Trần Xuân Triển – B20DCCN691

Câu1 : Hiện 5 dòng trong bộ dữ liệu

câu 1

In [1]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from keras.layers import Dropout
import pandas as pd

dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")
data = pd.read_csv("pima-indians-diabetes.csv", header = None)

X = dataset[:, 0:8] #lấy đối tượng huấn luyện
Y = dataset[:, 8] #lấy nhãn

print(data.head())
```

```
   0    1   2   3    4     5      6   7  8
0  6  148  72  35    0  33.6  0.627  50  1
1  1   85  66  29    0  26.6  0.351  31  0
2  8  183  64   0    0  23.3  0.672  32  1
3  1   89  66  23   94  28.1  0.167  21  0
4  0  137  40  35  168  43.1  2.288  33  1
```

Câu 2:

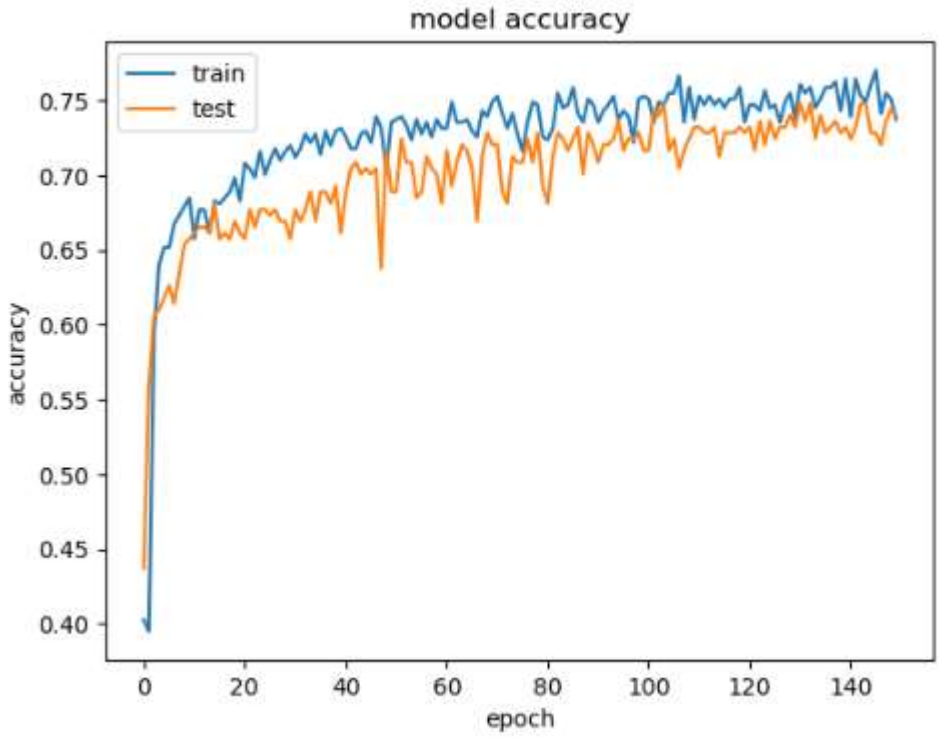
- Code:

câu 2

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from keras.layers import Dropout

# load pima indians dataset
dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables
X = dataset[:, 0:8] #lấy đối tượng huấn luyện
Y = dataset[:, 8] #lấy nhãn

# create model
model = Sequential() #mô hình tuần tự

# Định nghĩa các lớp layers
layer_1 = Dense(15, input_dim=8, activation='relu')
layer_2 = Dense(10, activation='relu')
layer_3 = Dense(1, activation='sigmoid')

# Thêm các lớp layers vào mô hình sử dụng model.add(layer)
model.add(layer_1) # thêm 1 lớp Dense có 15 nơ-ron
model.add(layer_2) # thêm 1 lớp Dense có 10 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt relu
model.add(layer_3) # thêm 1 lớp Dense có 1 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt sigmoid

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
#tạo đối tượng history
#tách 33% dữ liệu từ dữ liệu huấn luyện để xác thực, lặp lại 150 qua toàn bộ dữ liệu, kích thước mỗi batch 10
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10, verbose=0)

# list all data in history
print(history.history.keys())


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

