# NT522.O21.ANTT.2

# 21520155 - Nguyễn Triệu Thiên Bảo

# 21521195 - Trần Lê Minh Ngọc

# Lab 5

# Phát hiện bất thường mạng sử dụng mô hình LSTM

Trong bài thực hành này, ta sẽ tạo và huấn luyện mô hình LSTM để phát hiện điểm bất thường trên tập dữ liệu lưu lượng mạng KDD99.

# A. Hướng dẫn xây dựng mô hình phân loại 2 lớp

## 1. Đọc tập dữ liệu KDD99

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics

from tensorflow.keras.utils import get_file
try:
    path = get_file('kddcup.data_10_percent.gz', origin='http://kdd.ics.uci.edu/databases/k
except:
    print('Error downloading')
    raise

print(path)
```

```
/root/.keras/datasets/kddcup.data_10_percent.gz
```

```
df = pd.read_csv(path, header=None)
print("Read {} rows.".format(len(df)))
```

```
Read 494021 rows.
```

```
# CSV không có header
df.columns = ['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','w
              'num_failed_logins','logged_in','num_compromised','root_shell', 'su_attempted
              'num_access_files','num_outbound_cmds','is_host_login','is_guest_login','coun
              'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_hos
              'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rat
              'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate',
df.head()
```

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragment |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | tcp | http | SF | 181 | 5450 | 0 | 0 |
| **1** | 0 | tcp | http | SF | 239 | 486 | 0 | 0 |
| **2** | 0 | tcp | http | SF | 235 | 1337 | 0 | 0 |
| **3** | 0 | tcp | http | SF | 219 | 1337 | 0 | 0 |
| **4** | 0 | tcp | http | SF | 217 | 2032 | 0 | 0 |

5 rows × 42 columns

## 2. Xử lý dữ liệu

[ ] ↳ *3 cells hidden*

## 3. Encode dữ liệu số và chữ

```
# Encode cột số
def encode_numeric_zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name] - mean) / sd

# Encode cột chữ ([1,0,0],[0,1,0],[0,0,1] cho red,green,blue)
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = f"{name}-{x}"
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)


#encoding feature vector
text_col =['protocol_type', 'service', 'flag', 'land', 'logged_in', 'is_host_login', 'is_gu

for i in df.columns:
  if i not in text_col:
    if i != 'outcome':
      encode_numeric_zscore(df, i)

for x in text_col:
  encode_text_dummy(df, x)
```

```
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
```

```
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-35-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
```

```
df.dropna(inplace=True,axis=1)
df[0:5]
```

|   | duration | src_bytes | dst_bytes | wrong_fragment | urgent | hot | num_failed_login |
|---|---|---|---|---|---|---|---|
| **0** | -0.067792 | -0.002879 | 0.138664 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **1** | -0.067792 | -0.002820 | -0.011578 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **2** | -0.067792 | -0.002824 | 0.014179 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **3** | -0.067792 | -0.002840 | 0.014179 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **4** | -0.067792 | -0.002842 | 0.035214 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |

5 rows × 121 columns

```
df['protocol_type-tcp'].unique()
```

```
array([ True, False])
```

```
df.loc[df["outcome"] != "normal.", "outcome"] = 1
df.loc[df["outcome"] == "normal.", "outcome"] = 0
```

```
y = df['outcome']
df.drop('outcome',axis=1,inplace=True)
```

```
df = df.replace({True: 1, False: 0})
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df, y,  test_size=0.3, random_state=12)

print(f"Normal train count: {x_train.shape, y_train.shape}")
print(f"Normal test count: {x_test.shape, y_test.shape}")
```

```
Normal train count: ((345814, 120), (345814,))
Normal test count: ((148207, 120), (148207,))
```

```
y_train = tf.one_hot(y_train.values, 2)
y_test = tf.one_hot(y_test.values, 2)
```

## 4. Kiến trúc mô hình LSTM

```
model = keras.Sequential()
model.add(keras.layers.LSTM(units=64, input_shape=(x_train.shape[1],1)))
model.add(keras.layers.Dropout(rate=0.8))
model.add(keras.layers.Dense(units=y_train.shape[1], activation='softmax'))

