

BÁO CÁO LAB

Môn học: Phương pháp học máy trong an toàn thông tin

Tên chủ đề: Lab 1

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.O21.ATCL.1

STT	Họ và tên	MSSV	Email
1	Nguyễn Đại Nghĩa	21521182	21521182@gm.uit.edu.vn
2	Hoàng Gia Bảo	21521848	21521848@gm.uit.edu.vn
3	Trương Đặng Văn Linh	21520328	21520328@gm.uit.edu.vn
4	Mai Quốc Cường	21521901	21521901@gm.uit.edu.vn

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

BÁO CÁO CHI TIẾT

Câu 1: Sinh viên cho ví dụ về phép cộng, trừ hai ma trận numpy

Để cho ví dụ về phép cộng, trừ hai ma trận numpy thì trước hết em cần đảm bảo rằng 2 ma trận này có cùng kích thước, vì thế em sẽ tạo 2 ma trận có cùng kích thước như sau:

```
# Tạo ma trận A và B
A = np.array([[1, 1], [8, 2]])
B = np.array([[1, 8], [4, 8]])
```

Sau đó, em sử dụng toán tử + để cộng chúng lại với nhau và sử dụng toán tử - để trừ 2 ma trận:

```
# Cộng hai ma trận
C = A + B

# Trừ hai ma trận
D = A - B
```

Tổng thể sẽ trông như sau:

```
import numpy as np

# Tạo ma trận A và B
A = np.array([[1, 1], [8, 2]])
B = np.array([[1, 8], [4, 8]])

# Cộng hai ma trận
C = A + B

# Trừ hai ma trận
D = A - B

print("Ma trận A:\n", A)
print("Ma trận B:\n", B)
print("Tổng 2 ma trận:\n", C)
print("Hiệu 2 ma trận:\n", D)
```

Chạy đoạn code trên, kết quả nhận được là:

```
Ma trận A:  
[[1 1]  
 [8 2]]  
Ma trận B:  
[[1 8]  
 [4 8]]  
Tổng 2 ma trận:  
[[ 2  9]  
 [12 10]]  
Hiệu 2 ma trận:  
[[ 0 -7]  
 [ 4 -6]]
```

Câu 2: Sinh viên sử dụng pandas xử lý các yêu cầu.

Trước tiên em sẽ tạo tập tin CSV có cấu trúc như yêu cầu trong lab bằng đoạn code sau:

```
import pandas as pd  
  
# Dữ liệu từ bảng được cung cấp sẵn  
data = {  
    'X': [78, 85, 80, 96, 86],  
    'Y': [84, 94, 83, 94, 86],  
    'Z': [86, 97, 73, 96, 83]  
}  
  
# Tạo DataFrame từ dữ liệu  
df = pd.DataFrame(data, index=[1, 2, 3, 4, 5])  
  
# Lưu DataFrame vào tập tin CSV  
df.to_csv('Lab1_Bai2.csv', index=True)
```

Sau khi thực thi đoạn code trên, em tiến hành thực hiện lần lượt các yêu cầu sau:

1. Đọc CSV thành Dataframe và hiển thị:

```
import pandas as pd  
import numpy as np  
  
# Đọc tập tin CSV vào DataFrame  
df = pd.read_csv('Lab1_Bai2.csv', index_col=0)  
print("Đọc CSV thành Dataframe và hiển thị:")  
print(df)
```

Kết quả:

➡ Đọc CSV thành Dataframe và hiển thị:

	X	Y	Z
1	78	84	86
2	85	94	97
3	80	83	73
4	96	94	96
5	86	86	83

2. Hãy chuyển index mặc định thành giá trị cột id:

```
import pandas as pd

# Chuyển index mặc định thành giá trị cột id
df = pd.read_csv('Lab1_Bai2.csv')
df.rename(columns={'Unnamed: 0': 'id'}, inplace=True)
df.set_index('id', inplace=True)
print(df)
```

Kết quả:

⊗

	X	Y	Z
id			
1	78	84	86
2	85	94	97
3	80	83	73
4	96	94	96
5	86	86	83

3. Sắp xếp dữ liệu theo nhiều cột (sort):

Ở đây cụ thể em sẽ sắp xếp dữ liệu theo 2 cột là X và Y:

```
import pandas as pd

# Sắp xếp dữ liệu theo cột X và Y
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
df_sorted = df.sort_values(by=['X', 'Y'])
print("\nDataFrame sau khi sắp xếp theo X và Y:")
print(df_sorted)
```

Kết quả:



DataFrame sau khi sắp xếp theo X và Y:

	X	Y	Z
id			
1	78	84	86
3	80	83	73
2	85	94	97
5	86	86	83
4	96	94	96

4. Chọn một cột cụ thể và hiển thị nó:

Ở đây em sẽ chọn cụ thể là cột z để hiển thị:



```
import pandas as pd

#Chọn một cột cụ thể và hiển thị nó (chọn cột Z)
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
column_z = df['Z']
print("\nCột Z:")
print(column_z)
```

Kết quả:



```
Cột Z:
id
1      86
3      73
2      97
5      83
4      96
Name: Z, dtype: int64
```

5. Chọn 2 hàng đầu tiên và hiển thị chúng:

```

import pandas as pd

# Chọn 2 hàng đầu tiên và hiển thị chúng
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
first_two_rows = df.head(2)
print("\nHai hàng đầu tiên của DataFrame:")
print(first_two_rows)

```

Kết quả:

```

Hai hàng đầu tiên của DataFrame:
      X  Y  Z
id
1   78  84  86
3   80  83  73

```

6. Hãy chọn một hàng dựa trên một điều kiện giá trị của cột:

Ở đây em sẽ chọn điều kiện là ở cột z với giá trị z bằng 97:

```

import pandas as pd

# Chọn một hàng dựa trên một điều kiện cụ thể của cột (Z = 97)
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
row_condition = df[df['Z'] == 97]
print("\nHàng với điều kiện Z = 97:")
print(row_condition)

