TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

TÀI LIỆU HƯỚNG DẪN THỰC HÀNH
Môn: Khai thác dữ liệu

BÀI THỰC HÀNH
Nội dung: Neural Network

I. Mục tiêu

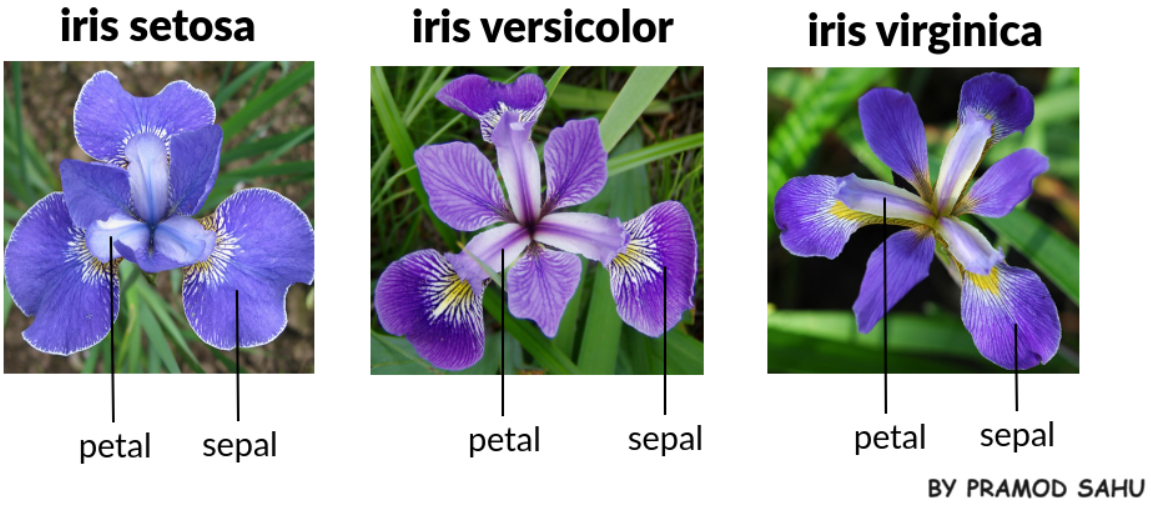
- Hiểu và vận dụng được Neural network đơn giản trong các bài toán phân lớp, dự đoán
- Hiểu và vận dụng được một số tiêu chí phổ biến đánh giá các mô hình phân lớp, dự đoán

II. Các bài tập

1. (Lập trình) Sử dụng Decision Tree, Naïve Bayes, K Nearest Neighbors, Neural Network để xây dựng mô hình phân lớp trên bộ dữ liệu Iris từ thư viện sklearn:

Mô tả bộ dữ liệu Iris: Bộ dữ liệu Iris bao gồm thông tin về 3 loài hoa Iris khác nhau: Iris setosa, Iris versicolor và Iris virginica. Gồm 4 thuộc tính đo lường (đơn vị cm): Chiều dài của lá đài (sepal length); Chiều rộng của lá đài (sepal width); Chiều dài của cánh hoa (petal length); Chiều rộng của cánh hoa (petal width).

Bộ dữ liệu Iris thường được sử dụng để thực hiện các tác vụ phân loại và nhận dạng loài hoa dựa trên các thuộc tính đã nêu trên.



Thực hiện các yêu cầu sau:

a) Đọc bộ dữ liệu Iris từ sklearn

Hướng dẫn: Từ thư viện sklearn import Iris và sử dụng hàm load_iris để đọc bộ dữ liệu

b) Khảo sát bộ dữ liệu Iris với yêu cầu sau:

- In ra tên các thuộc tính, thuộc tính phân lớp.
  Hướng dẫn: Sử dụng feature_names, target_names
- In ra số lượng mẫu, số lượng thuộc tính
  Hướng dẫn: Sử dụng data.shape, target.shape
- In ra 5 mẫu dữ liệu đầu tiên
  Hướng dẫn: Sử dụng data[], target[]
- Thống kê count, mean, std, min, max, tứ vị phân của bộ dữ liệu
  Hướng dẫn: tạo dataframe và sử dụng describe

c) Chuẩn hóa dữ liệu data về đoạn [0,1]