model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
model.summary()
```

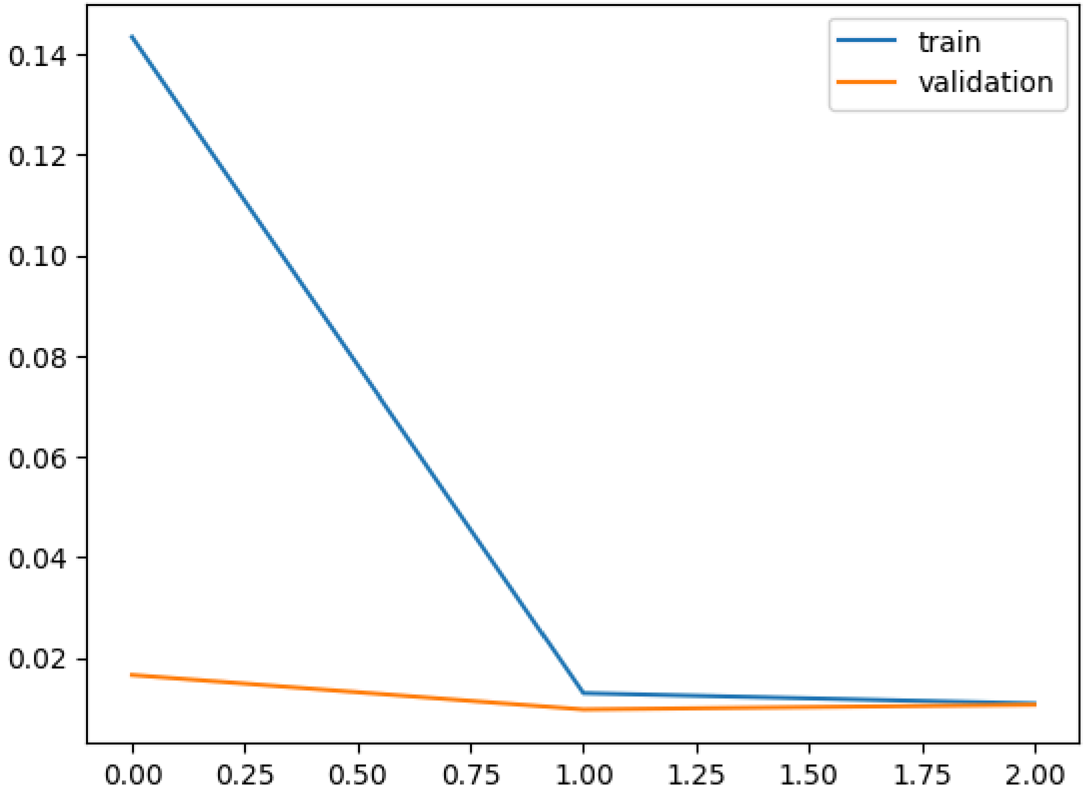
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 64)                16896

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 17026 (66.51 KB)
Trainable params: 17026 (66.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

## 5. Huấn luyện mô hình

```
history = model.fit(
    x_train, y_train,
    epochs=3,
    batch_size=1024,
    validation_split=0.2,
    shuffle = False
)
```

```
Epoch 1/3
271/271 [==============================] - 7s 20ms/step - loss: 0.1434 - accuracy: 0.82
Epoch 2/3
271/271 [==============================] - 5s 17ms/step - loss: 0.0129 - accuracy: 0.98
Epoch 3/3
271/271 [==============================] - 4s 16ms/step - loss: 0.0109 - accuracy: 0.98
```

```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend();
```



## 6. Đánh giá mô hình

[ ] ↳ *5 cells hidden*

# B. Bài tập

1. **Yêu cầu 1 (Tại lớp): Dựa trên hướng dẫn A hãy xây dựng một mô hình phân loại đa lớp (Multiclass Classification) với bộ dữ liệu KDD99.**
2. **Yêu cầu 2 (Về nhà): Sinh viên chạy lại tập dữ liệu CIC IDS 2018 trên mô hình bài lab này ở cả Multiclass Classification và Binary Classification.**

## Yêu cầu 1: Dựa trên hướng dẫn A hãy xây dựng một mô hình phân loại đa lớp (Multiclass Classification) với bộ dữ liệu KDD99.

## 1. Đọc tập dữ liệu KDD99

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics

from tensorflow.keras.utils import get_file
try:
    path = get_file('kddcup.data_10_percent.gz', origin='http://kdd.ics.uci.edu/databases/k
except:
    print('Error downloading')
    raise

print(path)
```

```
/root/.keras/datasets/kddcup.data_10_percent.gz
```

```
df = pd.read_csv(path, header=None)
print("Read {} rows.".format(len(df)))
```

```
Read 494021 rows.
```

```
# CSV không có header
df.columns = ['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','w
              'num_failed_logins','logged_in','num_compromised','root_shell', 'su_attempted
              'num_access_files','num_outbound_cmds','is_host_login','is_guest_login','coun
              'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_hos
              'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rat
              'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate',
df.head()
```

|   | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragment |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | tcp | http | SF | 181 | 5450 | 0 | 0 |
| **1** | 0 | tcp | http | SF | 239 | 486 | 0 | 0 |
| **2** | 0 | tcp | http | SF | 235 | 1337 | 0 | 0 |
| **3** | 0 | tcp | http | SF | 219 | 1337 | 0 | 0 |
| **4** | 0 | tcp | http | SF | 217 | 2032 | 0 | 0 |

5 rows × 42 columns

## 2. Xử lý dữ liệu

```
# loại bỏ NA
df.dropna(inplace=True,axis=1)
df.shape
```

```
(494021, 42)
```

```
df.dtypes
```

```
duration                         int64
protocol_type                   object
service                         object
flag                            object
src_bytes                        int64
dst_bytes                        int64
land                             int64
wrong_fragment                   int64
urgent                           int64
hot                              int64
num_failed_logins                int64
logged_in                        int64
num_compromised                  int64
root_shell                       int64
su_attempted                     int64
num_root                         int64
num_file_creations               int64
num_shells                       int64
num_access_files                 int64
num_outbound_cmds                int64
is_host_login                    int64
is_guest_login                   int64
count                            int64
srv_count                        int64
serror_rate                    float64
srv_serror_rate                float64
```

```
rerror_rate                    float64
srv_rerror_rate                float64
same_srv_rate                  float64
diff_srv_rate                  float64
srv_diff_host_rate             float64
dst_host_count                   int64
dst_host_srv_count               int64
dst_host_same_srv_rate         float64
dst_host_diff_srv_rate         float64
dst_host_same_src_port_rate    float64
dst_host_srv_diff_host_rate    float64
dst_host_serror_rate           float64
dst_host_srv_serror_rate       float64
dst_host_rerror_rate           float64
dst_host_srv_rerror_rate       float64
outcome                         object
dtype: object
```

```
df.groupby('outcome')['outcome'].count()
```

```
outcome
back.                 2203
buffer_overflow.        30
ftp_write.               8
guess_passwd.           53
imap.                   12
ipsweep.              1247
land.                   21
loadmodule.              9
multihop.                7
neptune.            107201
nmap.                  231
normal.              97278
perl.                    3
phf.                     4
pod.                   264
portsweep.            1040
rootkit.                10
satan.                1589
smurf.              280790
spy.                     2
teardrop.              979
warezclient.          1020
warezmaster.            20
Name: outcome, dtype: int64
```

