

BÁO CÁO THỰC HÀNH

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

Tên chủ đề: Set up your Machine Learning for Cybersecurity Arsenal

GVHD: Nguyễn Hữu Quyền

Nhóm: NT09

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.021.ANTT.2

STT	Họ và tên	MSSV	Email
1	Nguyễn Triệu Thiên Bảo	21520155	21520155@gm.uit.edu.vn
2	Trần Lê Minh Ngọc	21521195	21521195@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Nội dung	Tình trạng
1	Yêu cầu 1	100%
2	Yêu cầu 2	100%
3	Yêu cầu 3	100%
4	Yêu cầu 4	100%
5	Yêu cầu 5	100%
6	Yêu cầu 6	100%
7	Yêu cầu 7	100%
8	Yêu cầu 8	100%

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

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

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

- Đoạn code thực thi phép cộng trừ giữa hai ma trận

$$a = [-8 \quad 15] \text{ và } X = \begin{bmatrix} 1 & 5 \\ 3 & 4 \\ 2 & 3 \end{bmatrix}$$

```
import numpy as np
a = np.array([-8, 15])
X = np.array([[1, 5],
              [3, 4],
              [2, 3]])
y = np.add(X, a)
print("X + a = ")
print(y)

y = np.subtract(X, a)
print("X - a = ")
print(y)
```

Kết quả thực thi

```
X + a =
[[-7 20]
 [-5 19]
 [-6 18]]
X - a =
[[ 9 -10]
 [11 -11]
 [10 -12]]
```

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

- Tạo csv có tên là BT2.csv

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

- Đọc CSV thành Dataframe và hiển thị

```
print('Đọc CSV thành Dataframe và hiển thị')
df = pd.read_csv('/content/BT2.csv')
print(df, '\n')
```

Kết quả thực hiện

```
Đọc CSV thành Dataframe và hiển thị
Unnamed: 0    X    Y    Z
0            1   78   84   86
1            2   85   94   97
2            3   80   83   73
3            4   96   94   96
4            5   86   86   83
```

- Hãy chuyển index mặc định thành giá trị cột id

```
print('Chuyển index mặc định thành giá trị cột id')
df.set_index('Unnamed: 0', inplace = True)
print(df, '\n')
```

Kết quả thực hiện

```
Chuyển index mặc định thành giá trị cột id
      X    Y    Z
Unnamed: 0
1      78   84   86
2      85   94   97
3      80   83   73
4      96   94   96
5      86   86   83
```

- Sắp xếp dữ liệu theo nhiều cột (sort)

```
print('Sắp xếp dữ liệu theo nhiều cột (sort)')
df_sorted = df.sort_values(by = ['X', 'Y', 'Z'])
print(df_sorted, '\n')
```

Kết quả thực hiện

```
Sắp xếp dữ liệu theo nhiều cột (sort)
      X    Y    Z
Unnamed: 0
1      78   84   86
3      80   83   73
2      85   94   97
5      86   86   83
4      96   94   96
```

- Chọn một cột cụ thể và hiển thị nó

```
print('Chọn một cột cụ thể và hiển thị nó')
print(df['X'], '\n')
print(df_sorted, '\n')
```

Kết quả thực hiện

```
Chọn một cột cụ thể và hiển thị nó
Unnamed: 0
1      78
2      85
3      80
4      96
5      86
Name: X, dtype: int64
```

- Chọn 2 hàng đầu tiên và hiển thị chúng

```
print('Chọn 2 hàng đầu tiên và hiển thị chúng')
print(df.head(2), '\n')
```

Kết quả thực hiện

```
Chọn 2 hàng đầu tiên và hiển thị chúng
      X    Y    Z
Unnamed: 0
1      78   84   86
2      85   94   97
```

- Hãy chọn một hàng dựa trên một điều kiện giá trị của cột

```
print('Hãy chọn một hàng dựa trên một điều kiện giá trị của cột')
print('Chọn hàng mà X có giá trị bằng 85')
df_selectrow = df.loc[df['X'] == 85]
print(df_selectrow, '\n')
```

