

BÁO CÁO THỰC HÀNH

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

Lab 1: Set up your machine learning for Cybersecurity Arsenal

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT334.O21.ATCL.2

Nhóm: 7

STT	Họ và tên	MSSV	Email
1	Hồ Ngọc Thiện	21522620	21522620@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Bài tập 1	100%
2	Bài tập 2	100%
3	Bài tập 3	100%
4	Bài tập 4	100%
5	Bài tập 5	100%
6	Bài tập 6	100%
7	Bài tập 7	100%
8	Bài tập 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

® Bài tập (yêu cầu làm)

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

✓
0s



```
import numpy as np
```

✓
0s

```
[2] a = np.array([-8, 15])  
    b = np.array([9, -14])
```

✓
0s

```
[4] sum = a + b  
    print(sum)
```

```
[1 1]
```

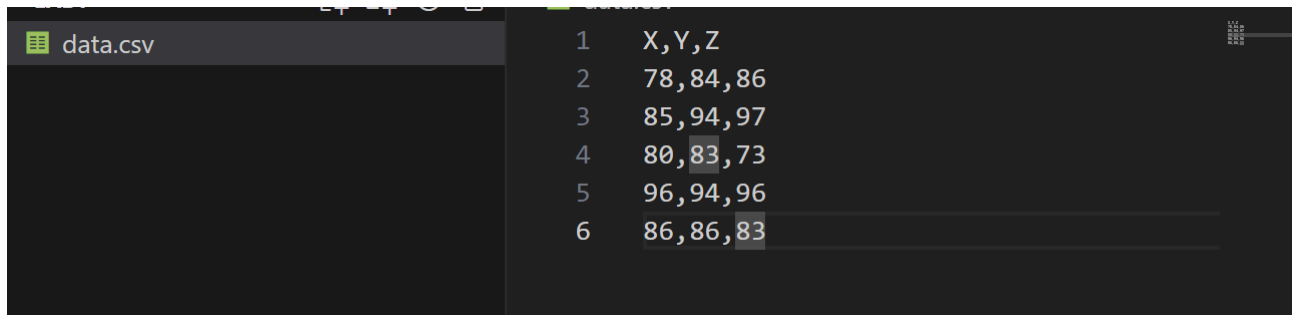
✓
0s

```
[5] subtraction = a - b  
    print(subtraction)
```

```
[-17 29]
```

Sinh viên tự tạo tập tin CSV có cấu trúc như sau:

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



```
1 X,Y,Z
2 78,84,86
3 85,94,97
4 80,83,73
5 96,94,96
6 86,86,83
```

Sau đó ta upload lên google colab



..



sample_data



data.csv

® Bài tập (yêu cầu làm)

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

- Đọc CSV thành Dataframe và hiển thị
- Hãy chuyển index mặc định thành giá trị cột id
- Sắp xếp dữ liệu theo nhiều cột (sort)
- Chọn một cột cụ thể và hiển thị nó
- Chọn 2 hàng đầu tiên và hiển thị chúng
- Hãy chọn một hàng dựa trên một điều kiện giá trị của cột
- Thay đổi một vài giá trị thành **NaN** ở CSV, sau đó đọc lên thành Dataframe và thay thế chúng bằng giá trị 0
- Ở cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn là False trong Dataframe
- 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
- Dùng tính năng thống kê hãy hiển thị kết quả thống kê các giá trị thuộc tính của Dataframe

- Đọc CSV thành Dataframe và hiển thị

✓
0s



```
input_file = 'data.csv'  
df = pd.read_csv(input_file)  
print(df)
```



	X	Y	Z
0	78	84	86
1	85	94	97
2	80	83	73
3	96	94	96
4	86	86	83

- Hãy chuyển index mặc định thành giá trị cột id

✓
0s

```
input_file = 'data.csv'  
df = pd.read_csv(input_file)  
#Đặt lại index cho dữ liệu bắt đầu từ 1  
df.reset_index(drop=True, inplace=True)  
df.index += 1  
print(df)
```



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

Ta sử dụng hàm `reset_index` để đặt lại index. Tiếp theo ta tăng index của DataFrame lên 1 bằng lệnh `df.index += 1`

- Sắp xếp dữ liệu theo nhiều cột (sort)

+ Xếp theo cột X

✓
0s

```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Sắp xếp dữ liệu theo nhiều cột
df_sorted = df.sort_values(by=['X'])

# Hiển thị DataFrame đã sắp xếp
print(df_sorted)
```