## 3. Encode dữ liệu số và chữ

```
# Encode cột số
def encode_numeric_zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name] - mean) / sd

# Encode cột chữ ([1,0,0],[0,1,0],[0,0,1] cho red,green,blue)
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = f"{name}-{x}"
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)


#encoding feature vector
text_col =['protocol_type', 'service', 'flag', 'land', 'logged_in', 'is_host_login', 'is_gu

for i in df.columns:
  if i not in text_col:
    if i != 'outcome':
      encode_numeric_zscore(df, i)

for x in text_col:
  encode_text_dummy(df, x)
```

```
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
```

```
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
<ipython-input-58-ede0120ec756>:16: PerformanceWarning: DataFrame is highly fragmented.
  df[dummy_name] = dummies[x]
```

```
df.dropna(inplace=True,axis=1)
df[0:5]
```

| | duration | src_bytes | dst_bytes | wrong_fragment | urgent | hot | num_failed_login |
|---|---|---|---|---|---|---|---|
| **0** | -0.067792 | -0.002879 | 0.138664 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **1** | -0.067792 | -0.002820 | -0.011578 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **2** | -0.067792 | -0.002824 | 0.014179 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **3** | -0.067792 | -0.002840 | 0.014179 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |
| **4** | -0.067792 | -0.002842 | 0.035214 | -0.04772 | -0.002571 | -0.044136 | -0.00978 |

5 rows × 121 columns

```
df['protocol_type-tcp'].unique()
```

```
array([ True, False])
```

```
labels = df["outcome"].unique()
for i, label in enumerate(labels):
    if label == "normal.":
        df.loc[df["outcome"] == label, "outcome"] = 0
    else:
        df.loc[df["outcome"] == label, "outcome"] = i+1
```

```
y = df['outcome']
df.drop('outcome',axis=1,inplace=True)
```

```
df = df.replace({True: 1, False: 0})
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df, y,  test_size=0.3, random_state=12)

print(f"Normal train count: {x_train.shape, y_train.shape}")
print(f"Normal test count: {x_test.shape, y_test.shape}")
```

```
Normal train count: ((345814, 120), (345814,))
Normal test count: ((148207, 120), (148207,))
```

```
y_train = tf.one_hot(y_train.values, 2)
y_test = tf.one_hot(y_test.values, 2)
```

## 4. Kiến trúc mô hình LSTM

```
model = keras.Sequential()
model.add(keras.layers.LSTM(units=64, input_shape=(x_train.shape[1],1)))
model.add(keras.layers.Dropout(rate=0.8))
model.add(keras.layers.Dense(units=y_train.shape[1], activation='softmax'))

model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                16896

 dropout_2 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 2)                 130

=================================================================
Total params: 17026 (66.51 KB)
```

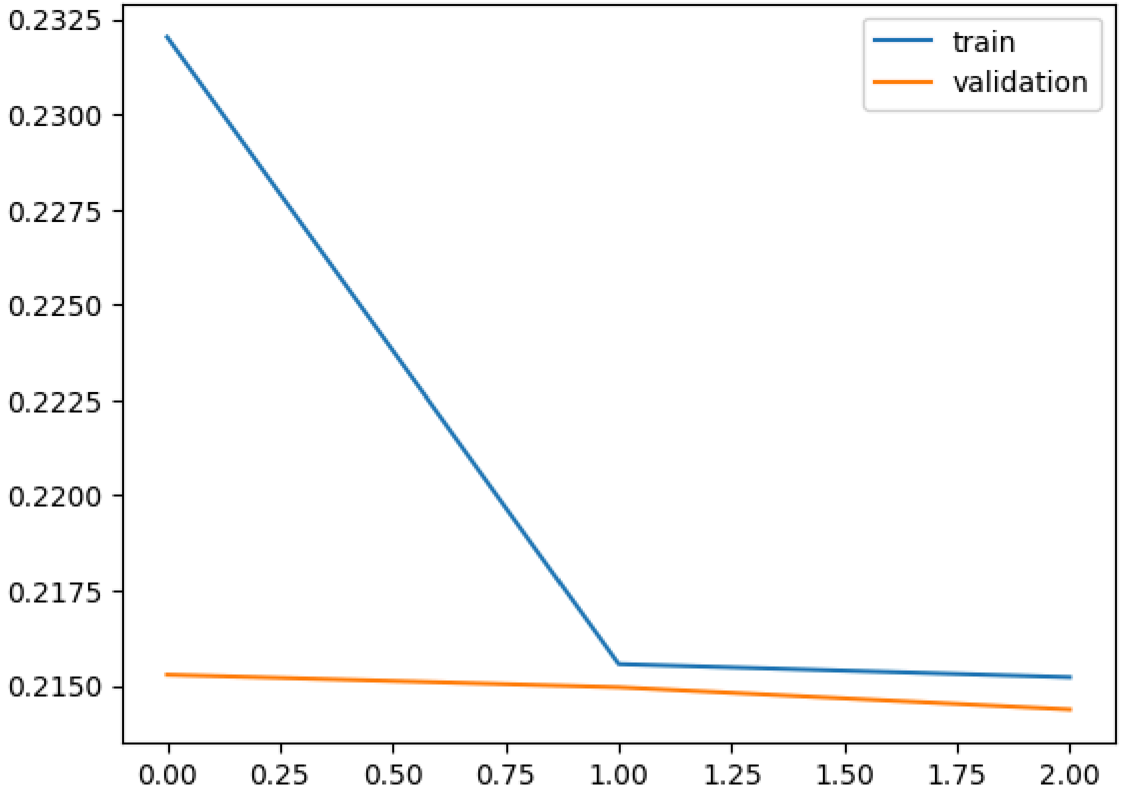
```
Trainable params: 17026 (66.51 KB)
Non-trainable params: 0 (0.00 Byte)
```

