BÁO CÁO LAB

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

Tên chủ đề: Lab 5

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

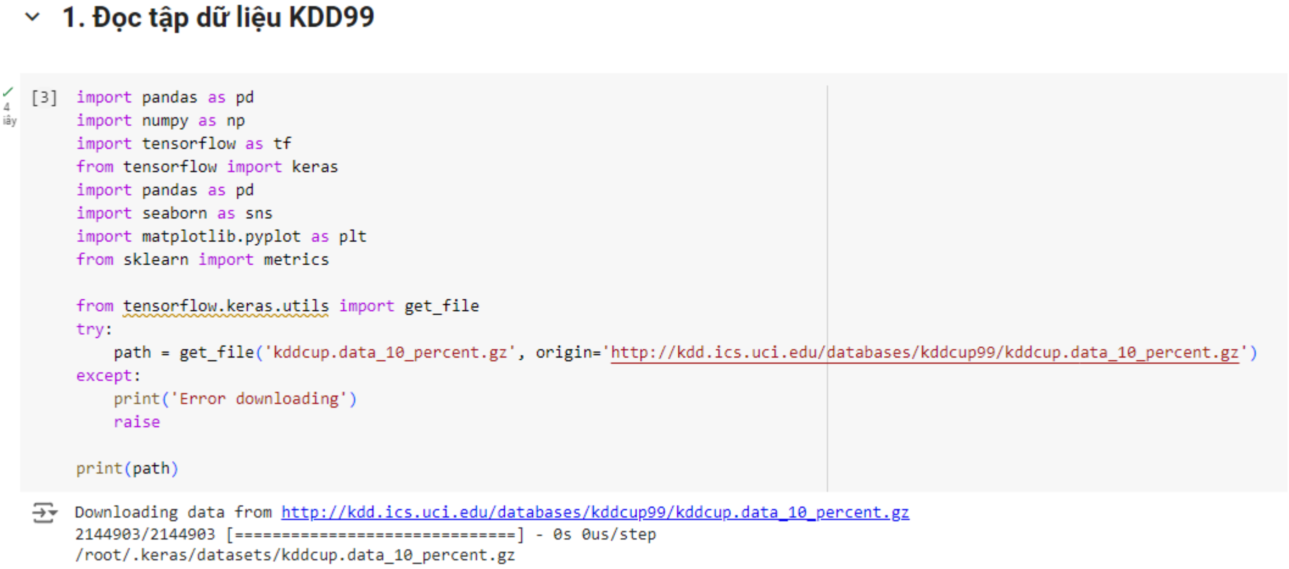
Lớp: NT522.O21.ATCL.1

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

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

BÁO CÁO CHI TIẾT

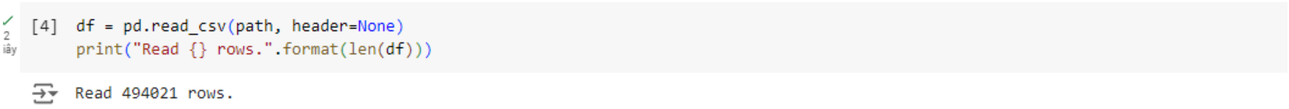
Yêu cầu 1: Dựa trên hướng dẫn A hãy xây dựng một mô hình phân loại đa lớp (Multiclass Classification) với bộ dữ liệu KDD99.



1. Đọc tập dữ liệu KDD99

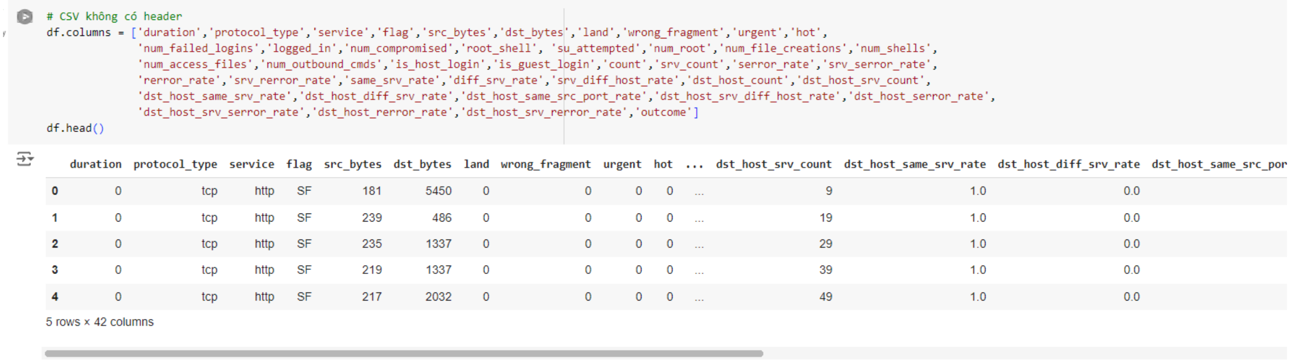
```
[3] import pandas as pd
    import numpy as np
    import tensorflow as tf
    from tensorflow import keras
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn import metrics

    from tensorflow.keras.utils import get_file
    try:
        path = get_file('kddcup.data_10_percent.gz', origin='http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz')
    except:
        print('Error downloading')
        raise

    print(path)
```

```
Downloading data from http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
2144903/2144903 [==============================] - 0s 0us/step
/root/.keras/datasets/kddcup.data_10_percent.gz
```



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

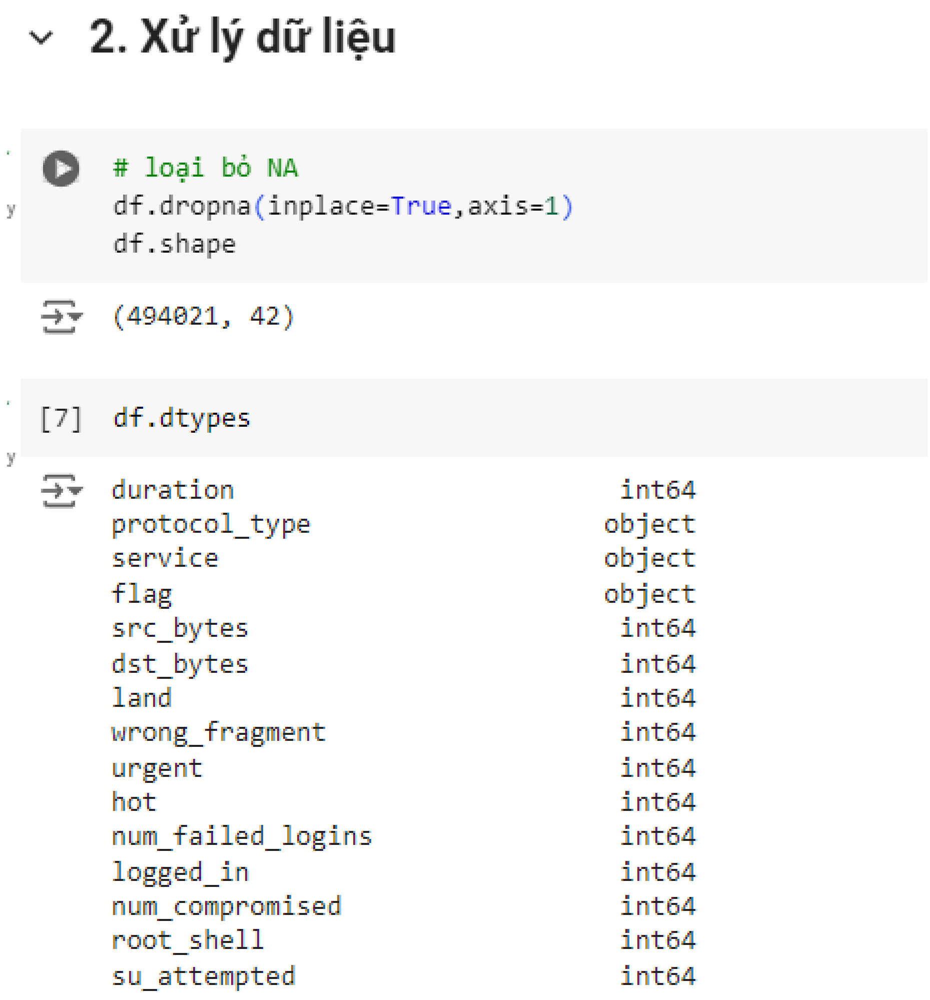
```
Read 494021 rows.
```



```
# CSV không có header
df.columns = ['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot',
              'num_failed_logins','logged_in','num_compromised','root_shell', 'su_attempted','num_root','num_file_creations','num_shells',
              'num_access_files','num_outbound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate',
              'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count',
              'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate',
              'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate','outcome']
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_por
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9	1.0	0.0	
1	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	0.0	
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	0.0	
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	0.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	0.0	

5 rows × 42 columns



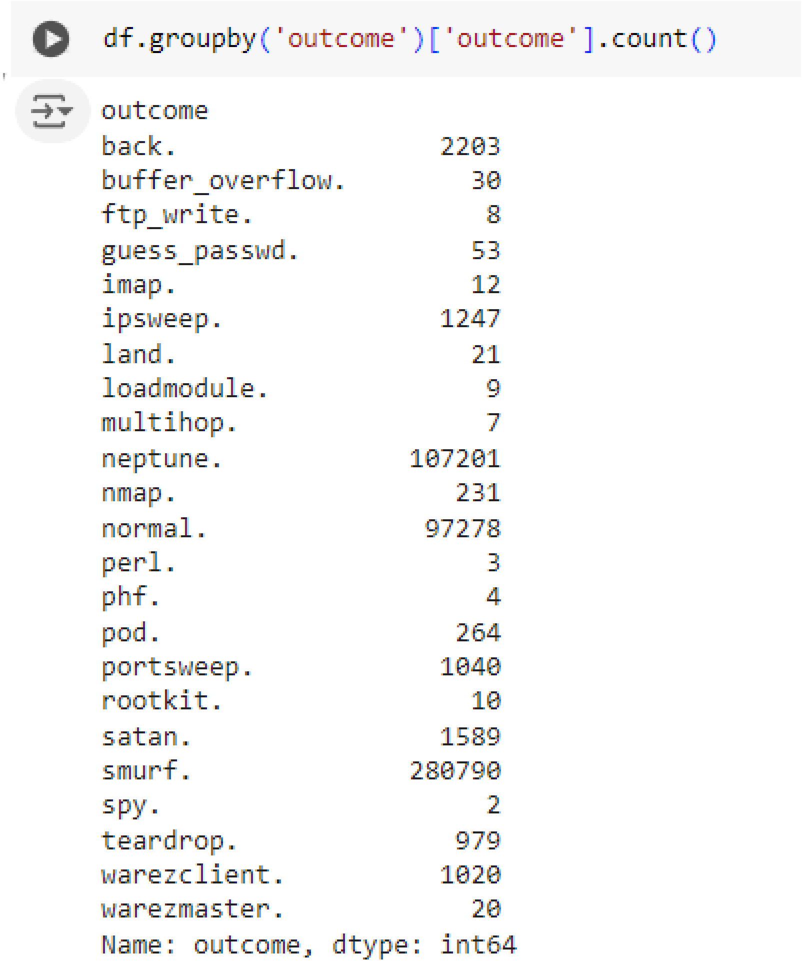
2. Xử lý dữ liệu

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

