Cấu Trúc Dữ Liệu Và Giải Thuật

Bài Tập Lớn 1

XỬ LÍ DỮ LIỆU MNIST VÀ HIỆN THỰC GIẢI THUẬT kNN BẰNG CẤU TRÚC DỮ LIỆU DANH SÁCH

nhóm thảo luận CSE
https://www.facebook.com/groups/211867931379013

Tp. Hồ Chí Minh, Tháng 2/2024

Mục lục

1 Kế hoạch làm bài

Phần Làm	Bắt đầu	Thời gian	Kết thúc	Kiến thức	Nộp lại discord anh
Task1	26/2	6 ngày	6:00 2/3	Danh sách liên kết	ID_task1
Task2	2/3	6 ngày	6:00 7/3	Danh sách liên kết	ID_task2
Task3(full)	7/3	6 ngày	6:00 13/3	Danh sách liên kết	ID_task3

Một số quy định của nhóm

- Nếu mấy bạn vô trễ thì thời gian làm có thể kéo dài theo ngày với thời gian Bắt đầu sẽ là ngày hôm đó
- Sau *deadline* thì anh không chỉ BTL phần đó nữa 
- Bạn nào trễ hạn 1 lần BTL2 anh sẽ không giảm giá
- Hỏi lý thuyết thì hỏi trong nhóm cho tiện
- Hỏi code thì nhắn riêng anh TA để bảo mật cho mấy bạn có thông báo trong nhóm
- Nộp bài thì gửi file code $kNN.cpp$ và $kNN.hpp$ chuyển thành tên $ID\_task2.cpp$ và $ID\_task2.hpp$ với ID trong excel nộp qua anh TA luôn

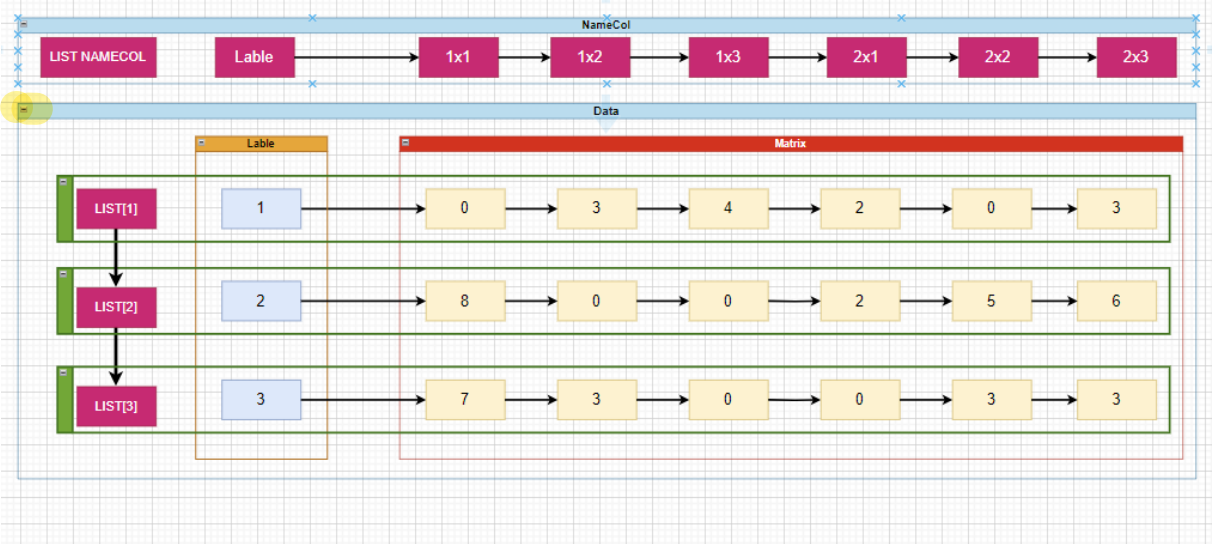
2 Task 2: Hiện thực Dataset

2.1 Lý Thuyết

Sinh viên được cung cấp file mnist.csv, trong đó:

