MACHINE LEARNING: BÀI THỰC HÀNH SỐ 3 – PHẦN LOGISTIC REGRESSION.

A. PHƯƠNG PHÁP K-NN CHO PHÂN LOẠI NHỊ PHÂN (K-NN BINARY CLASSIFIER)

Trong bài thực hành trước chúng ta đã áp dụng phương pháp K-NN cho bài toán hồi quy, trong đó với đầu vào **x**, đầu ra dự đoán $y = \frac{1}{K}\sum_{i=1}^{K} y_{n_i}$, trong đó tập hợp $\{x_{n_i}\}_{i=1}^{K} \subseteq X$ – là K điểm gần **x** nhất **trong tập dữ liệu huấn luyện**. Trong phần này chúng ta tiếp tục áp dụng phương pháp này cho bài toán phân loại nhị phân. Ở đây ta xét đầu ra **y** thuộc 01 trong 02 khả năng (02 loại): **0 hoặc 1**.

Để đơn giản ta sẽ sử dụng lại các thao tác như trường hợp Hồi quy. Các bước của phương pháp K-NN có thể mô tả:

(i) Với mỗi **x** – unseen ở đầu vào (các phần tử trong tập Validation)

(ii) Tính khoảng cách $\mathbf{d_i}$ từ **x** đến các $\mathbf{x_i}$ trong tập Training => Lập thành 1 mảng.

(iii) Sắp xếp các khoảng cách này theo thứ tự tăng dần, nhưng cần có tham chiếu để biết chỉ số phần tử ban đầu trong mảng $\mathbf{d_i}$

(iv) Tìm ra K chỉ số của các phần tử $\mathbf{x_i}$ trong tập Training ứng với các $\mathbf{d_i}$ nhỏ nhất.

(v) Lấy đầu ra dự đoán y_pred = 1 nếu $(\Sigma\, y_i)/K \geq 0.5$ và y_pred = 0 nếu $(\Sigma\, y_i)/K < 0.5$.

Giải thích bước (v): Do **y** chỉ nhận giá trị 0 hoặc 1, nên nếu $(\Sigma\, y_i)/K \geq 0.5$ chứng tỏ tỉ lệ xuất hiện của **1** nhiều hơn và ngược lại.

Ta sử dụng lại các phương thức tính khoảng cách và phương thức tìm K mẫu dữ liệu gần **x** nhất đã có trong bài trước.

Hàm tính khoảng cách:

Trường hợp d = 1 (dữ liệu 01 chiều – là biến đơn)

```
def distance(array, value):
    array = np.array(array)
    return np.absolute(array - value)
```

Trường hợp d > 1:

```
def distance(array, value):
    array = np.array(array)
    return np.linalg.norm(array - value, ord = 2, axis=<dim_axis>)
```

Ở đây <dim_axis> sẽ là chiều phù hợp, cần kiểm tra qua việc xét kích thước mỗi chiều (shape) như bài trước.

Phương thức để tìm K phần tử gần nhất với **x** chúng ta vẫn dùng: **indexes = np.argsort(array_D)[:k]** trong đó array_D là mảng khoảng cách từ **x** đến các mẫu dữ liệu trong tập huấn luyện (bước (ii) ở trên).

Giả sử mảng các khoảng cách từ **x** đến các phần tử trong tập training tương ứng là **Array_D.** Đoạn code dự đoán đầu ra **y** cho đầu vào **x** bất kỳ sẽ như sau (các bạn tự sửa để code gọn hơn):

```
indexes = np.argsort(array_D)[:k]
y_pred = 0
for i in range(k):
    y_pred = y_pred + y[indexes[i] ]
y_pred = y_pred/k
if (y_pred >= 0.5):
    y_pred = 1
else:
    y_pred = 0
```

Ví dụ A.1. Chúng ta bắt đầu với một dữ liệu vui, với đầu vào một biến, đầu ra phân loại 02 lớp (nguồn từ https://en.wikipedia.org/wiki/Logistic_regression): Một nhóm 20 sinh viên dành thời gian trong khoảng từ 0 đến 6 giờ cho việc ôn thi. Thời gian ôn thi này ảnh hưởng đến xác suất sinh viên vượt qua kỳ thi như thế nào?