Kết quả thực hiện

```
Hãy chọn một hàng dựa trên một điều kiện giá trị của cột
Chọn hàng mà X có giá trị bằng 85
      X    Y    Z
Unnamed: 0
2      85   94   97
```

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

Tạo bảng BT2.1.csv bị khuyết vài giá trị

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

```
print('Thay đổi một vài giá trị thành NaN ở CSV, sau đó đọc lên thành Dataframe và thay thế chúng bằng giá trị 0')
```

```
other_df = pd.read_csv('/content/BT2.1.csv')
print('Dataframe trước khi thay đổi')
print(other_df, '\n')
print('Dataframe sau khi thay đổi')
other_df.fillna(0, inplace = True)
print(other_df, '\n')
```

Kết quả thực hiện

```
Thay đổi một vài giá trị thành NaN ở CSV, sau đó đọc lên thành Dataframe và thay thế chúng bằng giá trị 0
Dataframe trước khi thay đổi
      Unnamed: 0    X    Y    Z
0              1   78   84.0  86.0
1              2   85   NaN  97.0
2              3   80   83.0  73.0
3              4   96   94.0   NaN
4              5   86   86.0  83.0
```

```
Dataframe sau khi thay đổi
      Unnamed: 0    X    Y    Z
0              1   78   84.0  86.0
1              2   85    0.0  97.0
2              3   80   83.0  73.0
3              4   96   94.0    0.0
4              5   86   86.0  83.0
```

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

```
print('Ở cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn là False trong Dataframe')
def greater_than_90(x):
    if x > 90:
        return True
    else:
```

```
return False
df['Z'] = df['Z'].apply(greater_than_90)
print(df, '\n')
```

Kết quả thực hiện (dựa trên BT2.csv)

```
Ở cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn là False trong Dataframe
      X  Y  Z
Unnamed: 0
1      78  84  False
2      85  94   True
3      80  83  False
4      96  94   True
5      86  86  False
```

- 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

```
print('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')
print('Dataframe d1')
d1 = df.drop(columns = ['Z'])
print(d1)
print('Dataframe d2')
d2 = df[['Z']]
print(d2)
print('Dataframe d3')
d3 = pd.concat([d1, d2], axis = 1)
print(d3, '\n')
```

Kết quả thực hiện

```
Chuyển Dataframe trên thành 2 Dataframe d1 và d2; d1 chứa cột X và Y, d2 chứa cột Z;
Dataframe d1
      X  Y
Unnamed: 0
1      78  84
2      85  94
3      80  83
4      96  94
5      86  86
Dataframe d2
      Z
Unnamed: 0
1  False
2   True
3  False
4   True
5  False
Dataframe d3
      X  Y  Z
Unnamed: 0
1      78  84  False
2      85  94   True
3      80  83  False
4      96  94   True
5      86  86  False
```

- Dùng tính năng thống kê hãy hiển thị kết quả thống kê các giá trị thuộc tính của Dataframe

```
print('Dùng tính năng thống kê hãy hiển thị kết quả thống kê các giá trị thuộc tính của Dataframe')
```

```
print(other_df.describe())
```

Kết quả thực hiện

Dùng tính năng thống kê hãy hiển thị kết quả thông kê các giá trị thuộc tính của Dataframe	Unnamed: 0	X	Y	Z
count	5.000000	5.0	5.000000	5.000000
mean	3.000000	85.0	69.400000	67.800000
std	1.581139	7.0	39.035881	38.854858
min	1.000000	78.0	0.000000	0.000000
25%	2.000000	80.0	83.000000	73.000000
50%	3.000000	85.0	84.000000	83.000000
75%	4.000000	86.0	86.000000	86.000000
max	5.000000	96.0	94.000000	97.000000

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 sau:

- TensorFlow
 - Keras
 - PyTorch
- Cho biết cảm nghĩ về việc dùng 4 thư viện này
- Sử dụng thư viện Scikit-learning

```
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] )
```