- Mỗi dòng là mô tả một hình ảnh chữ số, được mã hóa thành 28x28 = 784 cột và một cột nhãn.
- Cột nhãn (label) đánh nhãn cho số được thể hiện bởi hình ảnh. Đối với mỗi ảnh, khoảng giá trị của ô trong cột này là từ 0 đến 9.
- Các cột còn lại, có tiêu đề ixj trong đó i, j nằm trong khoảng [1,28] thể hiện tọa độ của điểm ảnh. Đối với mỗi ảnh, khoảng giá trị của ô trong cột này là từ 0 đến 255 (độ đậm của điểm ảnh)

Sau khi đưa dữ liệu vào *Dataset* ta được cấu trúc dữ liệu sau

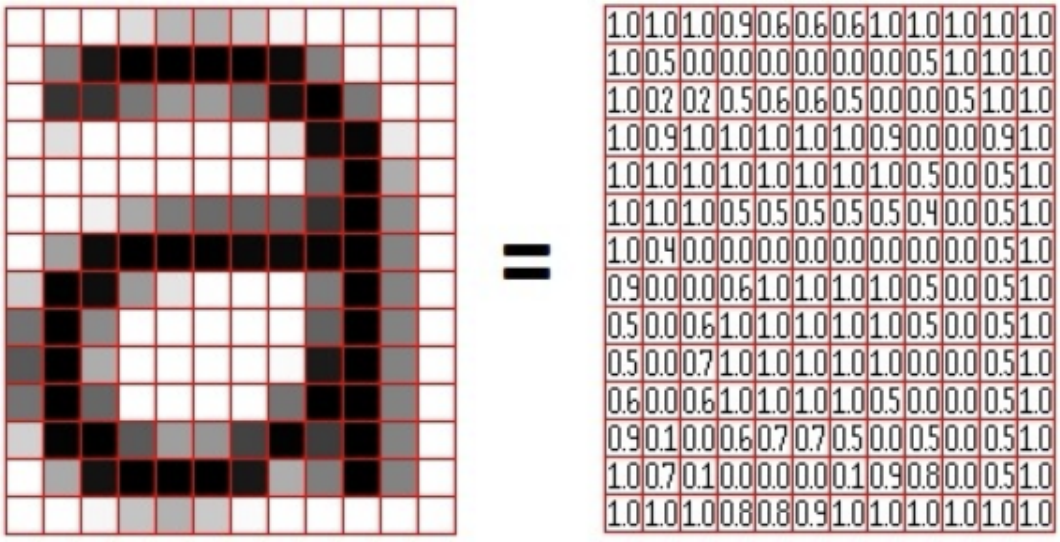


Hình 1: *Dataset ban đầu*

1. **NameCol :** là hàng đầu tiên trong *mnist.csv* là tên của của cột trong bảng được thiết kế như một danh sách liên kết được dùng ở bước trước

```
List<string>* nameCol;
// khởi tạo
nameCol = new Image<string>();
// các hàm hay dùng trong bài này đối với nameCol
List<T>* subList(int start, int end)
void printStartToEnd(int start, int end) const
void print() const
void push_back(T value)
int length() const
T& get(int index) const
```

2. **Data :** lầ phần còn lại của trong *mnist.csv* là danh sách ma trận 2 chiều với chiều được thiết kế bằng danh sách liên kết như hình vẽ với mỗi phần $List[N]$ trong *Data* là một danh sách liên kết con
   - *Lable* là kết quả của hình ảnh đang là số 1 đến số 9 phần sau ta sẽ cắt nó ra thành phần Y
   - *Matrix* là dữ liệu của ảnh đã được mã hóa thành một ma trận hai chiều đã được mã hóa thành một ma trận 1 chiều **(Xem cách chuyển array 2 chiều thành array)** phần sau ta sẽ cát nó ra thành phần X