```

Kết quả:

```

Hàng với điều kiện Z = 97:
      X  Y  Z
id
2   85  94  97

```

7. Thay đổi một vài giá trị thành NaN ở CSV, sau đó đọc lên thành Dataframe và thay thế chúng bằng giá trị 0:

Cụ thể em sẽ thay giá trị có id 3 ở cột X và id 2 ở cột Y thành NaN:

```

import pandas as pd

# Thay đổi một vài giá trị thành NaN và sau đó thay thế chúng bằng giá trị 0
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
df.loc[3, 'X'] = np.nan
df.loc[2, 'Y'] = np.nan
df = df.fillna(0).astype('Int64')
print("\nDataFrame sau khi thay thế NaN bằng 0:")
print(df)

```

Kết quả:

```

Dataframe sau khi thay thế NaN bằng 0:
      X  Y  Z
id
1    78  84  86
3     0  83  73
2    85   0  97
5    86  86  83
4    96  94  96

```

8. Ở cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn là False trong Dataframe:

```

import pandas as pd

# Trong cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn hoặc bằng là False
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
df['Z'] = df['Z'] > 90
print("\nDataFrame cột Z sau khi chuyển giá trị lớn hơn 90 là True và bé hơn là False:")
print(df)

```

Kết quả:

```

Dataframe cột Z sau khi chuyển giá trị lớn hơn 90 là True và bé hơn là False:
      X  Y  Z
id
1    78  84 False
3     0  83 False
2    85   0  True
5    86  86 False
4    96  94  True

```

9. Chuyển Dataframe trên thành 2 Dataframe d1 và d2; d1 chứa cột X và Y, d2 chứa cột Z; cuối cùng d3 là thành quả của nối 2 Dataframe d1 và d2:

```
import pandas as pd

# Chuyển DataFrame thành 2 DataFrame d1 và d2; d1 chứa cột X và Y, d2 chứa cột Z
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
d1 = df[['X', 'Y']]
d2 = df[['Z']]
print("\nDataFrame d1:")
print(d1)
print("\nDataFrame d2:")
print(d2)

# Nối d1 và d2 theo cột
df_concatenated = pd.concat([d1, d2], axis=1)
print("\nDataFrame sau khi ghép d1 và d2:")
print(df_concatenated)
```

Kết quả:

⊗

DataFrame d1:

	X	Y
id		
1	78	84
3	0	83
2	85	0
5	86	86
4	96	94

DataFrame d2:

	Z
id	
1	86
3	73
2	97
5	83
4	96

DataFrame sau khi ghép d1 và d2:

	X	Y	Z
id			
1	78	84	86
3	0	83	73
2	85	0	97
5	86	86	83
4	96	94	96

10. Dùng tính năng thống kê hãy hiển thị kết quả thống kê các giá trị thuộc tính của DataFrame:

```
import pandas as pd

# Dùng tính năng thống kê để hiển thị kết quả thống kê các giá trị thuộc tính của DataFrame
df = pd.read_csv('Lab1_Bai2.csv')
df.set_index('id', inplace=True)
statistics = df.describe()
print("\nThống kê cơ bản của DataFrame:")
print(statistics)
```

Kết quả:



Thống kê cơ bản của DataFrame:

	X	Y	Z
count	5.00000	5.000000	5.000000
mean	69.00000	69.400000	87.000000
std	39.10243	39.035881	9.924717
min	0.00000	0.000000	73.000000
25%	78.00000	83.000000	83.000000
50%	85.00000	84.000000	86.000000
75%	86.00000	86.000000	96.000000
max	96.00000	94.000000	97.000000

Câu 3: Sinh viên tự tìm hiểu thực hiện lại ví dụ dùng mô hình Linear Regression trong thư viện scikit-learning bằng các thư viện khác

Scikit-learn:

Tổng quan: Scikit-learn là một thư viện học máy phổ biến, cung cấp các thuật toán cơ bản trong machine learning như Decision Tree, Logistic Regression, và nhiều thuật toán khác.

Tính năng đáng chú ý: Cung cấp các thuật toán cơ bản trong machine learning và hỗ trợ các công cụ như cross-validation và feature selection.

Case sử dụng: Phù hợp khi cần áp dụng các thuật toán machine learning truyền thống trên dataset có kích thước nhỏ.

```
import numpy as np
from sklearn.linear_model import LinearRegression
# X is a matrix that represents the training dataset
# y is a vector of weights, to be associated with input dataset
X = np.array([[3], [5], [7], [9], [11]]).reshape(-1, 1)
y = [8.0, 9.1, 10.3, 11.4, 12.6]
lreg_model = LinearRegression()
lreg_model.fit(X, y)
# New data (unseen before)
new_data = np.array([[13]])
print('Model Prediction for new data: $%.2f' %
      lreg_model.predict(new_data)[0] )
```

Model Prediction for new data: \$13.73

TensorFlow:

Tổng quan: TensorFlow là một trong những thư viện deep learning hàng đầu, được sử dụng rộng rãi trong cả nghiên cứu và sản xuất.

Tính năng đáng chú ý: TensorFlow cung cấp một cách tiếp cận cấp thấp hơn so với Keras, cho phép người dùng tùy chỉnh mô hình một cách linh hoạt hơn. Ngoài ra, TensorFlow cũng hỗ trợ nhiều công cụ và tài nguyên để xử lý dữ liệu lớn và triển khai mô hình trên nhiều nền tảng.

Tốc độ và kích thước dữ liệu: TensorFlow có tốc độ thực thi tương đối nhanh và có khả năng xử lý được cả các dataset lớn.