Kết quả thực hiện

Model Prediction for new data: \$13.73

- Sử dụng thư viện Tensorflow

```
#Câu 3: Tensorflow
import tensorflow as tf

class SimpleLinearRegression:
    def __init__(self, initializer='random'):
        if initializer=='ones':
            self.var = 1.
        elif initializer=='zeros':
            self.var = 0.
        elif initializer=='random':
            self.var = tf.random.uniform(shape=[], minval=0., maxval=1.)

        self.m = tf.Variable(1., shape=tf.TensorShape(None))
        self.b = tf.Variable(self.var)
```

```
def predict(self, x):
    return tf.reduce_sum(self.m * x, 1) + self.b

def mse(self, true, predicted):
    return tf.reduce_mean(tf.square(true-predicted))

def update(self, X, y, learning_rate):
    with tf.GradientTape(persistent=True) as g:
        loss = self.mse(y, self.predict(X))

    print("Loss: ", loss)
    dy_dm = g.gradient(loss, self.m)
    dy_db = g.gradient(loss, self.b)

    self.m.assign_sub(learning_rate * dy_dm)
    self.b.assign_sub(learning_rate * dy_db)

def train(self, X, y, learning_rate=0.01, epochs=5):
    if len(X.shape)==1:
        X=tf.reshape(X,[X.shape[0],1])

    self.m.assign([self.var]*X.shape[-1])
    for i in range(epochs):
        print("Epoch: ", i)

        self.update(X, y, learning_rate)

x = np.array([[3], [5], [7], [9], [11]]).reshape(-1, 1)
y = np.array([8.0, 9.1, 10.3, 11.4, 12.6])

mean_label = y.mean(axis=0)
std_label = y.std(axis=0)
mean_feat = x.mean(axis=0)
std_feat = x.std(axis=0)
x = (x-mean_feat)/std_feat
y = (y-mean_label)/std_label

linear_model = SimpleLinearRegression('zeros')
linear_model.train(x, y, learning_rate = 0.01, epochs = 500)

x_test = [[13]]

# standardize
x_test = (x_test-mean_feat)/std_feat
```

```
# reverse standardization
pred = linear_model.predict(x_test)
pred *= std_label
pred += mean_label

print('Model Prediction for new data: $%.2f %pred)
```

Kết quả thực hiện

```
Loss: tf.Tensor(0.00022679355, shape=(), dtype=float32)
Epoch: 497
Loss: tf.Tensor(0.00022679262, shape=(), dtype=float32)
Epoch: 498
Loss: tf.Tensor(0.00022679342, shape=(), dtype=float32)
Epoch: 499
Loss: tf.Tensor(0.00022679335, shape=(), dtype=float32)
Model Prediction for new data: $13.73
```

- Sử dụng thư viện Keras

```
# Câu 3: Keras
import tensorflow as tf
import numpy as np

# Tạo dữ liệu mẫu
X_train = np.array([[3], [5], [7], [9], [11]], dtype=np.float32)
y_train = np.array([8.0, 9.1, 10.3, 11.4, 12.6], dtype=np.float32)

# Tạo model
model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])

# Compile model với loss function và optimizer
model.compile(loss='mean_squared_error',
optimizer=tf.keras.optimizers.Adam(learning_rate=0.1))

# Huấn luyện model
model.fit(X_train, y_train, epochs=500, verbose=0)

# Dự đoán giá trị cho dữ liệu mới
new_data = np.array([[13]], dtype=np.float32)
prediction = model.predict(new_data)[0][0]
print('Model Prediction for new data: $%.2f % prediction)
```

Kết quả thực hiện

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.>
1/1 [=====] - 0s 42ms/step
Model Prediction for new data: $13.73
```

- Sử dụng thư viện PyTorch

```
#Câu 3: Pytorch
import torch
from torch.autograd import Variable

x_data = Variable(torch.Tensor([[3], [5], [7], [9], [11]]))
```



```
y_data = Variable(torch.Tensor([[8.0], [9.1], [10.3], [11.4], [12.6]]))

class LinearRegressionModel(torch.nn.Module):

    def __init__(self):
        super(LinearRegressionModel, self).__init__()
        self.linear = torch.nn.Linear(1, 1) # One in and one out