```
(494021, 42)
```

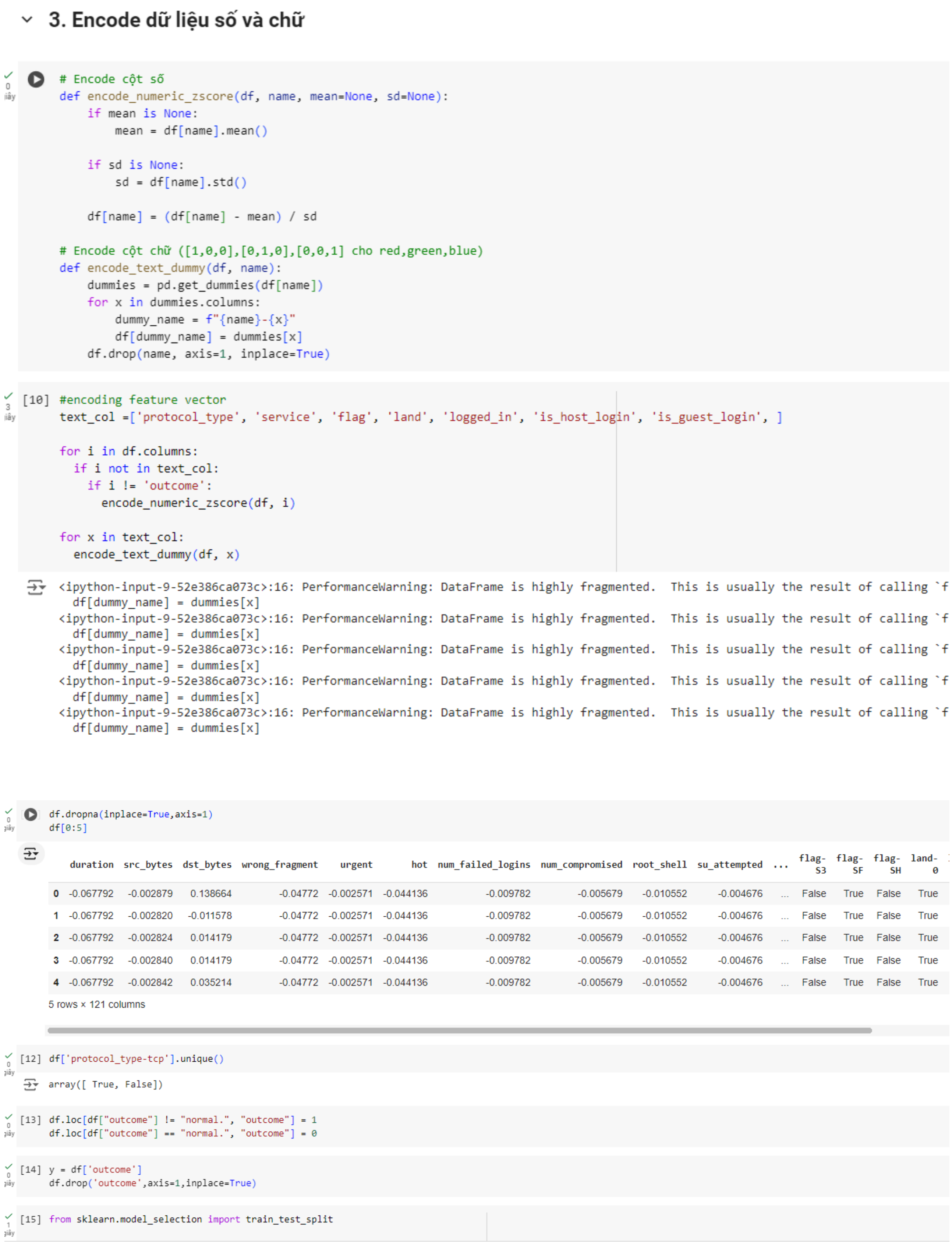
```
[7] df.dtypes
```

```
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
```



```
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)
```

```
[10] #encoding feature vector
text_col =['protocol_type', 'service', 'flag', 'land', 'logged_in', 'is_host_login', 'is_guest_login', ]

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-9-52e386ca073c>:16: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `f
  df[dummy_name] = dummies[x]
<ipython-input-9-52e386ca073c>:16: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `f
  df[dummy_name] = dummies[x]
<ipython-input-9-52e386ca073c>:16: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `f
  df[dummy_name] = dummies[x]
<ipython-input-9-52e386ca073c>:16: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `f
  df[dummy_name] = dummies[x]
<ipython-input-9-52e386ca073c>:16: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `f
  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_logins	num_compromised	root_shell	su_attempted	...	flag-S3	flag-SF	flag-SH	land-0
0	-0.067792	-0.002879	0.138664	-0.04772	-0.002571	-0.044136	-0.009782	-0.005679	-0.010552	-0.004676	...	False	True	False	True
1	-0.067792	-0.002820	-0.011578	-0.04772	-0.002571	-0.044136	-0.009782	-0.005679	-0.010552	-0.004676	...	False	True	False	True
2	-0.067792	-0.002824	0.014179	-0.04772	-0.002571	-0.044136	-0.009782	-0.005679	-0.010552	-0.004676	...	False	True	False	True
3	-0.067792	-0.002840	0.014179	-0.04772	-0.002571	-0.044136	-0.009782	-0.005679	-0.010552	-0.004676	...	False	True	False	True
4	-0.067792	-0.002842	0.035214	-0.04772	-0.002571	-0.044136	-0.009782	-0.005679	-0.010552	-0.004676	...	False	True	False	True

5 rows × 121 columns

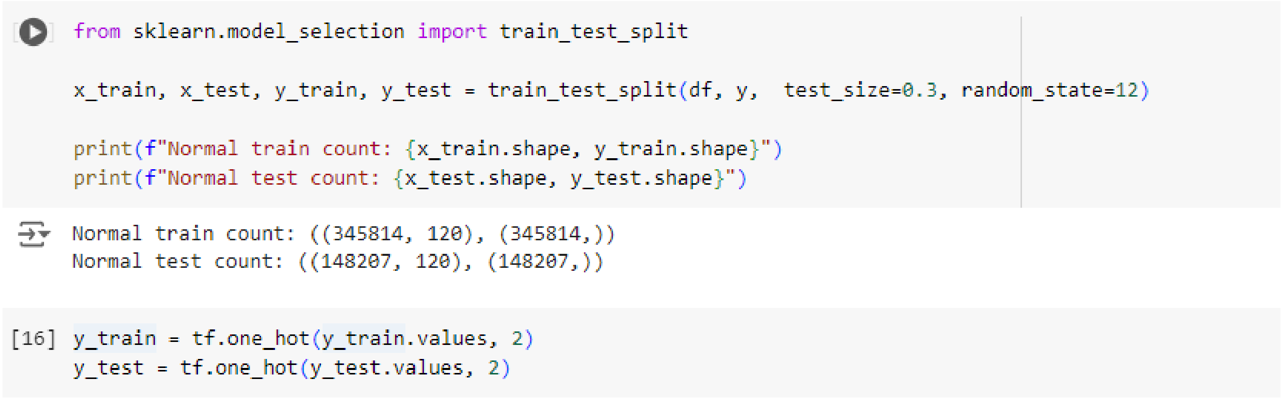
```
[12] df['protocol_type-tcp'].unique()
```