- Anh1:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Câu 3:

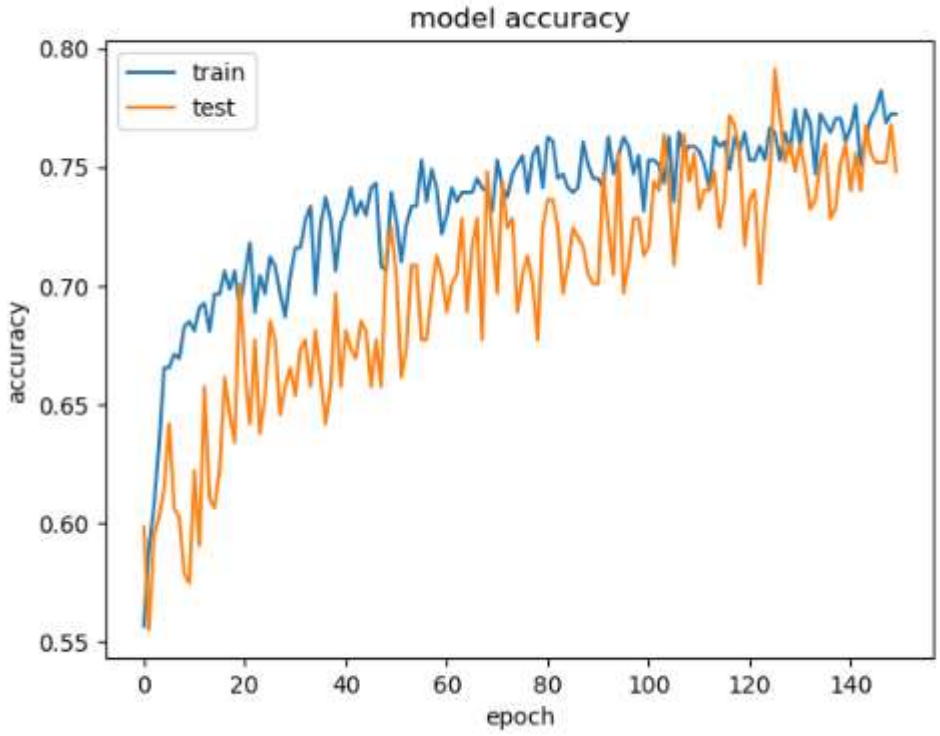
- Code:

câu 3

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from keras.layers import Dropout

# load pima indians dataset
dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables
X = dataset[:, 0:8] #lấy đối tượng huấn luyện
Y = dataset[:, 8] #lấy nhãn

# create model
model = Sequential() #mô hình tuần tự

# Định nghĩa các lớp layers
layer_1 = Dense(20, input_dim=8, activation='relu')
layer_2 = Dense(15, activation='relu')
layer_3 = Dense(10, activation='relu')
layer_4 = Dense(8, activation='relu')
layer_5 = Dense(1, activation='sigmoid')

# Thêm các lớp layers vào mô hình sử dụng model.add(layer)
model.add(layer_1) # thêm 1 lớp Dense có 15 nơ-ron
model.add(layer_2) # thêm 1 lớp Dense có 10 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt relu
model.add(layer_3) # thêm 1 lớp Dense có 1 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt sigmoid
model.add(layer_4)
model.add(layer_5)
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
#tạo đối tượng history
#tách 33% dữ liệu từ dữ liệu huấn luyện để xác thực, lặp lại 150 qua toàn bộ dữ liệu, kích thước mỗi batch 10
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10, verbose=0)

# list all data in history
print(history.history.keys())


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

– Anh2:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Câu 4:

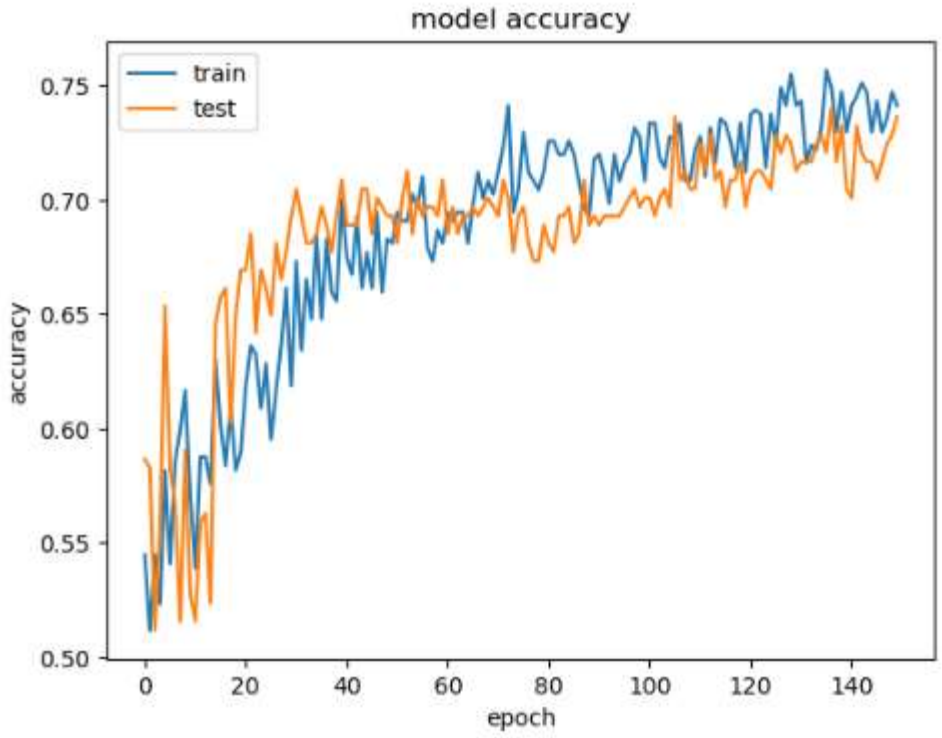
- Code:

câu 4

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from keras.layers import Dropout

# load pima indians dataset
dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables
X = dataset[:, 0:8] #lấy đối tượng huấn luyện
Y = dataset[:, 8] #lấy nhãn

# create model
model = Sequential() #mô hình tuần tự

# Định nghĩa các lớp layers
layer_1 = Dense(15, input_dim=8, activation='relu')

layer_2 = Dense(10, activation='relu')