## 5. Huấn luyện mô hình

```
history = model.fit(
    x_train, y_train,
    epochs=3,
    batch_size=1024,
    validation_split=0.2,
    shuffle = False
)
```

```
Epoch 1/3
271/271 [==============================] - 7s 18ms/step - loss: 0.2320 - accuracy: 0.94
Epoch 2/3
271/271 [==============================] - 5s 17ms/step - loss: 0.2156 - accuracy: 0.89
Epoch 3/3
271/271 [==============================] - 5s 17ms/step - loss: 0.2152 - accuracy: 0.90
```

```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend();
```



## 6. Đánh giá mô hình

```
score1 = model.evaluate(x_train, y_train, batch_size=1024)
```

```
338/338 [==============================] - 2s 7ms/step - loss: 0.2144 - accuracy: 0.824
```

# Yêu cầu 2: Sinh viên chạy lại tập dữ liệu CIC IDS 2018 trên mô hình bài lab này ở cả Multiclass Classification và Binary Classification.

## Cài đặt các công cụ cần thiết

```
!curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscli-exe-linux-x86_6
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 57.7M  100 57.7M    0     0   145M      0 --:--:-- --:--:-- --:--:--  145M
```

```
!unzip awscli-exe-linux-x86_64.zip
```

```
  inflating: aws/dist/awscli/examples/iotanalytics/cancel-pipeline-reprocessing.rst
  inflating: aws/dist/awscli/examples/omics/cancel-variant-import-job.rst
  inflating: aws/dist/awscli/examples/omics/create-share.rst
  inflating: aws/dist/awscli/examples/omics/delete-workflow.rst
  inflating: aws/dist/awscli/examples/omics/list-multipart-read-set-uploads.rst
  inflating: aws/dist/awscli/examples/omics/list-annotation-import-jobs.rst
  inflating: aws/dist/awscli/examples/omics/create-run-group.rst
  inflating: aws/dist/awscli/examples/omics/create-workflow.rst
  inflating: aws/dist/awscli/examples/omics/start-read-set-activation-job.rst
  inflating: aws/dist/awscli/examples/omics/start-read-set-export-job.rst
  inflating: aws/dist/awscli/examples/omics/list-run-groups.rst
  inflating: aws/dist/awscli/examples/omics/abort-multipart-read-set-upload.rst
  inflating: aws/dist/awscli/examples/omics/get-read-set.rst
  inflating: aws/dist/awscli/examples/omics/create-variant-store.rst
  inflating: aws/dist/awscli/examples/omics/delete-variant-store.rst
  inflating: aws/dist/awscli/examples/omics/delete-annotation-store-versions.rst
  inflating: aws/dist/awscli/examples/omics/update-annotation-store.rst
  inflating: aws/dist/awscli/examples/omics/cancel-run.rst
  inflating: aws/dist/awscli/examples/omics/get-reference-store.rst
  inflating: aws/dist/awscli/examples/omics/get-share.rst
  inflating: aws/dist/awscli/examples/omics/get-reference.rst
  inflating: aws/dist/awscli/examples/omics/start-annotation-import-job.rst
  inflating: aws/dist/awscli/examples/omics/get-annotation-store-version.rst
  inflating: aws/dist/awscli/examples/omics/list-reference-import-jobs.rst
  inflating: aws/dist/awscli/examples/omics/get-reference-import-job.rst
  inflating: aws/dist/awscli/examples/omics/batch-delete-read-set.rst
  inflating: aws/dist/awscli/examples/omics/get-annotation-store.rst
  inflating: aws/dist/awscli/examples/omics/upload-read-set-part.rst
  inflating: aws/dist/awscli/examples/omics/list-variant-stores.rst
  inflating: aws/dist/awscli/examples/omics/get-read-set-activation-job.rst
  inflating: aws/dist/awscli/examples/omics/cancel-annotation-import-job.rst
  inflating: aws/dist/awscli/examples/omics/get-workflow.rst
  inflating: aws/dist/awscli/examples/omics/tag-resource.rst
  inflating: aws/dist/awscli/examples/omics/list-reference-stores.rst
  inflating: aws/dist/awscli/examples/omics/create-annotation-store.rst
  inflating: aws/dist/awscli/examples/omics/delete-run.rst
```

```
!sudo ./aws/install
```

```
You can now run: /usr/local/bin/aws --version
```

```
!/usr/local/bin/aws --version
```

```
aws-cli/2.15.47 Python/3.11.8 Linux/6.1.58+ exe/x86_64.ubuntu.22 prompt/off
```

```
!aws s3 ls --no-sign-request --region us-east-2 "s3://cse-cic-ids2018/"
```

```
                           PRE Original Network Traffic and Log data/
                           PRE Processed Traffic Data for ML Algorithms/
```

```
!aws s3 ls --no-sign-request --region us-east-2 "s3://cse-cic-ids2018/Original Network Traf
```

```
                           PRE Friday-02-03-2018/
                           PRE Friday-16-02-2018/
                           PRE Friday-23-02-2018/
                           PRE Thursday-01-03-2018/
                           PRE Thursday-15-02-2018/
                           PRE Thursday-22-02-2018/
                           PRE Tuesday-20-02-2018/
                           PRE Wednesday-14-02-2018/
                           PRE Wednesday-21-02-2018/
                           PRE Wednesday-28-02-2018/
2018-10-10 11:52:09          0
```

```
!aws s3 ls --no-sign-request --region us-east-2 "s3://cse-cic-ids2018/Processed Traffic Dat
```

```
2018-10-11 16:02:25          0
2018-10-11 16:02:49  352368373 Friday-02-03-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:03:10  333723605 Friday-16-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:03:33  382840456 Friday-23-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:03:59 4054925350 Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:08:38  107842858 Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:08:48  375945899 Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:09:20  382636202 Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:09:44  358223333 Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:10:12  328893673 Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv
2018-10-11 16:10:33  209249758 Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv
```

```
!aws s3 cp --no-sign-request --region us-east-2 "s3://cse-cic-ids2018/Processed Traffic Dat
```

```
download: s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/Friday-16-02-20
```