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

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

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

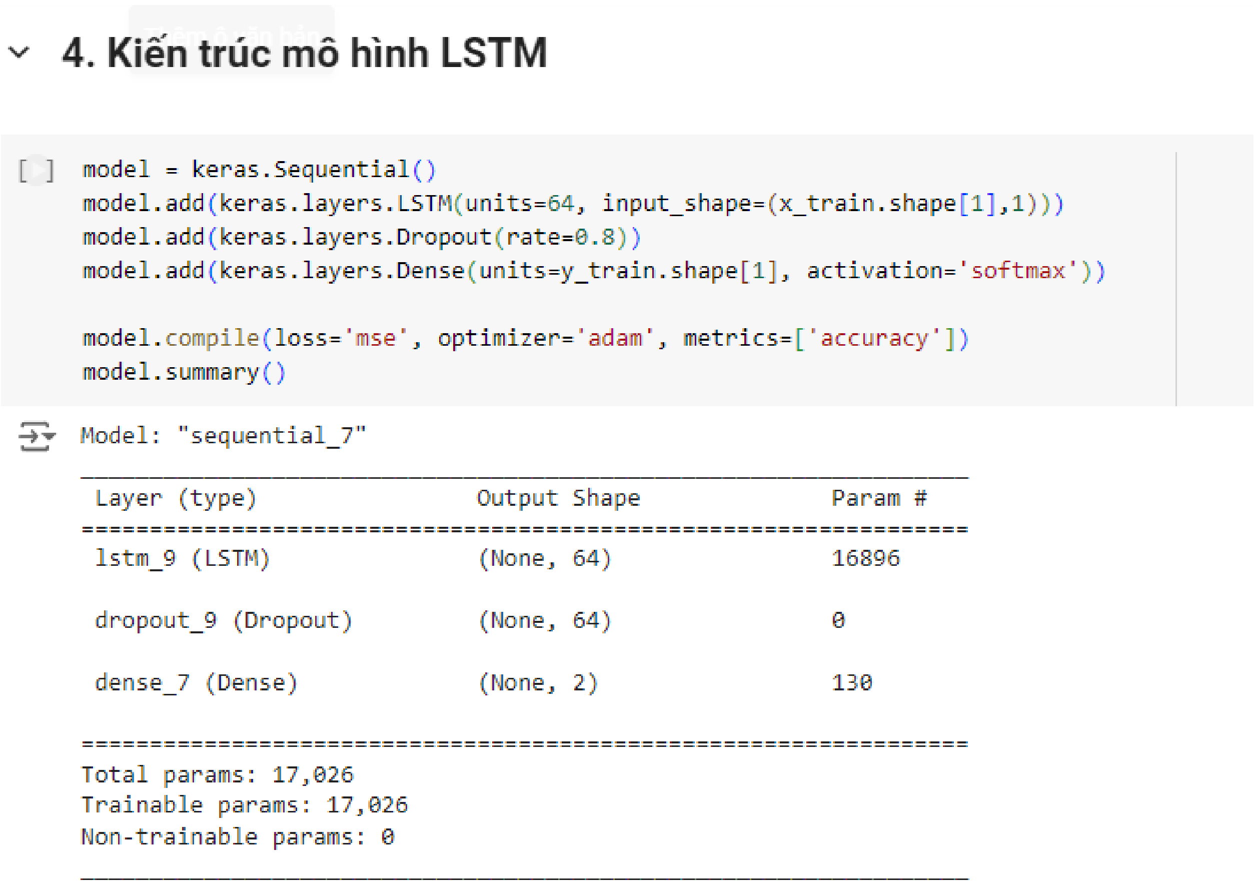
```
[15] from sklearn.model_selection import train_test_split
```



```
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,))
```

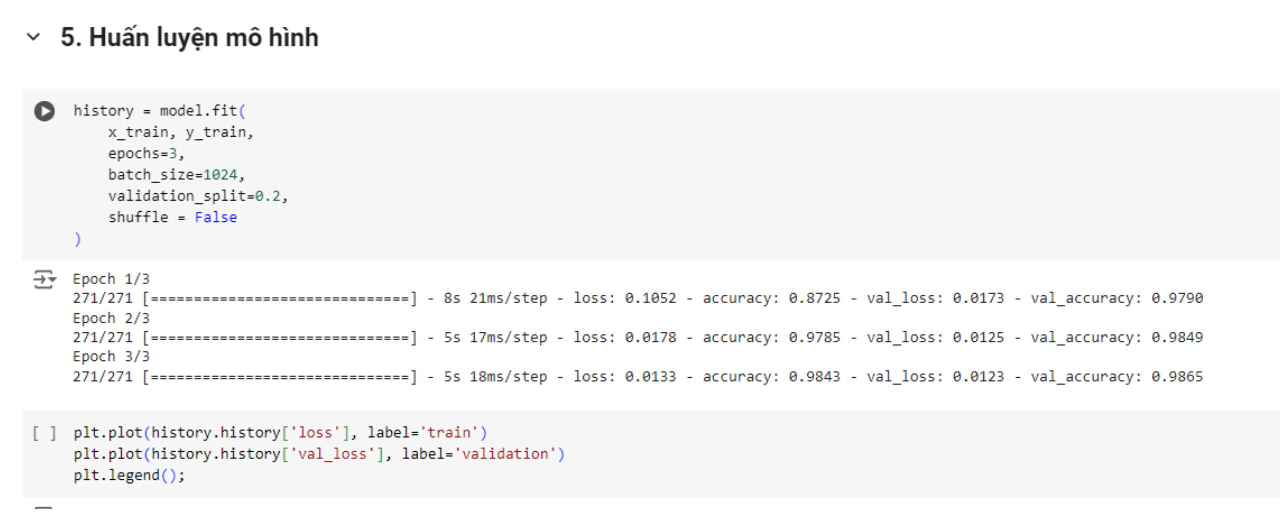
```
[16] y_train = tf.one_hot(y_train.values, 2)
     y_test = tf.one_hot(y_test.values, 2)
```



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_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_9 (LSTM)               (None, 64)                16896

 dropout_9 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 2)                 130

=================================================================
Total params: 17,026
Trainable params: 17,026
Non-trainable params: 0
_________________________________________________________________
```

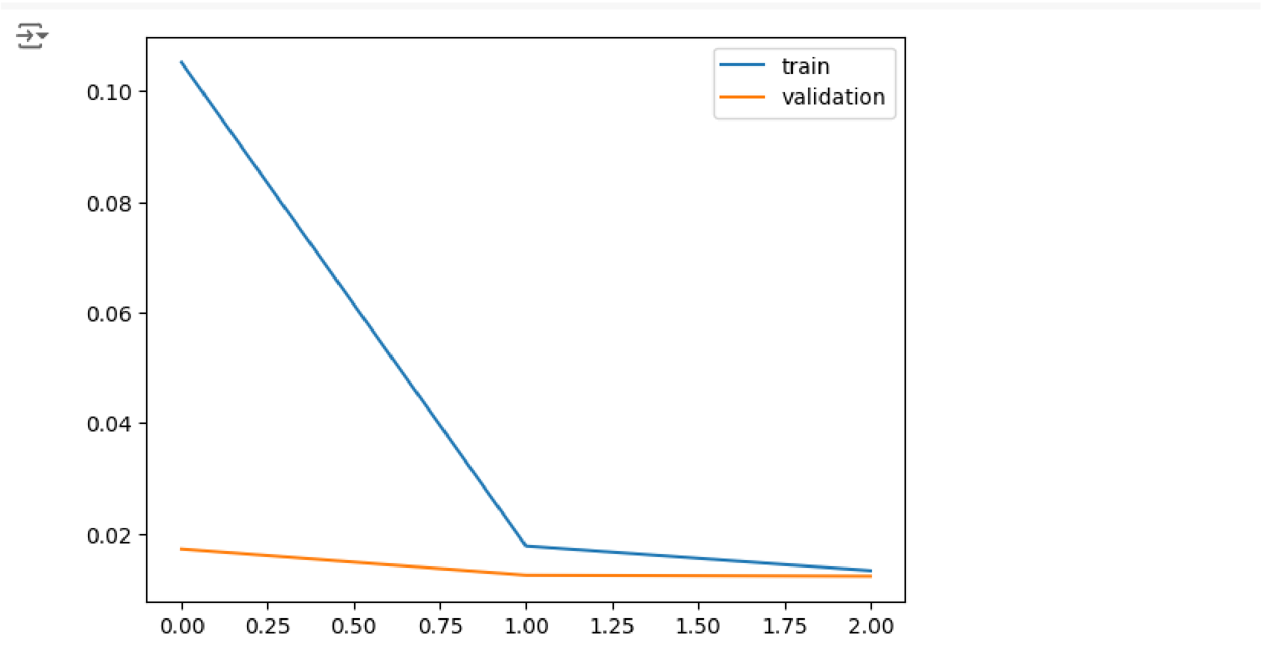


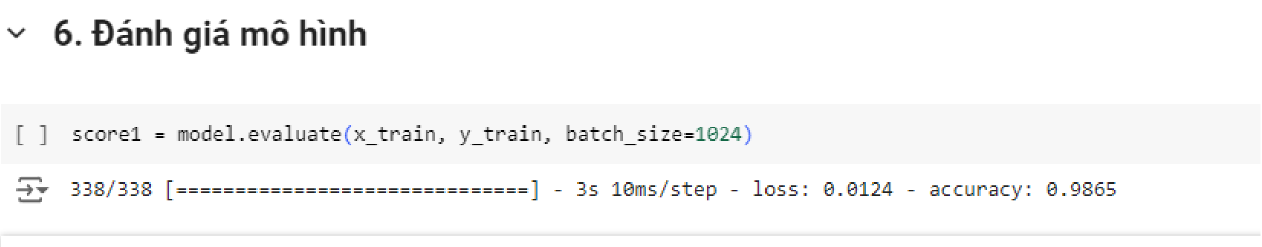
5. Huấn luyện mô hình

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

