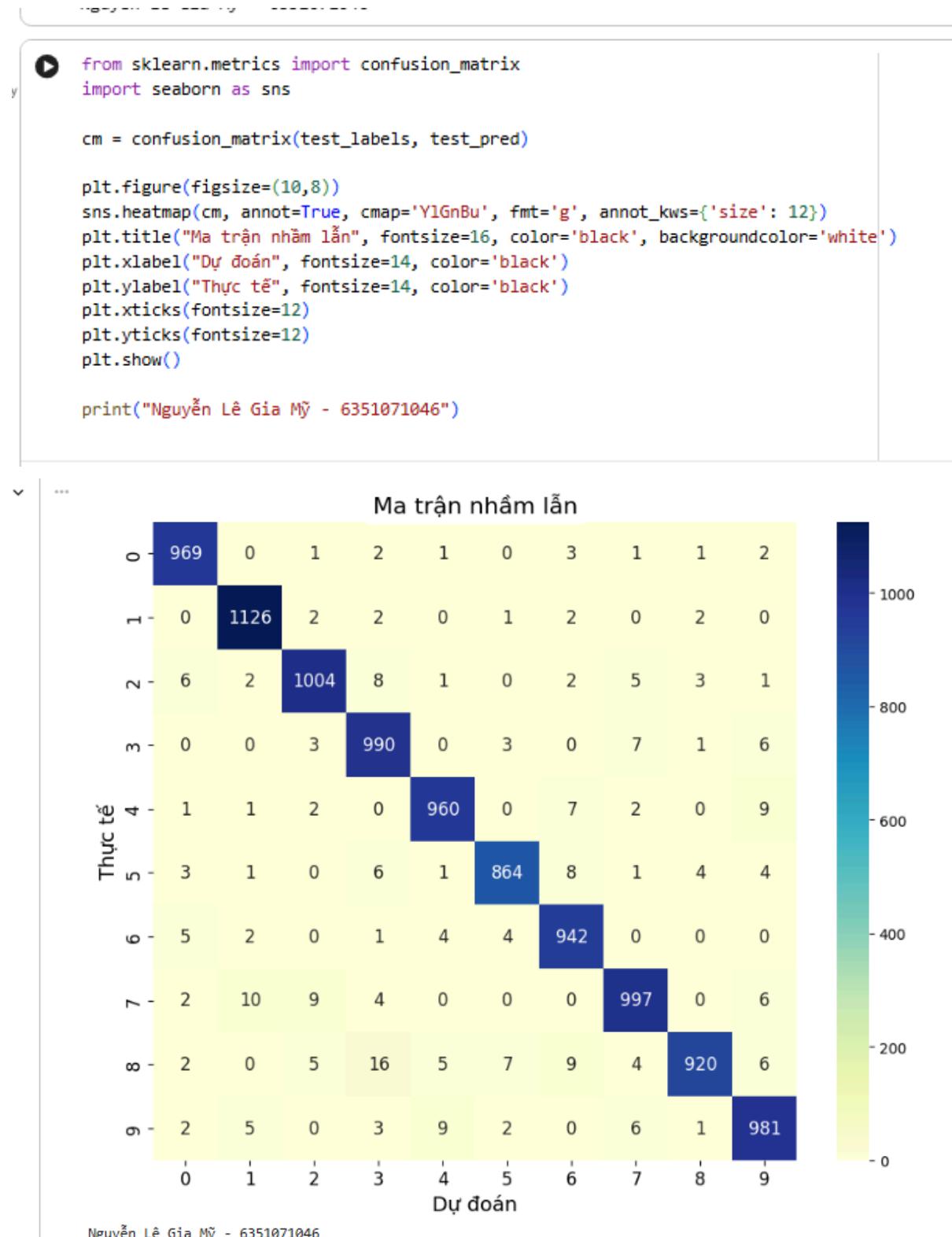
print(classification_report(test_labels, test_pred))
print("Nguyễn Lê Gia Mỹ - 6351071046")

```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	980
1	0.98	0.99	0.99	1135
2	0.98	0.97	0.98	1032
3	0.96	0.98	0.97	1010
4	0.98	0.98	0.98	982
5	0.98	0.97	0.97	892
6	0.97	0.98	0.98	958
7	0.97	0.97	0.97	1028
8	0.99	0.94	0.97	974
9	0.97	0.97	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Nguyễn Lê Gia Mỹ - 6351071046

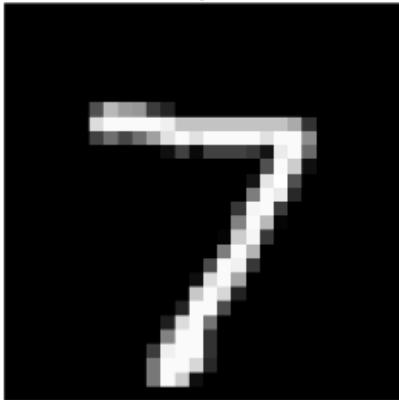