Theo đề bài chúng ta có 02 lớp là vượt qua kỳ thi (class 1) và không qua kỳ thi (class 0). Giả sử dữ liệu cho như trong bảng sau

Id	Hours	Pass	Id	Hours	Pass
1	0.50	0	11	2.75	1
2	0.75	0	12	3.00	0
3	1.00	0	13	3.25	1
4	1.25	0	14	3.50	0
5	1.50	0	15	4.00	1
6	1.75	0	16	4.25	1
7	1.75	1	17	4.50	1
8	2.00	0	18	4.75	1
9	2.25	1	19	5.00	1
10	2.50	0	20	5.50	1

Đoạn chương trình tạo dữ liệu như dưới đây:

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(2)
X = np.array([[0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 1.75, 2.00, 2.25, 2.50,
              2.75, 3.00, 3.25, 3.50, 4.00, 4.25, 4.50, 4.75, 5.00, 5.50]])
y = np.array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1])
```

Các bạn hãy sử dụng các đoạn code đã có và tham khảo lại phương pháp K-NN cho bài toán hồi quy để xây dựng chương trình dự đoán kết quả thi cho các mẫu dữ liệu đầu vào dưới đây:

Student ID	21	22	23	24	25
Hours	2.45	1.85	3.75	3.21	4.05

Hãy lấy K = 3, 4, 5 và so sánh kết quả thu được.

Ví dụ A.2. Xét lại **Ví dụ B.4** ở phần Hồi quy tuyến tính (bài thực hành số 2), với dữ liệu trong tệp vidu4_lin_reg.txt (tệp văn bản). Trong y sinh học, bề dày lớp nội trung mạc (NTM) phản ánh một số bệnh lý của cơ thể. Một nghiên cứu cho thấy nếu độ dày lớp nội trung mạc (NTM) lớn hơn hoặc bằng 1.0mm, bệnh nhân có nguy cơ cao mắc bệnh đái tháo đường Type 2.

Tham khảo phần đọc dữ liệu từ tệp văn bản đã có trong ví dụ trước. Các trường dữ liệu gồm:

ID	Mã bệnh nhân
TUOI	Tuổi
BIM	chỉ số khối lượng cơ thể (Body Mass Index)
HA	huyến áp tâm thu
GLUCOSE	đường huyết
CHOLESTEROL	độ Cholesterol trong máu
BEDAYNTM	độ dày NTM

Hãy đọc dữ liệu từ tệp, sau đó dựa vào độ dày NTM để tạo dữ liệu đầu ra Y tương ứng là nguy cơ cao mắc đái tháo đường Type2 vào phân lớp 1 (class 1); ngược lại vào class 0.

Tiếp theo hãy bỏ các trường dữ liệu ID và BEDAYNTM, phần còn lại sẽ là dữ liệu đầu vào. Chia dữ liệu thành: 80 dòng đầu dùng cho training; 20 dòng sau dùng cho Validation. Hãy áp dụng phương pháp K-NN cho bài toán phân loại này và kiểm tra độ chính xác của mô hình.

Hãy tham khảo lại phần Naïve Bayes để có các độ đo tính chính xác của mô hình phân loại nhị phân:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
```

B. PHƯƠNG PHÁP HỒI QUY LOGISTIC

Ví dụ B.1. Chúng ta sử dụng lại dữ liệu trong **Ví dụ A.1**. Tuy nhiên bây giờ ta sẽ áp dụng mô hình hồi quy Logistic cho ví dụ này.

a. Chương trình Python

- Trong ví dụ này, chúng ta sẽ tự xây dựng các phương thức chính của mô hình hồi quy Logistic, chỉ dựa vào các công cụ cơ bản trong thư viện Numpy.
- Khai báo thư viện và khởi tạo dữ liệu (sau đó vẽ ra để hình dung dữ liệu), **X** là số giờ học; **y** là nhãn với y = 1 nếu qua và y = 0 nếu trượt.

```
from __future__ import division, print_function, unicode_literals
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(2)
X = np.array([[0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 1.75, 2.00, 2.25, 2.50,
               2.75, 3.00, 3.25, 3.50, 4.00, 4.25, 4.50, 4.75, 5.00, 5.50]])
y = np.array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1])

# extended data by adding a column of 1s (x_0 = 1)
X = np.concatenate((np.ones((1, X.shape[1])), X), axis = 0)
```

- In dữ liệu ra màn hình để hình dung trực quan

```
X0 = X[1, np.where(y == 0)][0]
y0 = y[np.where(y == 0)]
X1 = X[1, np.where(y == 1)][0]
y1 = y[np.where(y == 1)]

plt.plot(X0, y0, 'ro', markersize = 8)
plt.plot(X1, y1, 'bs', markersize = 8)
plt.show()
```

- Nhìn vào hình vẽ (và cả từ bảng dữ liệu) tạo ra từ đoạn code trên ta có thể thấy dữ liệu không thực sự tách được (ví dụ sinh viên thứ 7 chỉ học 1.75h thì qua, trong lúc sinh viên 12 học 3h thì trượt). Tuy nhiên về tổng thể xu hướng sẽ là thời gian càng nhiều thì khả năng qua càng cao.
- Xây dựng các phương thức cần cho mô hình Hồi quy Logistic gồm: Hàm sigmoid (logistic) tính g(z) và hàm hồi quy logistic

```
def sigmoid(s):
    return 1/(1 + np.exp(-s))

def logistic_sigmoid_regression(X, y, w_init, eta, tol = 1e-4, max_count = 10000):
# method to calculate model logistic regression by Stochastic Gradient Descent method
# eta: learning rate; tol: tolerance; max_count: maximum iterates
    w = [w_init]
    it = 0
    N = X.shape[1]
    d = X.shape[0]
    count = 0
    check_w_after = 20
# loop of stochastic gradient descent
    while count < max_count:
        # shuffle the order of data (for stochastic gradient descent).
        # and put into mix_id
        mix_id = np.random.permutation(N)
        for i in mix_id:
            xi = X[:, i].reshape(d, 1)
            yi = y[i]
            zi = sigmoid(np.dot(w[-1].T, xi))
            w_new = w[-1] + eta*(yi - zi)*xi
            count += 1
            # stopping criteria
            if count%check_w_after == 0:
                if np.linalg.norm(w_new - w[-check_w_after]) < tol:
                    return w
            w.append(w_new)
    return w
```