```
Epoch 1/3
271/271 [==============================] - 8s 21ms/step - loss: 0.1052 - accuracy: 0.8725 - val_loss: 0.0173 - val_accuracy: 0.9790
Epoch 2/3
271/271 [==============================] - 5s 17ms/step - loss: 0.0178 - accuracy: 0.9785 - val_loss: 0.0125 - val_accuracy: 0.9849
Epoch 3/3
271/271 [==============================] - 5s 18ms/step - loss: 0.0133 - accuracy: 0.9843 - val_loss: 0.0123 - val_accuracy: 0.9865
```

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





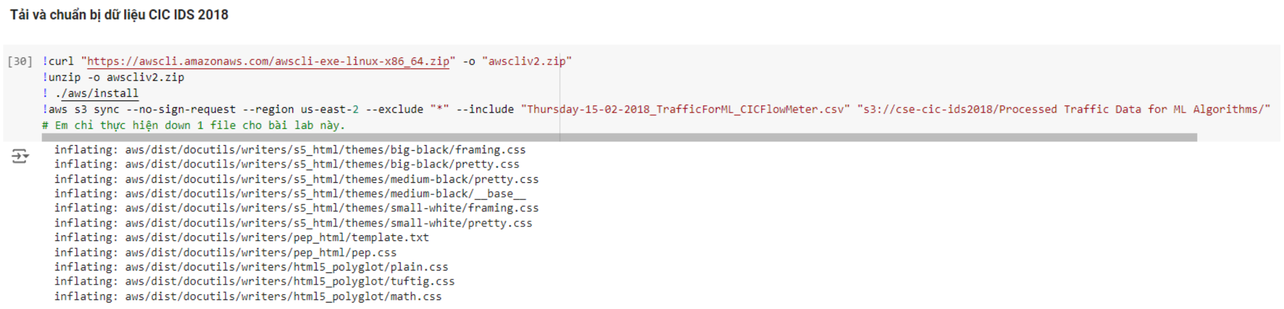
6. Đánh giá mô hình

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

```
338/338 [==============================] - 3s 10ms/step - loss: 0.0124 - accuracy: 0.9865
```

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.

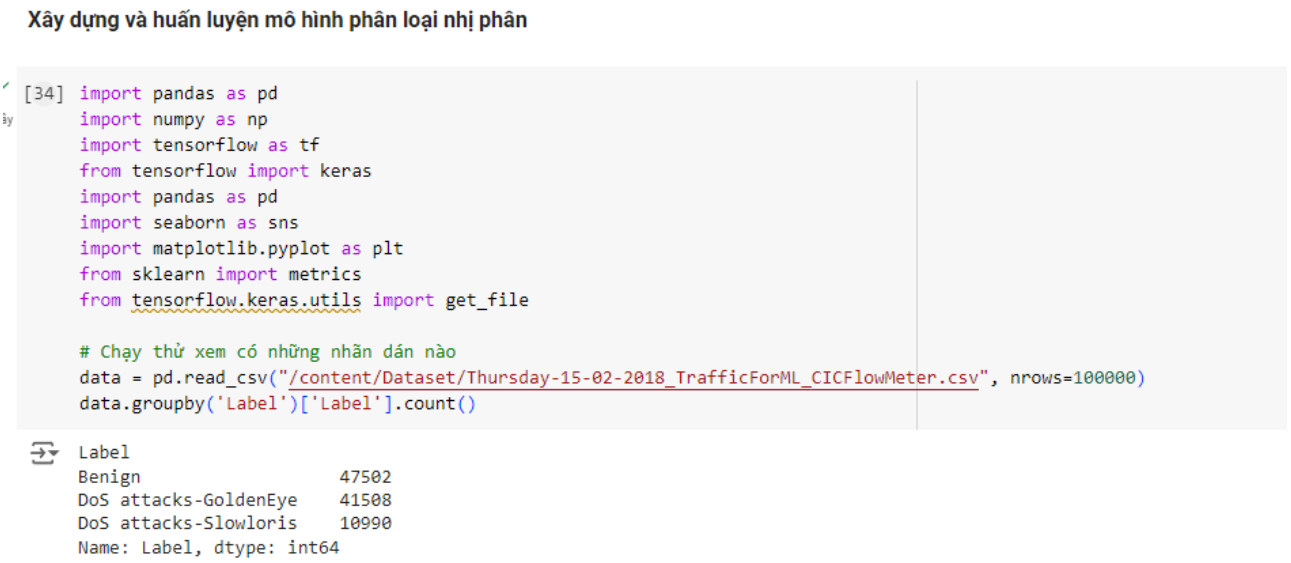
Binary Classification:



Tải và chuẩn bị dữ liệu CIC IDS 2018