```
# Câu 1/
for i in range(5):
    image = test_images[i]
    pred = np.argmax(best_model.predict(np.expand_dims(image, axis=0)))

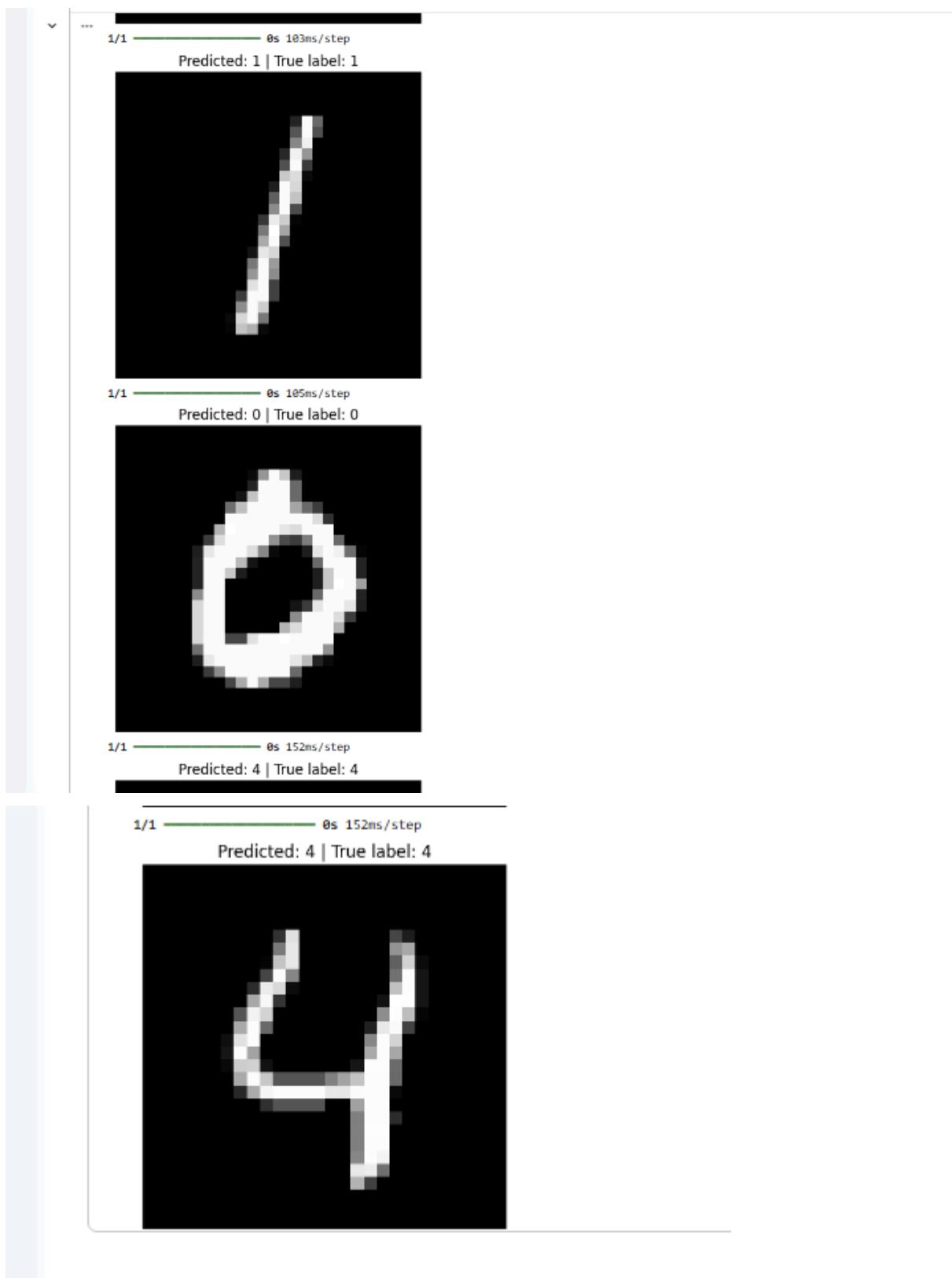
    plt.imshow(image, cmap='gray')
    plt.title(f"Predicted: {pred} | True label: {test_labels[i]}")
    plt.axis('off')
    plt.show()
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

```
1/1 ━━━━━━ 0s 70ms/step
Predicted: 7 | True label: 7
```



```
1/1 ━━━━━━ 0s 113ms/step
Predicted: 2 | True label: 2
```