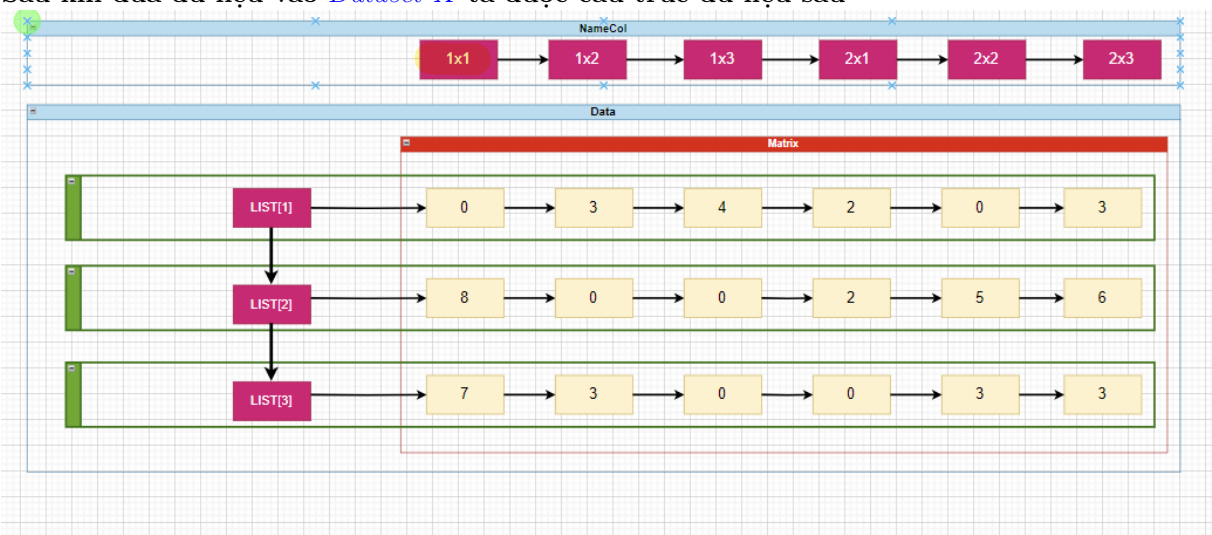
Hình 2: *Hình ảnh khi chuyển về ma trận với lable = a và matrix như hình bên phải*

2.1.1 Phần dữ liệu X (gồm X_train và X_test)

Phần dữ X là phần *matrix* phần này ta chưa xác định được ảnh là số nằm trong khoảng [1,9]

- X_train Bảng dữ liệu huấn luyện chứa các đặc trưng của ảnh. Số hàng là số lượng ảnh huấn luyện, Số cột là số lượng các đặc trưng. **có thể hiểu là phần này là phần dùng để cho con AI nó học thông minh lên**
- X_test Bảng dữ liệu dự đoán chứa các đặc trưng của ảnh. Số hàng là số lượng ảnh, Số cột là số lượng các đặc trưng. **có thể hiểu là phần này dùng để xem con AI nó có ngu hay không bằng cách đưa ảnh vào xem nó đoán được đúng số hay không**

Sau khi đưa dữ liệu vào *Dataset X* ta được cấu trúc dữ liệu sau



Hình 3: *Dataset X*

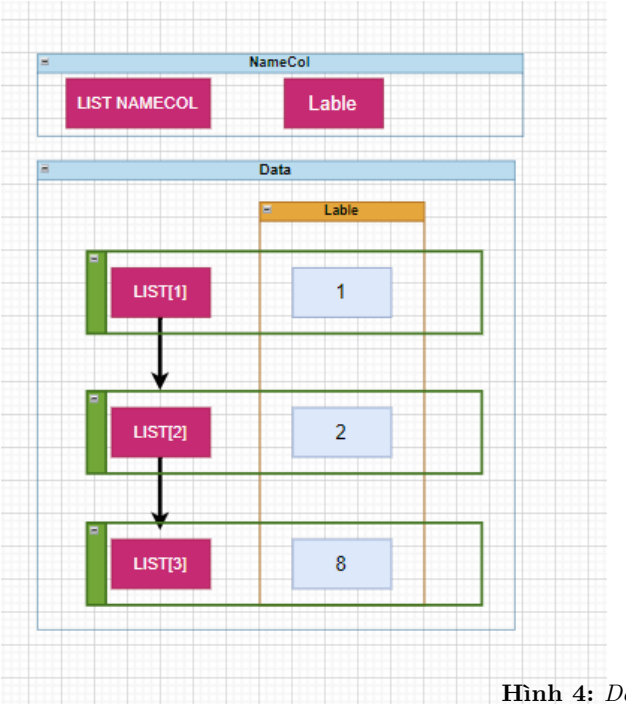
2.1.2 Phần dữ liệu Y (gồm Y_train và Y_test)