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

# our model
our_model = LinearRegressionModel()

criterion = torch.nn.MSELoss(size_average = False)
optimizer = torch.optim.Adam(our_model.parameters(), lr = 0.01)

for epoch in range(5000):

    # Forward pass: Compute predicted y by passing
    # x to the model
    pred_y = our_model(x_data)

    # Compute and print loss
    loss = criterion(pred_y, y_data)

    # Zero gradients, perform a backward pass,
    # and update the weights.
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print('epoch {}, loss {}'.format(epoch, loss.item()))

new_var = Variable(torch.Tensor([[13.0]]))
pred_y = our_model(new_var)
print('Model Prediction for new data: $%.2f %our_model(new_var).item())
```

Kết quả thực hiện

```
epoch 4997, loss 0.0030016775708645582
epoch 4998, loss 0.0030015481170266867
epoch 4999, loss 0.0030015548691153526
Model Prediction for new data: $13.73
```

Cảm nghĩ khi dùng cả bốn thư viện này:

- *Scikit-learning*: Là một thư viện cơ bản, thích hợp cho người mới học ML hoặc dùng cho các bài toán đơn giản.

- *Tensorflow*: Xử lý chậm hơn các thư viện còn lại, có ít các API hỗ trợ các thuật toán ML/DL phổ biến. Bù lại, TensorFlow có 1 cộng đồng người dùng đông đảo nên người dùng được hỗ trợ thường xuyên và thư viện được cập nhật liên tục.
- *Keras*: Keras tóm tắt nhiều chi tiết, làm cho mã trở nên đơn giản và ngắn gọn hơn so với trong PyTorch hoặc TensorFlow.
- *PyTorch*: hỗ trợ nhiều phương tiện liên quan cho Deep Learning. Có nhiều thuật toán tối ưu, đặc biệt là tối ưu Neural Network. Có tốc độ thực thi cao hơn, phù hợp cho hiệu suất cao.

4. Sinh viên hoàn thành code phát hiện spam với SVMs và Linear regression

- Sử dụng mô hình SVMs

Bước 1: Load dataset từ drive và tiền xử lý dataset. Với X là các thuộc tính, y là nhãn tương ứng.

#Câu 4: SVM

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
import pandas as pd
import numpy as np
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
df = pd.read_csv('/content/drive/MyDrive/PPHM/sms_spam_svm.csv')
y = df.iloc[:, 0].values
y = np.where(y == 'spam', -1, 1)
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)
```

Bước 2: Tạo và huấn luyện mô hình SVM dựa trên tập dataset X_train và y_train.

```
svm = SVC(kernel='linear', C=1.0, random_state=0)
svm.fit(X_train, y_train)
```

Bước 3: Sau khi huấn luyện mô hình, chúng ta sẽ test thử trên tập X_test và tính accuracy của mô hình. Ta thu được kết quả như sau.

```
y_pred = svm.predict(X_test)
```

```
print('Misclassified samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

Kết quả thực hiện

```
Mounted at /content/drive
Misclassified samples: 7
Accuracy: 0.84
```

- Sử dụng mô hình Linear Regression

#Câu 4: Linear Regression

```
import pandas as pd
```

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score

from google.colab import drive

drive.mount('/content/drive')

df = pd.read_csv('/content/drive/MyDrive/PPHM/sms_spam_svm.csv')
y = df.iloc[:, 0].values
y = np.where(y == 'spam', -1, 1)
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)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('Misclassified samples: %d' % (y_test != y_pred).sum())
```

Kết quả thực hiện

```
Mounted at /content/drive
Misclassified samples: 45
```

5. Sinh viên cho biết chức năng của phương thức `genfromtxt()` trong thư viện `numpy`.

```
# Câu 5
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/phishing_dataset.csv', delimiter=',',
dtype=np.int32)
print(phishing_dataset)
samples = phishing_dataset[:, :-1]
targets = phishing_dataset[:, -1]
from sklearn.model_selection import train_test_split
training_samples, testing_samples, training_targets, testing_targets = train_test_split(
samples, targets, test_size=0.2, random_state=0)
log_classifier = LogisticRegression()
log_classifier.fit(training_samples, training_targets)
predictions = log_classifier.predict(testing_samples)
accuracy = 100.0 * accuracy_score(testing_targets, predictions)
print ("Logistic Regression accuracy: " + str(accuracy))
```