Hướng dẫn: Gán X là data và Y là target của bộ dữ liệu. Chia mỗi giá trị của X cho giá trị lớn nhất trong cột tương ứng. Sử dụng max để lấy giá trị lớn nhất của data.

d) Chia dữ liệu thành tập train và tập test theo tỷ lệ 8:2

Hướng dẫn: Sử dụng train_test_split từ thư viện sklearn để chia X, Y thành X_train, X_test, y_train, y_test

e) Sử dụng Cây quyết định để phân lớp và thực hiện các yêu cầu sau:

- Huấn luyện mô hình phân lớp trên tập train
  Hướng dẫn: sử dụng DecisionTreeClassifier của thư viện sklearn
- Dự đoán nhãn lớp cho tập test
  Hướng dẫn: sử dụng predict
- Tính và in ra Accuracy của mô hình trên tập test
  Hướng dẫn: sử dụng accuracy_score từ thư viện sklearn để tính accuracy giữa y dự đoán từ mô hình và y_test
- Tính và in ra Precision, Recall, F1-score của từng lớp và trung bình của mô hình trên tập test

Hướng dẫn: sử dụng classification_report từ thư viện sklearn để tính các độ đo trên giữa y dự đoán từ mô hình và y_test

- Hiển thị confusion matrix bằng heat map (bản đồ nhiệt)

  Hướng dẫn: sử dụng confusion_matrix từ thư viện sklearn để tính các độ đo trên giữa y dự đoán từ mô hình và y_test. Sử dụng heatmap của seaborn và matplotlib để vẽ bản đồ.

f) Sử dụng Naïve Bayes để phân lớp và thực hiện các yêu cầu như câu e.

Hướng dẫn: sử dụng GaussianNB của thư viện sklearn

g) Sử dụng K Nearest Neighbors Classifier để phân lớp và thực hiện các yêu cầu như câu e.

Hướng dẫn: sử dụng KNeighborsClassifier của thư viện sklearn

h) Sử dụng Neural Network để phân lớp và thực hiện các yêu cầu:

- Xây dựng Neural Network gồm 02 hidden layers và 01 output layer:
  - 01 hidden layer: Gồm 10 neuron, sử dụng hàm kích hoạt Relu và các neuron được kết nối đầy đủ (Fully connected), với input_shape là shape của tập X_train (4 thuộc tính).
  - 01 hidden layer: Gồm 20 neuron, sử dụng hàm kích hoạt Relu và các neuron được kết nối đầy đủ với các neuron ở lớp trước đó (Fully connected)
  - Output layer: Gồm số lượng neuron là số class cần phân lớp, sử dụng hàm kích hoạt Softmax để tính xác suất của từng class và các neuron được kết nối đầy đủ (Fully connected).

  Hướng dẫn: Sử dụng Sequential và Dense từ thư viện Keras với TensorFlow backend.

  Sequential là một loại mô hình mạng phổ biến trong Keras, cho phép xây dựng một chuỗi tuần tự các layer mạng nơ-ron.

  Dense: là một fully connected layer trong mạng. Các đơn vị trong layer này kết nối tất cả các đơn vị từ layer trước đó đến tất cả các đơn vị của layer hiện tại.

- Biên dịch mô hình với tối ưu hóa adam để điều chỉnh các trọng số; sử dụng sparse_categorical_crossentropy loss để đo lường sự khác biệt giữa dự đoán và

nhãn thực tế trong quá trình train; sử dụng accuracy để đánh giá hiệu suất của mô hình sau mỗi lần lặp huấn luyện.