## Binary Classification

## 1. Đọc dữ liệu

```
df = pd.read_csv('/content/drive/MyDrive/NT522.O21.ANTT_Lab5/Friday-16-02-2018_TrafficForML
print("Read {} rows.".format(len(df)))
df
```

```
Read 200000 rows.
```

| | Dst Port | Protocol | Timestamp | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | .. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 16/02/2018 08:27:23 | 112640768 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **1** | 0 | 0 | 16/02/2018 08:30:12 | 112641773 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **2** | 35605 | 6 | 16/02/2018 08:26:55 | 20784143 | 23 | 44 | 2416 | 1344 | 240 | 64 | |
| **3** | 0 | 0 | 16/02/2018 08:33:01 | 112640836 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **4** | 23 | 6 | 16/02/2018 08:27:59 | 20 | 1 | 1 | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **199995** | 80 | 6 | 16/02/2018 01:45:49 | 4142 | 2 | 0 | 0 | 0 | 0 | 0 | |
| **199996** | 40012 | 6 | 16/02/2018 01:45:44 | 5393905 | 5 | 3 | 935 | 345 | 935 | 0 | |
| **199997** | 39978 | 6 | 16/02/2018 01:45:44 | 5414995 | 5 | 3 | 935 | 358 | 935 | 0 | |
| **199998** | 40022 | 6 | 16/02/2018 01:45:44 | 5156071 | 5 | 3 | 935 | 351 | 935 | 0 | |
| **199999** | 80 | 6 | 16/02/2018 01:45:49 | 1041 | 2 | 0 | 0 | 0 | 0 | 0 | |

200000 rows × 80 columns

## 2. Xử lý dữ liệu

```
# loại bỏ NA
df.dropna(inplace=True,axis=1)
df.shape
```

```
(200000, 80)
```

```
df.dtypes
```

```
Dst Port           int64
Protocol           int64
Timestamp         object
Flow Duration      int64
Tot Fwd Pkts       int64
                   ...
Idle Mean        float64
Idle Std         float64
Idle Max           int64
Idle Min           int64
Label             object
Length: 80, dtype: object
```

```
df.groupby('Label')['Label'].count()
```

```
Label
Benign                      41216
DoS attacks-Hulk            67350
DoS attacks-SlowHTTPTest    91434
Name: Label, dtype: int64
```

```
# Loại bỏ các cột chỉ có 1 giá trị để tiết kiệm thời gian tính toán
def remove_single_value_columns(dataframe):
    col_names = dataframe.columns
    unique_values = [len(dataframe[col].unique()) for col in col_names]

    col_unique_df = pd.DataFrame({'ColName': col_names, 'UniqueValues': unique_values})
    single_val_cols_df = col_unique_df[col_unique_df['UniqueValues'] == 1]
    return col_unique_df, single_val_cols_df


def sort_data_by_time(dataframe, time_col="Timestamp"):
    sorted_data = dataframe.sort_values(by=[time_col], ascending=True)
    return sorted_data

col_unique_df, single_val_cols_df = remove_single_value_columns(df)
df = df.drop(columns=single_val_cols_df['ColName'].values)

# Sắp xếp dữ liệu theo thời gian
df = sort_data_by_time(df)

df
```

|  | Dst Port | Protocol | Timestamp | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **91659** | 0 | 0 | 16/02/2018 01:00:32 | 112640723 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91658** | 22 | 6 | 16/02/2018 01:01:42 | 2146470 | 14 | 12 | 1335 | 2273 | 744 | 0 | ... |
| **91661** | 0 | 0 | 16/02/2018 01:03:21 | 112640737 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91689** | 67 | 17 | 16/02/2018 01:03:50 | 721 | 1 | 1 | 300 | 329 | 300 | 300 | ... |
| **91663** | 0 | 0 | 16/02/2018 01:06:10 | 112640647 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **91653** | 0 | 0 | 16/02/2018 12:52:05 | 112640636 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91654** | 0 | 0 | 16/02/2018 12:54:54 | 112640695 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91657** | 0 | 0 | 16/02/2018 12:57:43 | 112640664 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91655** | 22 | 6 | 16/02/2018 12:58:13 | 10162102 | 9 | 7 | 1063 | 1297 | 744 | 0 | ... |
| **91656** | 42453 | 6 | 16/02/2018 12:58:24 | 855 | 2 | 0 | 848 | 0 | 848 | 0 | ... |