- In ra kết quả (bộ hệ số w)

```
eta = .05
d = X.shape[0]
w_init = np.random.randn(d, 1)

w = logistic_sigmoid_regression(X, y, w_init, eta)
print(w[-1])
```

- In ra xác suất rơi vào lớp 1 (qua) của các phần tử dữ liệu trong tập training:

```
print(sigmoid(np.dot(w[-1].T, X)))
```

- In kết quả trực quan

```
X0 = X[1, np.where(y == 0)][0]
y0 = y[np.where(y == 0)]
X1 = X[1, np.where(y == 1)][0]
y1 = y[np.where(y == 1)]

plt.plot(X0, y0, 'ro', markersize = 8)
plt.plot(X1, y1, 'bs', markersize = 8)

xx = np.linspace(0, 6, 1000)
w0 = w[-1][0][0]
w1 = w[-1][1][0]
threshold = -w0/w1
yy = sigmoid(w0 + w1*xx)
plt.axis([-2, 8, -1, 2])
plt.plot(xx, yy, 'g-', linewidth = 2)
plt.plot(threshold, .5, 'y^', markersize = 8)
plt.xlabel('studying hours')
plt.ylabel('predicted probability of pass')
plt.show()
```

Hãy thực hiện lệnh để thu được bộ hệ số, quan sát kết quả chạy và dùng bộ hệ số thu được để chạy dự đoán cho kết quả thi của các sinh viên với số giờ học sau

Student ID	21	22	23	24	25
Hours	2.45	1.85	3.75	3.21	4.05

Hãy so sánh kết quả với phương pháp K-NN ở phần A.

Ví dụ B.2. Ta tiếp tục với một dữ liệu tự tạo trong không gian hai chiều (d = 2). Số điểm dữ liệu trong ví dụ này cũng là 20. Chúng ta sẽ sử dụng độ đậm của màu để minh họa xác suất một điểm rơi vào phân lớp nào, từ đó có cái nhìn trực quan về cách phân lớp của phương pháp hồi quy Logistic.

Đoạn code tạo dữ liệu như dưới đây, trong đó dòng lệnh cuối cùng ta đã tạo 2 cột ứng với tọa độ x_1, x_2 của dữ liệu, tức là mỗi điểm tọa độ $x = (x_1, x_2)$, và ở đây ta có N = 20 điểm.

```
from __future__ import division, print_function, unicode_literals
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist

# generate list of data points
np.random.seed(22)

means = [[2, 2], [4, 2]]
cov = [[.7, 0], [0, .7]]
N = 20
```

```
X1 = np.random.multivariate_normal(means[0], cov, N)
X2 = np.random.multivariate_normal(means[1], cov, N)
```

Dưới đây là đoạn code vẽ dữ liệu để minh họa trực quan

```
plt.plot(X1[:, 0], X1[:, 1], 'bs', markersize = 8, alpha = 1)
plt.plot(X2[:, 0], X2[:, 1], 'ro', markersize = 8, alpha = 1)
plt.axis('equal')
plt.ylim(0, 4)
plt.xlim(0, 5)

# hide tikcs
cur_axes = plt.gca()
cur_axes.axes.get_xaxis().set_ticks([])
cur_axes.axes.get_yaxis().set_ticks([])

plt.xlabel('$x_1$', fontsize = 20)
plt.ylabel('$x_2$', fontsize = 20)

# save the figure to an image first
plt.savefig('logistic_2d.png', bbox_inches='tight', dpi = 300)
plt.show()
```

Nhắc lại các hàm cần thiết sử dụng trong hồi quy logistic mà ta đã lập ở bài trên. Chú ý số chiều của dữ liệu sẽ được lấy từ tham đối của hàm, do đó chúng ta có thể sử dụng đoạn code này cho dữ liệu có số chiều bất kỳ

```
def sigmoid(s):
    return 1/(1 + np.exp(-s)) # calculate sigmoid function

def logistic_sigmoid_regression(X, y, w_init, eta, tol = 1e-4, max_count = 10000):
    w = [w_init]
    it = 0
    N = X.shape[1]
    d = X.shape[0]
    count = 0
    check_w_after = 20
    while count < max_count:
        # mix data for stochastic gradient descent method
        mix_id = np.random.permutation(N)
        for i in mix_id:
            xi = X[:, i].reshape(d, 1)
            yi = y[i]
            zi = sigmoid(np.dot(w[-1].T, xi))
            w_new = w[-1] + eta*(yi - zi)*xi
            count += 1
            # stopping criteria
            if count%check_w_after == 0:
                if np.linalg.norm(w_new - w[-check_w_after]) < tol:
                    return w
            w.append(w_new)
    return w
```