Hướng dẫn: Sử dụng compile

- Huấn luyện mô hình dựa trên tập train (X_train, y_train) với số vòng lặp (epoch) là 100; số lượng mẫu dữ liệu được sử dụng trong mỗi lần cập nhật trọng số (batch_size) là 1. Hiển thị thông tin tiến trình huấn luyện sau mỗi epoch (verbose = 1)

  Hướng dẫn: Sử dụng fit
- Dự đoán nhãn lớp cho tập test

  Hướng dẫn: sử dụng predict để dự đoán xác suất cho mỗi mẫu và mỗi class. Sử dụng argmax của thư viện numpy để chọn ra class có xác suất lớn nhất của từng mẫu dữ liệu.
- Tính và in ra Accuracy của mô hình trên tập test
- Tính và in ra Precision, Recall, F1-score của từng lớp và trung bình của mô hình trên tập test
- Hiển thị confusion matrix bằng heat map (bản đồ nhiệt)

  Hướng dẫn: Thực hiện tương tư như câu e

i) Sử dụng các mô hình phân lớp đã huấn luyện ở trên (câu e, f, g,h) cho 03 mẫu dữ liệu mới sau:

[6.2, 2.9, 4.3, 1.3]

[5.1, 3.5, 1.4, 0.2]

[7.3, 2.8, 6.4, 2.1]

Hướng dẫn: Đưa 3 mẫu dữ liệu trên vào array. Sử dụng predict để dự đoán từ các mô hình phân lớp đã huấn luyện ở các câu trên. In ra nhãn lớp dự đoán.

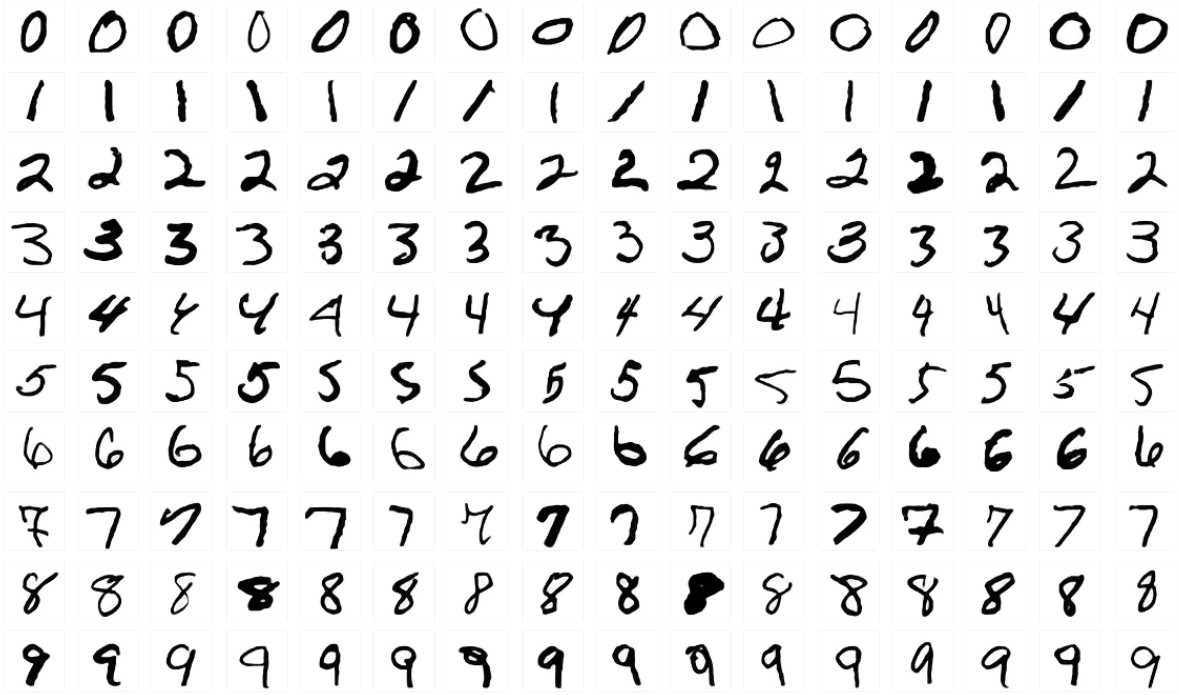
j) Nhận xét, đánh giá hiệu suất của các mô hình đã huấn luyện