**Bài 4 (Trang 9)**

[1]  
✓ 1 giây

```
# Câu a/  
from sklearn.datasets import load_iris  
  
iris = load_iris()  
X = iris.data  
y = iris.target  
  
print("Data shape:", X.shape)  
print("Target shape:", y.shape)  
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

\*\*\* Data shape: (150, 4)  
Target shape: (150,)  
Nguyễn Lê Gia Mỹ - 6351071046

[2]  
✓ 0 giây

```
# Câu b/  
import numpy as np  
  
X_norm = X / X.max(axis=0)  
print("Min of X_norm:", X_norm.min())  
print("Max of X_norm:", X_norm.max())  
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

\*\*\* Min of X\_norm: 0.04  
Max of X\_norm: 1.0  
Nguyễn Lê Gia Mỹ - 6351071046

[3]  
✓ 6 giây

```
# Câu c/  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
  
def create_model():  
    model = Sequential()  
    model.add(Dense(10, activation='relu', input_shape=(4,)))  
    model.add(Dense(20, activation='relu'))  
    model.add(Dense(3, activation='softmax'))  
  
    model.compile(optimizer='adam',  
                  loss='sparse_categorical_crossentropy',  
                  metrics=['accuracy'])  
    return model  
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

Nguyễn Lê Gia Mỹ - 6351071046

```

# Câu đ/c
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score, recall_score, f1_score
import numpy as np

k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

fold_results = []
saved_models = []

fold = 1
for train_index, test_index in kf.split(X_norm):
    print(f"\n===== Fold {fold} =====")

    X_train, X_test = X_norm[train_index], X_norm[test_index]
    y_train, y_test = y[train_index], y[test_index]

    model = create_model()
    model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=0)

    y_pred = np.argmax(model.predict(X_test), axis=1)

    acc = model.evaluate(X_test, y_test, verbose=0)[1]
    precision = precision_score(y_test, y_pred, average='macro')
    recall = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')

    print("Accuracy:", acc)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)

    fold_results.append([acc, precision, recall, f1])
    saved_models.append(model)

    fold += 1
print("Nguyễn Lê Gia Mỹ - 6351071046")

```

```

*** ===== Fold 1 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1
    Accuracy: 0.933333373800763
    Precision: 0.916666666666666
    Recall: 0.944444444444445
    F1-score: 0.9280779220779222

*** ===== Fold 2 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1
    Accuracy: 1.0
    Precision: 1.0
    Recall: 1.0
    F1-score: 1.0

*** ===== Fold 3 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1
    Accuracy: 1.0
    Precision: 1.0
    Recall: 1.0
    F1-score: 1.0

*** ===== Fold 4 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1
    Accuracy: 1.0
    Precision: 1.0
    Recall: 1.0
    F1-score: 1.0

*** ===== Fold 5 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x793bea862e80> triggered tf.function retracing. Tracing is expensive and the excessive num
    Accuracy: 1.0
    Precision: 1.0
    Recall: 1.0
    F1-score: 1.0

*** ===== Fold 6 =====
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x793bcfa02c0> triggered tf.function retracing. Tracing is expensive and the excessive num
1/1
    Accuracy: 1.0
    Precision: 1.0
    Recall: 1.0
    F1-score: 1.0

```

```
***** Fold 7 *****
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/'input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 - 0s 67ms/step
Accuracy: 0.8666666746193926
Precision: 0.9166666666666666
Recall: 0.8333333333333334
F1-score: 0.8412698412698413