Đoạn code sau đây bổ sung thêm cột X0 ≡1 vào bên trái để có dữ liệu Xbar = (1, X_1, X_2), khởi tạo xấp xỉ ban đầu cho bộ tham số w (chọn ngẫu nhiên bằng random), sau đó gọi hàm logistic_sigmoid_regression ở trên để thực hiện quá trình lặp gradient descent ngẫu nhiên tìm tham số tối ưu.

```
X = np.concatenate((X1, X2), axis = 0).T

y = np.concatenate((np.zeros((1, N)), np.ones((1, N))), axis = 1).T

# Xbar

X = np.concatenate((np.ones((1, 2*N)), X), axis = 0)

eta = 0.05

d = X.shape[0]

w_init = np.random.randn(d, 1) # initialize parameters w = w_init
```

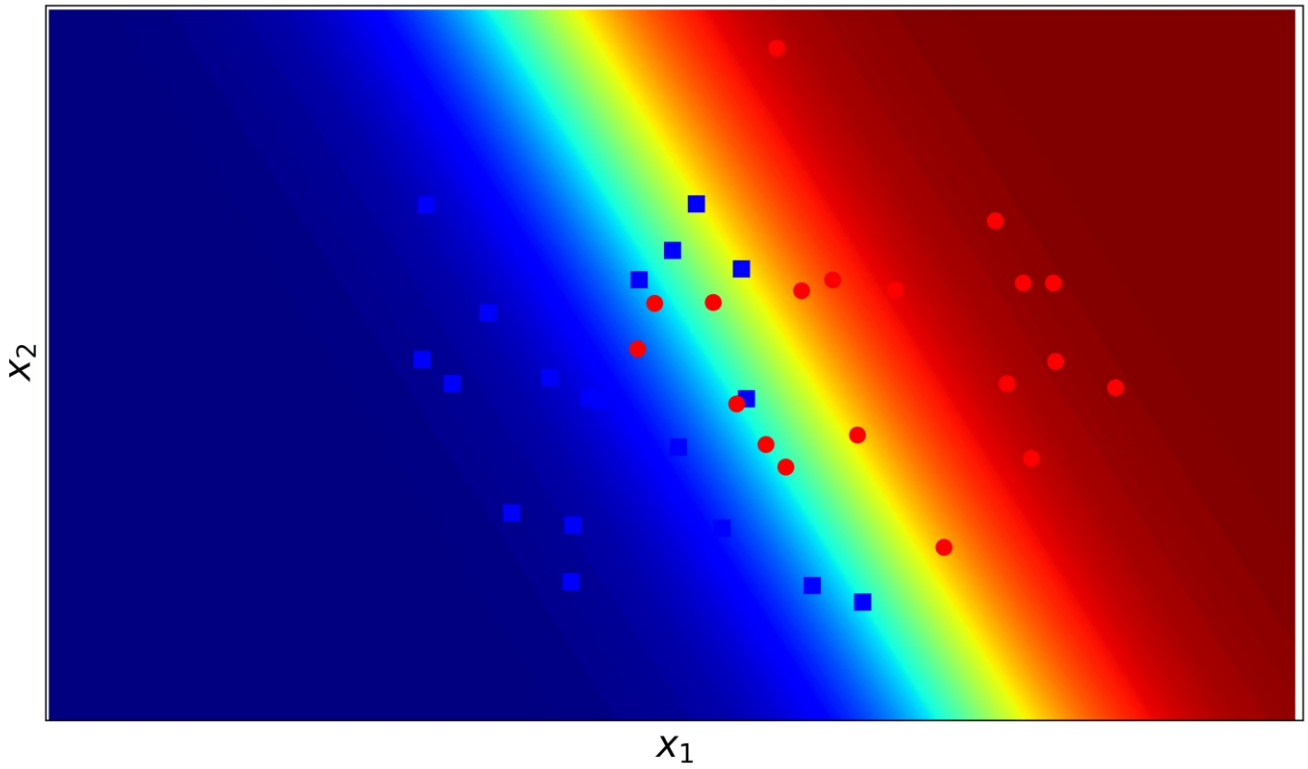
```
# call logistic_sigmoid_regression procedure

w = logistic_sigmoid_regression(X, y, w_init, eta, tol = 1e-4, max_count= 10000)

# print out the parameter

print(w[-1])
```