Phần dữ Y là phần *lable* phần này ta xác định được ảnh là số nằm trong khoảng [1,9]

- Y_train Bảng dữ liệu huấn luyện chứa nhãn của ảnh. Số hàng là số lượng ảnh huấn luyện, Số cột là 1. **có thể hiểu là phần này là phần dùng để cho con AI nó học thông minh lên**

- *Y_test* Bảng dữ liệu dự đoán chứa nhãn thực sự của ảnh. Số hàng là số lượng ảnh, Số cột là 1. **có thể hiểu là phần này dùng để xem con AI nó có ngu hay không bằng cách đưa ảnh vào xem nó đoán được đúng số hay không**

Sau khi đưa dữ liệu vào *Dataset Y* ta được cấu trúc dữ liệu sau



Hình 4: *Dataset Y*

2.2 Một số Hàm dùng trong BTL này

1. Khởi tạo và hủy *data*

```
// khởi tạo
this->nameCol = new Image<string>();
// hủy
delete data;
```

2. Lấy chiều rộng và chiều dài **Chú ý trường hợp rỗng**

```
nRows = this->data->length();
nCols = this->data->get(0)->length(); // không xảy ra với Th nRow = 0

// kết quả với dataset X
nRows = 3
nCols = 6

// kết quả với dataset Y
nRows = 1 // phần này đối với Y là cố định
nCols = 1
```

3. Print ra trong khoảng từ $[start, end]$

```
// print ra 5 phần tử hàng đầu tiên
data->get(0)->printStartToEnd(0, 5);
// kết quả với dataset X
0 3 7 5 0
// kết quả với dataset Y
1

// print ra 1 phần tử hàng thứ 3
data->get(2)->printStartToEnd(0, 1);
// kết quả với dataset X
7
// kết quả với dataset Y
8

// print ra tất cả hàng thứ 2
data->get(1)->print();
// kết quả với dataset X
8 0 0 2 5 6
// kết quả với dataset Y
2
```

4. Truy cập đến phần bất kì trong *matrix*

```
// lấy ra hàng thứ 2 là danh sách liên kết
data->get(1)

// lấy ra phần tử thứ 3 trong hàng thứ 3
data->get(2)->get(2)
```

5. Xóa một hàng hoặt một cột bất kì trong một *matrix*

```
// xóa đi hàng dầu tiên
data->remove(0)

// xóa đi cột thứ 2
for(int i = 0; i < row; i++) data->get(i)->remove(1);

// xóa phần tử hàng thứ 3 trong cột thứ 3
data->get(2)->remove(2);
```

6. Cắt ma trận

```
// lấy 2 hàng đầu tiên và 2 cột đầu tiên trong ma trận
Dataset result;
result.nameCol = nameCol->subList(0, 2);
for(int i = 0; i < 2; i++)
{
    result.data->push_back(data->get(i)->subList(0, 2));
}
```