layer_3 = Dense(1, activation='sigmoid')

# Thêm các lớp layers vào mô hình sử dụng model.add(layer)
model.add(layer_1) # thêm 1 lớp Dense có 15 nơ-ron
model.add(Dropout(0.1))
model.add(layer_2) # thêm 1 lớp Dense có 10 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt relu
model.add(Dropout(0.1))
model.add(layer_3) # thêm 1 lớp Dense có 1 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt sigmoid

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
#tạo đối tượng history
#tách 33% dữ liệu từ dữ liệu huấn luyện để xác thực, lặp lại 150 qua toàn bộ dữ liệu, kích thước mỗi batch 10
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10, verbose=0)

# list all data in history
print(history.history.keys())


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

– Anh1.1:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

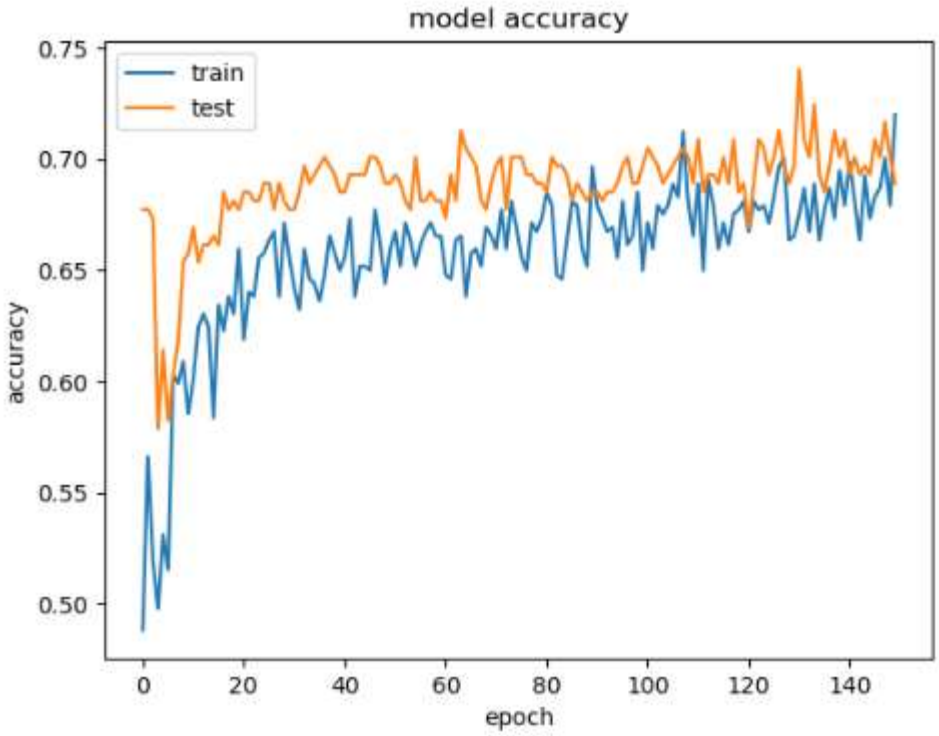


– Code:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from keras.layers import Dropout

# load pima indians dataset
dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables
X = dataset[:, 0:8] #lấy đối tượng huấn luyện
Y = dataset[:, 8] #lấy nhãn

# create model
model = Sequential() #mô hình tuần tự

# Định nghĩa các lớp layers
layer_1 = Dense(20, input_dim=8, activation='relu')
layer_2 = Dense(15, activation='relu')
layer_3 = Dense(10, activation='relu')
layer_4 = Dense(8, activation='relu')
layer_5 = Dense(1, activation='sigmoid')

# Thêm các lớp layers vào mô hình sử dụng model.add(layer)
model.add(layer_1) # thêm 1 lớp Dense có 15 nơ-ron

model.add(Dropout(0.1))
model.add(layer_2) # thêm 1 lớp Dense có 10 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt relu
model.add(Dropout(0.1))
model.add(layer_3) # thêm 1 lớp Dense có 1 nơ-ron, nhận đầu vào là đầu ra của lớp trước, sử dụng hàm kích hoạt sigmoid
model.add(Dropout(0.1))
model.add(layer_4)
model.add(Dropout(0.1))
model.add(layer_5)