Case sử dụng: Phù hợp khi cần triển khai các mô hình deep learning trên quy mô lớn và khi cần sự linh hoạt trong việc tùy chỉnh mô hình.

```
import numpy as np
import tensorflow as tf

# Dữ liệu mẫu
X_train = np.array([1, 2, 3, 4, 5])
y_train = np.array([2, 4, 6, 8, 10])


# Định nghĩa mô hình Linear Regression
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

# Compile mô hình
model.compile(optimizer='sgd', loss='mean_squared_error')

# Huấn luyện mô hình
model.fit(X_train, y_train, epochs=1000)

# Dự đoán
X_test = np.array([6, 7, 8, 9, 10])
predictions = model.predict(X_test)
print(predictions)
```

Kết quả:


 en ò chứa mã ở bên dưới
 Ctrl+H

```

Epoch 1/1000
1/1 [=====] - 0s 440ms/step - loss: 78.2816
Epoch 2/1000
1/1 [=====] - 0s 9ms/step - loss: 45.6538
Epoch 3/1000
1/1 [=====] - 0s 9ms/step - loss: 26.6398
Epoch 4/1000
1/1 [=====] - 0s 8ms/step - loss: 15.5590
Epoch 5/1000
1/1 [=====] - 0s 9ms/step - loss: 9.1015
Epoch 6/1000
1/1 [=====] - 0s 10ms/step - loss: 5.3382
Epoch 7/1000
1/1 [=====] - 0s 8ms/step - loss: 3.1448
Epoch 8/1000
1/1 [=====] - 0s 8ms/step - loss: 1.8664
Epoch 9/1000
1/1 [=====] - 0s 8ms/step - loss: 1.1212
Epoch 10/1000
1/1 [=====] - 0s 9ms/step - loss: 0.6867
Epoch 11/1000
1/1 [=====] - 0s 11ms/step - loss: 0.4333
Epoch 12/1000
1/1 [=====] - 0s 7ms/step - loss: 0.2854
Epoch 13/1000
1/1 [=====] - 0s 7ms/step - loss: 0.1990
Epoch 14/1000
1/1 [=====] - 0s 11ms/step - loss: 0.1484
Epoch 15/1000
1/1 [=====] - 0s 6ms/step - loss: 0.1187
  
```

```

1/1 [-----] - 0s 5ms/step - loss: 1.0505e-04
Epoch 991/1000
1/1 [=====] - 0s 6ms/step - loss: 1.0492e-04
Epoch 992/1000
1/1 [=====] - 0s 5ms/step - loss: 1.0421e-04
Epoch 993/1000
1/1 [=====] - 0s 5ms/step - loss: 1.0351e-04
Epoch 994/1000
1/1 [=====] - 0s 5ms/step - loss: 1.0281e-04
Epoch 995/1000
1/1 [=====] - 0s 6ms/step - loss: 1.0211e-04
Epoch 996/1000
1/1 [=====] - 0s 5ms/step - loss: 1.0142e-04
Epoch 997/1000
1/1 [=====] - 0s 8ms/step - loss: 1.0074e-04
Epoch 998/1000
1/1 [=====] - 0s 7ms/step - loss: 1.0007e-04
Epoch 999/1000
1/1 [=====] - 0s 10ms/step - loss: 9.9389e-05
Epoch 1000/1000
1/1 [=====] - 0s 5ms/step - loss: 9.8715e-05
1/1 [=====] - 0s 77ms/step
[[11.984638 ]
 [13.9782095]
 [15.971781 ]
 [17.965351 ]
 [19.958923 ]]

```

Keras:

Tổng quan: Keras là một khung học sâu cấp cao, giúp tóm tắt nhiều chi tiết, làm cho mã trở nên đơn giản và ngắn gọn hơn so với TensorFlow hoặc PyTorch.

Tính năng đáng chú ý: API nhất quán và đơn giản, giảm thiểu số lượng hành động của người dùng cần thiết cho các trường hợp sử dụng phổ biến.

Tốc độ và kích thước dữ liệu: Tốc độ thực thi chậm hơn so với PyTorch và phù hợp cho các tác vụ với dataset nhỏ.

```
[ ] import numpy as np
    from keras.models import Sequential
    from keras.layers import Dense

    # Dữ liệu mẫu
    X_train = np.array([1, 2, 3, 4, 5])
    y_train = np.array([2, 4, 6, 8, 10])

    # Định nghĩa mô hình Linear Regression
    model = Sequential([
        Dense(units=1, input_shape=[1])
    ])

    # Compile mô hình
    model.compile(optimizer='sgd', loss='mean_squared_error')

    # Huấn luyện mô hình
    model.fit(X_train, y_train, epochs=1000)

    # Dự đoán
    X_test = np.array([6, 7, 8, 9, 10])
    predictions = model.predict(X_test)
    print(predictions)
```

Kết quả:

Epoch 1/1000
1/1 [=====] - 1s 643ms/step - loss: 33.2852
Epoch 2/1000
1/1 [=====] - 0s 14ms/step - loss: 19.4119
Epoch 3/1000
1/1 [=====] - 0s 11ms/step - loss: 11.3272
Epoch 4/1000
1/1 [=====] - 0s 12ms/step - loss: 6.6157
Epoch 5/1000
1/1 [=====] - 0s 17ms/step - loss: 3.8699
Epoch 6/1000
1/1 [=====] - 0s 25ms/step - loss: 2.2698
Epoch 7/1000
1/1 [=====] - 0s 23ms/step - loss: 1.3372
Epoch 8/1000
1/1 [=====] - 0s 26ms/step - loss: 0.7936
Epoch 9/1000
1/1 [=====] - 0s 29ms/step - loss: 0.4767
Epoch 10/1000
1/1 [=====] - 0s 22ms/step - loss: 0.2920
Epoch 11/1000
1/1 [=====] - 0s 15ms/step - loss: 0.1842
Epoch 12/1000
1/1 [=====] - 0s 24ms/step - loss: 0.1213
Epoch 13/1000
1/1 [=====] - 0s 9ms/step - loss: 0.0846
Epoch 14/1000
1/1 [=====] - 0s 17ms/step - loss: 0.0631
Epoch 15/1000
1/1 [=====] - 0s 10ms/step - loss: 0.0505


```

1/1 [=====] - 0s 9ms/step - loss: 4.4612e-05
Epoch 992/1000
1/1 [=====] - 0s 9ms/step - loss: 4.4310e-05
Epoch 993/1000
1/1 [=====] - 0s 8ms/step - loss: 4.4011e-05
Epoch 994/1000
1/1 [=====] - 0s 8ms/step - loss: 4.3714e-05
Epoch 995/1000
1/1 [=====] - 0s 12ms/step - loss: 4.3419e-05
Epoch 996/1000
1/1 [=====] - 0s 8ms/step - loss: 4.3126e-05
Epoch 997/1000
1/1 [=====] - 0s 7ms/step - loss: 4.2835e-05
Epoch 998/1000
1/1 [=====] - 0s 7ms/step - loss: 4.2547e-05
Epoch 999/1000
1/1 [=====] - 0s 7ms/step - loss: 4.2259e-05
Epoch 1000/1000
1/1 [=====] - 0s 8ms/step - loss: 4.1974e-05
1/1 [=====] - 0s 102ms/step
[[11.989983]
 [13.985791]
 [15.981599]
 [17.977407]
 [19.973215]]