Trong đoạn code dưới đây, chúng ta sẽ dùng độ đậm của màu để minh họa xác suất một điểm sẽ thuộc class nào. Ở đây ta dùng 2 màu là xanh và đỏ, màu đỏ càng đậm nghĩa là xác suất điểm dữ liệu thuộc lớp đỏ càng cao; ngược lại màu xanh càng đậm nghĩa là xác suất điểm dữ liệu thuộc lớp xanh càng cao. Để thực hiện, trước hết ta sử dụng các hệ số tính được theo phương pháp hồi quy logistic với dữ liệu training ở trên. Sau đó ta tạo tập dữ liệu là toàn bộ các điểm (x1m, x2m) trong khoảng [-1, 6]×[0, 4] (thật ra ta lấy các điểm với bước lưới 0.025). Sau đó ta tính xác suất thuộc lớp 1 (đỏ) cho tất cả các điểm x = (x1m, x2m) trên lưới mà chúng ta vừa tạo ra: zm = P(c|x) = sigmoid($w^T\mathbf{x}$) = sigmoid($w_0 + w_1*x1m + w_2*x2m$). Cuối cùng chúng ta dùng hàm contourf để kẻ các đường đồng mức (cùng giá trị) zm với các màu tương ứng.

Cụ thể chúng ta theo dõi trong đoạn code dưới đây:

```
# Make data.
x1m = np.arange(-1, 6, 0.025) # generate data coord. X1
xlen = len(x1m)
x2m = np.arange(0, 4, 0.025)  # generate data coord. X2
x2en = len(x2m)
x1m, x2m = np.meshgrid(x1m, x2m) # create mesh grid X = (X1, X2)

# now assign the parameter w0, w1, w2 from array w which was computed above
w0 = w[-1][0][0]
w1 = w[-1][1][0]
w2 = w[-1][2][0]

# calculate probability zm=P(c|x)=sigmoid(w^Tx)=sigmoid(w0+w1*x1m+w2*x2m)
zm = sigmoid(w0 + w1*x1m + w2*x2m)

# plot contour of prob. zm by the saturation of blue and red
# more red <=> prob. that data point belong to red class is higher & vise versa
CS = plt.contourf(x1m, x2m, zm, 200, cmap='jet')

# finally, plot the data and take a look
plt.plot(X0[:, 0], X0[:, 1], 'bs', markersize = 8, alpha = 1)
plt.plot(X1[:, 0], X1[:, 1], 'ro', markersize = 8, alpha = 1)
plt.axis('equal')
plt.ylim(0, 4)
plt.xlim(0, 5)

# hide tikcs
cur_axes = plt.gca()
cur_axes.axes.get_xaxis().set_ticks([])
cur_axes.axes.get_yaxis().set_ticks([])

plt.xlabel('$x_1$', fontsize = 20)
plt.ylabel('$x_2$', fontsize = 20)
plt.savefig('logistic_2d_2.png', bbox_inches='tight', dpi = 300)
plt.show()
```

Nếu các bạn thực hiện thành công, kết quả sẽ như hình dưới đây



Ví dụ B.3 (Bài tập 1): Tiếp theo chúng ta xét một ví dụ được lấy từ dữ liệu tuyển sinh sau đại học (master) của Ấn độ. Link lấy dữ liệu ở đây: https://www.kaggle.com/mohansacharya/graduate-admissions hoặc từ tệp Admission_Predict.csv đính kèm.

Các trường dữ liệu như sau

- GRE (Graduate Record Exam) Scores (0..340): bảng điểm học tập đại học
- TOEFL Scores (0.. 120): Điểm tiếng Anh (toefl)
- University Rating (0.. 5): Điểm xếp loại đại học
- SOP (Statement of Purpose) Strength (0..5): Điểm bài viết tự giới thiệu
- LOR (Letter of Recommendation) Strength (0..5): Điểm cho thư giới thiệu
- Undergraduate GPA - CGPA (0..10): Điểm trung bình ĐH
- Research Experience (0 hoặc 1): kinh nghiệm nghiên cứu (chỉ 1 – có hoặc 0 – không)
- Chance of Admit (Số thực 0 .. 1): Khả năng được chọn

Mô tả chi tiết hơn về dữ liệu có thể tìm thấy trong link:

https://www.researchgate.net/publication/348433004_Graduate_Admission_Prediction_Using_Machine_Learning

Chú ý file online trên đường link có thể được update nên khác dữ liệu gửi kèm theo bài thực hành này. Trong tệp đính kèm, chúng ta có 400 bản ghi ứng với các hồ sơ.