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
#tạo đối tượng history
#tách 33% dữ liệu từ dữ liệu huấn luyện để xác thực, lặp lại 150 qua toàn bộ dữ liệu, kích thước mỗi batch 10
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10, verbose=0)

# list all data in history
print(history.history.keys())


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

– Anh2.1:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Câu 5:

Nhận xét:

- Tất cả các mô hình đều có độ chính xác trong khoảng 0.55-0.8.
- Mô hình không sử dụng dropout có độ chính xác trên bộ huấn luyện cao hơn so với độ chính xác trên bộ kiểm tra, điều này cho thấy mô hình có hiện tượng overfitting.
- Mô hình sử dụng dropout có độ chính xác trên bộ kiểm tra cao hơn so với độ chính xác trên bộ huấn luyện, điều này cho thấy mô hình có khả năng giảm hiện tượng overfitting.
- Mô hình không sử dụng dropout sẽ học các đặc trưng chi tiết của tập dữ liệu huấn luyện, dẫn đến việc mô hình có thể ghi nhớ chính xác các mẫu dữ liệu trong tập huấn luyện nhưng lại không thể áp dụng cho các mẫu dữ liệu mới. Điều này được thể hiện qua độ chính xác trên bộ kiểm tra thấp hơn so với độ chính xác trên bộ huấn luyện.
- Mô hình sử dụng dropout sẽ loại bỏ một số neuron ngẫu nhiên trong quá trình huấn luyện, điều này giúp mô hình trở nên linh hoạt hơn và ít phụ thuộc vào các đặc trưng chi tiết của tập dữ liệu huấn luyện. Do đó, mô hình

có khả năng giảm hiện tượng overfitting và có độ chính xác cao hơn trên bộ kiểm tra.

- Kết quả trên cho thấy, lớp dropout có khả năng giảm hiện tượng overfitting, giúp mô hình có độ chính xác cao hơn trên bộ kiểm tra.
- Một số lưu ý khi sử dụng dropout:
  + Hệ số dropout (p) cần được điều chỉnh phù hợp với từng mô hình. Hệ số dropout quá cao sẽ khiến mô hình trở nên không ổn định và có độ chính xác thấp.
  + Dropout cần được sử dụng kết hợp với các kỹ thuật khác để cải thiện hiệu quả của mô hình.