```

PyTorch:

Tổng quan: PyTorch là một thư viện hỗ trợ nhiều phương tiện liên quan cho Deep Learning, với tính linh hoạt và khả năng tính toán hiệu quả.

Tính năng đáng chú ý: Có autograd để tự động tính toán độ dốc của các hàm và các quy trình tối ưu hóa dựa trên gradient để tối ưu hóa mạng thần kinh.

Tốc độ và kích thước dữ liệu: PyTorch có tốc độ thực thi cao hơn, phù hợp cho hiệu suất cao và có thể vận hành trên dataset lớn.

```

import torch
import torch.nn as nn
import numpy as np

# Dữ liệu mẫu
X_train = torch.tensor([[1], [2], [3], [4], [5]], dtype=torch.float32)
y_train = torch.tensor([[2], [4], [6], [8], [10]], dtype=torch.float32)

# Định nghĩa mô hình Linear Regression
class LinearRegression(nn.Module):
    def __init__(self):
        super(LinearRegression, self).__init__()
        self.linear = nn.Linear(1, 1)

    def forward(self, x):
        return self.linear(x)

model = LinearRegression()

# Định nghĩa hàm loss và optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# Huấn luyện mô hình
num_epochs = 1000
for epoch in range(num_epochs):
    # Forward pass
    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    # Backward và optimize
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 100 == 0:
        print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch+1, num_epochs, loss.item()))

# Dự đoán
X_test = torch.tensor([[6], [7], [8], [9], [10]], dtype=torch.float32)
predictions = model(X_test)
print(predictions.detach().numpy())

```

Kết quả:

```
Epoch [100/1000], Loss: 0.1623
Epoch [200/1000], Loss: 0.0825
Epoch [300/1000], Loss: 0.0419
Epoch [400/1000], Loss: 0.0213
Epoch [500/1000], Loss: 0.0108
Epoch [600/1000], Loss: 0.0055
Epoch [700/1000], Loss: 0.0028
Epoch [800/1000], Loss: 0.0014
Epoch [900/1000], Loss: 0.0007
Epoch [1000/1000], Loss: 0.0004
[[11.97044 ]
 [13.95807 ]
 [15.9457  ]
 [17.93333 ]
 [19.920961]]
```

```
import torch
import torch.nn as nn
import time

# Define the model
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.linear = nn.Linear(1, 1)

    def forward(self, x):
        return self.linear(x)

# Initialize model and other necessary components
model = Model()
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# Sample data
X_train = torch.tensor([[1], [2], [3], [4], [5]], dtype=torch.float32)
y_train = torch.tensor([[2], [4], [6], [8], [10]], dtype=torch.float32)

# Training loop
num_epochs = 1000
start_time = time.time()
for epoch in range(num_epochs):
    # Forward pass
    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    # Backward and optimize
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Calculate and print time per step
    time_per_step = (time.time() - start_time) / (epoch + 1)
    print('Epoch [{}/{}], Loss: {:.4f}, Time per step: {:.4f}'.format(epoch+1, num_epochs, loss.item(), time_per_step))

# Print final time
end_time = time.time()
print('Total training time:', end_time - start_time)
```

Kết quả:

```
Epoch [983/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [984/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [985/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [986/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [987/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [988/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [989/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [990/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [991/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [992/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [993/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [994/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [995/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [996/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [997/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [998/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [999/1000], Loss: 0.0005, Time per step: 0.0011
Epoch [1000/1000], Loss: 0.0004, Time per step: 0.0011
Total training time: 1.089134693145752
```

```

import torch
import torch.nn as nn
import numpy as np
import time

# Dữ liệu mẫu
X_train = torch.tensor([[1], [2], [3], [4], [5]], dtype=torch.float32)
y_train = torch.tensor([[2], [4], [6], [8], [10]], dtype=torch.float32)

# Định nghĩa mô hình Linear Regression
class LinearRegression(nn.Module):
    def __init__(self):
        super(LinearRegression, self).__init__()
        self.linear = nn.Linear(1, 1)

    def forward(self, x):
        return self.linear(x)

model = LinearRegression()

# Định nghĩa hàm loss và optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# Huấn luyện mô hình
num_epochs = 1000
start_time = time.time()
for epoch in range(num_epochs):
    # Forward pass
    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    # Backward và optimize
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 100 == 0:
        end_time = time.time()
        time_per_epoch = end_time - start_time
        start_time = time.time()
        print('Epoch [{}/{}], Loss: {:.4f}, Time: {:.4f} seconds'.format(epoch+1, num_epochs, loss.item(), time_per_epoch))

# Dự đoán
X_test = torch.tensor([[6], [7], [8], [9], [10]], dtype=torch.float32)
predictions = model(X_test)
print(predictions.detach().numpy())

```

Kết quả:

```
Epoch [100/1000], Loss: 0.0510, Time: 0.0480 seconds
Epoch [200/1000], Loss: 0.0259, Time: 0.0484 seconds
Epoch [300/1000], Loss: 0.0132, Time: 0.0516 seconds
Epoch [400/1000], Loss: 0.0067, Time: 0.0442 seconds
Epoch [500/1000], Loss: 0.0034, Time: 0.0443 seconds
Epoch [600/1000], Loss: 0.0017, Time: 0.0494 seconds
Epoch [700/1000], Loss: 0.0009, Time: 0.0462 seconds
Epoch [800/1000], Loss: 0.0004, Time: 0.0517 seconds
Epoch [900/1000], Loss: 0.0002, Time: 0.0422 seconds
Epoch [1000/1000], Loss: 0.0001, Time: 0.0423 seconds
[[11.983424]
 [13.976488]
 [15.969552]
 [17.962616]
 [19.955679]]
```

Cảm nghĩ sau khi sử dụng qua 4 thư viện trên như sau:

Keras thích hợp cho các tác vụ Deep Learning cơ bản và có tính linh hoạt cao trong việc tạo và huấn luyện mô hình.