2.2.1 Bài Tập Lớn

```
class Dataset {
private:
    List<List<int>*>* data;
    List<string>* NameCol;
    //You may need to define more
public:
    //! Hàm khởi tạo
    Dataset::Dataset()
    {
        this->NameCol = new Image<string>();
        this->data = new Image<List<int>*>();
    }
    //! Hàm hủy
    Dataset::~Dataset()
    {
        delete data;
        delete nameCol;
    }
    Dataset::Dataset(const Dataset& other);
    Dataset& Dataset::operator=(const Dataset& other);
    bool Dataset::loadFromCSV(const char* fileName);
    void Dataset::getShape(int& nRows, int& nCols) const;
    void Dataset::columns() const;
    void printHead(int nRows = 5, int nCols = 5) const;
    void printTail(int nRows = 5, int nCols = 5) const;
    bool drop(int axis = 0, int index = 0, std::string columns = "");
    Dataset extract(int startRow = 0, int endRow = -1, int startCol = 0, int endCol
     ↪  = -1) const;


    Dataset predict(const Dataset& X_train, const Dataset& Y_train, const int k)
     ↪  const;
    double score(const Dataset& y_test) const;
};
```

1. **data** và **nameCol** đã được định nghĩa trong phần lý thuyết
2. Hàm khởi tạo rỗng sẽ gán các giá trị **data** và **nameCol** cấp phát bộ trớ trong vùng nhớ *heap*
3. Hàm hủy là hủy đi **data** và **nameCol**
4. Hàm khởi tạo có tham số đầu vào *other* hàm này yêu cầu *copy* đối tượng *other* sang cho đối tượng cần được khởi tạo
5. Hàm **operator=** hàm này dùng để *copy* giống hàm trên
6. Hàm **loadFromCSV** khi chạy hết test case các bạn nên hiện thực lại không bị chấm đạo văn, Đoạn code trong đó chỉ mang tính tham khảo

7. Hàm **getShape(int& nRows, int& nCols)** gán *nRows* và *nCols* là chiều dài và chiều rỗng của *matrix*

```
X_train.getShape(nRows, nCols)
// kết quả
nRows = 3
nCols = 6

Y_train.getShape(nRows, nCols)
// kết quả
nRows = 3
nCols = 1 // luôn luôn = 1 chỉ có một cột lable
```

8. Hàm **columns()** in ra tất cả các tên

```
X_train.columns();
// kết quả
1x1 1x2 1x3 2x1 2x2 2x3

Y_train.columns();
// kết quả
lable
```

9. Hàm **printHead(int nRows = 5, int nCols = 5)** In ra nRows dòng đầu tiên, và chỉ in nCols cột đầu tiên của bảng dữ liệu.
   - Dòng đầu tiên, in ra tên các cột của bảng dữ liệu (dùng *nameCol*)
   - Từ dòng thứ 2 trở đi, in giá trị của mỗi ô trong bảng, mỗi phần tử cách nhau bằng khoảng trắng, không có khoảng trắng dư ở cuối dòng (dùng *data* và hàm *printStartToEnd* để xử lí)
   - Nếu nRows lớn hơn số lượng dòng trong bảng dữ liệu, in hết dòng trong bảng dữ liệu. Nếu nCols lớn hơn số lượng cột trong bảng dữ liệu, in hết cột trong bảng dữ liệu. Nếu nRows hoặc nCols nhỏ hơn 0, không in gì cả.

```
X_train.printHead(2, -1);
// kết quả

X_train.printHead(2, 2);
// kết quả
1x1 2x2
0 3
8 0

Y_train.printHead(2, 2);
// kết quả
lable
1
2
```