200000 rows × 69 columns

```
# Xóa cột timestamp
# Timestamp không đóng góp vào việc training model vì vậy xóa nó đi để giảm kích thước dữ l
df.drop('Timestamp', axis=1, inplace=True)
df
```

```
                                                                Fwd  Fwd
         Dst                Flow  Tot  Tot TotLen TotLen  Pkt  Pkt   Fwd Pkt
        Port Protocol   Duration  Fwd  Bwd    Fwd    Bwd  Len  Len  Len Mean ...
                                 Pkts Pkts   Pkts   Pkts  Max  Min
91659      0        0  112640723    3    0      0      0    0    0  0.000000 ...
91658     22        6    2146470   14   12   1335   2273  744    0 95.357143 ...
91661      0        0  112640737    3    0      0      0    0    0  0.000000 ...
91689     67       17        721    1    1    300    329  300  300 300.000000 ...
91663      0        0  112640647    3    0      0      0    0    0  0.000000 ...
  ...    ...      ...        ...  ...  ...    ...    ...  ...  ...       ... ...
91653      0        0  112640636    3    0      0      0    0    0  0.000000 ...
91654      0        0  112640695    3    0      0      0    0    0  0.000000 ...
91657      0        0  112640664    3    0      0      0    0    0  0.000000 ...
91655     22        6   10162102    9    7   1063   1297  744    0 118.111111 ...
91656  42453        6        855    2    0    848      0  848    0 424.000000 ...

200000 rows × 68 columns
```

## 3. Encode dữ liệu số và chữ

```python
# Encode cột số
def encode_numeric_zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name] - mean) / sd

# Encode cột chữ ([1,0,0],[0,1,0],[0,0,1] cho red,green,blue)
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = f"{name}-{x}"
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)
```

```
#encoding feature vector
text_col =[ ]

for i in df.columns:
  if i not in text_col:
    if i != 'Label':
      encode_numeric_zscore(df, i)

for x in text_col:
  encode_text_dummy(df, x)
```

```
df.dropna(inplace=True,axis=1)
df[0:5]
```

|  | Dst Port | Protocol | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max |
|---|---|---|---|---|---|---|---|---|
| **91659** | -0.500746 | -36.506422 | 32.154667 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |
| **91658** | -0.499600 | 0.016953 | 0.274845 | 7.166526 | 4.317151 | 2.962521 | 0.600508 | 1.407392 |
| **91661** | -0.500746 | -36.506422 | 32.154671 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |
| **91689** | -0.497257 | 66.976473 | -0.344246 | -0.761037 | -0.172995 | 0.219894 | 0.048766 | 0.223401 |
| **91663** | -0.500746 | -36.506422 | 32.154645 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |

5 rows × 68 columns

```
df.loc[df["Label"] != "Benign.", "Label"] = 1
df.loc[df["Label"] == "Benign.", "Label"] = 0
```

```
y = df['Label']
df.drop('Label',axis=1,inplace=True)
```

```
df = df.replace({True: 1, False: 0})
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(df, y,  test_size=0.3, random_state=12)

print(f"Normal train count: {X_train.shape, Y_train.shape}")
print(f"Normal test count: {X_test.shape, Y_test.shape}")
```

```
Normal train count: ((140000, 67), (140000,))
Normal test count: ((60000, 67), (60000,))
```

```
# Convert Y_train and Y_test to one-hot encoding
Y_train = tf.one_hot(Y_train.values, 2)
Y_test = tf.one_hot(Y_test.values, 2)
```

## 4. Kiến trúc mô hình LSTM

```
model = keras.Sequential()
model.add(keras.layers.Dense(units=32, activation='relu', input_shape=(X_train.shape[1],)))
model.add(keras.layers.Dropout(rate=0.5))
model.add(keras.layers.Dense(units=2, activation='softmax'))

model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
model.summary()
```