Đoạn lệnh dưới đây ví dụ về việc đọc tệp csv, sau đó lấy các cột dữ liệu vào các mảng X1, X2, … Ở đây chúng ta sẽ bỏ qua trường đầu tiên (‘Serial No.’). Nếu mô hình của các bạn cần các trường này, hãy tự bổ sung lệnh cần thiết.

```
# importing module
import numpy as np
from pandas import *
# reading CSV file
data = read_csv("Admission_Predict.csv")
# converting column data to list, then convert list to array
```

```
sn = data['Serial No.'].tolist()


gre = data['GRE Score'].tolist()
X1 = np.asarray(gre)


tfl = data['TOEFL Score'].tolist()
X2 = np.asarray(tfl)


unirt = data['University Rating'].tolist()
X3 = np.asarray(unirt)


sop = data['SOP'].tolist()
X4 = np.asarray(sop)


lor1 = data['LOR '].tolist()
X5 = np.asarray(lor1)


cgpa1 = data['CGPA'].tolist()
X6 = np.asarray(cgpa1)


research_exp = data['Research'].tolist()
X7 = np.asarray(research_exp)


prob_Admit = data['Chance of Admit'].tolist()
Yt = np.asarray(prob_Admit)
# printing list data
```

a) Phân loại bằng phương pháp hồi quy Logistic

Trong ví dụ này ta sẽ giả thiết 'Chance of Admit' ≥ 0.75 thì ứng viên tương ứng được chọn (thuộc class 1); ngược lại sẽ không được chọn (thuộc class 0).

Chúng ta tham khảo đoạn code trên để đọc dữ liệu, sau đó sử dụng các hàm **sigmoid** và **logistic_sigmoid_regression** đã viết ở ví dụ trước (chú ý các hàm này không phụ thuộc số chiều dữ liệu) để thực hiện tìm tham số cho siêu phẳng tách. Các bước thực hiện như sau:

i. Đọc dữ liệu, chọn ra 350 dòng đầu làm dữ liệu training, phần còn lại là dữ liệu test. Đổi các dòng dữ liệu sang dạng cột (thêm .T vào sau – đọc các đoạn code trong ví dụ trước).

ii. Sắp xếp dữ liệu để có ma trận dữ liệu X = (X1, X2, …, X7).

iii. Bổ sung cột X0 ≡ 1 vào bên trái của ma trận X để được Xbar.

iv. Bỏ qua các đoạn code vẽ dữ liệu vì trong ví dụ này chúng ta có số chiều là 7 > 2. Gọi các hàm để thực hiện quá trình tính hệ số bằng hồi quy logistic.

v. In các hệ số kết quả ra màn hình.

vi. Sử dụng các hệ số kết quả, dự đoán cho dữ liệu test và đối chiếu kết quả với dữ liệu đúng. Chú ý ở đây ta chỉ đối chiếu phân loại, không so sánh giá trị của khả năng trúng tuyển.

Tính toán các đại lượng đo độ chính xác: Accuracy, Precission, Recall.

b) Bài tập thực hành: Dự đoán khả năng bằng hồi quy tuyến tính

Vẫn chia tập dữ liệu training – test như trên. Hãy sử dụng phương pháp hồi quy tuyến tính để ước lượng khả năng trúng tuyển của các hồ sơ có dữ liệu trong tập test.

Tính trung bình bình phương sai số bằng cách lấy tổng bình phương sai khác giữa kết quả tính theo hồi quy và kết quả thực tế (cột cuối) sau đó chia cho số mẫu.

c) Bài tập tự thực hành: Hãy sử dụng phương pháp Naïve Bayes phù hợp để phân loại dữ liệu nói trên. Tìm hiểu và sử dụng hàm đo thời gian để tính

- Thời gian chạy của phương pháp Naïve Bayes và Logistic Regression;
- Độ chính xác tính theo Accuracy, Recall, Precision của mỗi phương pháp nói trên.

d) Sử dụng thư viện Scikit-Learn:Lớp mô hình Logistic Regression cũng thuộc gói các lớp mô hình tuyến tính linear_model trong thư viện Scikit-Learn. Tương tự mô hình hồi quy tuyến tính, để sử dụng thư viện, chúng ta cần import các gói tương ứng:

```
# Sử dụng thư viện scikit-learn
from sklearn import linear_model
logReg = linear_model.LogisticRegression(penalty='none')
# Training & compute weights w
logReg.fit(X_train, Y_train)


# For new data X_test, predict
logReg.predict(X_test)
# For showing of w
print(logReg.coef_)
```

Ở đây penalty là tùy chọn cho phần hiệu chỉnh, mặc định là hiệu chỉnh L_2. Hãy sử dụng thư viện scikit-learn để thực nghiệm lại các ví dụ 1 và 2, sau đó so sánh kết quả (bộ tham số tối ưu; kết quả dự đoán; độ chính xác) của trường hợp code tự xây dựng và sử dụng thư viện. Nếu có sự khác nhau, hãy giải thích lý do.