Kết quả thực hiện

```
[[-1  1  1 ...  1 -1 -1]
 [ 1  1  1 ...  1  1 -1]
 [ 1  0  1 ...  0 -1 -1]
 ...
 [ 1 -1  1 ...  0  1 -1]
 [-1 -1  1 ...  1  1 -1]
 [-1 -1  1 ...  1 -1 -1]]
Logistic Regression accuracy: 91.67797376752601
```

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

- Phương thức `numpy.genfromtxt()` được sử dụng để đọc dữ liệu từ tệp văn bản và tạo ra một mảng `numpy` từ các dữ liệu đó, cho phép bạn thực hiện các phép toán và xử lý dữ liệu tiếp theo.
- `numpy.genfromtxt()` có khả năng xử lý các trường hợp dữ liệu thiếu trong tệp bằng cách thêm giá trị mặc định hoặc bỏ qua chúng tùy thuộc vào cách cài đặt.
- `numpy.genfromtxt()` có cấu hình linh hoạt: người dùng có thể cấu hình phương thức này để đọc dữ liệu theo cách mà họ muốn, bằng cách chuyển đổi số như `delimiter` (dấu phân cách), `dtype` (kiểu dữ liệu), `skip_header` (bỏ qua dòng đầu tiên), và nhiều thiết lập khác.
- Mảng `numpy` có kiểu dữ liệu là `np.int32` tạo ra được từ file “`phishing_dataset.csv`”:

```
+ Code + Text

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/phishing_dataset.csv', delimiter=',', dtype=np.int32)
print(phishing_dataset)

[[-1  1  1 ...  1 -1 -1]
 [ 1  1  1 ...  1  1 -1]
 [ 1  0  1 ...  0 -1 -1]
 ...
 [ 1 -1  1 ...  0  1 -1]
 [-1 -1  1 ...  1  1 -1]
 [-1 -1  1 ...  1 -1 -1]]
```

6. Sinh viên hoàn thiện code Decision trees trên và đánh giá kết quả nhận được so với phương pháp Logistic regression.

- Bước 1: Tiền xử lý dataset như mô hình Logistic Regression

```
#Câu 6
import pandas as pd
import numpy as np
from sklearn import *
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

phishing_dataset = np.genfromtxt('/content/phishing_dataset.csv', delimiter=',',
dtype=np.int32)
```

```
samples = phishing_dataset[:, :-1]
targets = phishing_dataset[:, -1]
print('Phishing dataset')
print(phishing_dataset, '\n')
```

```
training_samples, testing_samples, training_targets, testing_targets =
train_test_split(samples, targets, test_size=0.2, random_state=0)
```

- Bước 2: Tạo và huấn luyện mô hình Decision Tree dựa trên tập dataset training_samples và training_targets

```
# Tạo và huấn luyện mô hình Decision Tree
tree_classifier = tree.DecisionTreeClassifier()
tree_classifier.fit(training_samples, training_targets)
```

- Bước 3: Sau khi huấn luyện mô hình, chúng ta sẽ test thử trên tập dataset testing

```
# Dự đoán trên tập kiểm tra
tree_predictions = tree_classifier.predict(testing_samples)
```

```
# Đánh giá kết quả sử dụng độ chính xác
tree_accuracy = 100.0 * accuracy_score(testing_targets, tree_predictions)
print("Decision Tree accuracy:", tree_accuracy)
```

Kết quả thực hiện

```
➡ Phishing dataset
[[-1  1  1 ...  1 -1 -1]
 [ 1  1  1 ...  1  1 -1]
 [ 1  0  1 ...  0 -1 -1]
 ...
 [ 1 -1  1 ...  0  1 -1]
 [-1 -1  1 ...  1  1 -1]
 [-1 -1  1 ...  1 -1 -1]]

Decision Tree accuracy: 96.47218453188603
```

=> Ta có thể thấy kết quả thu được của mô hình Decision Tree (96.47) tốt hơn kết quả của mô hình Logistic Regression (91.68).