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

+ Xếp theo cột Y

✓
0s

```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Sắp xếp dữ liệu theo nhiều cột
df_sorted = df.sort_values(by=['Y'])

# Hiển thị DataFrame đã sắp xếp
print(df_sorted)
```



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

+ Xếp theo cột Z

✓
0s

```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Sắp xếp dữ liệu theo nhiều cột
df_sorted = df.sort_values(by=['Z'])

# Hiển thị DataFrame đã sắp xếp
print(df_sorted)
```

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

- Chọn một cột cụ thể và hiển thị nó

✓
0s

```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Chọn một cột cụ thể và hiển thị nó
selected_column = df['Y']
print(selected_column)
```

```
1      84
2      94
3      83
4      94
5      86
Name: Y, dtype: int64
```

- Chọn 2 hàng đầu tiên và hiển thị chúng

✓
0s

```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Chọn một cột cụ thể và hiển thị nó
selected_rows = df.head(2)
print(selected_rows)
```

	X	Y	Z
1	78	84	86
2	85	94	97

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

Chọn hàng sao cho cột Z chứa giá trị 73

✓
0s



```
import pandas as pd

# Đường dẫn và tên file CSV
input_file = 'data.csv'

# Đọc file CSV vào DataFrame
df = pd.read_csv(input_file)
df.reset_index(drop=True, inplace=True)
df.index += 1

# Chọn một cột cụ thể và hiển thị nó
selected_rows = df[df['Z'] == 73]
print(selected_rows)
```

	X	Y	Z
3	80	83	73

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

data.csv	
1	X,Y,Z
2	78,84,NaN
3	NaN,94,97
4	80,83,73
5	96,NaN,96
6	NaN,86,83

Ta sử dụng hàm `fillna(0, inplace=True)` để thực hiện thay thế các giá trị Nan thành 0

```
input_file = 'data.csv'
df = pd.read_csv(input_file)
df.fillna(0, inplace=True)
print(df)
```

	X	Y	Z
0	78.0	84.0	0.0
1	0.0	94.0	97.0
2	80.0	83.0	73.0
3	96.0	0.0	96.0
4	0.0	86.0	83.0

- Ở cột Z chuyển giá trị lớn hơn 90 là True và nhỏ hơn là False trong Dataframe
Ta sử dụng hàm apply để thay thế True False dựa theo điều kiện đã cho

```
df['Z'] = df['Z'].apply(lambda x: True if pd.notnull(x) and x > 90 else False)  
  
print(df)
```

	X	Y	Z
0	78.0	84.0	False
1	0.0	94.0	True
2	80.0	83.0	False
3	96.0	0.0	True
4	0.0	86.0	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

Ta sử dụng hàm loc để tiến hành lọc các cột tương ứng với d1 và d2. Sau đó ta sử dụng hàm concat để nối d1 và d2

```
import pandas as pd  
  
# Đọc dữ liệu từ tệp CSV vào DataFrame  
df = pd.read_csv('data.csv')  
  
# Tạo DataFrame d1 chứa cột 'X' và 'Y'  
d1 = df.loc[:, ['X', 'Y']]  
  
# Tạo DataFrame d2 chứa cột 'Z'  
d2 = df.loc[:, ['Z']]  
  
# Kết hợp d1 và d2 để tạo DataFrame d3  
d3 = pd.concat([d1, d2], axis=1)  
  
print("d1:")  
print(d1)  
print("d2:")  
print(d2)  
print("d3:")  
print(d3)
```

Kết quả:



d1:

	X	Y
0	78.0	84.0
1	NaN	94.0
2	80.0	83.0
3	96.0	NaN
4	NaN	86.0

d2:

	Z
0	NaN
1	97.0
2	73.0
3	96.0
4	83.0

d3:

	X	Y	Z
0	78.0	84.0	NaN
1	NaN	94.0	97.0
2	80.0	83.0	73.0
3	96.0	NaN	96.0
4	NaN	86.0	83.0

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

Ta sử dụng hàm describe() để hiển thị các kết quả thống kê



```
import pandas as pd
```

```
# Đọc dữ liệu từ tệp CSV vào DataFrame
df = pd.read_csv('data.csv')
```

```
# Hiển thị kết quả thống kê mô tả cho các cột số học trong DataFrame
statistics = df.describe()
print(statistics)
```



	X	Y	Z
count	3.000000	4.000000	4.000000
mean	84.666667	86.750000	87.250000
std	9.865766	4.991666	11.441882
min	78.000000	83.000000	73.000000
25%	79.000000	83.750000	80.500000
50%	80.000000	85.000000	89.500000
75%	88.000000	88.000000	96.250000
max	96.000000	94.000000	97.000000