```
[30] !curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
     !unzip -o awscliv2.zip
     ! ./aws/install
     !aws s3 sync --no-sign-request --region us-east-2 --exclude "*" --include "Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv" "s3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms/"
     # Em chỉ thực hiện down 1 file cho bài lab này.
```

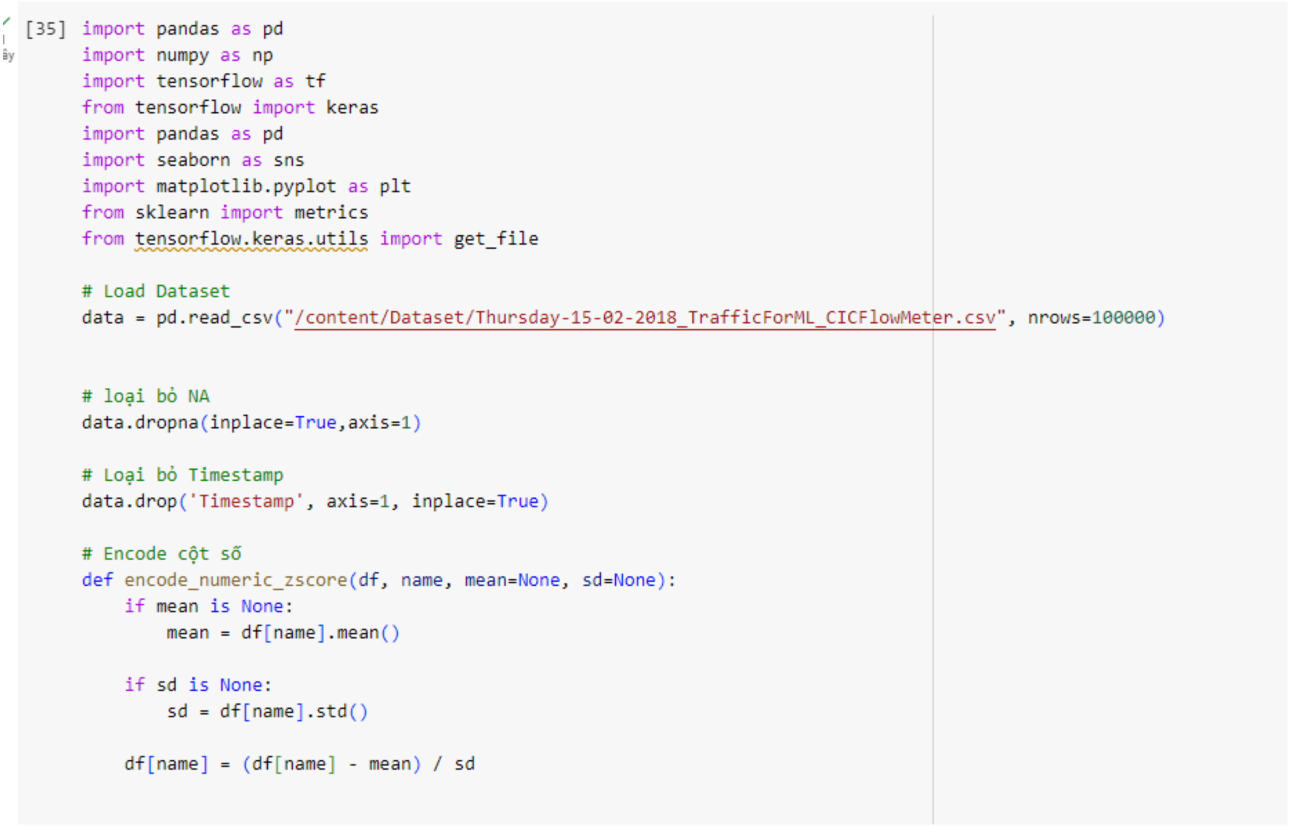
```
  inflating: aws/dist/docutils/writers/s5_html/themes/big-black/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/big-black/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-black/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-black/__base__
  inflating: aws/dist/docutils/writers/s5_html/themes/small-white/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/small-white/pretty.css
  inflating: aws/dist/docutils/writers/pep_html/template.txt
  inflating: aws/dist/docutils/writers/pep_html/pep.css
  inflating: aws/dist/docutils/writers/html5_polyglot/plain.css
  inflating: aws/dist/docutils/writers/html5_polyglot/tuftig.css
  inflating: aws/dist/docutils/writers/html5_polyglot/math.css
```



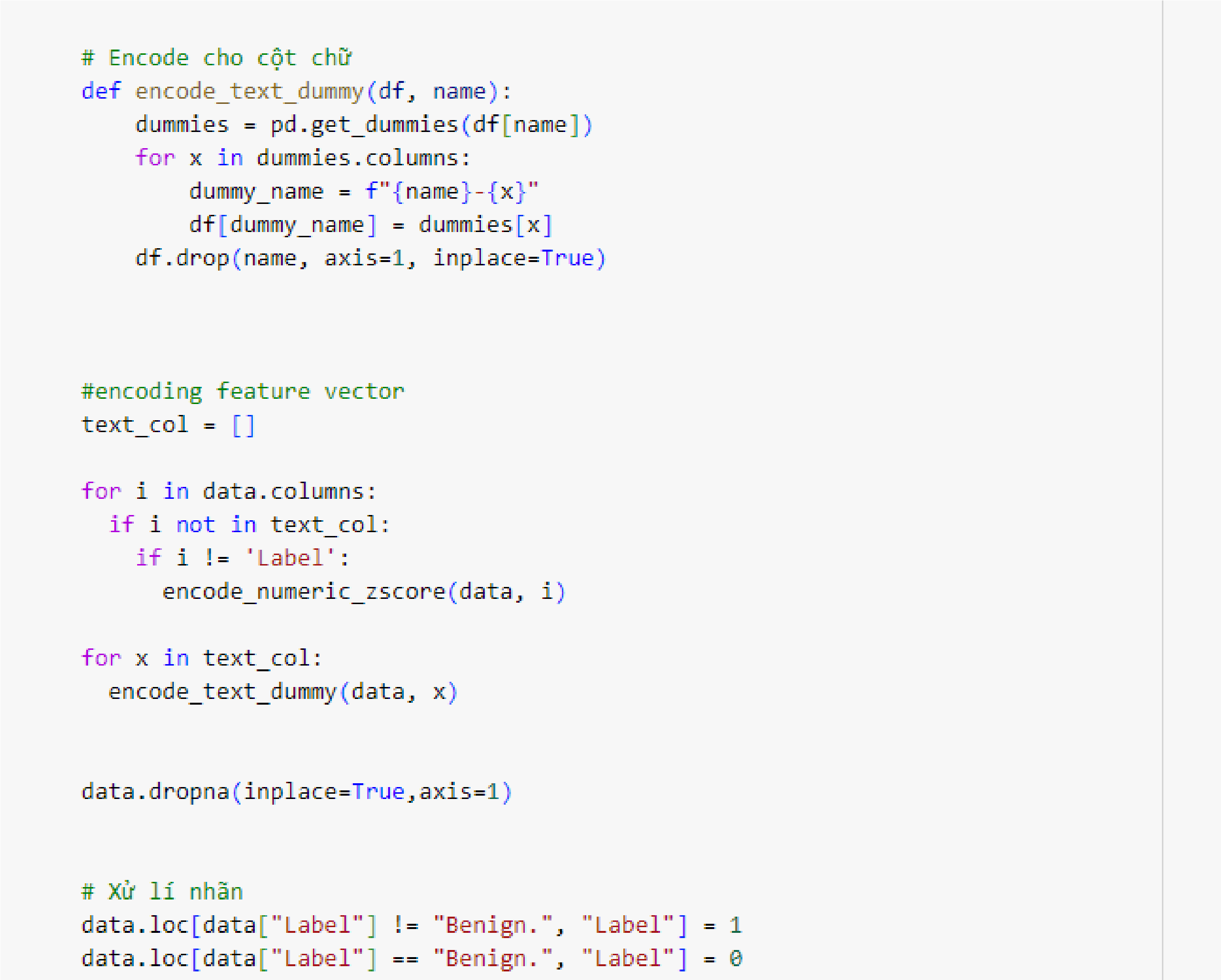
Xây dựng và huấn luyện mô hình phân loại nhị phân

```
[34] 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

     # Chạy thử xem có những nhãn dán nào
     data = pd.read_csv("/content/Dataset/Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv", nrows=100000)
     data.groupby('Label')['Label'].count()
```

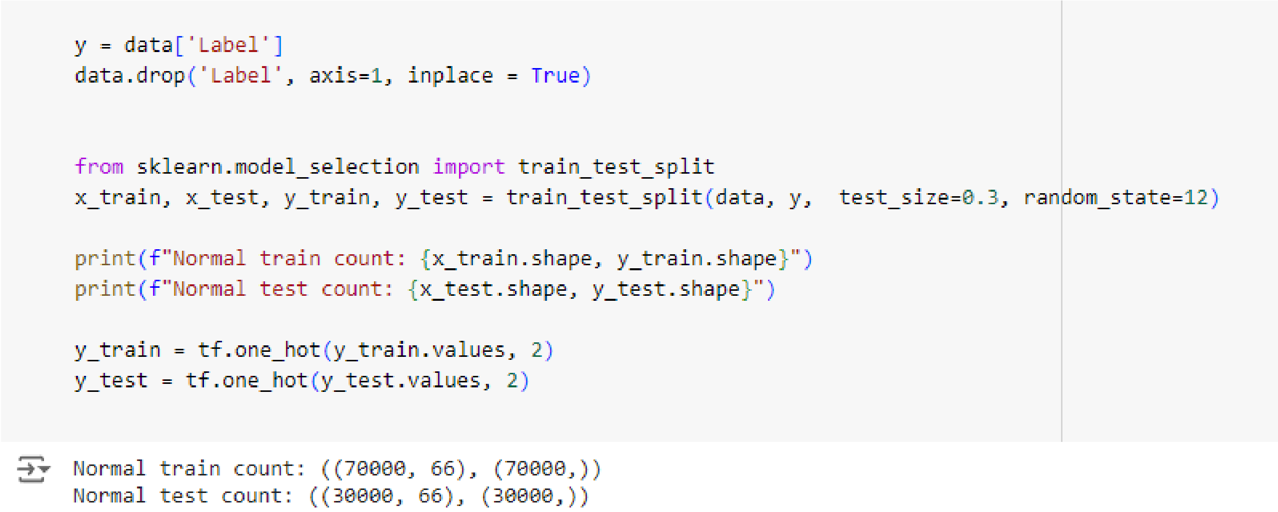
```
Label
Benign                  47502
DoS attacks-GoldenEye   41508
DoS attacks-Slowloris   10990
Name: Label, dtype: int64
```



```
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

# Load Dataset
data = pd.read_csv("/content/Dataset/Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv", nrows=100000)