PyTorch được ưa chuộng với cộng đồng nghiên cứu và phát triển mô hình Deep Learning do tính linh hoạt và hiệu suất cao.

Scikit-learn vẫn là lựa chọn hàng đầu khi cần áp dụng các thuật toán machine learning truyền thống trên các dataset nhỏ.

TensorFlow là một lựa chọn mạnh mẽ cho các dự án deep learning đòi hỏi tính linh hoạt cao và hiệu suất tốt.

So với Keras, TensorFlow thường được ưa chuộng hơn trong các dự án phức tạp và quy mô lớn hơn.

TensorFlow cung cấp các công cụ và tài nguyên hỗ trợ mạnh mẽ cho việc triển khai mô hình trên nhiều nền tảng khác nhau.

Câu 4: Phát hiện spam với SVMs và Linear regression

1. Phát hiện spam với SVMs

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Đọc file csv
df = pd.read_csv('/content/Dataset_ML_Newbie/Lab_1/Datasets/sms_spam_svm.csv')
y = df.iloc[:, 0].values
y = np.where(y == 'spam', -1, 1)
X = df.iloc[:, [1, 2]].values
# Chia tập dataset với tỉ lệ 7:3 (train:test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)

# Sử dụng SVMs để phát hiện spam
svm = SVC(kernel='linear', C=1.0, random_state=0)
# Huấn luyện mô hình
svm.fit(X_train, y_train)
# Đánh giá mô hình
y_pred = svm.predict(X_test)
print('Misclassified samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

Misclassified samples: 7
Accuracy: 0.84

2. Phát hiện spam với Linear Regression


```
[ ] import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
# Đọc dataset
df = pd.read_csv('/content/Dataset_ML_Newbie/Lab_1/Datasets/sms_spam_svm.csv')
y = df.iloc[:, 0].values
# Nếu giá trị của y[index] là spam thì mang giá trị 1 ngược lại là 0
y = np.where(y == 'spam', 1, 0)
# Ma trận X sẽ lấy giá trị của tất cả các hàng với cột 1 và cột 2 của dataset
X = df.iloc[:, [1, 2]].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)

# Sử dụng model với LinearRegression
model = LinearRegression().fit(X_train, y_train)

# Lưu giá trị mà model đã dự đoán
y_pred = model.predict(X_test)

# Đánh giá model với MSE và MAE
# MSE
print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
# MAE
print(f'Mean Absolute Error: {mean_absolute_error(y_test, y_pred)}')

Mean Square Error: 0.14445008988999872
Mean Absolute Error: 0.29867295165055574
```

Câu 5: Chức năng của phương thức `genfromtxt()` trong thư viện `numpy`.

```
[ ] import pandas as pd
import numpy as np
from sklearn import *
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

phishing_dataset = np.genfromtxt('/content/Dataset_ML_Newbie/Lab_1/Datasets/phishing_dataset.csv',
                                delimiter=',',
                                dtype=np.int32)

samples = phishing_dataset[:, :-1]
targets = phishing_dataset[:, -1]

from sklearn.model_selection import train_test_split

training_samples_lr, testing_samples_lr, training_targets_lr, testing_targets_lr = train_test_split(
    samples, targets, test_size=0.2, random_state=0)

log_classifier = LogisticRegression()
log_classifier.fit(training_samples_lr, training_targets_lr)
predictions_lr = log_classifier.predict(testing_samples_lr)

accuracy_lr = 100.0 * accuracy_score(testing_targets_lr, predictions_lr)
print('Logistic Regression accuracy:' + str(accuracy_lr))

Logistic Regression accuracy:91.67797376752601
```

Hàm `np.genfromtxt` sẽ đọc file text và csv và làm 1 trong 2 chức năng dưới đây

- Sẽ trả về 1 masked array, chúng sẽ che đi những giá trị bị thiếu hay không hợp lệ (nếu như `usemask = True`)
- Hoặc điền vào giá trị còn thiếu hay không hợp lệ được chỉ định trong `fill_values` (mặc định là `np.nan` cho float, `-1` cho int)

Tóm lại hàm `genfromtxt` sẽ giúp dữ liệu đầu vào không bị nhiễu bởi các giá trị không mong muốn

Câu 6: Hoàn thiện code Decision trees trên và đánh giá kết quả nhận được so với phương pháp Logistic regression

```
[ ] import pandas as pd
import numpy as np
from sklearn import *
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

phishing_dataset = np.genfromtxt('/content/Dataset_ML_Newbie/Lab_1/Datasets/phishing_dataset.csv',
                                delimiter=',',
                                dtype=np.int32)

samples = phishing_dataset[:, :-1]
targets = phishing_dataset[:, -1]

training_samples_tree, testing_samples_tree, training_targets_tree, testing_targets_tree = train_test_split(
    samples, targets, test_size=0.2, random_state=0)
# Chọn thuật toán DecisionTreeClassifier()
tree_classifier = tree.DecisionTreeClassifier()
# Huấn luyện mô hình
tree_classifier.fit(training_samples_tree, training_targets_tree)

predictions_tree = tree_classifier.predict(testing_samples_tree)

# Đánh giá mô hình
print("---Decision Tree---")
print(classification_report(testing_targets_tree, predictions_tree))
print("---Logistic Regression---")
print(classification_report(testing_targets_lr, predictions_lr))
```