Ví dụ B.4 (Bài tập 2): Ngân hàng Bồ Đào Nha đã có sự sụt giảm doanh thu và họ muốn biết cần thực hiện những cải cách gì. Sau khi điều tra, họ phát hiện ra rằng nguyên nhân sâu xa là do khách hàng của họ không đầu tư đủ cho các khoản tiền gửi dài hạn. Vì vậy, ngân hàng muốn xác định những khách hàng hiện tại có khả năng đăng ký tiền gửi dài hạn cao hơn và tập trung nỗ lực tiếp thị vào những khách hàng đó.Trong tệp **banking.csv** đính kèm có 41188 bản ghi liên quan đến thông tin khách hàng, gồm 20 trường dữ liệu quan sát và 01 trường y là đầu ra (dự báo) tương ứng: $y = 1$ nếu khách hàng CÓ đăng ký tiền gửi dài hạn; $y = 0$ nếu ngược lại. Các trường dữ liệu quan sát (đầu vào) mô tả như dưới đây

Feature	Feature_Type	Description
age	numeric	age of a person
job	Categorical,nominal	type of job ('admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
marital	categorical,nominal	marital status ('divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
education	categorical,nominal	('basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

Feature	Feature_Type	Description
default	categorical,nominal	has credit in default? ('no','yes','unknown')
housing	categorical,nominal	has housing loan? ('no','yes','unknown')
loan	categorical,nominal	has personal loan? ('no','yes','unknown')
contact	categorical,nominal	contact communication type ('cellular','telephone')
month	categorical,ordinal	last contact month of year ('jan', 'feb', 'mar', ..., 'nov', 'dec')
day*of*week	categorical,ordinal	last contact day of the week ('mon','tue','wed','thu','fri')
duration	numeric	last contact duration, in seconds . Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no')
campaign	numeric	number of contacts performed during this campaign and for this client (includes last contact)
pdays	numeric	number of days that passed by after the client was last contacted from a previous campaign (999 means client was not previously contacted)
previous	numeric	number of contacts performed before this campaign and for this client
poutcome	categorical,nominal	outcome of the previous marketing campaign ('failure','nonexistent','success')

Có thể tham khảo chi tiết tại link: https://www.kaggle.com/janiobachmann/bank-marketing-dataset .

Trước hết cần chuyển các trường dữ liệu kiểu text-categories/nomial (dữ liệu dạng phân loại/danh nghĩa) như day_of_week; month; contact… sang dạng danh sách các số định danh, ví dụ yes/no chuyển thành 1/0; 'jan', 'feb', 'mar'… chuyển thành 1, 2, 3…

Chú ý có 2 loại dữ liệu Categories/Nomial. Kiểu thứ nhất như yes/no; month; dayofweek ta có thể chuyển thành các số 0, 1, 2…

Loại thứ 2 như education; marital… chúng ta cần chuyển thành dạng vector kiểu như trong bag of words, tức là lập vector có số chiều = tổng số danh nghĩa, sau đó ứng với danh nghĩa nào, ta đặt vị trí thành phần đó = 1, còn lại đặt là 0 (one host coding)

Đoạn code đọc dữ liệu và chuyển đổi sang thuần số (cho một số trường minh họa) như sau:

```
import pandas as pd
# change to your data's path
data=pd.read_csv("D:\\Teach_n_Train\\Machine
Learning\\exam_n_practice\\Log_Reg\\banking.csv")
data.head()

# convert field of 'month'
dict_month = {'jan' : 1, 'feb' : 2, 'mar' : 3, 'apr' : 4, 'may' : 5, 'jun' : 6,
              'jul' : 7, 'aug' : 8, 'sep' : 9, 'oct' : 10, 'nov' : 11, 'dec' : 12}
data['month'] = data['month'].map(dict_month)

# convert field of dayOfweek
dict_day = {'sun' : 1, 'mon' : 2, 'tue' : 3, 'wed' : 4, 'thu' : 5, 'fri' : 6,
            'sat' : 7}
data['day_of_week'] = data['day_of_week'].map(dict_day)

# conver binary fields
#default :
data.default.replace({'no' : 0, 'yes' : 1}, inplace = True)
```

```
#housing :
data.housing.replace({'no' : 0, 'yes' : 1}, inplace = True)
#loan :
data.loan.replace({'no' : 0, 'yes' : 1}, inplace = True)

# convert categories field by one host coding
marital_dummies = pd.get_dummies(data['marital'], prefix = 'marital')
marital_dummies.drop('marital_divorced', axis=1, inplace=True)
data = pd.concat([data, marital_dummies], axis=1)

job_dummies = pd.get_dummies(data['job'], prefix = 'job')
job_dummies.drop('job_unknown', axis=1, inplace=True)
data= pd.concat([data, job_dummies], axis=1)

education_dummies = pd.get_dummies(data['education'], prefix = 'education')
education_dummies.drop('education_unknown', axis=1, inplace=True)
data = pd.concat([data, education_dummies], axis=1)

contact_dummies = pd.get_dummies(data['contact'], prefix = 'contact')
#contact_dummies.drop('contact_unknown', axis=1, inplace=True)
data = pd.concat([data, contact_dummies], axis=1)

poutcome_dummies = pd.get_dummies(data['poutcome'], prefix = 'poutcome')
#poutcome_dummies.drop('poutcome_unknown', axis=1, inplace=True)
data = pd.concat([data, poutcome_dummies], axis=1)

data['pdays'] = data['pdays'].apply(lambda row: 0 if row == -1 else 1)

data.drop(['job', 'education', 'marital', 'contact', 'poutcome'], axis=1,inplace=True)
```

Tự hoàn thành việc chuyển đổi các trường dữ liệu categories/nomial dạng text còn lại.