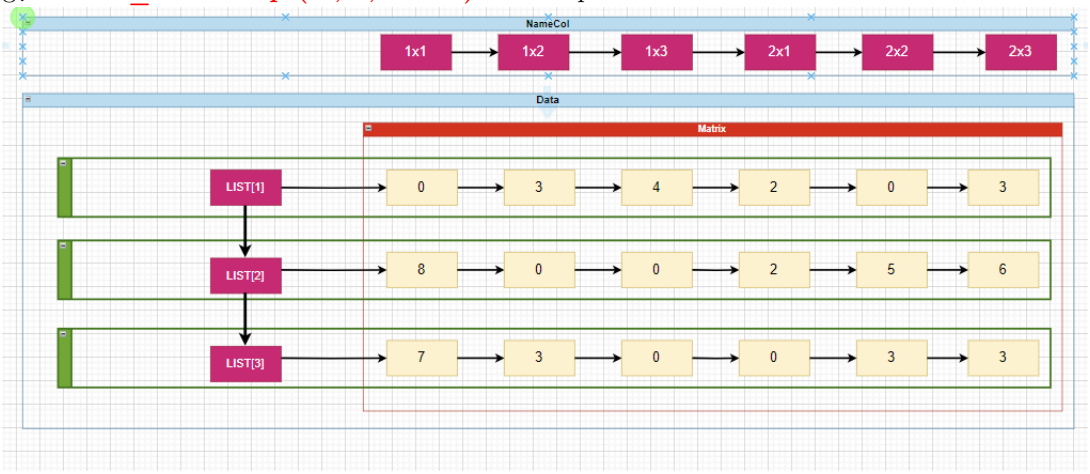
10. Hàm **printTail(int nRows = 5, int nCols = 5)** In ra nRows dòng cuối cùng, và chỉ in nCols cột cuối cùng của bảng dữ liệu..
    - Dòng đầu tiên, in ra tên các cột của bảng dữ liệu (dùng *nameCol*)

- Từ dòng thứ 2 trở đi, in giá trị của mỗi ô trong bảng, mỗi phần tử cách nhau bằng khoảng trắng, không có khoảng trắng dư ở cuối dòng (dùng *data* và hàm *printStartToEnd* để xử lí)
- Nếu nRows lớn hơn số lượng dòng trong bảng dữ liệu, in hết dòng trong bảng dữ liệu. Nếu nCols lớn hơn số lượng cột trong bảng dữ liệu, in hết cột trong bảng dữ liệu. Nếu nRows hoặc nCols nhỏ hơn 0, không in gì cả.

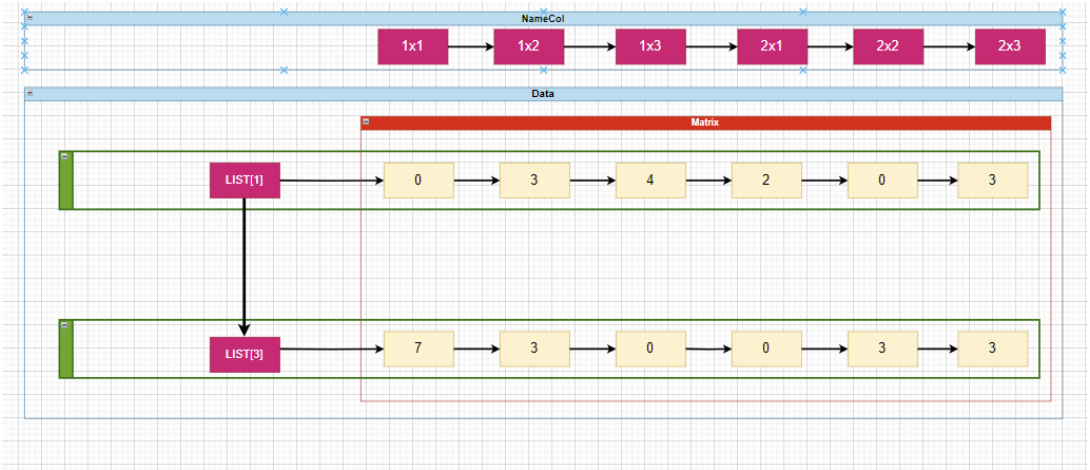
```
X_train.printTail(2, -1);
// kết quả

X_train.printTail(2, 2);
// kết quả
2x2 2x3
5 6
3 3

Y_train.printTail(2, 2);
// kết quả
lable
2
8
```

11. Hàm **drop (int axis = 0, int index = 0, std::string columns = "")**
    - Xóa một hàng hoặc cột của bảng dữ liệu.
    - Nếu axis khác 0 hoặc 1, hàm không làm gì cả và trả về false.
    - Nếu axis == 0, thực hiện xóa một hàng tại vị trí index, sau đó trả về true. Nếu index >= số dòng hoặc < 0, hàm không làm gì cả và trả về false.
    - Nếu axis == 1, thực hiện xóa một cột có tên trùng với columnName, sau đó trả về true. Nếu columnName không nằm trong danh sách tên các cột, hàm không làm gì cả và trả về false.

