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 |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-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.**

BÁO CÁO CHI TIẾT

**A close-up of a text

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A table with numbers and letters

Description automatically generated**A screenshot of a computer

Description automatically generated

Sau đó ta upload lên google colab

**A white background with black dots

Description automatically generated  
A screenshot of a computer

Description automatically generated**

**- Đọc CSV thành Dataframe và hiển thị**

A screenshot of a computer code

Description automatically generated

**- Hãy chuyển index mặc định thành giá trị cột id**

A screenshot of a computer code

Description automatically generated

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**

A screenshot of a computer

Description automatically generated

**+ Xếp theo cột Y**

**A screenshot of a computer program

Description automatically generated**

**+ Xếp theo cột Z**

**A screenshot of a computer

Description automatically generated**

**- Chọn một cột cụ thể và hiển thị nó**

A screenshot of a computer

Description automatically generated

**- Chọn 2 hàng đầu tiền và hiển thị chúng**

A screenshot of a computer program

Description automatically generated

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

Chọn hàng sao cho cột Z chứa giá trị 73

**A screenshot of a computer code

Description automatically generated**

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

**A screenshot of a computer

Description automatically generated**

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

**A white rectangular object with a black border

Description automatically generated**

**- 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

**A group of colorful text boxes

Description automatically generated with medium confidence**

**- Chuyển Dataframe trên thành 2 Dataframe d1 và d2; d1 chứ 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

**A screenshot of a computer program

Description automatically generated**

Kết quả:**A white background with black text

Description automatically generated**

**- 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

A screenshot of a computer

Description automatically generated

A close-up of a message

Description automatically generated

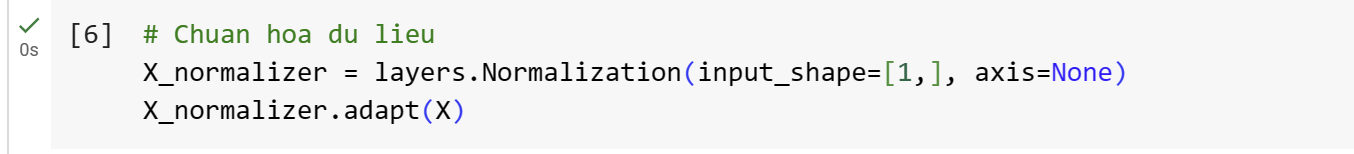
Sample Data

A computer screen shot of a computer code

Description automatically generated

- TensorFlow, Keras





A screenshot of a computer program

Description automatically generated

A black and white text

Description automatically generated with medium confidence

- Pytorch

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Scikit-leaning | 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. | |

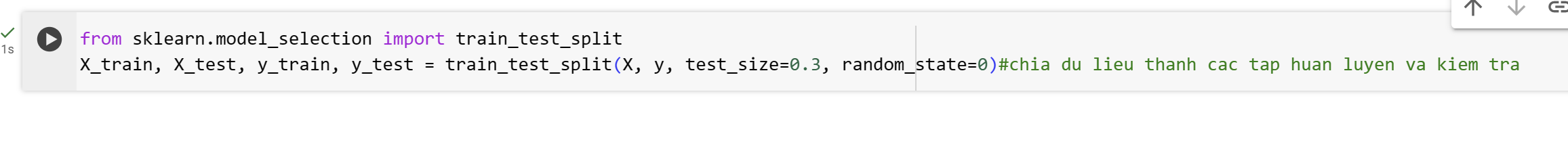
A close-up of a message

Description automatically generated

A screenshot of a computer code

Description automatically generated

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



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

A screenshot of a computer

Description automatically generated

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

A close-up of a computer code

Description automatically generated

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.

A screenshot of a computer program

Description automatically generated

A close-up of a sign

Description automatically generatedChức 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.

A close-up of a message

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

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

EX7:

A close-up of a web page

Description automatically generated

-Sử dụng các thư viện như sau:

A screen shot of a computer

Description automatically generated

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



-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.

A blue and white text on a black background

Description automatically generated

-Kết quả:

A screenshot of a computer program

Description automatically generated

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



-Kết quả:

A screenshot of a computer

Description automatically generated

-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.

A computer screen shot of words

Description automatically generated

-Kết quả:

A screenshot of a computer

Description automatically generated

-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).

A screen shot of a computer

Description automatically generated

-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.

A screen shot of a computer code

Description automatically generated

+) 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.

A screen shot of a computer code

Description automatically generated

+) 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.

A screen shot of a computer code

Description automatically generated

+) Kết quả của 2 mô hình và so sánh



-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 (coeficients) 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:

A close up of text

Description automatically generated

-Các thư viện sử dụng:

A screen shot of a computer code

Description automatically generated

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

A computer code with white text

Description automatically generated with medium confidence

-Kết quả:

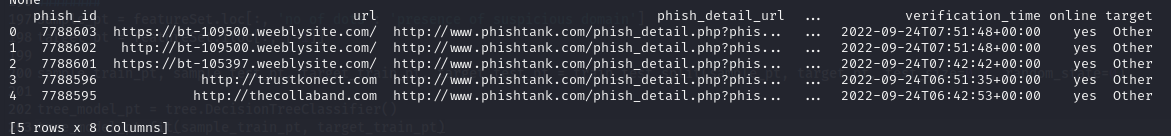
A screen shot of a computer

Description automatically generated

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



-Kết quả:



-Lấy một mẫu ngẫu nhiên từ tập dữ liệu:



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



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

A screen shot of a computer code

Description automatically generated

-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:

A computer code with colorful text

Description automatically generated with medium confidence

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

A screen shot of a computer

Description automatically generated

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

A black background with white text

Description automatically generated

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','deepspacer.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 delimeters*
* *def countdelim(url):*
* *count = 0*
* *delim=[';','\_','?','=','&']*
* *for each in url:*
* *if each in delim:*
* *count = count + 1*
* *return count*
* *# Method to count number of delimeters*
* *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["isPhising"][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["isPhising"][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["isPhising"][i:i+1].values[0])*
* *featureSet.loc[i] = features*
* *featureSet*

Chọn Các Đặc Trưng và Biến Mục Tiêu:



* 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:



* 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:

A computer screen shot of white text

Description automatically generated

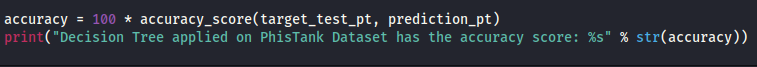
* tree\_model\_pt: Khởi tạo một bộ phân loại cây quyết định.
* fit: Huấn luyện bộ phân loại cây quyết định bằng cách sử dụng dữ liệu huấn luyện (sample\_train\_pt, target\_train\_pt).

Dự Đoán:



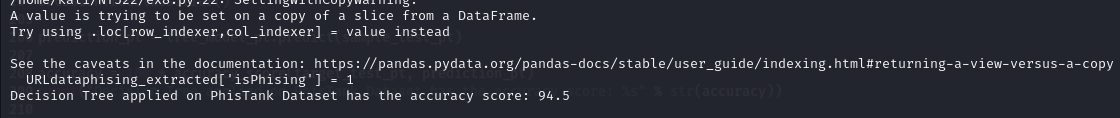
* predict: Sử dụng mô hình đã huấn luyện (tree\_model\_pt) để dự đoán trên dữ liệu kiểm tra (sample\_test\_pt). Các nhãn dự đoán được lưu trong prediction\_pt.

Đánh Giá Độ Chính Xác của Mô Hình:



* accuracy\_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:



-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, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)