2. (Lập trình) Sử dụng Neural Network để xây dựng mô hình phân lớp trên bộ dữ liệu MNIST từ thư viện Keras (chia train – test – validation)

Mô tả bộ dữ liệu MNIST: được tạo ra từ một tập hợp các ảnh kỹ thuật số của các chữ số viết tay từ 0 đến 9. Gồm 70.000 ảnh kỹ thuật số, trong đó 60.000 ảnh được sử dụng cho train và 10.000 ảnh được sử dụng cho test.

Mỗi ảnh trong bộ dữ liệu MNIST có kích thước 28x28 pixel và là ảnh đen trắng (grayscale). Các pixel được mã hóa dưới dạng giá trị từ 0 đến 255, trong đó 0 đại diện cho màu đen tuyệt đối và 255 đại diện cho màu trắng tuyệt đối.

Mỗi ảnh được gán một nhãn, tương ứng với chữ số mà nó đại diện. Nhãn được biểu diễn dưới dạng số nguyên từ 0 đến 9.



Thực hiện các yêu cầu sau:

a) Đọc bộ dữ liệu train, test của MNIST từ Keras

Hướng dẫn: Từ thư viện Kerast import MNIST và sử dụng hàm load_data để đọc bộ dữ liệu và ghi vào (train_images, train_labels), (test_images, test_labels)

b) In ra kích thước của tập train, test:

Hướng dẫn: Sử dụng shape

c) Chuẩn hóa các giá trị pixel của ảnh về đoạn [0,1]

Hướng dẫn: Chia 255.0

d) Chia tập train ban đầu thành tập train và tập validaion theo tỷ lệ 8:2

Hướng dẫn: Sử dụng train_test_split từ thư viện sklearn để chia train_images, train_labels thành chia train_images, train_labels, validation_images, validation_labels

In shape của các trập train và validation.

e) Xây dựng Neural Network gồm các layers:

- Flatten layer: Làm phẳng, chuyển đổi dữ liệu từ định dạng 2D (28x28 pixel) thành vector 1D (784 chiều)
- Dense layer: Gồm 128 neural, sử dụng hàm kích hoạt Relu và các neural được kết nối đầy đủ với các neural ở lớp trước đó (Fully connected)
- Drop out layer: ngẫu nhiên "tắt" một số neuron trong quá trình huấn luyện, nhằm giảm thiểu hiện tượng overfitting. Mỗi neuron sẽ có xác suất 20% để bị tắt.
- Dense layer: Gồm số lượng neural là số class cần phân lớp, sử dụng hàm kích hoạt Softmax để tính xác suất của từng class và các neural được kết nối đầy đủ (Fully connected).

Hướng dẫn: Sử dụng Sequential, Flatten, Dense, Dropout từ thư viện Keras với TensorFlow backend.

f) Biên dịch mô hình sử dụng optimizer: adam; loss: sparse_categorical_crossentropy; metric: accuracy

Hướng dẫn: Sử dụng compile

g) Huấn luyện mô hình dựa trên tập train, đánh giá với tập validation; epoch: 5; batch_size: 32.

Hướng dẫn: Sử dụng fit

h) Vẽ biểu đồ thể hiện accuracy, loss sau mỗi epoch.

Hướng dẫn: Sử dụng plot trong matplotlib

i) Dự đoán phân lớp cho tập test

Hướng dẫn: sử dụng predict để dự đoán xác suất cho mỗi mẫu và mỗi class. Sử dụng argmax của thư viện numpy để chọn ra class có xác suất lớn nhất của từng mẫu dữ liệu.

j) Tính và in ra Accuracy của mô hình trên tập test

k) Tính và in ra Precision, Recall, F1-score của từng lớp và trung bình của mô hình trên tập test

l) Hiển thị confusion matrix bằng heat map (bản đồ nhiệt)

m) In ra kết quả phân lớp của 05 ảnh đầu tiên trong tập test

Hướng dẫn:

Chạy image index trong range(5) để phân lớp 05 ảnh đầu tiên trong tập test.