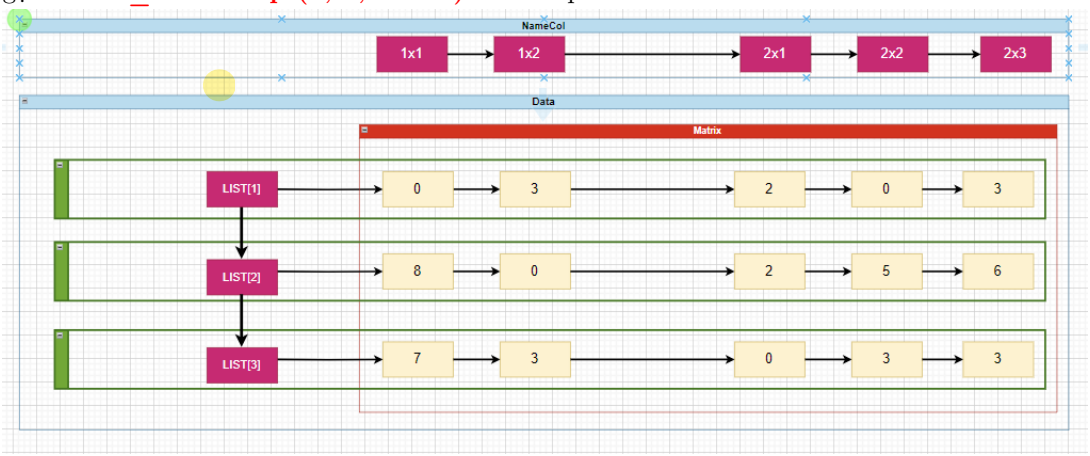
    (a) gọi hàm **X_train.drop (-1, 0, "1x3")** thì kết quả sẽ như hình bên dưới



    (b) gọi hàm **X_train.drop (0, 1, "1x3")** thì kết quả sẽ như hình bên dưới



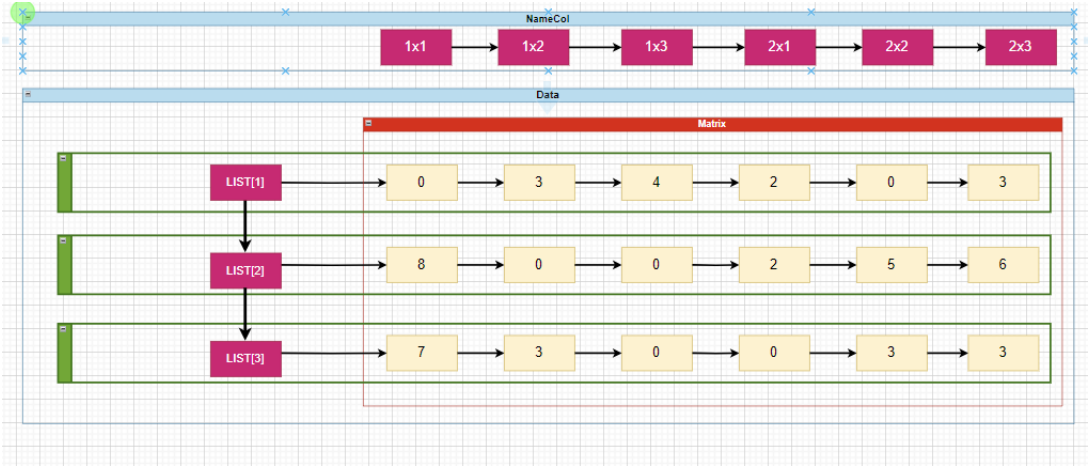
(c) gọi hàm **X_train.drop (1, 1, "1x3")** thì kết quả sẽ như hình bên dưới