Kết quả:

```

---Decision Tree---
              precision    recall  f1-score   support

     -1       0.97       0.95       0.96       1014
     1       0.95       0.97       0.96       1197

   accuracy                0.96       2211
  macro avg       0.96       0.96       0.96       2211
 weighted avg       0.96       0.96       0.96       2211

---Logistic Regression---
              precision    recall  f1-score   support

     -1       0.92       0.89       0.91       1014
     1       0.91       0.94       0.92       1197

   accuracy                0.92       2211
  macro avg       0.92       0.91       0.92       2211
 weighted avg       0.92       0.92       0.92       2211

```

Từ hai điều trên ta có thể thấy rằng phương pháp Decision Tree Classifier cho độ chính xác cao hơn và hiệu quả hơn so với Logistic Regression.

Câu 7: Sinh viên thực hiện code phát hiện phishing website bằng mô hình học máy Logistic regression và Decision trees với train và test trên tập dữ liệu

```
[ ] import pandas as pd
    from sklearn import datasets
    iris = datasets.load_iris()
    iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
    iris_df.head()
    iris_df.describe()
    phisLegitimate_data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Phishing_Legitimate_full.csv')
    phisLegitimate_data.info()
```

Kết quả là:

```

▶ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   id                                         10000 non-null  int64
1   NumDots                                   10000 non-null  int64
2   SubdomainLevel                           10000 non-null  int64
3   PathLevel                                10000 non-null  int64
4   UrlLength                                 10000 non-null  int64
5   NumDash                                   10000 non-null  int64
6   NumDashInHostname                        10000 non-null  int64
7   AtSymbol                                  10000 non-null  int64
8   TildeSymbol                              10000 non-null  int64
9   NumUnderscore                             10000 non-null  int64
10  NumPercent                               10000 non-null  int64
11  NumQueryComponents                       10000 non-null  int64
12  NumAmpersand                             10000 non-null  int64
13  NumHash                                   10000 non-null  int64
14  NumNumericChars                          10000 non-null  int64
15  NoHttps                                   10000 non-null  int64
16  RandomString                             10000 non-null  int64
17  IPAddress                                10000 non-null  int64
18  DomainInSubdomains                       10000 non-null  int64
19  DomainInPaths                            10000 non-null  int64
20  HttpsInHostname                          10000 non-null  int64
21  HostnameLength                           10000 non-null  int64
22  PathLength                               10000 non-null  int64
23  QueryLength                              10000 non-null  int64
24  DoubleSlashInPath                       10000 non-null  int64
25  NumSensitiveWords                        10000 non-null  int64
26  EmbeddedBrandName                       10000 non-null  int64
27  PctExtHyperlinks                        10000 non-null  float64
28  PctExtResourceUrls                      10000 non-null  float64
29  ExtFavicon                              10000 non-null  int64
30  InsecureForms                           10000 non-null  int64
31  RelativeFormAction                      10000 non-null  int64
32  ExtFormAction                           10000 non-null  int64
33  AbnormalFormAction                      10000 non-null  int64
34  PctNullSelfRedirectHyperlinks           10000 non-null  float64
35  FrequentDomainNameMismatch              10000 non-null  int64
36  FakeLinkInStatusBar                     10000 non-null  int64
37  RightClickDisabled                       10000 non-null  int64
38  PopUpWindow                             10000 non-null  int64
39  SubmitInfoToEmail                       10000 non-null  int64
40  IframeOrFrame                           10000 non-null  int64
41  MissingTitle                             10000 non-null  int64

```

```

42 ImagesOnlyInForm          10000 non-null int64
43 SubdomainLevelRT          10000 non-null int64
44 UrlLengthRT                10000 non-null int64
45 PctExtResourceUrlsRT       10000 non-null int64
46 AbnormalExtFormActionR     10000 non-null int64
47 ExtMetaScriptLinkRT        10000 non-null int64
48 PctExtNullSelfRedirectHyperlinksRT 10000 non-null int64
49 CLASS_LABEL                10000 non-null int64
dtypes: float64(3), int64(47)
memory usage: 3.8 MB

```

A) LOGISTIC REGRESSION

import class LogisticRegression từ module linear_model trong thư viện sklearn (scikit-learn)

```
[ ] from sklearn.linear_model import LogisticRegression
```

Chia dữ liệu thành hai phần, một là các mẫu (samples_phisLegit) chứa các đặc trưng (features) và hai là các mục tiêu (targets_phisLegit) chứa biến mục tiêu (target variable). iloc được sử dụng để truy cập phần tử dựa trên chỉ mục của chúng trong DataFrame.

```
[ ] from sklearn.linear_model import LogisticRegression
    samples_phisLegit = phisLegitimate_data.iloc[:, :-2]
    targets_phisLegit = phisLegitimate_data.iloc[:, -1]
```

Chia dữ liệu thành hai tập dữ liệu riêng biệt: một tập dữ liệu dùng để huấn luyện mô hình (train set) và một tập dữ liệu dùng để kiểm tra hiệu suất của mô hình (test set)

```
from sklearn.model_selection import train_test_split
train_samples_phisLegit, test_samples_phisLegit, train_targets_phisLegit, test_targets_phisLegit = train_test_split(samples_phisLegit, targets_phisLegit, test_size=0.3, random_state=0)
```

Thực hiện huấn luyện mô hình logistic regression trên tập dữ liệu huấn luyện

```

▶ phishing_model_log = LogisticRegression(max_iter=120)
  phishing_model_log.fit(train_samples_phisLegit, train_targets_phisLegit)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
    LogisticRegression
    LogisticRegression(max_iter=120)

```

Chúng ta sử dụng mô hình đã huấn luyện để dự đoán các nhãn cho tập dữ liệu kiểm tra (test_samples_phisLegit), và lưu kết quả vào biến predict_phisLegit_log.

```
[ ] predict_phisLegit_log = phishing_model_log.predict(test_samples_phisLegit)
```