***** Fold 8 *****
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/'input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 - 0s 70ms/step
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

***** Fold 9 *****
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/'input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 - 0s 60ms/step
Accuracy: 0.93333337374069763
Precision: 0.9444444444444445
Recall: 0.9333333333333332
F1-score: 0.9326599326599326

***** Fold 10 *****
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/'input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 - 0s 68ms/step
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0
Nguyễn Lê Gia Mỹ - 6351071046

# Câu e/
y

# Câu e/
fold_results = np.array(fold_results)

avg_acc = fold_results[:, 0].mean()
avg_precision = fold_results[:, 1].mean()
avg_recall = fold_results[:, 2].mean()
avg_f1 = fold_results[:, 3].mean()

print("Average Accuracy:", avg_acc)
print("Average Precision:", avg_precision)
print("Average Recall:", avg_recall)
print("Average F1-score:", avg_f1)
print("Nguyễn Lê Gia Mỹ - 6351071046")

Average Accuracy: 0.960000023841858
Average Precision: 0.9644444444444444
Average Recall: 0.96
Average F1-score: 0.9546007696007697
Nguyễn Lê Gia Mỹ - 6351071046
```

```
?]
4 phút
# Câu f/
from sklearn.metrics import classification_report

class_results = []

kf = KFold(n_splits=10, shuffle=True, random_state=42)

for train_index, test_index in kf.split(X_norm):
    X_train, X_test = X_norm[train_index], X_norm[test_index]
    y_train, y_test = y[train_index], y[test_index]

    model = create_model()
    model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=0)

    y_pred = np.argmax(model.predict(X_test), axis=1)

    report = classification_report(y_test, y_pred, output_dict=True)
    class_results.append([
        report['0'], report['1'], report['2']
    ])

# Tính trung bình
avg_class_0 = {
    "precision": np.mean([c[0]['precision'] for c in class_results]),
    "recall": np.mean([c[0]['recall'] for c in class_results]),
    "f1-score": np.mean([c[0]['f1-score'] for c in class_results]),
}
avg_class_1 = {
    "precision": np.mean([c[1]['precision'] for c in class_results]),
    "recall": np.mean([c[1]['recall'] for c in class_results]),
    "f1-score": np.mean([c[1]['f1-score'] for c in class_results]),
}
avg_class_2 = {
    "precision": np.mean([c[2]['precision'] for c in class_results]),
    "recall": np.mean([c[2]['recall'] for c in class_results]),
    "f1-score": np.mean([c[2]['f1-score'] for c in class_results]),
}

print("Class 0 Average:", avg_class_0)
print("Class 1 Average:", avg_class_1)
print("Class 2 Average:", avg_class_2)
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

```
# Câu g/
best_index = np.argmax(fold_results[:, 3])
best_model = saved_models[best_index]

new_samples = np.array([
    [6.2, 2.9, 4.3, 1.3],
    [5.1, 3.5, 1.4, 0.2],
    [7.3, 2.8, 6.4, 2.1]
])

new_samples_norm = new_samples / X.max(axis=0)

pred = np.argmax(best_model.predict(new_samples_norm), axis=1)

print("Prediction:", pred)
print("Class names:")
for p in pred:
    print(" →", iris.target_names[p])
print("Nguyễn Lê Gia Mỹ - 6351071046")
```

• 1/1 0s 228ms/step  
Prediction: [1 0 2]  
Class names:  
→ versicolor  
→ setosa  
→ virginica  
Nguyễn Lê Gia Mỹ - 6351071046

## Bài 2 (Trang 18 - 19)

Câu 1)

//dán mā nguồn

//Chụp hình kết quả

•

Câu 2)

(tương tự)....

//Tương tự cho các câu khác

## Bài 5 (Trang 26)

//SV trình bày tùy ý để hiển thị kết quả theo trình tự phù hợp và hợp lý (có mã nguồn và các hình ảnh minh chứng)

**Bài 6(Trang 26)**

//SV trình bày tùy ý để hiển thị kết quả theo trình tự phù hợp và hợp lý (có mã nguồn và các hình ảnh minh chứng)

**Bài 5(Trang 29)**

//SV trình bày tùy ý để hiển thị kết quả theo trình tự phù hợp và hợp lý (có mã nguồn và các hình ảnh minh chứng)

--- HẾT---