=> Lí do: mô hình Decision Tree có khả năng học được các mối quan hệ phức tạp và không tuyến tính giữa các đặc trưng và nhãn. Trong khi đó, mô hình Logistic Regression tạo ra các mối quan hệ giữa các đặc trưng và nhãn là tuyến tính. Với tập dataset được cho, kết quả trả về chỉ có hai nhãn là -1 và 1 (có thể tương ứng với “đúng” và “sai”), không có tính tuyến tính, vì vậy, mô hình Decision Tree phù hợp hơn.

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

<https://www.kaggle.com/shashwatwork/phishing-dataset-for-machine-learning>

- Sử dụng mô hình Logistic Regression

Bước 1: Tiền xử lý dataset như các mô hình trước

```
#Câu 7: Logistic Regression
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier

#Load dataset từ drive
df = pd.read_csv("/content/drive/MyDrive/PPHM/Phishing_Legitimate_full.csv")

#Tiền xử lí dataset
samples = df.iloc[:, :-1]
targets = df.iloc[:, -1]

#Tách thành 2 phần: training và testing với tỉ lệ 7:3
X_train, X_test, y_train, y_test = train_test_split(samples, targets, test_size=0.2,
random_state=42)
```

Bước 2: Tạo và huấn luyện mô hình Logistic Regression dựa trên tập dataset samples và targets

```
# Tạo và huấn luyện mô hình Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Bước 3: Sau khi huấn luyện mô hình, chúng ta sẽ test thử trên tập dataset testing và tính accuracy của mô hình. Ta thu được kết quả như sau

```
#Kiểm tra mô hình trên tập X_test và tính accuracy
lr_preds = lr.predict(X_test)

accuracy_lr = 100.0 * accuracy_score(y_test, lr_preds)
print('Logistic Regression accuracy:' + str(accuracy_lr))
```

Kết quả thực hiện

```
Logistic Regression accuracy:97.39999999999999
```

- Sử dụng mô hình Decision Tree

Bước 1: Lập lại giống bước 1 của mô hình Logistic Regression nhưng là với mô hình Decision Tree

```
#Câu 7: Decision Tree
```

```
# Tạo và huấn luyện mô hình Logistic Regression
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Bước 5: Lập lại giống bước 2 của mô hình Logistic Regression nhưng là với mô hình Decision Tree. Ta thu được kết quả như sau

```
#Kiểm tra mô hình trên tập X_test và tính accuracy
dt_preds = dt.predict(X_test)

accuracy_dt = 100.0 * accuracy_score(y_test, dt_preds)
print('Decision Tree accuracy:' + str(accuracy_dt))
```

Kết quả thực hiện

Decision Tree accuracy:100.0

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 phishtank. Tham khảo cách xử lý và trích xuất thuộc tính <https://github.com/surajr/URL-Classification>

Bước 1: Trích xuất dữ liệu từ tập dataset phishtank, lấy các url và thêm thuộc tính “isPhishing” mặc định bằng 1 cho các url này.

```
# Câu 8
import pandas as pd
# Trích xuất dữ liệu
URLphishing = pd.read_csv('/content/verified_online.csv')
URLphishing.sample().reset_index(drop=True)
URLphishing.head()
URLphishing['target'].value_counts()
URLphishing_extract = URLphishing[['url']]
URLphishing_extract['isPhishing'] = 1
URLphishing_extract.head()
```

	url	isPhishing
0	https://zb912.app.link/e/z1GDWPu15Hb	1
1	https://att-login810.weeblysite.com/	1
2	http://att-login810.weeblysite.com/	1
3	https://www.declaratia-de-valoare.ro/chdashb/r...	1
4	https://dev-arprovincia-home.pantheonsite.io/	1