Bài tập về nhà (yêu cầu làm)

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

Sample Data

```
✓ [1] import numpy as np
0s      X = np.array([2, 4, 6, 8, 10], dtype=np.float32)
      y = np.array([8.0, 9.2, 11.3, 12.4, 13.6], dtype=np.float32)
      test = np.array([13], dtype=np.float32)
```

- TensorFlow, Keras

```
✓ [2] from tensorflow.keras import layers
7s      import tensorflow as tf
```

```
✓ [6] # Chuẩn hóa dữ liệu
0s      X_normalizer = layers.Normalization(input_shape=[1,], axis=None)
      X_normalizer.adapt(X)
```

```
✓ # Build a model
0s      X_model = tf.keras.Sequential([
      X_normalizer,
      layers.Dense(units=1)
      ])
      X_model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
normalization_1 (Normaliza tion)	(None, 1)	3
dense_1 (Dense)	(None, 1)	2
=====		
Total params: 5 (24.00 Byte)		
Trainable params: 2 (8.00 Byte)		
Non-trainable params: 3 (16.00 Byte)		

✓ [8] X_model.predict(test)

0s

```
1/1 [=====] - 0s 146ms/step
array([[ -1.0243225]], dtype=float32)
```

- Pytorch

▶ pip install torch

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.2.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.10.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.2.1)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
    23.7/23.7 MB 29.8 MB/s eta 0:00:00
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
    823.6/823.6 kB 32.4 MB/s eta 0:00:00
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
    14.1/14.1 MB 44.4 MB/s eta 0:00:00
```

✓ ▶ from torch.autograd import Variable
import torch

5s

```
x_data = Variable(torch.Tensor([[2.0], [4.0], [6.0], [6.0], [8.0]]))
y_data = Variable(torch.Tensor([[3.0], [5.0], [7.0], [9.0], [11.0]]))

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 = LinearRegressionModel()

criterion = torch.nn.MSELoss(reduction='sum')
```



```
▶ for epoch in range(100):  
  
    # 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()))
```

```
⇒ epoch 0, loss 696.2560424804688  
epoch 1, loss 696.2560424804688  
epoch 2, loss 696.2560424804688  
epoch 3, loss 696.2560424804688  
epoch 4, loss 696.2560424804688  
epoch 5, loss 696.2560424804688  
epoch 6, loss 696.2560424804688  
epoch 7, loss 696.2560424804688  
epoch 8, loss 696.2560424804688  
epoch 9, loss 696.2560424804688  
epoch 10, loss 696.2560424804688  
epoch 11, loss 696.2560424804688  
epoch 12, loss 696.2560424804688  
epoch 13, loss 696.2560424804688  
epoch 14, loss 696.2560424804688  
epoch 15, loss 696.2560424804688  
epoch 16, loss 696.2560424804688  
epoch 17, loss 696.2560424804688  
epoch 18, loss 696.2560424804688  
epoch 19, loss 696.2560424804688  
epoch 20, loss 696.2560424804688  
epoch 21, loss 696.2560424804688  
epoch 22, loss 696.2560424804688  
epoch 23, loss 696.2560424804688  
epoch 24, loss 696.2560424804688  
epoch 25, loss 696.2560424804688  
epoch 26, loss 696.2560424804688  
epoch 27, loss 696.2560424804688  
epoch 28, loss 696.2560424804688  
epoch 29, loss 696.2560424804688
```

```

new_var = Variable(torch.Tensor([[13.0]]))
pred_y = our_model(new_var)
print("predict (after training)", our_model(new_var).item())