Sau khi có dữ liệu thuần số, hãy chia dữ liệu thành phần Training và phần Test theo tỷ lệ 8:2, trong đó với phần Test, trường *y* sẽ không dùng đến.

a) Sau đó sử dụng mô hình hồi quy logistic với phần dữ liệu Training, và áp dụng để dự đoán phần Test. Cuối cùng sử dụng các độ đo Accuracy, precision, recall và F1-score ($\beta = 1$) để kiểm tra độ chính xác của mô hình nói trên.

b) Sử dụng mô hình Naïve Bayes phù hợp với tập huấn luyện như trên, sau đó chạy dự đoán với tập Test và tính độ chính xác của mô hình, như ý a.

c) So sánh thời gian chạy cũng như độ chính xác của hai phương pháp.

Ví dụ B.6 (Bài tập 3). Chúng ta sử dụng dữ liệu chứa các thông tin về nhân khẩu, hành vi thói quen và lịch sử bệnh lý để xác định khả năng bị trụy tim ở một bệnh nhân. Bộ dữ liệu được cung cấp công khai trên trang web Kaggle và nó là từ một nghiên cứu tim mạch đang diễn ra trên cư dân của thị trấn Framingham, Massachusetts. Mục tiêu phân loại là dự đoán liệu bệnh nhân có nguy cơ mắc bệnh tim mạch vành (CHD) trong tương lai (10 năm) hay không. Bộ dữ liệu cung cấp thông tin của bệnh nhân, có thể tìm thấy tại link https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset/data hoặc lấy từ tệp đính kèm framingham.csv. Nó bao gồm hơn 4.000 bản ghi và 15 thuộc tính.

Mỗi thuộc tính là một yếu tố rủi ro tiềm ẩn. Có cả các yếu tố rủi ro về nhân khẩu học, hành vi và y tế.

Nhân khẩu học:

• Giới tính: male - nam hoặc nữ (nomial)

• Tuổi: age - Tuổi của bệnh nhân;(Liên tục - Mặc dù tuổi được ghi đã bị cắt bớt thành số nguyên nhưng khái niệm tuổi là liên tục)

Hành vi

• Người hút thuốc hiện tại: currentSmoke - bệnh nhân có hút thuốc hay không (Nomial)

• Cigs Per Day: số điếu thuốc mà một người hút trung bình trong một ngày. (có thể coi là liên tục vì một người có thể hút bao nhiêu điếu, thậm chí nửa điếu.)

Tiền sử y tế/bệnh

• Thuốc BP: BPMed - bệnh nhân có dùng thuốc huyết áp hay không (yes/no)

• Đột quỵ trước đó: Prevalent Stroke - trước đó bệnh nhân có bị đột quỵ hay không (yes/no)

• Prevalent Hyp: bệnh nhân có tăng huyết áp hay không (yes/no)

• Đái tháo đường: Diabetes - bệnh nhân có bị đái tháo đường hay không (Danh định)

Tình trạng y tế (hiện tại)

• Tot Chol: mức cholesterol toàn phần (Liên tục)

• Sys HA: huyết áp tâm thu (Liên tục)

• Dia BP: huyết áp tâm trương (Liên tục)

• BMI: Chỉ số khối cơ thể (Liên tục)

• Heart Rate: nhịp tim (Liên tục - Trong nghiên cứu y học, các biến số như nhịp tim mặc dù trên thực tế là rời rạc, nhưng vẫn được coi là liên tục vì số lượng lớn các giá trị có thể có.)

• Glucose: mức glucose (Liên tục)

Biến dự đoán (đầu ra y)

• Nguy cơ mắc bệnh mạch vành CHD trong 10 năm (nhị phân: “1”, nghĩa là “Có”, “0” nghĩa là “Không”)

Một số trường dữ liệu chứa giá trị lỗi N/A. Giả sử dữ liệu đã được đọc vào đối tượng df (pandas data). Câu lệnh sau thống kê các trường có N/A trong dữ liệu:

```
df = pd.read_csv("D:\\Teach_n_Train\\Machine Learning\\exam_n_practice\\Log_Reg\\framingham
.csv")
df.head()
df.isnull().sum()
```

Đoạn lệnh dưới đây loại bỏ các thành phần N/A trong dữ liệu:

```
df=df.dropna(how="any", axis=0)
```

Chia dữ liệu thành các tập Training – Validation theo tỉ lệ 7:3. Hãy sử dụng mô hình hồi quy logistic đã huấn luyện trên tập Training để dự đoán cho tập Validation. Tính độ chính xác của mô hình theo các độ đo Accuracy, Recall và Precision.