Bước 2: Trích xuất dữ liệu từ tập dataset các url an toàn và cho thuộc tính “isPhishing” bằng 0.

```
URLbenign = pd.read_csv('/content/1.Benign_list_big_final.csv')
URLbenign.columns = ['url']
URLbenign['isPhishing'] = 0
URLbenign.head()
```

	url	isPhishing
0	http://1337x.to/torrent/1110018/Blackhat-2015-...	0
1	http://1337x.to/torrent/1122940/Blackhat-2015-...	0
2	http://1337x.to/torrent/1124395/Fast-and-Furio...	0
3	http://1337x.to/torrent/1145504/Avengers-Age-o...	0
4	http://1337x.to/torrent/1160078/Avengers-age-o...	0

Bước 3: Lấy ngẫu nhiên khoảng 1000 url phishing và benign là tập dataset huấn luyện cho mô hình.

```
URLdataset_phishing = URLphishing_extract.sample(n = 1000)
URLdataset_benign = URLbenign.sample(n = 1000)
URLdataset = pd.concat([URLdataset_phishing, URLdataset_benign])
```


Bước 4: Liệt kê các danh sách từ ngữ, tên miền và TLD (Top Level Domain) bị nghi ngờ là độc hại.

```
Suspicious_TLD=['zip','cricket','link','work','party','gq','kim','country','science','tk']
Suspicious_Domain=['luckytime.co.kr','mattfoll.eu.interia.pl','trafficholder.com','dl.baixaki.com.br','bembed.redtube.comr','tags.expo9.exponential.com','deepspacer.com','funad.co.kr','trafficconverter.biz']
```

Bước 5: Thực hiện các hàm tính toán.

```
!pip install tldextract
```

```
def countdots(url):
    return url.count('.')
```

```
def countdelim(url):
    count = 0
    delim=[';','_','?','=' '&']
    for each in url:
        if each in delim:
            count = count + 1

    return count
```

```
def countdelim(url):
    count = 0
    delim=[';','_','?','=' '&']
    for each in url:
        if each in delim:
            count = count + 1

    return count
```

```
import ipaddress as ip
```

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

```
def isPresentHyphen(url):
    return url.count('-')
```



```
def isPresentAt(url):
    return url.count('@')

def isPresentDSlash(url):
    return url.count('//')

def countSubDir(url):
    return url.count('/')

def get_ext(url):
    """Return the filename extension from url, or ""."""
    root, ext = splitext(url)
    return ext

def countSubDomain(subdomain):
    if not subdomain:
        return 0
    else:
        return len(subdomain.split('.'))

def countQueries(query):
    if not query:
        return 0
    else:
        return len(query.split('&'))

featureSet = pd.DataFrame(columns=['url','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','label'], dtype=np.int32)
featureSet

def getFeatures(url, label):
    result = []
    url = str(url)

    result.append(url)
    path = urlparse(url)
    ext = tldextract.extract(url)
```

```
result.append(countdots(ext.subdomain))
result.append(isPresentHyphen(path.netloc))
result.append(len(url))
result.append(isPresentAt(path.netloc))
result.append(isPresentDSlash(path.path))
result.append(countSubDir(path.path))
result.append(countSubDomain(ext.subdomain))
result.append(len(path.netloc))
result.append(len(path.query))
result.append(isip(ext.domain))
result.append(1 if ext.suffix in Suspicious_TLD else 0)
result.append(1 if '.'.join(ext.domain) in Suspicious_Domain else 0)
result.append(label)
return result
```

```
URLdataset["isPhising"][0:1].values[0]
```

```
URLdataset["url"][0:1]
```

```
features = getFeatures(URLdataset["url"][0:1], URLdataset["isPhising"][0:1].values)
len(features)
```

```
for i in range(len(URLdataset)):
    features = getFeatures(URLdataset["url"][i:i+1],
URLdataset["isPhising"][i:i+1].values[0])
    featureSet.loc[i] = features
featureSet
```

Kết quả thực hiện

Decision Tree applied on PhisTank Dataset has the accuracy score: 79.33333333333333