Tính toán độ chính xác của mô hình dựa trên so sánh giữa nhãn dự đoán (predict_phisLegit_log) và nhãn thực tế (test_targets_phisLegit)

```

[ ] from sklearn.metrics import accuracy_score
    accuracy = 100.0 * accuracy_score(test_targets_phisLegit, predict_phisLegit_log)
    print("ĐỘ CHÍNH XÁC <Logistic Regression>: %s" % str(accuracy))

ĐỘ CHÍNH XÁC <Logistic Regression>: 97.46666666666667

```

B) DECISION TREES

Decision Tree là một thuật toán học máy được sử dụng cho cả bài toán phân loại và hồi quy.

phising_model_tree = tree.DecisionTreeClassifier(): Tạo ra một đối tượng mô hình Decision Tree Classifier.

phising_model_tree.fit(train_samples_phisLegit, train_targets_phisLegit): Huấn luyện mô hình Decision Tree trên tập dữ liệu huấn luyện (train_samples_phisLegit, train_targets_phisLegit). Trong quá trình huấn luyện, mô hình sẽ học cách phân loại các mẫu dựa trên các đặc trưng và nhãn tương ứng của chúng.

```
from sklearn import tree
phishing_model_tree = tree.DecisionTreeClassifier()
phishing_model_tree.fit(train_samples_phisLegit, train_targets_phisLegit)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Sử dụng mô hình Decision Tree đã huấn luyện để dự đoán nhãn cho tập dữ liệu kiểm tra (test_samples_phisLegit). Kết quả dự đoán được lưu vào biến predict_phisLegit_tree.

```
predict_phisLegit_tree = phishing_model_tree.predict(test_samples_phisLegit)
```

Tính toán độ chính xác của mô hình Decision Tree dựa trên so sánh giữa nhãn dự đoán (predict_phisLegit_tree) và nhãn thực tế (test_targets_phisLegit).

```
accuracy = 100.0 * accuracy_score(test_targets_phisLegit, predict_phisLegit_tree)
print("ĐỘ CHÍNH XÁC <Decision Tree>: %s" % str(accuracy))
```

```
ĐỘ CHÍNH XÁC <Decision Tree>: 100.0
```

Câu 8: Sinh viên thực hiện code phát hiện phishing website bằng mô hình học máy Logistic regression hoặc Decision trees với train và test trên tập dữ liệu


```

import pandas as pd

df = pd.read_csv('verified_online.csv')

df.info()

```

<class 'pandas.core.frame.DataFrame'>
 RangeIndex: 54826 entries, 0 to 54825
 Data columns (total 8 columns):
 # Column Non-Null Count Dtype --- --- ---
 0 phish_id 54826 non-null int64
 1 url 54826 non-null object
 2 phish_detail_url 54826 non-null object
 3 submission_time 54826 non-null object
 4 verified 54826 non-null object
 5 verification_time 54826 non-null object
 6 online 54826 non-null object
 7 target 54826 non-null object
 dtypes: int64(1), object(7)
 memory usage: 3.3+ MB

df

	phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	target
0	8513997	https://www.mizuhubsnk.top	http://www.phishtank.com/phish_detail.php?phis...	2024-04-01T05:00:21+00:00	yes	2024-04-01T05:03:52+00:00	yes	Other
1	8513987	https://cliont.clicketcloud.com/limaclik.php?h...	http://www.phishtank.com/phish_detail.php?phis...	2024-04-01T03:31:05+00:00	yes	2024-04-01T03:52:59+00:00	yes	Banco De Brasil
2	8513975	http://dmgkryhpuj.duckdns.org	http://www.phishtank.com/phish_detail.php?phis...	2024-04-01T03:19:01+00:00	yes	2024-04-01T03:22:53+00:00	yes	Apple
3	8513974	http://djhikmapvc.duckdns.org	http://www.phishtank.com/phish_detail.php?phis...	2024-04-01T03:18:23+00:00	yes	2024-04-01T03:22:53+00:00	yes	Apple
4	8513973	https://misteros.pages.dev/check/dapp/	http://www.phishtank.com/phish_detail.php?phis...	2024-04-01T03:14:49+00:00	yes	2024-04-01T03:22:53+00:00	yes	Other
...
54821	3573703	http://www.ezblox.site/free/jennifer111/helpdesk	http://www.phishtank.com/phish_detail.php?phis...	2015-11-03T14:41:38+00:00	yes	2016-03-15T18:51:08+00:00	yes	Other
54822	2042606	http://www.formbuddy.com/cgi-bin/formdisp.pl?u...	http://www.phishtank.com/phish_detail.php?phis...	2013-09-30T13:24:39+00:00	yes	2013-10-01T13:33:17+00:00	yes	Other
54823	1865971	http://www.formbuddy.com/cgi-bin/formdisp.pl?u...	http://www.phishtank.com/phish_detail.php?phis...	2013-05-28T15:59:31+00:00	yes	2013-05-28T22:13:12+00:00	yes	Other
54824	1460953	http://www.habbocreditosparati.blogspot.com/	http://www.phishtank.com/phish_detail.php?phis...	2012-06-14T15:19:26+00:00	yes	2012-06-14T15:29:11+00:00	yes	Sulake Corporation
54825	1123978	http://creditiperhabbogratisicuro100.blogspot...	http://www.phishtank.com/phish_detail.php?phis...	2011-02-18T10:50:32+00:00	yes	2011-02-19T00:05:43+00:00	yes	Sulake Corporation

54826 rows x 8 columns

```
[ ] from sklearn.preprocessing import LabelEncoder

Label = LabelEncoder().fit_transform(Label)

print("===LABEL===")
print(Label)

===LABEL==
[0 1 1 ... 0 1 1]

[ ] #2016's top most suspicious TLD and words
Suspicious_TLD=['zip','cricket','link','work','party','gq','kim','country','science','tk']

#trend micro's top malicious domains
Suspicious_Domain=['luckytime.co.kr','mattfoll.eu.interia.pl','trafficholder.com','dl.baixaki.com.br','bembred.redtube.com','tags.expo9.exponential.com','deepspacer.com','funad.co.kr','trafficonverter.biz']
```

```
▶ # Method to count number of dots
def countdots(url):
    return url.count('.')