# loại bỏ NA
data.dropna(inplace=True,axis=1)

# Loại bỏ Timestamp
data.drop('Timestamp', axis=1, inplace=True)

# 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
```

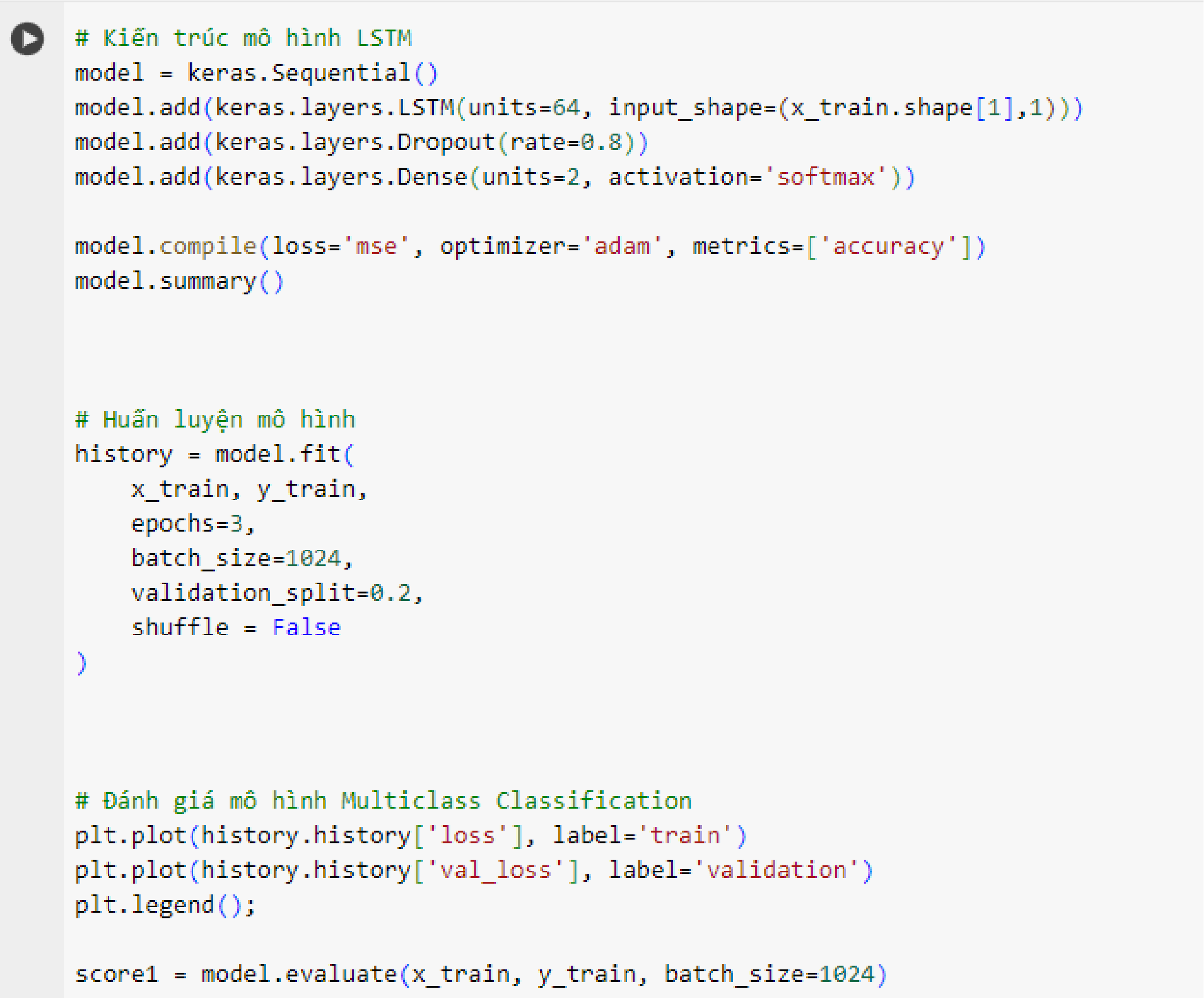


```
# Encode cho cột chữ
def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = f"{name}-{x}"
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)



#encoding feature vector
text_col = []

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

for x in text_col:
  encode_text_dummy(data, x)


data.dropna(inplace=True,axis=1)


# Xử lí nhãn
data.loc[data["Label"] != "Benign.", "Label"] = 1
data.loc[data["Label"] == "Benign.", "Label"] = 0
```

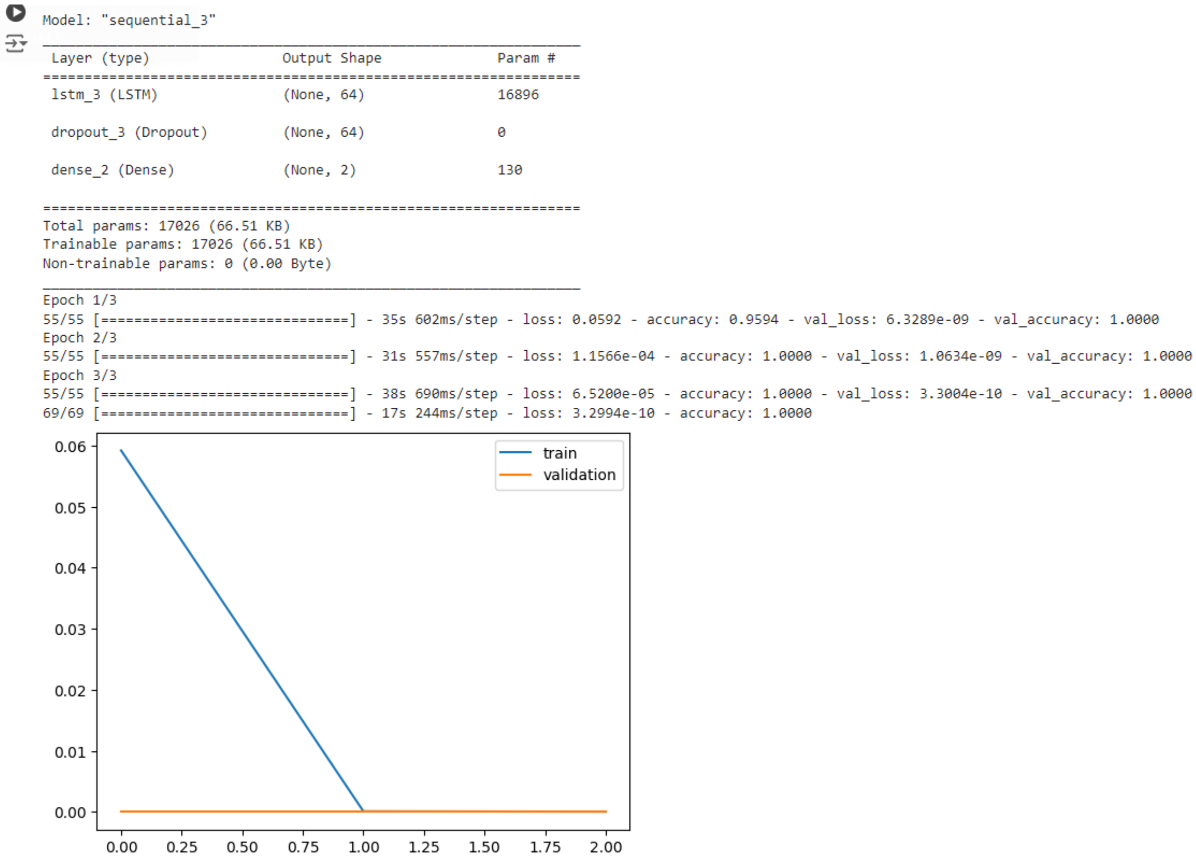


```
y = data['Label']
data.drop('Label', axis=1, inplace = True)


from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, 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}")

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

```
Normal train count: ((70000, 66), (70000,))
Normal test count: ((30000, 66), (30000,))
```