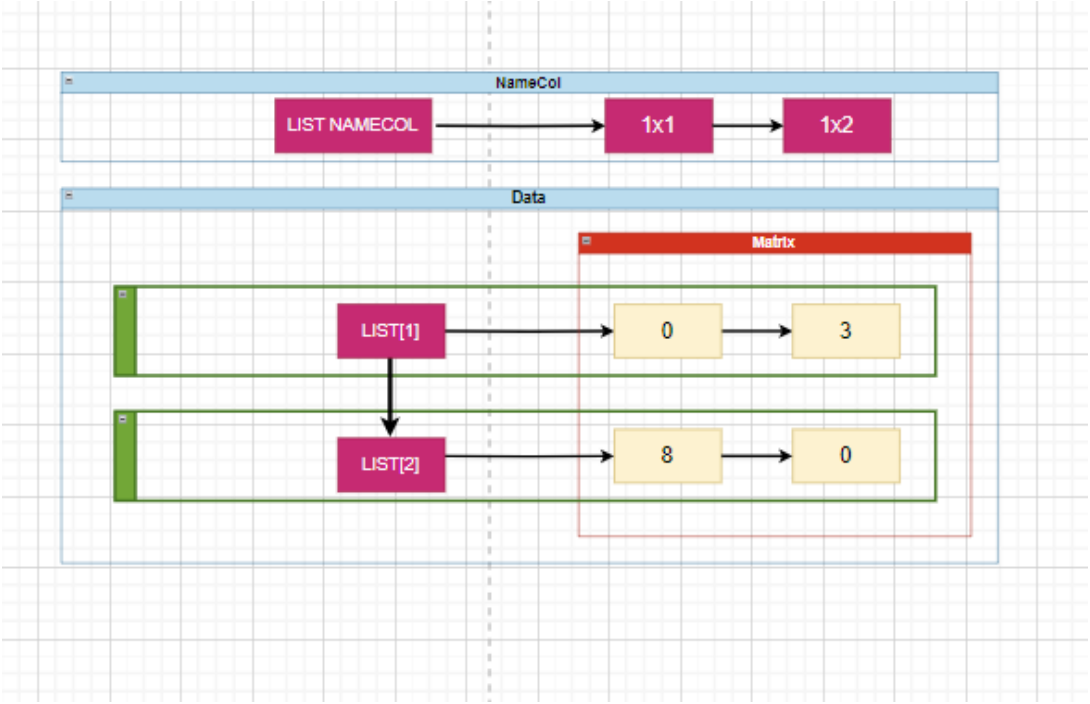
12. Hàm **extract(int startRow = 0, int endRow = -1, int startCol = 0, int endCol = -1)**

- Phương thức dùng để trích xuất một phần của bảng dữ liệu, sau đó trả về bảng dữ liệu đã trích xuất.
- Trong đó: startRow là hàng bắt đầu, endRow là hàng cuối cùng, startCol là cột bắt đầu, endCol là cột cuối cùng.
- Nếu endRow == -1, ta sẽ lấy tất cả các hàng. Tương tự với endCol == -1. **và trong trường hợp endRow, endCol lớn hơn kích thước thì cũng lấy hết tới cuối**
- Các testcase sẽ đảm bảo startRow, endRow, startCol và endCol nằm trong khoảng giá trị hợp lệ (từ 0 đến số hàng/số cột) và start <= end

(a) gọi hàm **X_train.extract (0,-1,0,-1)** thì kết quả sẽ như hình bên dưới



(b) gọi hàm **X_train.extract (0, 1, 0, 1)** thì kết quả sẽ như hình bên dưới



13. Hàm **distanceEuclidean(const List<int>* x, const List<int>* y)** Copy từ class *Image* xuống và chỉnh lại. Và bỏ *distanceEuclidean* trong class *Image*

3 Các Khóa Học HK232

nhóm thảo luận CSE

https://www.facebook.com/groups/211867931379013

- Lớp BTL1 + GK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL2 + CK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL1 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL2 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL1 + BTL2 + GK + Harmony của môn PPL HK232
- Lớp BTL3 + BTL4 + CK + Harmony của môn PPL HK232

CHÚC CÁC EM HỌC TỐT