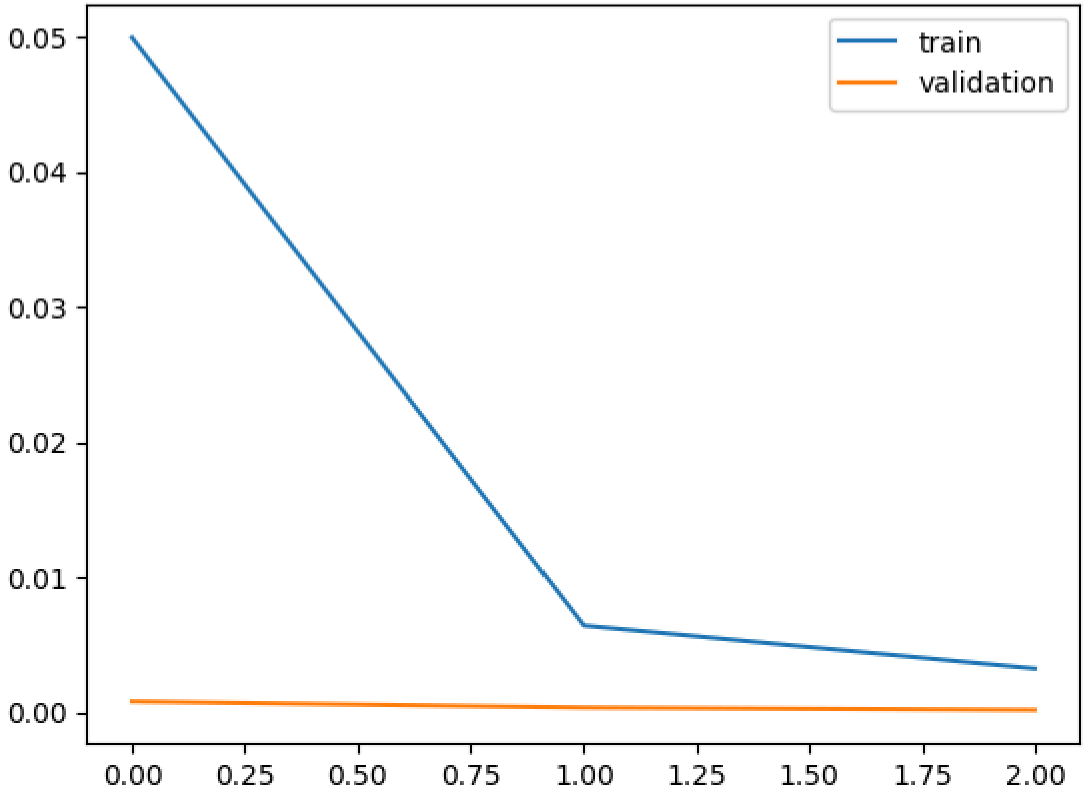
```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 32)                2176

 dropout_6 (Dropout)         (None, 32)                0

 dense_10 (Dense)            (None, 2)                 66

=================================================================
Total params: 2242 (8.76 KB)
Trainable params: 2242 (8.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

## 5. Huấn luyện mô hình

```
history = model.fit(
    X_train, Y_train,
    epochs=3,
    batch_size=1024,
    validation_split=0.2,
    shuffle=False
)
```

```
Epoch 1/3
110/110 [==============================] - 1s 5ms/step - loss: 0.0499 - accuracy: 0.944
Epoch 2/3
110/110 [==============================] - 0s 4ms/step - loss: 0.0064 - accuracy: 0.997
Epoch 3/3
110/110 [==============================] - 0s 4ms/step - loss: 0.0032 - accuracy: 0.999
```

```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend();
```



## 6. Đánh giá mô hình

```
score1 = model.evaluate(X_train, Y_train, batch_size=1024)
```

```
137/137 [==============================] - 1s 6ms/step - loss: 1.0434e-04 - accuracy: 0
```

# Multiclass Classification

## 1. Đọc dữ liệu

```
df = pd.read_csv('/content/drive/MyDrive/NT522.O21.ANTT_Lab5/Friday-16-02-2018_TrafficForML
print("Read {} rows.".format(len(df)))
df
```

```
Read 200000 rows.
```

| | Dst Port | Protocol | Timestamp | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | .. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 16/02/2018 08:27:23 | 112640768 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **1** | 0 | 0 | 16/02/2018 08:30:12 | 112641773 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **2** | 35605 | 6 | 16/02/2018 08:26:55 | 20784143 | 23 | 44 | 2416 | 1344 | 240 | 64 | |
| **3** | 0 | 0 | 16/02/2018 08:33:01 | 112640836 | 3 | 0 | 0 | 0 | 0 | 0 | |
| **4** | 23 | 6 | 16/02/2018 08:27:59 | 20 | 1 | 1 | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **199995** | 80 | 6 | 16/02/2018 01:45:49 | 4142 | 2 | 0 | 0 | 0 | 0 | 0 | |
| **199996** | 40012 | 6 | 16/02/2018 01:45:44 | 5393905 | 5 | 3 | 935 | 345 | 935 | 0 | |
| **199997** | 39978 | 6 | 16/02/2018 01:45:44 | 5414995 | 5 | 3 | 935 | 358 | 935 | 0 | |
| **199998** | 40022 | 6 | 16/02/2018 01:45:44 | 5156071 | 5 | 3 | 935 | 351 | 935 | 0 | |
| **199999** | 80 | 6 | 16/02/2018 01:45:49 | 1041 | 2 | 0 | 0 | 0 | 0 | 0 | |

200000 rows × 80 columns

## 2. Xử lý dữ liệu

```
# loại bỏ NA
df.dropna(inplace=True,axis=1)
df.shape
```

```
(200000, 80)
```

```
df.dtypes
```

```
Dst Port              int64
Protocol              int64
Timestamp            object
Flow Duration         int64
Tot Fwd Pkts          int64
                      ...
Idle Mean           float64
Idle Std            float64
Idle Max              int64
Idle Min              int64
Label                object
Length: 80, dtype: object
```

```
df.groupby('Label')['Label'].count()
```

```
Label
Benign                        41216
DoS attacks-Hulk              67350
DoS attacks-SlowHTTPTest      91434
Name: Label, dtype: int64
```