```

predict (after training) -4.168074131011963

Ý kiến cá nhân về những mô hình đã sử dụng bên trên

	Scikit-learning	Keras	Pytorch
Mô tả	Một thư viện học máy nói chung, cung cấp các thuật toán cơ bản.	Keras là một khung học sâu cao cấp hơn, nó tóm tắt nhiều chi tiết, làm cho mã trở nên đơn giản và ngắn gọn hơn so với trong PyTorch hoặc TensorFlow	Một thư viện hỗ trợ nhiều phương tiện cho Deep Learning
Tính năng	Cung cấp các thuật toán chuyên về học máy như Decision Tree, Logistic Regression, ...	Cung cấp các API nhất quán và đơn giản, giảm thiểu số lượng hành động của users cần thiết cho các trường hợp sử dụng phổ biến	- Autograd – một thuật toán có thể tự động tính toán độ dốc của các hàm của bạn, được xác định theo các hoạt động cơ bản - Các quy trình tối ưu hóa dựa trên Gradient để tối ưu hóa quy mô lớn, dành riêng cho tối ưu hóa mạng thần kinh
Tốc độ		Keras chậm hơn so với Pytorch	Pytorch có tốc độ thực thi cao hơn, phù hợp cho hiệu suất cao
Dataset size		Bởi lý do trên nên Keras phù hợp cho dataset nhỏ	Pytorch có thể vận hành tốt trên dataset có kích thước lớn
Các trường hợp sử dụng	Chủ yếu khi cần sử dụng các thuật toán Machine Learning truyền thống	- Phù hợp khi sử dụng Deep Learning, cả 2 thư viện đều update các mô hình Neuron Network hiện đại nhất khá tốt và nhanh	

		- Tùy vào từng ngữ cảnh và một số tiêu chí đánh giá để chọn lọc thư viện, không có thư viện nào hoàn toàn tốt.
--	--	--

Bài tập (yêu cầu làm)**4. Sinh viên hoàn thành code phát hiện spam với SVMs và Linear regression**

```

import pandas as pd
import numpy as np
input_file = "sms_spam_perceptron.csv"
df = pd.read_csv(input_file)
y = df.iloc[:, 0].values #lay du lieu cua cot dau tien luu vao mang y(ham or spam)
y = np.where(y == 'spam', -1, 1) #gan ham = 1, spam = -1
X = df.iloc[:, [1, 2]].values # lay du lieu cua cot 2 va 3 luu vao mang x

```

Ta thực hiện chia tập dataset thành tập huấn luyện và tập kiểm tra. Trong đó tham số X đầu tiên là mảng chứa các đặc trưng và tham số y chứa các nhãn của dataset

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0) #chia du lieu thanh cac tap huan luyen va kiem tra

```

Tiếp theo, ta khởi tạo 1 đối tượng perceptron với các tham số chỉ số lần lặp để huấn luyện mô hình, tỷ lệ học tập ban đầu

```

from sklearn.linear_model import Perceptron
p = Perceptron(max_iter=40, eta=0.1, random_state=0)
p.fit(X_train, y_train)

```

Perceptron
Perceptron(eta=0.1, max_iter=40)

Ta tiến hành dự đoán nhãn của dữ liệu kiểm tra(X_test) bằng mô hình perceptron đã được huấn luyện, sau đó lưu vào y_pred

```

y_pred = p.predict(X_test)

```

Cuối cùng, ta kiểm tra số mẫu bị phân loại sai bằng cách so sánh nhãn thực tế (y_test) và nhãn dự đoán (y_pred). Biểu thức (y_test != y_pred) sẽ tạo ra một mảng Boolean, trong đó các phần tử có giá trị True đại diện cho các mẫu bị phân loại sai., hàm sum() được sử dụng để tính tổng các giá trị True trong mảng đó, tức là số lượng mẫu bị phân loại sai. Sau đó, ta in ra độ chính xác của mô hình trên dữ liệu kiểm tra.

- Sử dụng mô hình SVM cho việc phân loại email

Ta tạo 1 đối tượng svm để tiến hành huấn luyện. Sau đó, ta truyền vào hàm fit() tập X_train và y_train để huấn luyện mô hình SVM. Tiếp theo ta dự đoán nhãn cho tập X_test và lưu vào y_pred và cuối cùng là in ra số mẫu sai và độ chính xác của mô hình cho tập dữ liệu đã cung cấp.

```
✓ 0s ▶ from sklearn.svm import SVC
svm = SVC(kernel='linear', random_state=0)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
from sklearn.metrics import accuracy_score
print('Missclassified sample: %d' % (y_test != y_pred).sum())
print('Accuracy Score: %.2f' % accuracy_score(y_test, y_pred))
```

Missclassified sample: 3
Accuracy Score: 0.90

Bài tập (yêu cầu làm)

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

Chức năng : genfromtxt() được sử dụng để load data từ file text. Hàm này có các tham số ví dụ như delimiter = ',' được sử dụng để tách các dữ liệu ngăn cách nhau bởi dấu "," hay dtype=np.int32 để chuyển dữ liệu thành số nguyên 32 bit và các tham số khác để thực hiện các chức năng khác nhau.

Bài tập (yêu cầu làm)

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