# Method to count number of delimiters
def countdelim(url):
    count = 0
    delim=[';', '_', '?', '=', '&']
    for each in url:
        if each in delim:
            count = count + 1

    return count

# Method to check if an URL is presented as IP
import ipaddress as ip

def isip(uri):
    try:
        if ip.ip_address(uri):
            return 1
    except:
        return 0

# Method to check the presence of hyphens
def isPresentHyphen(url):
    return url.count('-')

# Method to check the presence of @
def isPresentAt(url):
    return url.count('@')

# Method to check double slash (Using to redirect to another website)
def isPresentDSlash(url):
    return url.count('//')

# Method to count the sub directory
def countSubDir(url):
    return url.count('/')

# Method to check the exxtension from URL
from os.path import splitext
```

```
def get_ext(url):
    root, ext = splitext(url)
    return ext

# Method to count sub-domain
def countSubDomain(subdomain):
    if not subdomain:
        return 0
    else:
        return len(subdomain.split('.'))

# Method to count queries
def countQueries(query):
    if not query:
        return 0
    else:
        return len(query.split('&'))
```

```
▶ from urllib.parse import urlparse
import tldextract
def getFeatures(url):
    result = []
    url = str(url)

    # Parse the URL and extract the domain information
    path = urlparse(url)
    ext = tldextract.extract(url)

    # Counting number of dots in subdomain
    result.append(countdots(ext.subdomain))

    # Checking hyphen in domain
    result.append(isPresentHyphen(path.netloc))

    # Length of URL
    result.append(len(url))

    # Checking @ in the url
    result.append(isPresentAt(path.netloc))

    # Checking presence of double slash
    result.append(isPresentDSlash(path.path))

    # Count number of subdir
    result.append(countSubDir(path.path))

    # Number of sub domain
    result.append(countSubDomain(ext.subdomain))

    # Length of domain name
    result.append(len(path.netloc))

    # Count number of queries
    result.append(len(path.query))

    # Adding domain information

    # If IP address is being used as a URL
    result.append(isip(ext.domain))
```

```

    # Presence of Suspicious_TLD
    result.append(1 if ext.suffix in Suspicious_TLD else 0)

    # Presence of suspicious domain
    result.append(1 if path.netloc[5:] in Suspicious_Domain else 0 )

    return result

```

```

▶ featureExtraction = pd.DataFrame(columns=('no of dots','presence of hyphen','len of url',
                                           'presence of at','presence of double slash','no of subdir',
                                           'no of subdomain','len of domain','no of queries','is IP',
                                           'presence of Suspicious_TLD','presence of suspicious domain'))

for url in URL:
    features = getFeatures(url)
    featureExtraction.loc[len(featureExtraction)] = features

```

```
[ ] featureExtraction.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Index: 54826 entries, 0 to 54825
Data columns (total 12 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   no of dots                            54826 non-null  int64
 1   presence of hyphen                    54826 non-null  int64
 2   len of url                            54826 non-null  int64
 3   presence of at                        54826 non-null  int64
 4   presence of double slash              54826 non-null  int64
 5   no of subdir                          54826 non-null  int64
 6   no of subdomain                       54826 non-null  int64
 7   len of domain                         54826 non-null  int64
 8   no of queries                         54826 non-null  int64
 9   is IP                                54826 non-null  int64
10   presence of Suspicious_TLD            54826 non-null  int64
11   presence of suspicious domain         54826 non-null  int64
dtypes: int64(12)
memory usage: 5.4 MB

```

featureExtraction												
	no of dots	presence of hyphen	len of url	presence of at	presence of double slash	no of subdir	no of subdomain	len of domain	no of queries	is IP	presence of Suspicious_TLD	presence of suspicious domain
0	0	0	26	0	0	0	1	18	0	0	0	0
1	0	0	68	0	0	1	1	23	23	0	0	0
2	0	0	29	0	0	0	1	22	0	0	0	0
3	0	0	29	0	0	0	1	22	0	0	0	0
4	0	0	39	0	0	3	1	19	0	0	0	0
...
54821	0	0	48	0	0	3	1	15	0	0	0	0
54822	0	0	62	0	0	2	1	17	17	0	0	0
54823	0	0	76	0	0	2	1	17	31	0	0	0
54824	1	0	44	0	0	1	2	36	0	0	0	0
54825	0	0	95	0	0	3	1	43	0	0	0	0
54826 rows x 12 columns												

```

from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

X_train, X_test, y_train, y_test = train_test_split(featureExtraction.values, Label, test_size=0.3, random_state=19)

model_lr = LogisticRegression()
model_dt = DecisionTreeClassifier()

model_lr.fit(X_train, y_train)
y_pred = model_lr.predict(X_test)
print("---Logistic Regression---")
print(classification_report(y_test, y_pred))
print()

model_dt.fit(X_train, y_train)
y_pred = model_dt.predict(X_test)
print("---Decision Tree---")
print(classification_report(y_test, y_pred))

```

```

---Logistic Regression---
              precision    recall  f1-score   support

     0       0.91      1.00      0.96      15045
     1       0.00      0.00      0.00       1403

 accuracy          0.91      16448
 macro avg          0.46      0.50      0.48      16448
 weighted avg          0.84      0.91      0.87      16448

```

```

---Decision Tree---
              precision    recall  f1-score   support

     0       0.93      0.98      0.95      15045
     1       0.50      0.25      0.33       1403

 accuracy          0.91      16448
 macro avg          0.72      0.61      0.64      16448
 weighted avg          0.90      0.91      0.90      16448

```

/home/vanlinh/miniconda3/envs/CyberAI/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

Link giải thích bài tập 8