```
# Loại bỏ các cột chỉ có 1 giá trị để tiết kiệm thời gian tính toán
def remove_single_value_columns(dataframe):
    col_names = dataframe.columns
    unique_values = [len(dataframe[col].unique()) for col in col_names]

    col_unique_df = pd.DataFrame({'ColName': col_names, 'UniqueValues': unique_values})
    single_val_cols_df = col_unique_df[col_unique_df['UniqueValues'] == 1]
    return col_unique_df, single_val_cols_df


def sort_data_by_time(dataframe, time_col="Timestamp"):
    sorted_data = dataframe.sort_values(by=[time_col], ascending=True)
    return sorted_data

col_unique_df, single_val_cols_df = remove_single_value_columns(df)
df = df.drop(columns=single_val_cols_df['ColName'].values)

# Sắp xếp dữ liệu theo thời gian
df = sort_data_by_time(df)

df
```

|  | Dst Port | Protocol | Timestamp | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **91659** | 0 | 0 | 16/02/2018 01:00:32 | 112640723 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91658** | 22 | 6 | 16/02/2018 01:01:42 | 2146470 | 14 | 12 | 1335 | 2273 | 744 | 0 | ... |
| **91661** | 0 | 0 | 16/02/2018 01:03:21 | 112640737 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91689** | 67 | 17 | 16/02/2018 01:03:50 | 721 | 1 | 1 | 300 | 329 | 300 | 300 | ... |
| **91663** | 0 | 0 | 16/02/2018 01:06:10 | 112640647 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **91653** | 0 | 0 | 16/02/2018 12:52:05 | 112640636 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91654** | 0 | 0 | 16/02/2018 12:54:54 | 112640695 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91657** | 0 | 0 | 16/02/2018 12:57:43 | 112640664 | 3 | 0 | 0 | 0 | 0 | 0 | ... |
| **91655** | 22 | 6 | 16/02/2018 12:58:13 | 10162102 | 9 | 7 | 1063 | 1297 | 744 | 0 | ... |
| **91656** | 42453 | 6 | 16/02/2018 12:58:24 | 855 | 2 | 0 | 848 | 0 | 848 | 0 | ... |

200000 rows × 69 columns

```
# Xóa cột timestamp
# Timestamp không đóng góp vào việc training model vì vậy xóa nó đi để giảm kích thước dữ l
df.drop('Timestamp', axis=1, inplace=True)
df
```

|  | Dst Port | Protocol | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | Fwd Pkt Len Mean | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **91659** | 0 | 0 | 112640723 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **91658** | 22 | 6 | 2146470 | 14 | 12 | 1335 | 2273 | 744 | 0 | 95.357143 | ... |
| **91661** | 0 | 0 | 112640737 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **91689** | 67 | 17 | 721 | 1 | 1 | 300 | 329 | 300 | 300 | 300.000000 | ... |
| **91663** | 0 | 0 | 112640647 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **91653** | 0 | 0 | 112640636 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **91654** | 0 | 0 | 112640695 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **91657** | 0 | 0 | 112640664 | 3 | 0 | 0 | 0 | 0 | 0 | 0.000000 | ... |
| **91655** | 22 | 6 | 10162102 | 9 | 7 | 1063 | 1297 | 744 | 0 | 118.111111 | ... |
| **91656** | 42453 | 6 | 855 | 2 | 0 | 848 | 0 | 848 | 0 | 424.000000 | ... |

200000 rows × 68 columns

## 3. Encode dữ liệu số và chữ

```
# Encode cột số
def encode_numeric_zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name] - mean) / sd

# Encode cột chữ ([1,0,0],[0,1,0],[0,0,1] cho red,green,blue)
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = f"{name}-{x}"
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)
```

```
#encoding feature vector
text_col =[ ]

for i in df.columns:
  if i not in text_col:
    if i != 'Label':
      encode_numeric_zscore(df, i)

for x in text_col:
  encode_text_dummy(df, x)
```

```
df.dropna(inplace=True,axis=1)
df[0:5]
```

| | Dst Port | Protocol | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max |
|---|---|---|---|---|---|---|---|---|
| **91659** | -0.500746 | -36.506422 | 32.154667 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |
| **91658** | -0.499600 | 0.016953 | 0.274845 | 7.166526 | 4.317151 | 2.962521 | 0.600508 | 1.407392 |
| **91661** | -0.500746 | -36.506422 | 32.154671 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |
| **91689** | -0.497257 | 66.976473 | -0.344246 | -0.761037 | -0.172995 | 0.219894 | 0.048766 | 0.223401 |
| **91663** | -0.500746 | -36.506422 | 32.154645 | 0.458588 | -0.581190 | -0.575070 | -0.044611 | -0.576593 |

5 rows × 68 columns

```
labels = df["Label"].unique()
for i, label in enumerate(labels):
    if label == "Benign":
        df.loc[df["Label"] == label, "Label"] = 0
    else:
        df.loc[df["Label"] == label, "Label"] = i+1
```

```
y = df['Label']
df.drop('Label',axis=1,inplace=True)
```

```
df = df.replace({True: 1, False: 0})
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(df, y,  test_size=0.3, random_state=12)

print(f"Normal train count: {X_train.shape, Y_train.shape}")
print(f"Normal test count: {X_test.shape, Y_test.shape}")
```

```
Normal train count: ((140000, 67), (140000,))
Normal test count: ((60000, 67), (60000,))
```