```
✓ 2s [1] import pandas as pd
import numpy as np
from sklearn import *
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
✓ 47s [2] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
✓ 1s [3] phishing_dataset = np.genfromtxt('/content/drive/MyDrive/Classroom/NT522.021.ATCL/Lab1/phishing_dataset.csv', delimiter=',',
dtype=np.int32)
```

```
✓ 0s [4] samples = phishing_dataset[:, :-1]
targets = phishing_dataset[:, -1]
```

```
✓ 0s [5] training_samples, testing_samples, training_targets, testing_targets = train_test_split(samples, targets, test_size=0.2, random_state=0)
```

```

✓ 0s [6] # Logistic Regression
      log_classifier = LogisticRegression()
      log_classifier.fit(training_samples, training_targets)
      log_predictions = log_classifier.predict(testing_samples)

✓ 0s # Decision Tree
      tree_classifier = DecisionTreeClassifier()
      tree_classifier.fit(training_samples, training_targets)
      tree_predictions = tree_classifier.predict(testing_samples)

✓ 0s log_accuracy = 100.0 * accuracy_score(testing_targets, log_predictions)
      tree_accuracy = 100.0 * accuracy_score(testing_targets, tree_predictions)
      print ("Logistic Regression accuracy: " + str(log_accuracy))
      print ("Decision Tree accuracy: " + str(tree_accuracy))

      Logistic Regression accuracy: 91.67797376752601
      Decision Tree accuracy: 96.38172772501132

```

Đánh giá : độ chính xác của mô hình decision tree cao hơn mô hình logistic regression .

EX7:

Bài tập về nhà (yêu cầu làm)

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 các thư viện như sau:

```

1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.tree import DecisionTreeClassifier
5 from sklearn.metrics import accuracy_score

```

-đọc dữ liệu từ tệp CSV và lưu trữ nó vào một DataFrame bằng cách sử dụng thư viện Pandas.

```

7 # Đọc dữ liệu từ file CSV
8 phisLegitimate_data = pd.read_csv('Phishing_Legitimate_full.csv')
9

```

-In ra một số thông tin cơ bản về tập dữ liệu, bao gồm số lượng dòng, số lượng cột và thông tin về các cột.

```

10 # Hiển thị một số dòng đầu tiên của tập dữ liệu
11 print("First few rows of the dataset:")
12 print(phisLegitimate_data.info())
13

```

-Kết quả:

```
First few rows of the dataset:
<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
```

-In ra một số thống kê mô tả về các cột trong DataFrame

```
14 print(phishing_data.describe())
15
```

-Kết quả:

```
dtypes: float64(3), int64(47)
memory usage: 3.8 MB
None
```

	id	NumDots	SubdomainLevel	...	ExtMetaScriptLinkRT	PctExtNullSelfRedirectHyperlinksRT	CLASS_LABEL
count	10000.00000	10000.00000	10000.00000	...	10000.00000	10000.00000	10000.00000
mean	5000.50000	2.445100	0.586800	...	0.173400	0.314100	0.500000
std	2886.89568	1.346836	0.751214	...	0.755771	0.897843	0.500025
min	1.00000	1.000000	0.000000	...	-1.000000	-1.000000	0.000000
25%	2500.75000	2.000000	0.000000	...	0.000000	-1.000000	0.000000
50%	5000.50000	2.000000	1.000000	...	0.000000	1.000000	0.500000
75%	7500.25000	3.000000	1.000000	...	1.000000	1.000000	1.000000
max	10000.00000	21.000000	14.000000	...	1.000000	1.000000	1.000000

[8 rows x 50 columns]

-Kiểm tra xem tập dữ liệu có chứa giá trị thiếu (NULL) không bằng cách sử dụng phương thức isnull() của DataFrame và sau đó gọi any() để kiểm tra xem có cột nào chứa giá trị thiếu hay không.

```
17 # Kiểm tra xem tập dữ liệu có chứa giá trị NULL không
18 print("\nCheck for missing values:")
19 print(phishing_data.isnull().any())
20
```

-Kết quả:


```

Check for missing values:
id                False
NumDots           False
SubdomainLevel   False
PathLevel        False
UrlLength         False
NumDash          False
NumDashInHostname False
AtSymbol         False
TildeSymbol      False
NumUnderscore    False
NumPercent       False
NumQueryComponents False
NumAmpersand     False
NumHash          False
NumNumericChars  False
NoHttps          False
RandomString     False
IpAddress        False
DomainInSubdomains False
DomainInPaths    False
HttpsInHostname  False
HostnameLength   False
PathLength       False
QueryLength      False
DoubleSlashInPath False
NumSensitiveWords False
EmbeddedBrandName False
PctExtHyperlinks False
PctExtResourceUrls False
ExtFavicon       False
InsecureForms    False
RelativeFormAction False
ExtFormAction    False
AbnormalFormAction False

```

-Chia dữ liệu thành features (mẫu) và target (nhãn). Trong đó, `samples_phisLegit` chứa tất cả các cột ngoại trừ hai cột cuối cùng, và `targets_phisLegit` chỉ chứa cột cuối cùng (nhãn).