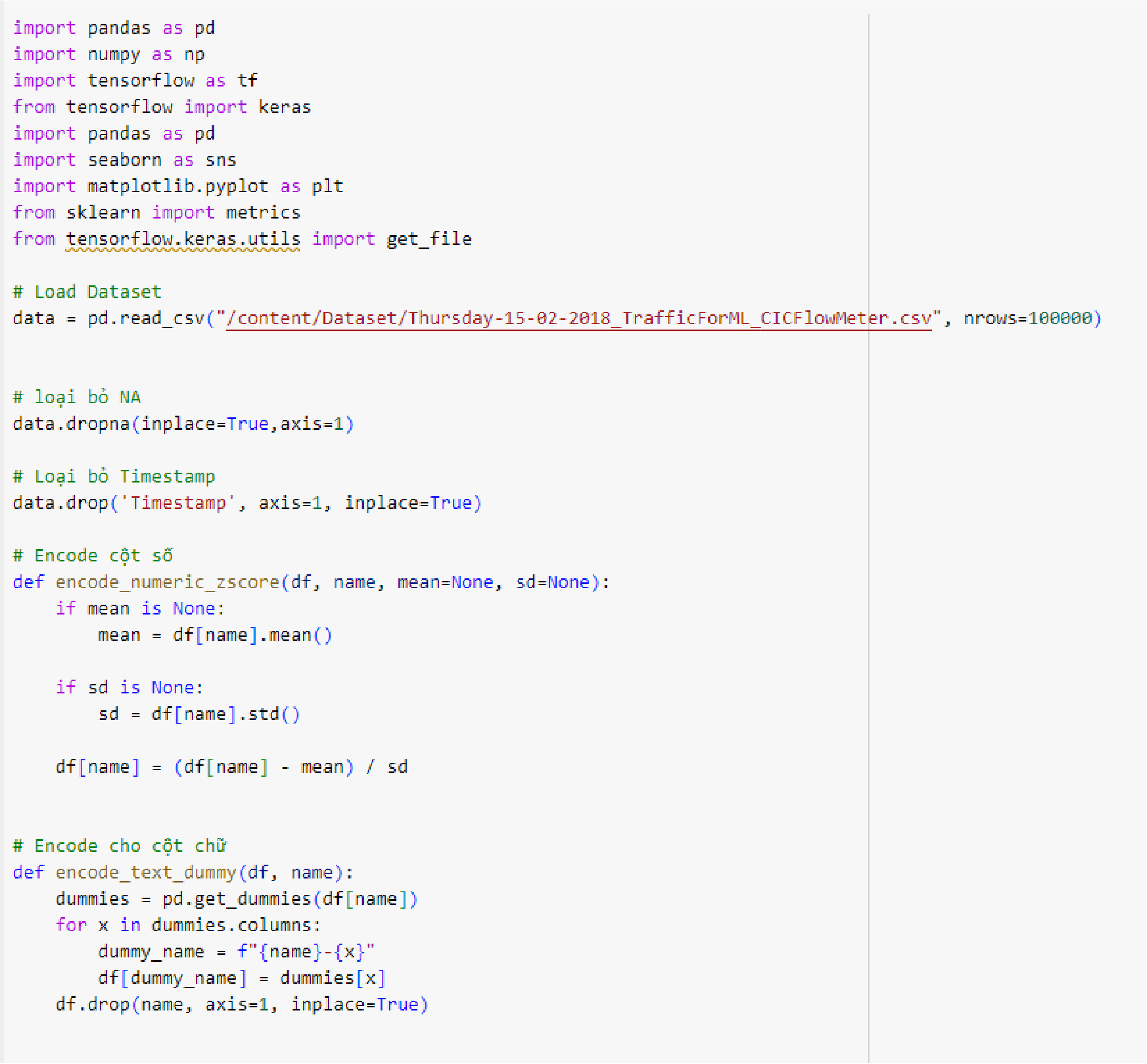


```
# Kiến trúc mô hình LSTM
model = keras.Sequential()
model.add(keras.layers.LSTM(units=64, input_shape=(x_train.shape[1],1)))
model.add(keras.layers.Dropout(rate=0.8))
model.add(keras.layers.Dense(units=2, activation='softmax'))

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



# Huấn luyện mô hình
history = model.fit(
    x_train, y_train,
    epochs=3,
    batch_size=1024,
    validation_split=0.2,
    shuffle = False
)



# Đánh giá mô hình Multiclass Classification
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend();

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

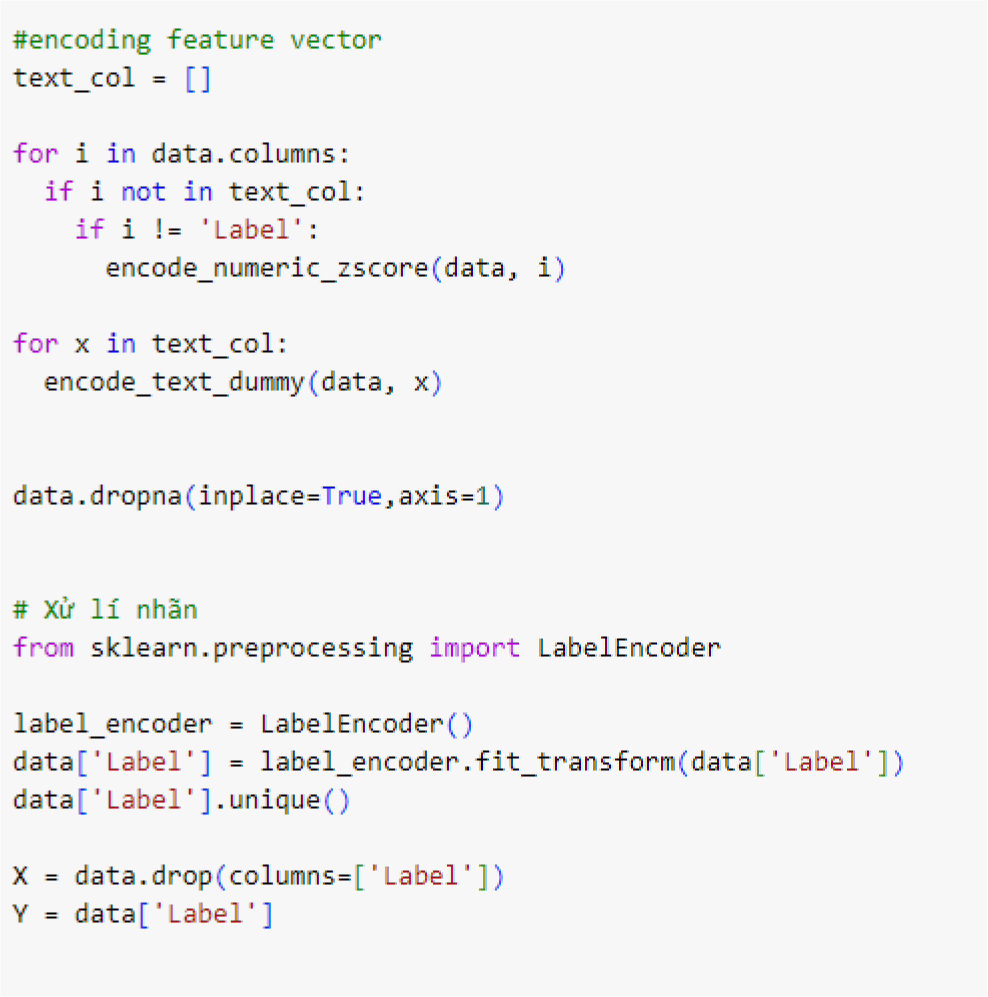


```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 64)                16896

 dropout_3 (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)
_________________________________________________________________
Epoch 1/3
55/55 [==============================] - 35s 602ms/step - loss: 0.0592 - accuracy: 0.9594 - val_loss: 6.3289e-09 - val_accuracy: 1.0000
Epoch 2/3
55/55 [==============================] - 31s 557ms/step - loss: 1.1566e-04 - accuracy: 1.0000 - val_loss: 1.0634e-09 - val_accuracy: 1.0000
Epoch 3/3
55/55 [==============================] - 38s 690ms/step - loss: 6.5200e-05 - accuracy: 1.0000 - val_loss: 3.3004e-10 - val_accuracy: 1.0000
69/69 [==============================] - 17s 244ms/step - loss: 3.2994e-10 - accuracy: 1.0000
```

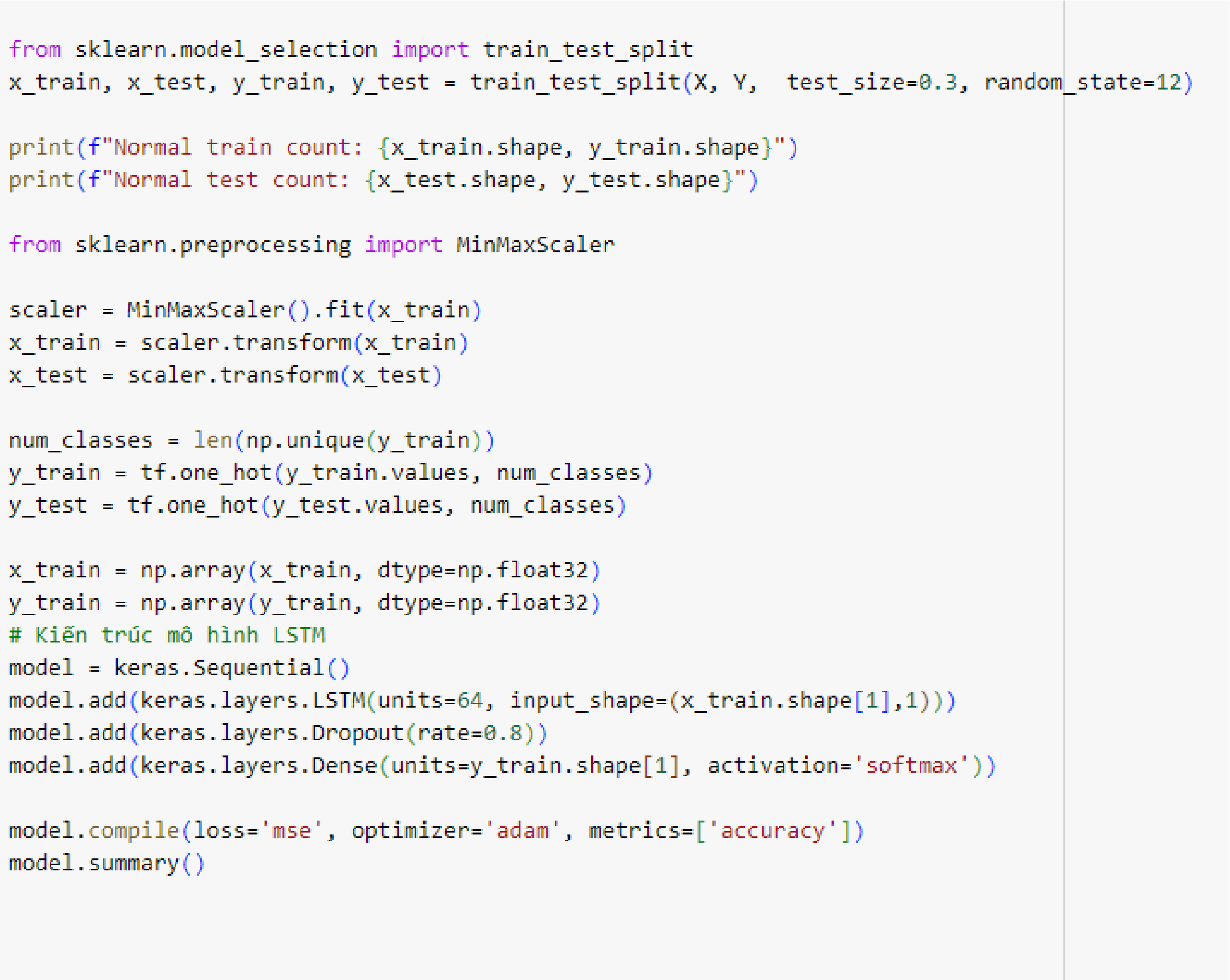
Multiclass Classification:



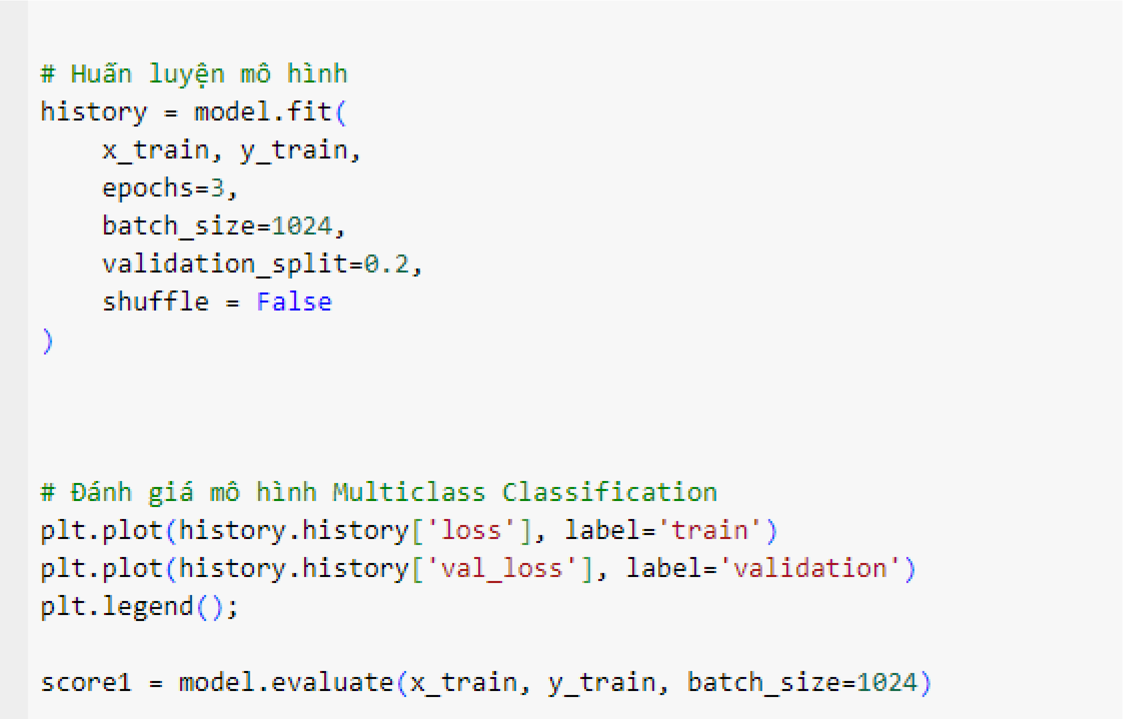
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from tensorflow.keras.utils import get_file

# Load Dataset
data = pd.read_csv("/content/Dataset/Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv", nrows=100000)


# loại bỏ NA
data.dropna(inplace=True,axis=1)

# Loại bỏ Timestamp
data.drop('Timestamp', axis=1, inplace=True)

# 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 cho cột chữ
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)
```