Sử dụng expand_dims trong numpy để lấy ảnh từ tập test.

Sử dụng predict, argmax để phân lớp.

Sử dụng plot trong matplotlib để hiển thị ảnh.

n) In ra 05 kết quả đầu tiên phân lớp sai so với nhãn thực tế trong tập test

3. (Lập trình) Sử dụng Neural Network để xây dựng mô hình phân lớp trên bộ dữ liệu MNIST từ thư viện Keras (k-fold cross validation)

a) Đọc bộ dữ liệu train, test của MNIST từ Keras

b) In ra kích thước của tập train, test:

c) Chuẩn hóa các giá trị pixel của ảnh về đoạn [0,1]

d) Định nghĩa hàm create_model, bao gồm:

- Model là Neural Network tương tự như câu 2
- Biên dịch mô hình sử dụng optimizer: adam; loss: sparse_categorical_crossentropy; metric: accuracy

e) Huấn luyện mô hình bằng k-fold cross validation (với k=10).

Trong quá trình huấn luyện lưu và in ra kết quả loss và accuracy của từng fold, lưu lại mô hình sau mỗi fold.

Hướng dẫn:

- *Sử dụng KFold trong sklearn để định nghĩa kfold với n_spilts = 10, shutfle = True, random_state=42*

- *Với mỗi fold: train_index, val_index in kfold.split(train_images)*
    - *Gán model là kết quả của hàm create_model()*
    - *Gán X_train_fold, X_val_fold, y_train_fold, y_val_fold*
    - *Huấn luyện mô hình dựa trên tập train (X_train_fold, y_train_fold), đánh giá với tập validation (X_val_fold, y_val_fold); epoch: 5; batch_size: 32.*
    - *Lưu model huấn luyện được vào mảng models*
    - *Tính và in accuracy và loss của fold. Lưu vào mảng accuracy_per_fold[] và loss_per_fold[].*

f) Tính và in ra Mean Accuracy, Standard deviation accuracy (độ lệch chuẩn của Accuracy), Mean Loss, Standard deviation loss của mô hình trong quá trình huấn luyện

Hướng dẫn: Sử dụng mean, std trong numpy

g) Vẽ biểu đồ thể hiện accuracy, loss sau mỗi fold.

Hướng dẫn: Sử dụng plot trong matplotlib

h) Sử dụng mô hình có Accuracy tốt nhất giữa các fold trong quá trình huấn luyện và phân lớp cho tập test.

Hướng dẫn:

Sử dụng argmax của thư viện numpy để chọn ra index có accuracy lớn nhất trong accuracy_per_fold. Sau đó lấy ra models[index]

i) Tính và in ra Accuracy của mô hình trên tập test

j) Tính và in ra Precision, Recall, F1-score của từng lớp và trung bình của mô hình trên tập test

k) Hiển thị confusion matrix bằng heat map (bản đồ nhiệt)

l) In ra kết quả phân lớp của 05 ảnh đầu tiên trong tập test

Hướng dẫn: Thực hiện tương tự như câu 2

4. **(Lập trình) Sử dụng Neural Network để xây dựng mô hình phân lớp trên bộ dữ liệu Iris từ thư viện sklearn (k-fold cross validation):**

Thực hiện các yêu cầu sau:

a) **Đọc bộ dữ liệu Iris từ sklearn**

b) **Chuẩn hóa dữ liệu data về đoạn [0,1]**

c) **Định nghĩa hàm create_model**, bao gồm:

- Model là Neural Network tương tự như câu 1.h
- Biên dịch mô hình sử dụng optimizer: adam; loss: sparse_categorical_crossentropy; metric: accuracy

d) **Huấn luyện mô hình bằng k-fold cross validation (với k=10)**

Trong quá trình huấn luyện lưu và in ra kết quả accuracy, precision, recall, f1-score của từng fold. Lưu lại mô hình sau mỗi fold.

e) **In ra Average Accuracy, Average Recall, Average Precision, Average F1-score của mô hình sau quá trình huấn luyện với k-fold cross validation**

f) **In ra Average Recall, Average Precision, Average F1-score cho từng class của mô hình sau quá trình huấn luyện với k-fold cross validation.**

g) **Sử dụng các mô hình phân lớp có F1-score tốt nhất đã huấn luyện ở trên cho 03 mẫu dữ liệu mới sau:**

[6.2, 2.9, 4.3, 1.3]

[5.1, 3.5, 1.4, 0.2]

[7.3, 2.8, 6.4, 2.1]

5. **(Lập trình) Sử dụng Neural Network để dự đoán doanh thu của việc kinh doanh game.**

Bộ dữ liệu bao gồm tập train: "sales_data_training.csv" và tập test: "sales_data_training.csv", chứa thông tin về doanh số bán hàng của các tựa game trong quá khứ, cũng như các đặc điểm và thuộc tính của từng tựa game:

critic_rating: Điểm đánh giá trung bình của game từ các nhà phê bình, được đánh giá bằng số sao trên tổng số 5 sao.

is_action: Biến nhị phân (0 hoặc 1) cho biết tựa game có phải là game hành động hay không.

is_exclusive_to_us: Biến nhị phân cho biết tựa game có phải là độc quyền của cửa hàng bán game này hay không.

is_portable: Biến nhị phân cho biết tựa game có thể chơi trên hệ máy chơi game cầm tay hay không.

is_role_playing: Biến nhị phân cho biết tựa game có phải là game nhập vai hay không.

is_sequel: Biến nhị phân cho biết tựa game có phải là phần tiếp theo của một tựa game trước đó và là một phần của một loạt game liên tục hay không.

is_sports: Biến nhị phân cho biết liệu game có phải là game thể thao hay không.

suitable_for_kids: Biến nhị phân cho biết tựa game có phù hợp cho trẻ em mọi lứa tuổi hay không.

total_earnings: Tổng doanh thu mà cửa hàng đã kiếm được từ việc bán tựa game cho tất cả khách hàng.

unit_price: Giá bán lẻ cho mỗi bản sao của tựa game.

Thực hiện các yêu cầu sau:

a) **Đọc dữ liệu từ tập train và tập test (File csv)**

b) **Tách dữ liệu với thuộc tính total_earnings là thuộc tính cần dự đoán**

c) **Chuyển đổi dữ liệu train và test về [0, 1]**

d) **Chia tập train thành train và validation với tỷ lệ 8:2**

e) **Xây dựng và biên dịch mô hình Neural Network:**

- Mạng gồm các layers sau:
    - Dense layer: Gồm 50 neuron, sử dụng hàm kích hoạt Relu, với input_shape là 9 thuộc tính.
    - Dense layer: Gồm 100 neuron, sử dụng hàm kích hoạt Relu và các neuron được kết nối đầy đủ với các neuron ở lớp trước đó (Fully connected)
    - Dense layer: Gồm 50 neuron, sử dụng hàm kích hoạt Relu và các neuron được kết nối đầy đủ với các neuron ở lớp trước đó (Fully connected)
    - Dense layer (output): 1 neuron.

- Biên dịch mô hình:
    - Optimizer với Adam sử dụng learning_rate = 0.001
    - Loss = ‘mean_squared_error’
    - Metric = ‘mean_absolute_error’

f) Training mô hình với epochs = 100 và đánh giá với tập validation trong quá trình huấn luyện. In ra giá trị train_loss và val_loss sau mỗi epoch

g) Lưu model tốt nhất (val_loss thấp nhất) vào ổ cứng

h) Đánh giá Loss và Accuracy của mô hình trên tập test