```

21 # Phân chia dữ liệu thành features và target
22 samples_phisLegit = phisLegitimate_data.iloc[:, :-2]
23 targets_phisLegit = phisLegitimate_data.iloc[:, -1]
24

```

-Phân chia dữ liệu thành tập huấn luyện và tập kiểm tra bằng cách sử dụng hàm `train_test_split()` từ `sklearn.model_selection`. Dữ liệu được chia thành 70% cho tập huấn luyện và 30% cho tập kiểm tra.

```

25 # Chia dữ liệu thành tập train và tập test
26 train_samples_phisLegit, test_samples_phisLegit, train_targets_phisLegit,
27 test_targets_phisLegit = train_test_split(samples_phisLegit,
28 targets_phisLegit, test_size=0.3, random_state=0)
29

```

+) Huấn luyện với mô hình Logictics Regression

-Tạo một mô hình Logistic Regression và huấn luyện nó trên tập dữ liệu huấn luyện sử dụng phương thức `fit()`

-Dự đoán nhãn cho tập dữ liệu kiểm tra bằng cách sử dụng phương thức `predict()` và tính toán độ chính xác bằng cách so sánh nhãn dự đoán với nhãn thực tế, sau đó in ra độ chính xác.

```
# Huấn luyện mô hình Logistic Regression
phishing_model_log = LogisticRegression(max_iter=1000, solver='saga')
phishing_model_log.fit(train_samples_phisLegit, train_targets_phisLegit)
predict_phisLegit_log = phishing_model_log.predict(test_samples_phisLegit)
accuracy = 100.0 * accuracy_score(test_targets_phisLegit, predict_phisLegit_log)
print("\nThe accuracy score performed by Logistic Regression: %s" % str(accuracy))
```

+) Huấn luyện với mô hình Decision Trees

-Tạo một mô hình Decision Tree và huấn luyện nó trên tập dữ liệu huấn luyện.

-Dự đoán nhãn cho tập dữ liệu kiểm tra bằng cách sử dụng phương thức predict() của mô hình Decision Tree và tính toán độ chính xác tương tự như trước đó với mô hình Logistic Regression, sau đó in ra độ chính xác.

```
# Huấn luyện mô hình Decision Trees
phishing_model_tree = DecisionTreeClassifier()
phishing_model_tree.fit(train_samples_phisLegit, train_targets_phisLegit)
predict_phisLegit_tree = phishing_model_tree.predict(test_samples_phisLegit)
accuracy1 = 100.0 * accuracy_score(test_targets_phisLegit, predict_phisLegit_tree)
print("The accuracy score performed by Decision Trees: %s" % str(accuracy1))
```

+) Kết quả của 2 mô hình và so sánh

```
The accuracy score performed by Logistic Regression: 92.2
The accuracy score performed by Decision Trees: 100.0
```

-Logistic Regression:

- Độ chính xác: 92.2%
- Loại mô hình: Mô hình tuyến tính
- Điểm mạnh: Độ chính xác khá cao (92.2%).
- Điểm yếu: Một số hệ số (coefficients) có thể chưa hội tụ, điều này có thể làm giảm tính ổn định và tin cậy của mô hình.

-Decision Trees:

- Độ chính xác: 100.0%
- Loại mô hình: Mô hình cây quyết định
- Điểm mạnh: Độ chính xác đạt 100%, có vẻ như mô hình đã học dữ liệu huấn luyện hoàn hảo.
- Điểm yếu: Có thể xảy ra overfitting, nghĩa là mô hình có thể quá mức học thuộc lòng dữ liệu huấn luyện mà không thể tổng quát hóa tốt cho dữ liệu mới.

-So sánh:

- Độ chính xác: Mô hình Decision Trees có độ chính xác cao hơn so với Logistic Regression trên tập dữ liệu kiểm tra.
- Tính ổn định: Logistic Regression có cảnh báo về việc hệ số không hội tụ, trong khi Decision Trees có thể có dấu hiệu của overfitting.

- Phù hợp với dữ liệu: Logistic Regression thường phù hợp tốt hơn khi có mối quan hệ tuyến tính giữa các đặc trưng và nhãn, trong khi Decision Trees có thể phù hợp tốt hơn với các mối quan hệ phức tạp và phi tuyến tính.
- Tính diễn giải: Logistic Regression có thể dễ dàng diễn giải hơn so với Decision Trees, vì Decision Trees thường khó hiểu hơn khi cần giải thích quyết định của mô hình.

EX8:

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](https://github.com/surajir/URL-Classification). Tham khảo cách xử lý và trích xuất thuộc tính <https://github.com/surajir/URL-Classification>

-Các thư viện sử dụng:

```
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
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from urllib.parse import urlparse
import tldextract
import ipaddress as ip #works only in python 3
```

-Đọc dữ liệu từ tập tin CSV phishingwebsite_phishtank.csv và hiển thị thông tin về tập dữ liệu:

```
0 URLdataphishing = pd.read_csv('phishingwebsite_phishtank.csv')
1 print(URLdataphishing.info())
2
```

-Kết quả:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64550 entries, 0 to 64549
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   phish_id             64550 non-null  int64
 1   url                  64550 non-null  object
 2   phish_detail_url     64550 non-null  object
 3   submission_time      64550 non-null  object
 4   verified             64550 non-null  object
 5   verification_time    64550 non-null  object
 6   online              64550 non-null  object
 7   target              64550 non-null  object
dtypes: int64(1), object(7)
memory usage: 3.9+ MB
None
```

-Hiển thị vài dòng đầu tiên của tập dữ liệu:

```
5 print(URLdataphishing.head())
6
```

-Kết quả:

```

none
phish_id      url      phish_detail_url  ...  verification time  online  target
0  7788603  https://bt-109500.weeblysite.com/  http://www.phishtank.com/phish_detail.php?phis...  ...  2022-09-24T07:51:48+00:00  yes  Other
1  7788602  http://bt-109500.weeblysite.com/  http://www.phishtank.com/phish_detail.php?phis...  ...  2022-09-24T07:51:48+00:00  yes  Other
2  7788601  https://bt-105397.weeblysite.com/  http://www.phishtank.com/phish_detail.php?phis...  ...  2022-09-24T07:42:42+00:00  yes  Other
3  7788596  http://trustkonec.com  http://www.phishtank.com/phish_detail.php?phis...  ...  2022-09-24T06:51:35+00:00  yes  Other
4  7788595  http://thecollaband.com  http://www.phishtank.com/phish_detail.php?phis...  ...  2022-09-24T06:42:53+00:00  yes  Other
[5 rows x 8 columns]

```

-Lấy một mẫu ngẫu nhiên từ tập dữ liệu:

```
URLdataphising.sample().reset_index(drop=True)
```

-Đếm số lượng mục tiêu trong cột "target":

```
URLdataphising['target'].value_counts()
```

-Chọn chỉ cột 'url' từ tập dữ liệu và tạo một DataFrame mới với cột 'isPhising' được thiết lập thành 1:

```

1 # Only extract 'url' feature for future training
2 URLdataphising_extracted = URLdataphising[['url']]
3 URLdataphising_extracted['isPhising'] = 1
4 URLdataphising_extracted.head()

```

-Đọc dữ liệu từ tập tin CSV 1.Benign_list_big_final.csv, đổi tên cột 'url' và tạo một cột mới 'isPhising' với giá trị 0:

```

5 URLdatabenign = pd.read_csv('1.Benign_list_big_final.csv')
6 URLdatabenign.columns = ['url']
7 URLdatabenign['isPhising'] = 0
8 URLdatabenign.head()

```

-Lấy mẫu ngẫu nhiên 1000 dòng từ mỗi tập dữ liệu (phishing và benign):

```

# Get random first 10,000 rows as a sample for each set
URLdataset_phising = URLdataphising_extracted.sample(n = 1000)
URLdataset_benign = URLdatabenign.sample(n = 1000)

```

-Kết hợp hai tập dữ liệu để tạo một tập dữ liệu tổng hợp:

```

URLdataset = pd.concat([URLdataset_phising, URLdataset_benign])
URLdataset.head(2000)

```

Handling the dataset (Not using)

Define functions

- #2016's top most suspicious TLD and words
- 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','deepspac er.com','funad.co.kr','trafficconverter.biz']
- #trend micro's top malicious domains
-
- # 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 count number of delimiters
• def countdelim(url):
•     count = 0
•     delim=[';', '_', '?', '=', '&']
•     for each in url:
•         if each in delim:
•             count = count + 1
•     return count
•
• # Is IP addr present as th hostname, let's validate
• 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('@')
•

```