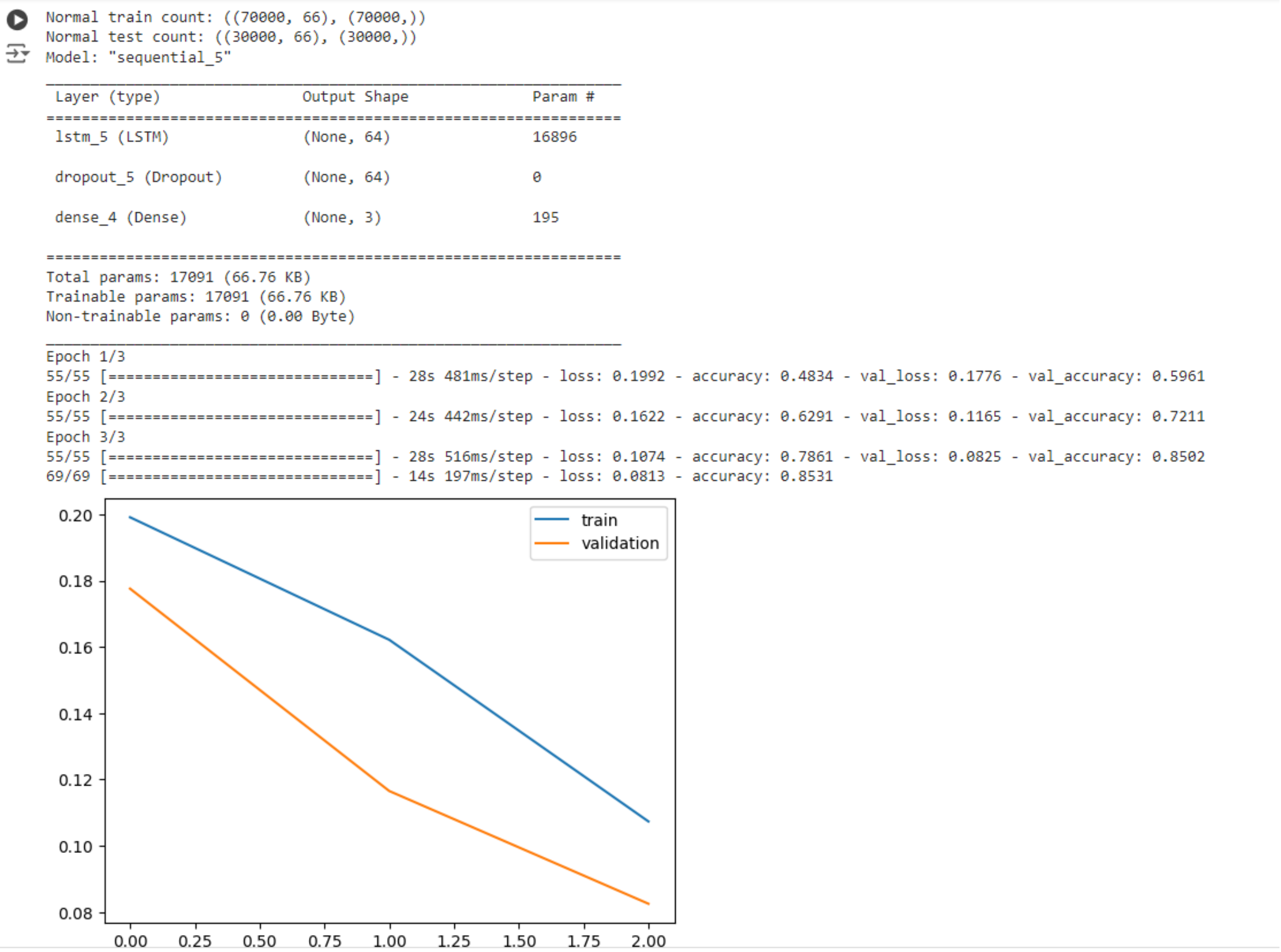
```
#encoding feature vector
text_col = []

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

for x in text_col:
  encode_text_dummy(data, x)


data.dropna(inplace=True,axis=1)


# Xử lí nhãn
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['Label'] = label_encoder.fit_transform(data['Label'])
data['Label'].unique()

X = data.drop(columns=['Label'])
Y = data['Label']
```



```python
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=12)

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

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler().fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

num_classes = len(np.unique(y_train))
y_train = tf.one_hot(y_train.values, num_classes)
y_test = tf.one_hot(y_test.values, num_classes)

x_train = np.array(x_train, dtype=np.float32)
y_train = np.array(y_train, dtype=np.float32)
# 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()
```



```python
# Huấn luyện mô hình
history = model.fit(
    x_train, y_train,
    epochs=3,
    batch_size=1024,
    validation_split=0.2,
    shuffle = False
)



# Đánh giá mô hình Multiclass Classification
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend();

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



```
Normal train count: ((70000, 66), (70000,))
Normal test count: ((30000, 66), (30000,))
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_5 (LSTM)               (None, 64)                16896     
                                                                 
 dropout_5 (Dropout)         (None, 64)                0         
                                                                 
 dense_4 (Dense)             (None, 3)                 195       
                                                                 
=================================================================
Total params: 17091 (66.76 KB)
Trainable params: 17091 (66.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/3
55/55 [==============================] - 28s 481ms/step - loss: 0.1992 - accuracy: 0.4834 - val_loss: 0.1776 - val_accuracy: 0.5961
Epoch 2/3
55/55 [==============================] - 24s 442ms/step - loss: 0.1622 - accuracy: 0.6291 - val_loss: 0.1165 - val_accuracy: 0.7211
Epoch 3/3
55/55 [==============================] - 28s 516ms/step - loss: 0.1074 - accuracy: 0.7861 - val_loss: 0.0825 - val_accuracy: 0.8502
69/69 [==============================] - 14s 197ms/step - loss: 0.0813 - accuracy: 0.8531
```