```

• 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)
•
•     #add the url to feature set
•     result.append(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 '.'.join(ext.domain) in Suspicious_Domain else 0)`
-
- `result.append(label)`
- `return result`
-
- `#Yay! finally done!`
-
- `URLdataset["isPhishing"][0:1].values[0]`
-
- `URLdataset["url"][0:1] # Just for testing, get a row in our data in "url" column`
- `features = getFeatures(URLdataset["url"][0:1],`
`URLdataset["isPhishing"][0:1].values) # Just for testing, get all the values for 14`
`features in a URL`
- `len(features)`
- `for i in range(len(URLdataset)): # Loop over our dataset, set 14 defined features`
- `features = getFeatures(URLdataset["url"][i:i+1],`
`URLdataset["isPhishing"][i:i+1].values[0])`
- `featureSet.loc[i] = features`
- `featureSet`

Chọn Các Đặc Trưng và Biến Mục Tiêu:

```
sample_pt = featureSet.loc[:, 'no of dots': 'presence of suspicious domain']
target_pt = featureSet.iloc[:, [-1]]
```

- `sample_pt`: Lựa chọn tập hợp các đặc trưng từ DataFrame `featureSet`. Nó bao gồm các cột từ 'no of dots' đến 'presence of suspicious domain'. Điều này giả định rằng các cột này đại diện cho các đặc trưng được sử dụng để huấn luyện mô hình.
- `target_pt`: Lựa chọn biến mục tiêu từ DataFrame `featureSet`. Nó lựa chọn cột cuối cùng, giả định rằng nó đại diện cho biến mục tiêu chỉ ra liệu một URL có phải là lừa đảo hay không.

Phân Chia Dữ Liệu thành Tập Huấn Luyện và Tập Kiểm Tra:

```
sample_train_pt, sample_test_pt, target_train_pt, target_test_pt = train_test_split(sample_pt, target_pt, test_size=0.3, random_state=0)
```

- `train_test_split`: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra. Nó lấy tập hợp đặc trưng (`sample_pt`) và biến mục tiêu (`target_pt`) làm đầu vào và chia chúng thành các tập huấn luyện và kiểm tra (`sample_train_pt`, `sample_test_pt`, `target_train_pt`, `target_test_pt`). Tham số `test_size` xác định tỷ lệ của tập dữ liệu được bao gồm trong tập kiểm tra, và `random_state` đảm bảo tính nhất quán.

Huấn Luyện Bộ Phân Loại Cây Quyết Định:

```
tree_model_pt = tree.DecisionTreeClassifier()
tree_model_pt.fit(sample_train_pt, target_train_pt)
DecisionTreeClassifier()
```

- tree_model_pt: Khởi tạo một bộ phân loại cây quyết định.
- fit: Huấn luyện bộ phân loại cây quyết định bằng cách sử dụng dữ liệu huấn luyện (sample_train_pt, target_train_pt).

Dự Đoán:

```
prediction_pt = tree_model_pt.predict(sample_test_pt)
```

- predict: Sử dụng mô hình đã huấn luyện (tree_model_pt) để dự đoán trên dữ liệu kiểm tra (sample_test_pt). Các nhãn dự đoán được lưu trong prediction_pt.

Đánh Giá Độ Chính Xác của Mô Hình:

```
accuracy = 100 * accuracy_score(target_test_pt, prediction_pt)
print("Decision Tree applied on PhisTank Dataset has the accuracy score: %s" % str(accuracy))
```

- accuracy_score: Tính toán độ chính xác của các dự đoán của mô hình bằng cách so sánh chúng với các nhãn thực (target_test_pt). Điểm chính xác được nhân với 100 để biểu diễn nó dưới dạng phần trăm.
- In ra điểm chính xác của bộ phân loại cây quyết định được áp dụng vào tập dữ liệu PhisTank.

-Kết quả:

```
/home/kati/17322/ex0.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
URLdataphishing_extracted['isPhishing'] = 1
Decision Tree applied on PhisTank Dataset has the accuracy score: 94.5
```

-Dựa vào kết quả trên ta thấy mô hình phát hiện phishing website bằng Decision Tree với tập dữ liệu PhisTank có độ chính xác xấp xỉ 94.5%

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.DOCX và .PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-ExeX_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).
- Ví dụ: [NT101.K11.ANTT]-Exe01_Group03.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài nộp.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trể, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT