

21520155 - Nguyễn Triệu Thiên Bảo

21521195 - Trần Lê Minh Ngọc

LAB 4.1

A. Phát hiện trang web lừa đảo bằng kỹ thuật học máy

(Phishing Website Detection by Machine Learning Techniques)

1. Mục tiêu:

Website lừa đảo (phishing website) là một tấn công phi kỹ thuật (social engineering) bằng cách bắt trước các URL tin cậy. Mục tiêu bài thực hành sẽ đào tạo mô hình học máy và học sâu trên tập dữ liệu để dự đoán website lừa đảo. Cả 2 URL lừa đảo và lành tính (benign) được trích xuất thuộc tính từ thành phần cơ bản của website thành tập dữ liệu. Dataset sử dụng trong Lab:

<https://drive.google.com/drive/folders/1vD0i34RVMFKXdvDmnZmejmtQ-tNm5xcB?usp=sharing>

2. Tải dữ liệu

Các tính năng được trích xuất và lưu trữ trong tập tin csv.

```
from google.colab import drive
drive.mount('/content/drive')
data_dir = '/content/drive/My Drive/PPHM/Lab4/'
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
#import packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Loading the data
data0 = pd.read_csv(data_dir + 'urldata.csv')
data0.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	Ti
0	graphicriver.net	0	0	1	1	0	0	
1	ecnavi.jp	0	0	1	1	1	0	
2	hubpages.com	0	0	1	1	0	0	
3	extratorrent.cc	0	0	1	3	0	0	
4	icicibank.com	0	0	1	3	0	0	

3. Làm quen với dữ liệu

Dùng Dataframe để xem qua dữ liệu.

```
#Check shape
data0.shape
```

```
(10000, 18)
```

```
#List features
data0.columns

Index(['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth',
      'Redirection', 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record',
      'Web_Traffic', 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over',
      'Right_Click', 'Web_Forwards', 'Label'],
      dtype='object')
```

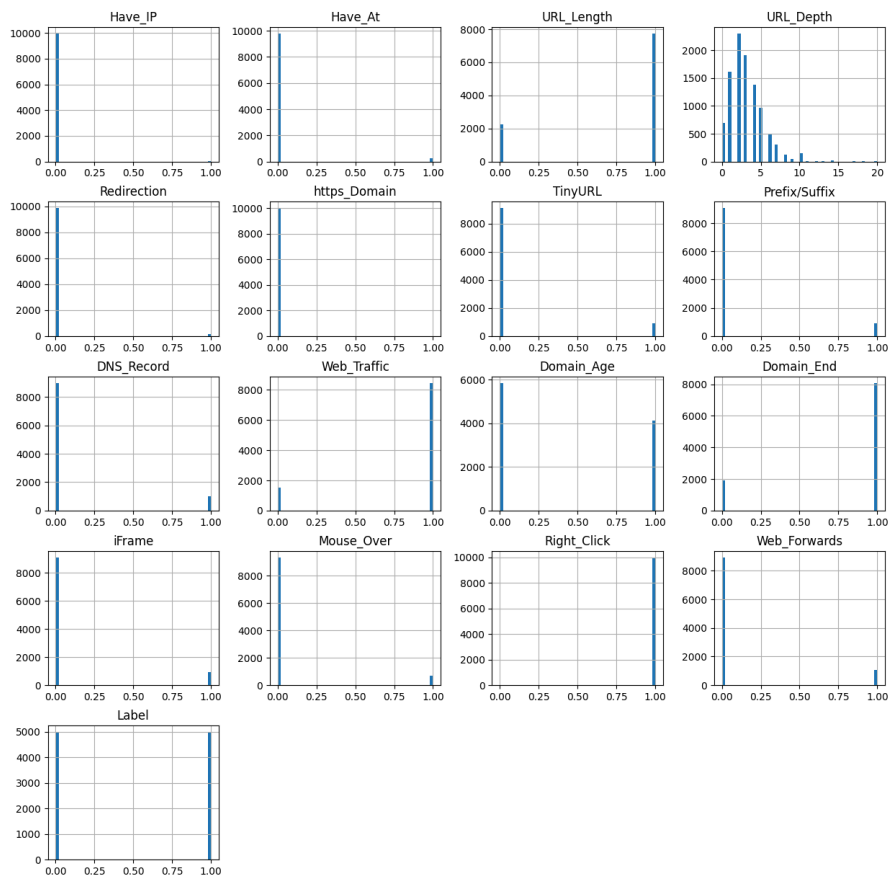
```
#Information
data0.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Domain           10000 non-null   object
1   Have_IP          10000 non-null   int64
2   Have_At          10000 non-null   int64
3   URL_Length       10000 non-null   int64
4   URL_Depth        10000 non-null   int64
5   Redirection      10000 non-null   int64
6   https_Domain     10000 non-null   int64
7   TinyURL          10000 non-null   int64
8   Prefix/Suffix    10000 non-null   int64
9   DNS_Record       10000 non-null   int64
10  Web_Traffic       10000 non-null   int64
11  Domain_Age        10000 non-null   int64
12  Domain_End        10000 non-null   int64
13  iFrame           10000 non-null   int64
14  Mouse_Over        10000 non-null   int64
15  Right_Click       10000 non-null   int64
16  Web_Forwards      10000 non-null   int64
17  Label            10000 non-null   int64
dtypes: int64(17), object(1)
memory usage: 1.4+ MB
```

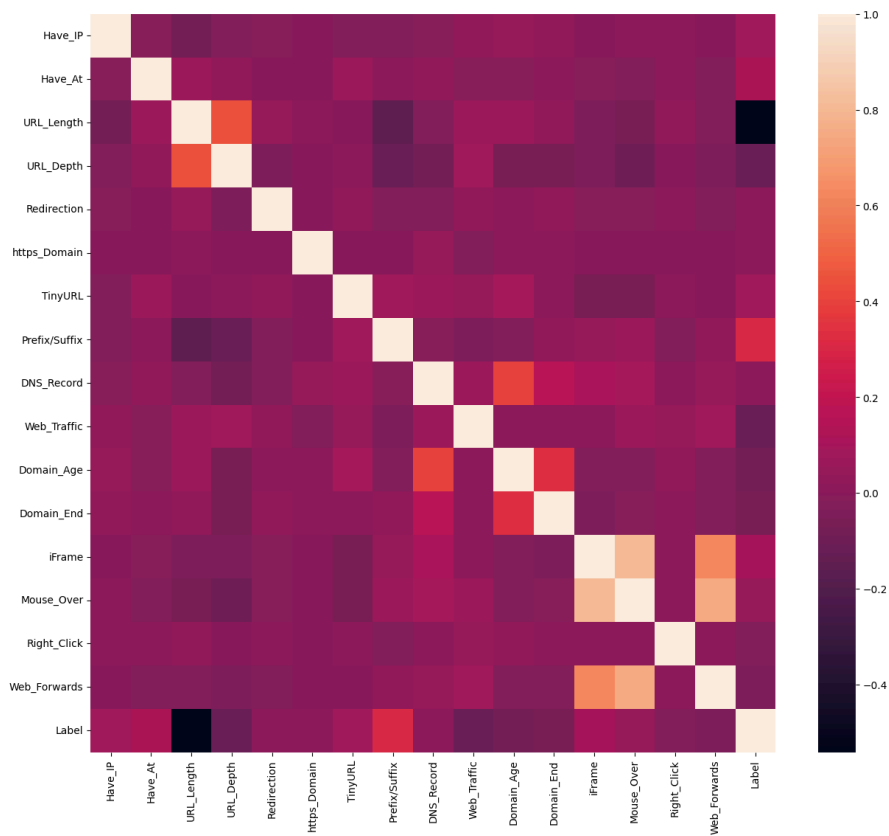
✓ 4. Trực quan hóa dữ liệu

Một số biểu đồ hiển thị để tìm các phân phối dữ liệu và cách mỗi liên quan với nhau giữa các đối tượng.

```
#Plotting the data distribution
data0.hist(bins = 50,figsize = (15,15))
plt.show()
```



```
#Correlation heatmap
plt.figure(figsize=(15,13))
sns.heatmap(data0.corr(numeric_only = True))
plt.show()
```



✓ 5. Tiền xử lý dữ liệu & EDA

Làm sạch dữ liệu bằng cách áp dụng các kỹ thuật tiền xử lý dữ liệu và chuyển đổi dữ liệu để sử dụng nó trong các mô hình.

```
data0.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000

Kết quả thu được ở trên cho thấy rằng hầu hết dữ liệu được tạo từ 0 và 1 ngoại trừ cột 'Domain' & 'URL_Depth'. cột Domain hoàn toàn không có ý nghĩa gì để đào tạo mô hình nên loại nó.

```
#Drop Domain column
data = data0.drop(['Domain'], axis = 1).copy()
```

Ta còn lại 16 thuộc tính và 1 cột phân loại. Giá trị tối đa của 'URL_Depth' là 20. Ta cần chuẩn hoá cột này.

```
#check data null
data.isnull().sum()
```

```
Have_IP      0
Have_At      0
URL_Length   0
URL_Depth    0
Redirection  0
https_Domain 0
TinyURL      0
Prefix/Suffix 0
DNS_Record   0
Web_Traffic  0
Domain_Age   0
Domain_End   0
iFrame       0
Mouse_Over   0
Right_Click  0
Web_Forwards 0
Label        0
dtype: int64
```

Trong tập tin thuộc tính được trích xuất, các thuộc tính được trích xuất của tập url lừa đảo và lành tính không có bất kỳ xáo trộn nào.

```
# shuffling the rows
data = data.sample(frac=1).reset_index(drop=True)
data.head()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/S
0	0	0	0	0	0	0	0	
1	0	0	1	7	0	0	0	
2	0	0	1	3	0	0	0	
3	0	0	0	1	0	0	0	
4	0	0	1	3	0	0	0	

✓ 6. Chia dữ liệu

```
#features and target columns to X & y
y = data['Label']
X = data.drop('Label',axis=1)
X.shape, y.shape

((10000, 16), (10000,))

# train and test sets: 80-20 split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2, random_state = 12)
X_train.shape, X_test.shape

((8000, 16), (2000, 16))
```

✓ 7. Đào tạo mô hình học máy

Dữ liệu phù hợp với việc học có giám sát và hai bài toán nên áp dụng là classification - phân loại và regression - hồi quy. Ta sẽ có 2 nhãn phishing (1) và legitimate (0), với các mô hình học giám sát sau:

- Decision Tree
- Random Forest
- Multilayer Perceptrons
- XGBoost
- Autoencoder Neural Network
- Support Vector Machines

```
from sklearn.metrics import accuracy_score

# Creating model results
ML_Model = []
acc_train = []
acc_test = []

#storing the results
def storeResults(model, a,b):
    ML_Model.append(model)
    acc_train.append(round(a, 3))
    acc_test.append(round(b, 3))
```

✓ 7.1 Decision Tree Classifier

```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate
tree = DecisionTreeClassifier(max_depth = 5)
# fit
tree.fit(X_train, y_train)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

```
#predicting
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
```

Đánh giá hiệu suất

```
#computing the accuracy
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
```

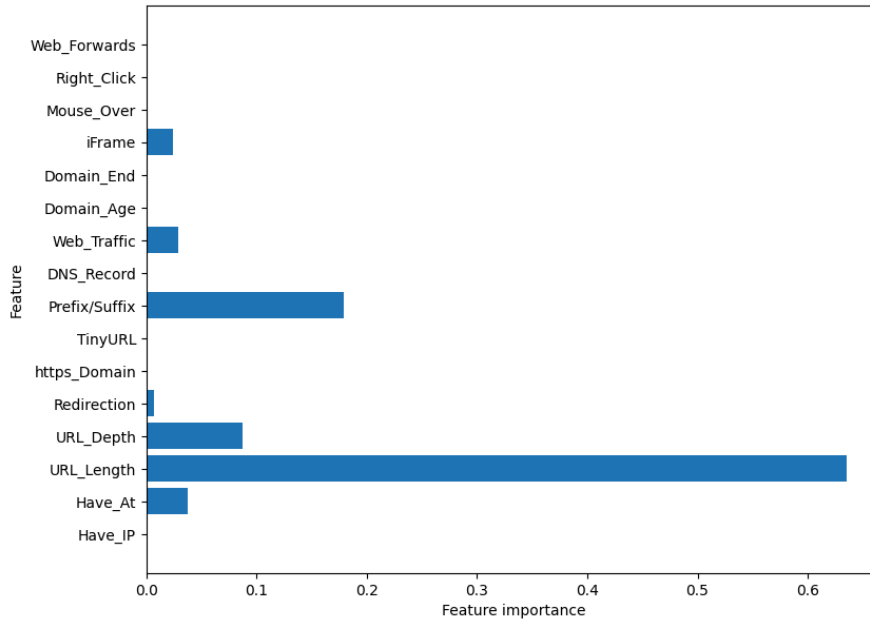
```
print(Decision Tree: Accuracy on test Data: {:.3f} .format(acc_test_tree))
```

Decision Tree: Accuracy on training Data: 0.812

Decision Tree: Accuracy on test Data: 0.810

#checking the feature importance in the model

```
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), tree.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```



Lưu trữ kết quả

```
storeResults('Decision Tree', acc_train_tree, acc_test_tree)
```

✓ 7.2 Random Forest Classifier

```
# Random Forest model
from sklearn.ensemble import RandomForestClassifier
```

```
# instantiate
forest = RandomForestClassifier(max_depth=5)
```

```
# fit
forest.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5)
```

```
#predicting
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
```

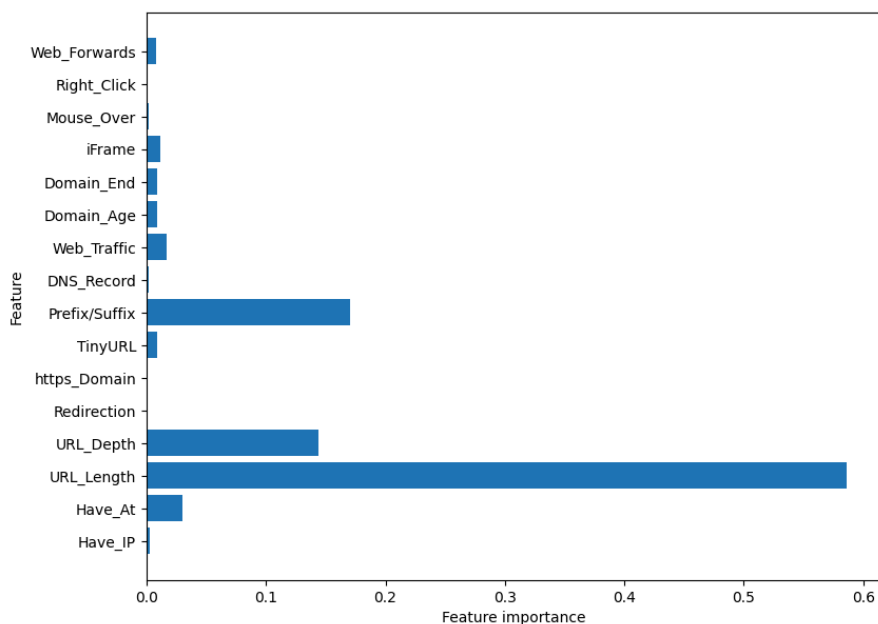
Đánh giá hiệu suất

```
#computing the accuracy
acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)

print("Random forest: Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random forest: Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random forest: Accuracy on training Data: 0.814
Random forest: Accuracy on test Data: 0.811
```

```
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), forest.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```



Lưu trữ kết quả

```
storeResults('Random Forest', acc_train_forest, acc_test_forest)
```

✓ 7.3. Multilayer Perceptrons (MLPs) - Học sâu

```
# Multilayer Perceptrons model
from sklearn.neural_network import MLPClassifier

# instantiate
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=([100,100,100]))

# fit
mlp.fit(X_train, y_train)
```

```
MLPClassifier(alpha=0.001, hidden_layer_sizes=[100, 100, 100])
```



```
#predicting
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)
```

Đánh giá hiệu suất

```
#computing the accuracy
acc_train_mlp = accuracy_score(y_train,y_train_mlp)
acc_test_mlp = accuracy_score(y_test,y_test_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc_test_mlp))

Multilayer Perceptrons: Accuracy on training Data: 0.864
Multilayer Perceptrons: Accuracy on test Data: 0.862
```

Lưu trữ kết quả

```
storeResults('Multilayer Perceptrons', acc_train_mlp, acc_test_mlp)
```

✓ 7.4. XGBoost Classifier

```
#XGBoost Classification model
from xgboost import XGBClassifier

# instantiate
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)
#fit
xgb.fit(X_train, y_train)
```

```
▼ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.4, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=7, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, random_state=None, ...)
```

```
#predicting
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)
```

Đánh giá hiệu suất

```
#computing the accuracy
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

XGBoost: Accuracy on training Data: 0.867
XGBoost : Accuracy on test Data: 0.861
```

Lưu trữ kết quả

```
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)
```

✓ 7.5. Autoencoder Neural Network

Bài tập số 1: Hoàn thành huấn luyện mô hình ANN.

```
#importing required packages
import keras
from keras.layers import Input, Dense
from keras import regularizers
import tensorflow as tf
from keras.models import Model
from sklearn import metrics

#building autoencoder model

input_dim = X_train.shape[1]
encoding_dim = input_dim

input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="relu",
                activity_regularizer=regularizers.l1(10e-4))(input_layer)
encoder = Dense(int(encoding_dim), activation="relu")(encoder)

encoder = Dense(int(encoding_dim-2), activation="relu")(encoder)
code = Dense(int(encoding_dim-4), activation='relu')(encoder)
decoder = Dense(int(encoding_dim-2), activation='relu')(code)

decoder = Dense(int(encoding_dim), activation='relu')(encoder)
decoder = Dense(input_dim, activation='sigmoid')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.summary()

Model: "model_10"

```

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 16)]	0
dense_70 (Dense)	(None, 16)	272
dense_71 (Dense)	(None, 16)	272
dense_72 (Dense)	(None, 14)	238
dense_75 (Dense)	(None, 16)	240
dense_76 (Dense)	(None, 16)	272

```

Total params: 1294 (5.05 KB)
Trainable params: 1294 (5.05 KB)
Non-trainable params: 0 (0.00 Byte)

#compiling the model
autoencoder.compile(optimizer='adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

#Training the model
history = autoencoder.fit(X_train, X_train, epochs=10, batch_size=64, shuffle=False, validation_split=0.2)

Epoch 1/10
100/100 [=====] - 2s 5ms/step - loss: 0.1252 - accuracy: 0.3469 - val_loss: -1.3938 - val_accuracy: 0.8494
Epoch 2/10
100/100 [=====] - 0s 3ms/step - loss: -23.1541 - accuracy: 0.8428 - val_loss: -82.0420 - val_accuracy: 0.8494
Epoch 3/10
100/100 [=====] - 0s 3ms/step - loss: -391.7335 - accuracy: 0.8428 - val_loss: -970.9565 - val_accuracy: 0.849
Epoch 4/10
100/100 [=====] - 0s 5ms/step - loss: -2759.9543 - accuracy: 0.8428 - val_loss: -5498.3530 - val_accuracy: 0.8
Epoch 5/10
100/100 [=====] - 0s 4ms/step - loss: -11686.4062 - accuracy: 0.8428 - val_loss: -19985.5195 - val_accuracy: 0
Epoch 6/10
100/100 [=====] - 0s 5ms/step - loss: -35712.4375 - accuracy: 0.8428 - val_loss: -55051.9453 - val_accuracy: 0
Epoch 7/10
100/100 [=====] - 0s 4ms/step - loss: -88014.4922 - accuracy: 0.8428 - val_loss: -126065.3125 - val_accuracy:
Epoch 8/10
100/100 [=====] - 0s 4ms/step - loss: -186697.9375 - accuracy: 0.8428 - val_loss: -253275.3438 - val_accuracy:
Epoch 9/10
100/100 [=====] - 0s 5ms/step - loss: -354893.7188 - accuracy: 0.8428 - val_loss: -461960.8125 - val_accuracy:
Epoch 10/10
```

```
100/100 [=====] - 1s 5ms/step - loss: -620643.2500 - accuracy: 0.8428 - val_loss: -782088.7500 - val_accuracy:
```

```
acc_train_auto = autoencoder.evaluate(X_train, X_train)[1]
acc_test_auto = autoencoder.evaluate(X_test, X_test)[1]

print('\nAutoencoder: Accuracy on training Data: {:.3f}'.format(acc_train_auto))
print('Autoencoder: Accuracy on test Data: {:.3f}'.format(acc_test_auto))

250/250 [=====] - 1s 2ms/step - loss: -793228.3750 - accuracy: 0.8441
63/63 [=====] - 0s 2ms/step - loss: -774432.0000 - accuracy: 0.8455

Autoencoder: Accuracy on training Data: 0.844
Autoencoder: Accuracy on test Data: 0.845
```

```
#storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('AutoEncoder', acc_train_auto, acc_test_auto)
```

✓ 7.6. Support Vector Machines

Bài tập số 2: Hoàn thành huấn luyện mô hình SVM.

```
#Support vector machine model
from sklearn.svm import SVC

# instantiate the model
svm = SVC(kernel='linear', C=1.0, random_state=0, gamma=0.5)
#fit the model
svm.fit(X_train, y_train)

v SVC
SVC(gamma=0.5, kernel='linear', random_state=0)

#predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)

#computing the accuracy of the model performance
acc_train_svm = accuracy_score(y_train,y_train_svm)
acc_test_svm = accuracy_score(y_test,y_test_svm)

print("SVM: Accuracy on training Data: {:.3f}".format(acc_train_svm))
print("SVM : Accuracy on test Data: {:.3f}".format(acc_test_svm))

SVM: Accuracy on training Data: 0.803
SVM : Accuracy on test Data: 0.798

#storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('SVM', acc_train_svm, acc_test_svm)
```

✓ 8. So sánh các mô hình

```
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.812	0.810
1	Random Forest	0.814	0.812
2	Multilayer Perceptrons	0.864	0.862
3	XGBoost	0.867	0.860
4	AutoEncoder	0.844	0.845
5	SVM	0.803	0.798

```
#Sort accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

	ML Model	Train Accuracy	Test Accuracy
2	Multilayer Perceptrons	0.864	0.862
3	XGBoost	0.867	0.860
4	AutoEncoder	0.844	0.845
1	Random Forest	0.814	0.812
0	Decision Tree	0.812	0.810
5	SVM	0.803	0.798

Bài tập số 3: Cho biết kết quả so sánh mô hình, sau đó viết code lưu trữ mô hình.

```
import joblib

# save
joblib.dump(mlp, "MultilayerPerceptrons.pkl")

temp_model = joblib.load("MultilayerPerceptrons.pkl")

print(temp_model)

MLPClassifier(alpha=0.001, hidden_layer_sizes=[100, 100, 100])

joblib.dump(xgb, "XGBBoost.pkl")
temp_model = joblib.load("XGBBoost.pkl")
print(temp_model)

XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.4, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=7, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, random_state=None, ...)

# saving whole model
autoencoder.save('AutoEncoder.h5')

# loading whole model
from keras.models import load_model
temp_model = load_model('AutoEncoder.h5')
print(temp_model.summary())
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `
  saving_api.save_model(
Model: "model_10"
```

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 16)]	0
dense_70 (Dense)	(None, 16)	272

dense_71 (Dense)	(None, 16)	272
dense_72 (Dense)	(None, 14)	238
dense_75 (Dense)	(None, 16)	240
dense_76 (Dense)	(None, 16)	272

```
=====
Total params: 1294 (5.05 KB)
Trainable params: 1294 (5.05 KB)
Non-trainable params: 0 (0.00 Byte)
```

None

```
joblib.dump(forest, "RandomForest.pkl")
temp_model = joblib.load("RandomForest.pkl")
print(temp_model)
```

```
RandomForestClassifier(max_depth=5)
```

```
joblib.dump(tree, "DecisionTree.pkl")
temp_model = joblib.load("DecisionTree.pkl")
print(temp_model)
```

```
DecisionTreeClassifier(max_depth=5)
```

```
joblib.dump(svm, "SVM.pkl")
temp_model = joblib.load("SVM.pkl")
print(temp_model)
```

```
SVC(gamma=0.5, kernel='linear', random_state=0)
```

9. Trích dẫn

<https://blog.keras.io/building-autoencoders-in-keras.html> <https://en.wikipedia.org/wiki/Autoencoder> <https://mc.ai/a-beginners-guide-to-build-stacked-autoencoder-and-tying-weights-with-it/>
https://github.com/shreyagopal/t81_558_deep_learning/blob/master/t81_558_class_14_03_anomaly.ipynb
<https://machinelearningmastery.com/save-gradient-boosting-models-xgboost-python/>

LAB 4.2

B. Trích xuất thuộc tính trang web lừa đảo

(Phishing Website Detection Feature Extraction)

✓ 1. Thu thập dữ liệu

Ta cần 2 nhãn hợp pháp - legitimate (0) và lừa đảo - phishing (1). Việc thu thập các url lừa đảo khá dễ dàng do nguồn mở có tên PhishTank.

Dịch vụ này cung cấp một tập hợp các URL lừa đảo ở nhiều định dạng như csv, json, v.v. được cập nhật hàng giờ. Để tải xuống dữ liệu:

https://www.phishtank.com/developer_info.php Đối với các url hợp pháp, nguồn từ trường ĐH New Brunswick có sẵn các dạng benign, spam, phishing, malware & defacement URL <https://www.unb.ca/cic/datasets/url-2016.html>. Số lượng URL hợp pháp là 35,300 'Benign_list.csv'

✓ 1.1. Phishing URLs:

```
import pandas as pd
```

```
#Download phishing URLs
data0 = pd.read_csv("/content/drive/MyDrive/PPHM/Lab4/online-valid.csv")
data0.head()
```

	phish_id	url	phish_det
0	1624122	http://www.thegrillonthesquare.co.uk/language/...	http://www.phishtank.com/phish_det
1	1624120	http://sistemafidelidade2012.com/cadastramento...	http://www.phishtank.com/phish_det
2	1624113	http://www.cordonneriedaujourdhui-evreux.com/c...	http://www.phishtank.com/phish_det
3	1624112	http://65.60.51.66/~direccion/arqA/htmls/www.ca...	http://www.phishtank.com/phish_det
4	1624107	http://174.37.144.224/~credicon/includes/js/dt...	http://www.phishtank.com/phish_det

```
data0.shape
(7411, 8)
```

Chọn ngẫu nhiên 5000 mẫu.

```
phishurl = data0.sample(n = 5000, random_state = 12).copy()
phishurl = phishurl.reset_index(drop=True)
phishurl.head()
```

	phish_id	url	phish_det
0	1616193	http://ambcf.net/v3/images/masterpro/	http://www.phishtank.com/phish_de
1	1543768	http://chelf5089.home.comcast.net/~chelf5089/	http://www.phishtank.com/phish_de
2	1585549	http://www.mcbolos.com.br/libraries/joomla/log...	http://www.phishtank.com/phish_de
3	1620991	http://www.matchmai.com/img/us.match.com- login...	http://www.phishtank.com/phish_de
4	1502817	http://app.qmailer.co.uk/link/8556/3621624/179552	http://www.phishtank.com/phish_de

```
phishurl.shape
(5000, 8)
```

✓ 1.2. Legitimate URLs:

```
#Load legitimate files
data1 = pd.read_csv(data_dir + "Benign_list.csv")
data1.columns = ['URLs']
data1.head()
```

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

```
#Collec 5,000 Legitimate URLs randomly
legiurl = data1.sample(n = 5000, random_state = 12).copy()
legiurl = legiurl.reset_index(drop=True)
legiurl.head()
```

- | | |
|---|---|
| 0 | http://graphicriver.net/search?date=this-month... |
| 1 | http://ecnavi.jp/redirect?url=http://www.cros... |
| 2 | https://hubpages.com/signin?explain=follow+Hub... |
| 3 | http://extratorrent.cc/torrent/4190536/AOMEI+B... |
| 4 | http://icicibank.com/Personal-Banking/offers/o... |

```
legiurl.shape
```

```
(5000, 1)
```

✓ 2. Trích xuất thuộc tính

Các thuộc tính sẽ được trích xuất phân loại thành:

- Address Bar
- Domain
- HTML & Javascript

✓ 2.1 Thuộc tính Address Bar

Ta chỉ chọn một số thuộc tính cơ bản sau:

- Domain of URL
- IP Address in URL
- "@" Symbol in URL
- Length of URL
- Depth of URL
- Redirection "/" in URL
- "http/https" in Domain name
- Using URL Shortening Services "TinyURL"
- Prefix or Suffix "-" in Domain

```
from urllib.parse import urlparse,urlencode
import ipaddress
import re
```

```
# 1.Domain
def getDomain(url):
    domain = urlparse(url).netloc
    if re.match(r"^www.",domain):
        domain = domain.replace("www.", "")
    return domain
```

```
# 2.Have_IP
def havingIP(url):
    try:
        ipaddress.ip_address(url)
        ip = 1
    except:
        ip = 0
    return ip
```

```
# 3.Have_At (@)
def haveAtSign(url):
    if "@" in url:
        at = 1
    else:
        at = 0
    return at
```

```
# 4 URL Length
```

```

# 4.URL_Length
def getLength(url):
    if len(url) < 54:
        length = 0
    else:
        length = 1
    return length

# 5.URL_Depth '/'
def getDepth(url):
    s = urlparse(url).path.split('/')
    depth = 0
    for j in range(len(s)):
        if len(s[j]) != 0:
            depth = depth+1
    return depth

# 6.Redirection '/'
def redirection(url):
    pos = url.rfind('/')
    if pos > 6:
        if pos > 7:
            return 1
        else:
            return 0
    else:
        return 0

# 7. https_Domain
def httpDomain(url):
    domain = urlparse(url).netloc
    if 'https' in domain:
        return 1
    else:
        return 0

# 8. Checking Tiny_URL
shortening_services = r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|" \
    r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|" \
    r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|" \
    r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|db\.tt|" \
    r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd|" \
    r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co|" \
    r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|" \
    r"tr\.im|link\.zip\.net"

def tinyURL(url):
    match=re.search(shortening_services,url)
    if match:
        return 1
    else:
        return 0

# 9.Check Prefix or Suffix "-" in Domain
def prefixSuffix(url):
    if '-' in urlparse(url).netloc:
        return 1          # phishing
    else:
        return 0          # legitimate

```

✓ 2.2 Thuộc tính Domain

Các thể loại thuộc tính được trích xuất:

- DNS Record
- Website Traffic
- Age of Domain
- End Period of Domain

!pip install python-whois


```
Requirement already satisfied: python-whois in /usr/local/lib/python3.10/dist-packages (0.9.4)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from python-whois) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil->python-whois) (1.16.0)
```

```
import re
from bs4 import BeautifulSoup
import whois
import urllib
import urllib.request
from datetime import datetime
```

```
# 11.DNS_Record
# obtained in the featureExtraction function
```

```
!wget "https://tranco-list.eu/download/66GWX/1000000"
```

```
--2024-05-01 14:54:58-- https://tranco-list.eu/download/66GWX/1000000
Resolving tranco-list.eu (tranco-list.eu)... 104.21.77.59, 172.67.204.217, 2606:4700:3037::6815:4d3b, ...
Connecting to tranco-list.eu (tranco-list.eu)|104.21.77.59|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/csv]
Saving to: '1000000.1'

1000000.1          [ <=>          ] 21.52M  99.5MB/s   in 0.2s

2024-05-01 14:54:59 (99.5 MB/s) - '1000000.1' saved [22564504]
```

```
web_csv = pd.read_csv('1000000', header=None)
web_csv.head()
```

	0	1
0	1	google.com
1	2	facebook.com
2	3	a-msedge.net
3	4	amazonaws.com
4	5	microsoft.com

```
import requests
```

```
def web_traffic(url):
    try:
        #Tím domain tür URLs
        domain_url = extract_domain(url)
        rank = web_csv[web_csv[1]==domain_url]
        if (rank.empty):
            return 1
    except TypeError:
        return 1
    if (int(rank.iat[0, 0]) < 100000):
        return 0
    else:
        return 1
```

```
def extract_domain(url):
    domain = urllib.parse.urlsplit(url).netloc
    return domain
```

```

#Nếu age of domain > 12 tháng, giá trị của tính năng này là 1 (lừa đảo) khác 0 (hợp pháp)
# 13.Survival time of domain (Domain_Age)
def domainAge(domain_name):
    creation_date = domain_name.creation_date
    expiration_date = domain_name.expiration_date
    if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
        try:
            creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
            expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
        except:
            return 1
    if ((expiration_date is None) or (creation_date is None)):
        return 1
    elif ((type(expiration_date) is list) or (type(creation_date) is list)):
        return 1
    else:
        ageofdomain = abs((expiration_date - creation_date).days)
        if ((ageofdomain/30) < 6):
            age = 1
        else:
            age = 0
    return age

# Nếu period of domain > 6 tháng, giá trị của tính năng này là 1 (lừa đảo) khác 0 (hợp pháp)
# 14.End time of domain (Domain_End)
def domainEnd(domain_name):
    expiration_date = domain_name.expiration_date
    if isinstance(expiration_date,str):
        try:
            expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
        except:
            return 1
    if (expiration_date is None):
        return 1
    elif (type(expiration_date) is list):
        return 1
    else:
        today = datetime.now()
        end = abs((expiration_date - today).days)
        if ((end/30) < 6):
            end = 0
        else:
            end = 1
    return end

```

✓ 2.3 Thuộc tính HTML and JavaScript

Các tập thuộc tính được trích xuất sau:

- IFrame Redirection
- Status Bar Customization
- Disabling Right Click
- Website Forwarding

```
import requests
```

```

# 15. IFrame
def iframe(response):
    if response == "":
        return 1
    else:
        if re.findall(r"<iframe>|<frameBorder>", response.text):
            return 0
        else:
            return 1

```

```

# 16.Check Mouse_Over
def mouseOver(response):
    if response == "" :
        return 1
    else:
        if re.findall("<script>.+onmouseover.+</script>", response.text):
            return 1
        else:
            return 0

# 17.Check Right_Click
def rightClick(response):
    if response == "":
        return 1
    else:
        if re.findall(r"event.button ?== ?2", response.text):
            return 0
        else:
            return 1

# 18.Check Web_Forwards
def forwarding(response):
    if response == "":
        return 1
    else:
        if len(response.history) <= 2:
            return 0
        else:
            return 1

```

✓ 3. Tính toán thuộc tính URLs

Tạo danh sách và một hàm gọi các hàm khác và lưu trữ tất cả các thuộc tính của URL vào list.

```

def featureExtraction(url,label):

    features = []
    #Address bar features (10)
    features.append(getDomain(url))
    features.append(havingIP(url))
    features.append(haveAtSign(url))
    features.append(getLength(url))
    features.append(getDepth(url))
    features.append(redirection(url))
    features.append(httpDomain(url))
    features.append(tinyURL(url))
    features.append(prefixSuffix(url))

    #Domain features (4)
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    features.append(dns)
    features.append(web_traffic(url))
    features.append(1 if dns == 1 else domainAge(domain_name))
    features.append(1 if dns == 1 else domainEnd(domain_name))

    # HTML & Javascript features (4)
    try:
        response = requests.get(url)
    except:
        response = ""
    features.append(iframe(response))
    features.append(mouseOver(response))
    features.append(rightClick(response))
    features.append(forwarding(response))
    features.append(label)

    return features

```

3.1 URL hợp pháp

```
legiurl.shape

(5000, 1)

legi_features = []
label = 0

for i in range(0, 150):
    url = legiurl['URLs'][i]
    legi_features.append(featureExtraction(url,label))

2024-05-01 15:14:12,951 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:55:44,818 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:56:41,147 - whois.whois - ERROR - Error trying to connect to socket: closing socket - timed out
ERROR:whois.whois:Error trying to connect to socket: closing socket - timed out
2024-05-01 15:56:46,898 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:57:13,520 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:57:20,400 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:57:50,346 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:58:04,439 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:58:10,875 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
```

```
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
                 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Right_Click', 'Web_Forwards', 'Label']

legitimate = pd.DataFrame(legi_features, columns= feature_names)
legitimate.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	Ti
0	graphicriver.net	0	0	1	1	0	0	
1	ecnavi.jp	0	0	1	1	1	0	
2	hubpages.com	0	0	1	1	0	0	
3	extratorrent.cc	0	0	1	3	0	0	
4	icicibank.com	0	0	1	3	0	0	

```
legitimate.shape

(150, 18)

legitimate.to_csv('legitimate.csv', index= False)
```

3.2 URL lừa đảo

```
phishurl.shape

(5000, 8)

phish_features = []
label = 1
for i in range(0, 150):
    url = phishurl['url'][i]
    phish_features.append(featureExtraction(url,label))
```

```
2024-05-01 15:00:16,916 - whois.whois - ERROR - Error trying to connect to socket: closing socket - [Errno -2] Name or service not know
ERROR:whois.whois:Error trying to connect to socket: closing socket - [Errno -2] Name or service not known
2024-05-01 15:00:29,421 - whois.whois - ERROR - Error trying to connect to socket: closing socket - timed out
ERROR:whois.whois:Error trying to connect to socket: closing socket - timed out
```

```
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth', 'Redirection',
                 'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
                 'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over', 'Right_Click', 'Web_Forwards', 'Label']
```

```
phishing = pd.DataFrame(phish_features, columns= feature_names)
phishing.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https
0	ambcf.net	0	0	0	3	0	
1	chelf5089.home.comcast.net	0	0	0	1	0	
2	mcbolos.com.br	0	0	1	5	0	
3	matchmai.com	0	0	0	2	0	
4	app.qmailer.co.uk	0	0	0	4	0	

```
phishing.to_csv('phishing.csv', index= False)
```

4. Datasets

```
urldata = pd.concat([legitimate, phishing]).reset_index(drop=True)
urldata.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	Ti
0	graphicriver.net	0	0	1	1	0	0	
1	ecnavi.jp	0	0	1	1	1	0	
2	hubpages.com	0	0	1	1	0	0	
3	extratorrent.cc	0	0	1	3	0	0	
4	icicibank.com	0	0	1	3	0	0	

```
urldata.tail()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_
295	alexbucatarladomiciliu.ro	0	0	1	3	0	
296	ksra.ca	0	0	0	4	0	
297	sinaem.net	0	0	0	0	0	
298	hncxin.com	0	0	1	1	1	
299	vidros-curitiba.com	0	0	1	3	0	

```
urldata.shape
```

```
(300, 18)
```

```
urldata.to_csv('urldata.csv', index=False)
```

Ta trích xuất được 18 thuộc tính với 10,000 rows trong đó có 5000 phishing & 5000 legitimate URL

Bài tập số 3: Dựa vào mục trích xuất dữ liệu Lab 4.2, chọn một tập dataset khác thực hiện lại Lab 4.1

Sử dụng dataset Web page phishing detection từ <https://data.mendeley.com/datasets/c2gw7fy2j4/3> file: dataset_B_05_2020.csv

```
!wget "https://data.mendeley.com/public-files/datasets/c2gw7fy2j4/files/575316f4-ee1d-453e-a04f-7b950915b61b/file_downloaded" -O /content/d
--2024-05-01 16:03:48-- https://data.mendeley.com/public-files/datasets/c2gw7fy2j4/files/575316f4-ee1d-453e-a04f-7b950915b61b/file_dow
Resolving data.mendeley.com (data.mendeley.com)... 162.159.133.86, 162.159.130.86
Connecting to data.mendeley.com (data.mendeley.com)|162.159.133.86|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-1.amazonaws.com/67a1578a-90d9-4183-8e50-3ec9416561f0 [following]
--2024-05-01 16:03:49-- https://prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-1.amazonaws.com/67a1578a-90d9-4183-8e50-3ec9416561
Resolving prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-1.amazonaws.com (prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-1.am
Connecting to prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-1.amazonaws.com (prod-dcd-datasets-public-files-eu-west-1.s3.eu-west-
HTTP request sent, awaiting response... 200 OK
Length: 3661166 (3.5M) [text/csv]
Saving to: '/content/drive/MyDrive/PPHM/Lab4/url_phis_legit.csv'

/content/drive/MyDr 100%[=====] 3.49M 4.43MB/s in 0.8s

2024-05-01 16:03:50 (4.43 MB/s) - '/content/drive/MyDrive/PPHM/Lab4/url_phis_legit.csv' saved [3661166/3661166]
```

```
dataset = pd.read_csv("/content/drive/MyDrive/PPHM/Lab4/url_phis_legit.csv")
dataset.head()
```

	url	length_url	length_hostname	ip	nb_dots
0	http://www.crestonwood.com/router.php	37	19	0	3
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4
3	http://rgipt.ac.in	18	11	0	2
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2

5 rows × 89 columns

```
dataset.shape
```

(11430, 89)

```
url = dataset.iloc[:,0]
url.head()
```

```
0      http://www.crestonwood.com/router.php
1      http://shadetreetechnology.com/V4/validation/a...
2      https://support-appleld.com.secureupdate.duila...
3      http://rgipt.ac.in
4      http://www.iracing.com/tracks/gateway-motorspo...
Name: url, dtype: object
```

```
# thêm cột 'label' từ cột 'status' (legitimate = 0, phishing = 1)
import numpy as np
```

```
label_list = np.array(dataset.iloc[:,1])
convertSeries = np.where(dataset.status.values == "phishing", 1, 0)
label = pd.Series(convertSeries)
label.name = "label"
label.head()
```

```
0      0
1      1
2      1
3      0
4      0
Name: label, dtype: int64
```

```
full_df = pd.concat([url, label], axis=1)
full_df
```

	url	label
0	http://www.crestonwood.com/router.php	0
1	http://shadetreetechnology.com/V4/validation/a...	1
2	https://support-appleld.com.secureupdate.duila...	1
3	http://rgipt.ac.in	0
4	http://www.iracing.com/tracks/gateway-motorspo...	0
...
11425	http://www.fontspace.com/category/blackletter	0
11426	http://www.budgetbots.com/server.php/Server%20...	1
11427	https://www.facebook.com/Interactive-Televisio...	0
11428	http://www.mypublicdomainpictures.com/	0
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	1

11430 rows x 2 columns

Chọn ngẫu nhiên 150 URL. Quá nhiều URL thì vượt phiên làm việc của colab

```
df = full_df.sample(n=150, random_state=12)
df = df.reset_index(drop=True)
df.shape

(150, 2)

feature_list = []

for i in df.index:
    url = df['url'][i]
    label = df['label'][i]
    feature_list.append(featureExtraction(url,label))

feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
                  'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
                  'Domain_Age', 'Domain_End', 'iFrame', 'Mouse_Over','Right_Click', 'Web_Forwards', 'Label']

df_phis_legit = pd.DataFrame(feature_list, columns= feature_names)
df_phis_legit.head()
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirec
0	sapient-flare-279107.df.r.appspot.com	0	0	1	1	
1	support-appleld.com.secureupdate.duilawyeryork...	0	0	1	2	
2	centromusicalpaternense.es:443	0	0	1	2	
3	urbanmeisters.ovh	0	0	1	6	
4	immobiliereroseland.fr	0	0	0	0	

Tiến hành training cho các mô hình như 4.1

- Làm quen với dữ liệu

```
df_phis_legit.shape

(150, 18)
```

```
df_phis_legit.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 18 columns):
#   Column          Non-Null Count  Dtype
#  ...  ...  ...
```

```

---  -----  -----  -----
0   Domain      150 non-null  object
1   Have_IP     150 non-null  int64
2   Have_At     150 non-null  int64
3   URL_Length  150 non-null  int64
4   URL_Depth   150 non-null  int64
5   Redirection 150 non-null  int64
6   https_Domain 150 non-null  int64
7   TinyURL     150 non-null  int64
8   Prefix/Suffix 150 non-null  int64
9   DNS_Record  150 non-null  int64
10  Web_Traffic 150 non-null  int64
11  Domain_Age  150 non-null  int64
12  Domain_End  150 non-null  int64
13  iFrame      150 non-null  int64
14  Mouse_Over  150 non-null  int64
15  Right_Click 150 non-null  int64
16  Web_Forwards 150 non-null  int64
17  Label       150 non-null  int64
dtypes: int64(17), object(1)
memory usage: 21.2+ KB

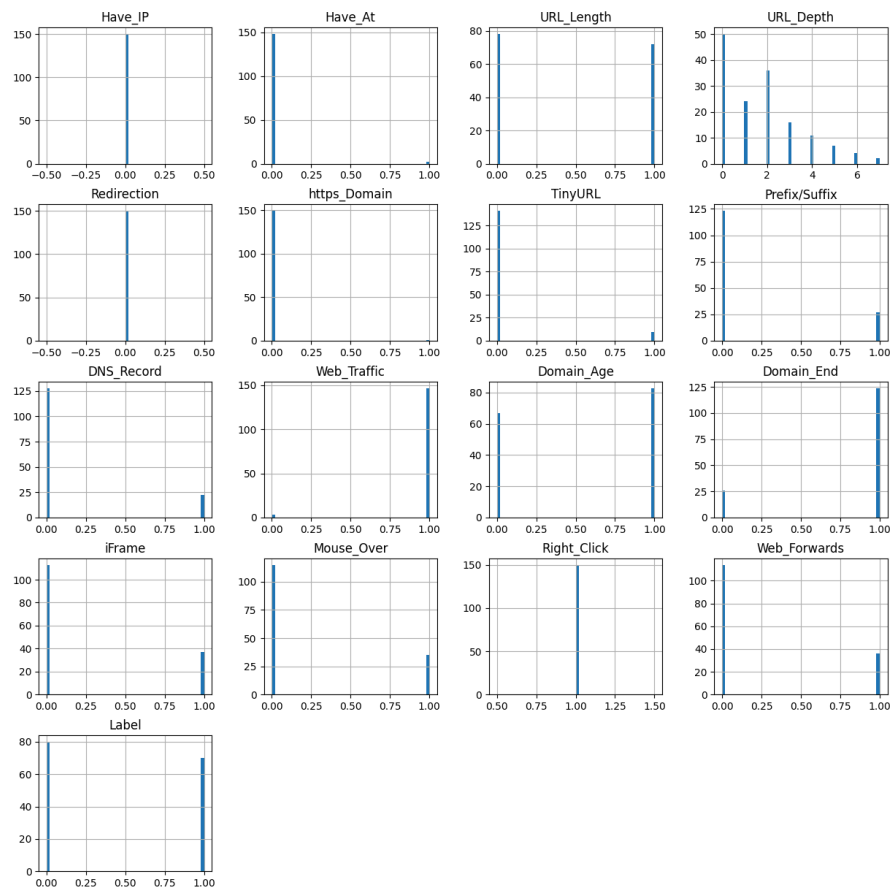
```

- Trực quan hóa dữ liệu

```

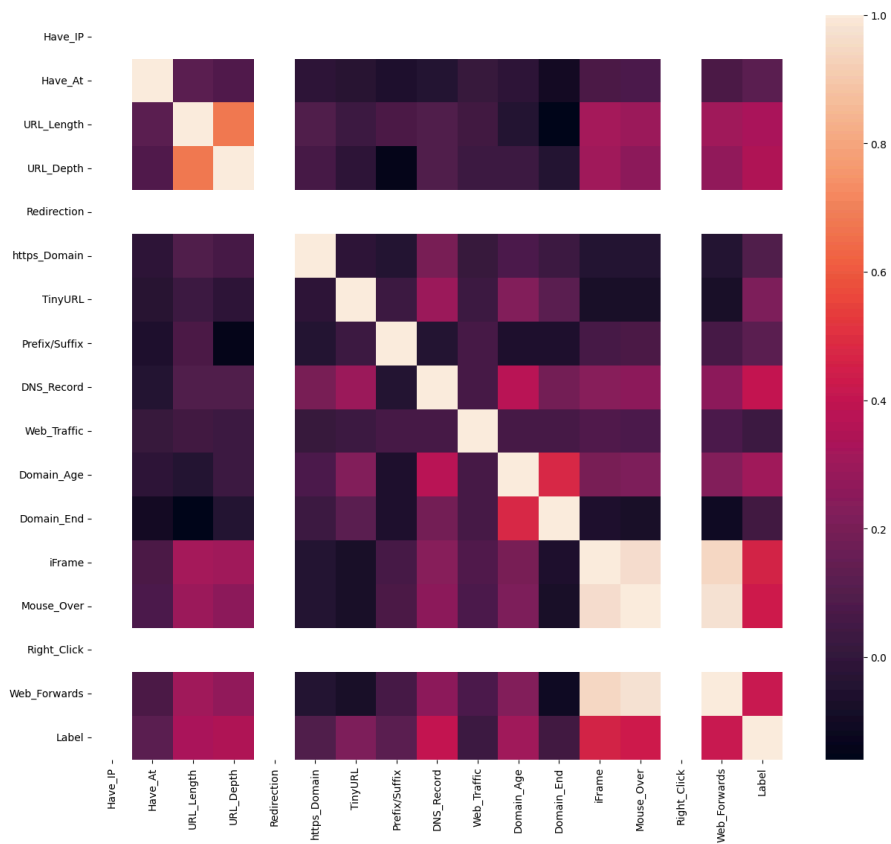
#Plotting the data distribution
df_phis_legit.hist(bins = 50,figsize = (15,15))
plt.show()

```

```
#Correlation heatmap
numeric_data = df_phis_legit.select_dtypes(include=['number']) # Select numeric columns

# Create correlation heatmap
plt.figure(figsize=(15,13))
sns.heatmap(numeric_data.corr())
plt.show()
```



- Tiền xử lý dữ liệu & EDA

```
df_phis_legit.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL
count	150.0	150.000000	150.000000	150.000000	150.0	150.000000	150.000000
mean	0.0	0.013333	0.480000	1.740000	0.0	0.006667	0.060000
std	0.0	0.115082	0.501274	1.739513	0.0	0.081650	0.238282
min	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
50%	0.0	0.000000	0.000000	2.000000	0.0	0.000000	0.000000
75%	0.0	0.000000	1.000000	3.000000	0.0	0.000000	0.000000
max	0.0	1.000000	1.000000	7.000000	0.0	1.000000	1.000000

```
#Drop Domain column
data = df_phis_legit.drop(['Domain'], axis = 1).copy()
```

```
#check data null
data.isnull().sum()
```

```
Have_IP      0
Have_At      0
URL_Length   0
URL_Depth    0
Redirection  0
https_Domain 0
TinyURL      0
Prefix/Suffix 0
DNS_Record   0
Web_Traffic  0
Domain_Age   0
Domain_End   0
iFrame       0
Mouse_Over   0
Right_Click  0
Web_Forwards 0
Label        0
dtype: int64
```

```
# shuffling the rows
data = data.sample(frac=1).reset_index(drop=True)
data.head()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/S
0	0	0	0	0	0	0	0	
1	0	0	1	3	0	1	0	
2	0	0	0	0	0	0	0	
3	0	0	1	2	0	0	0	
4	0	0	0	0	0	0	0	

- Chia dữ liệu

```
#features and target columns to X & y
y = data['Label']
X = data.drop('Label',axis=1)
X.shape, y.shape

((150, 16), (150,))
```

```
# train and test sets: 80-20 split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2, random_state = 12)
X_train.shape, X_test.shape
```

((120, 16), (30, 16))

- Đào tạo các mô hình học máy

```
from sklearn.metrics import accuracy_score
```

```
# Creating model results
ML_Model = []
acc_train = []
acc_test = []

#storing the results
def storeResults(model, a,b):
    ML_Model.append(model)
    acc_train.append(round(a, 3))
    acc_test.append(round(b, 3))
```

Decision Tree

```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate
tree = DecisionTreeClassifier(max_depth = 5)
# fit
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

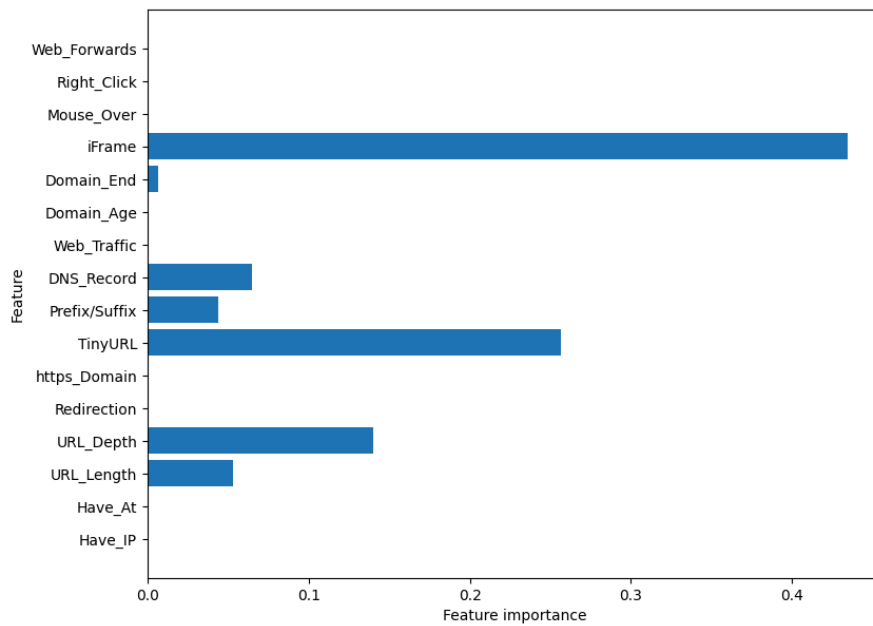
```
#predicting
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
```

```
#computing the accuracy
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)
```

```
print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
```

```
Decision Tree: Accuracy on training Data: 0.842
Decision Tree: Accuracy on test Data: 0.767
```

```
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), tree.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```



```
storeResults('Decision Tree', acc_train_tree, acc_test_tree)
```

Random Forest

```
# Random Forest model
from sklearn.ensemble import RandomForestClassifier
```

```
# instantiate
forest = RandomForestClassifier(max_depth=5)
```

```
# fit
forest.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5)
```

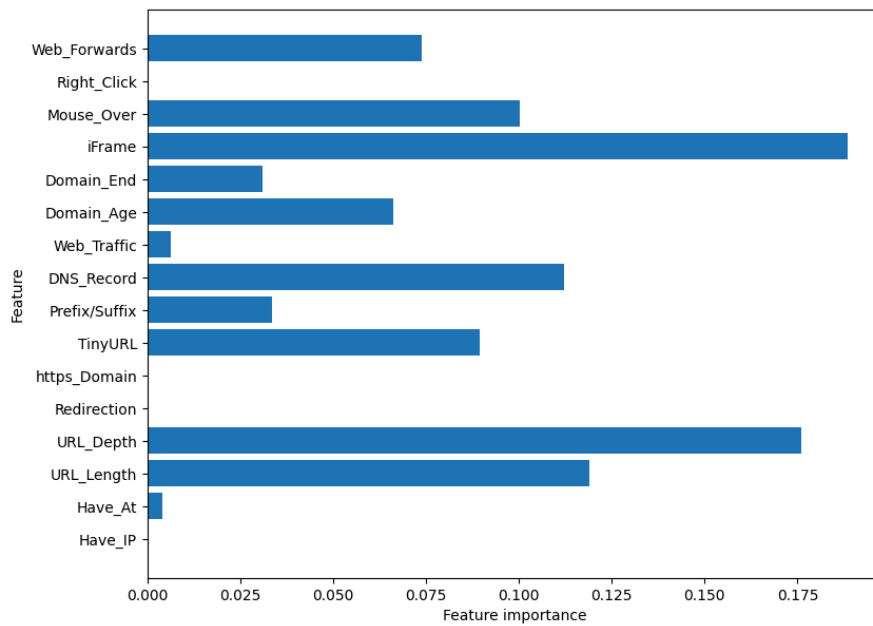
```
#predicting
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
```

```
#computing the accuracy
acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)
```

```
print("Random forest: Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random forest: Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random forest: Accuracy on training Data: 0.858
Random forest: Accuracy on test Data: 0.767
```

```
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), forest.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```



```
storeResults('Random Forest', acc_train_forest, acc_test_forest)
```

Multilayer Perceptrons (MLPs) - Học sâu

```
# Multilayer Perceptrons model
from sklearn.neural_network import MLPClassifier

# instantiate
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=([100,100,100]))

# fit
mlp.fit(X_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.p
warnings.warn(
    """
    MLPClassifier
    MLPClassifier(alpha=0.001, hidden_layer_sizes=[100, 100, 100])
    """
)
```

```
#predicting
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)

#computing the accuracy
acc_train_mlp = accuracy_score(y_train,y_train_mlp)
acc_test_mlp = accuracy_score(y_test,y_test_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc_test_mlp))
```

```
Multilayer Perceptrons: Accuracy on training Data: 0.908
Multilayer Perceptrons: Accuracy on test Data: 0.700
```

```
storeResults('MLP', acc_train_mlp, acc_test_mlp)
```

XGBoost

```
#XGBoost Classification model
from xgboost import XGBClassifier

# instantiate
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)
#fit
xgb.fit(X_train, y_train)
```

```
▼ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.4, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=7, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, random_state=None, ...)
```

```
#predicting
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

#computing the accuracy
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

XGBoost: Accuracy on training Data: 0.883
XGBoost : Accuracy on test Data: 0.667
```

```
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)
```

Autoencoder Neural Network

```
#importing required packages
import keras
from keras.layers import Input, Dense
from keras import regularizers
import tensorflow as tf
from keras.models import Model
from sklearn import metrics

#building autoencoder model

input_dim = X_train.shape[1]
encoding_dim = input_dim

input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="relu",
                activity_regularizer=regularizers.l1(10e-4))(input_layer)
encoder = Dense(int(encoding_dim), activation="relu")(encoder)

encoder = Dense(int(encoding_dim-2), activation="relu")(encoder)
code = Dense(int(encoding_dim-4), activation='relu')(encoder)
decoder = Dense(int(encoding_dim-2), activation='relu')(code)

decoder = Dense(int(encoding_dim), activation='relu')(decoder)
decoder = Dense(input_dim, activation='sigmoid')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.summary()
```

Model: "model_4"

Layer (type)	Output Shape	Param #
=====		
input_5 (InputLayer)	[(None, 16)]	0
dense_28 (Dense)	(None, 16)	272

dense_29 (Dense)	(None, 16)	272
dense_30 (Dense)	(None, 14)	238
dense_33 (Dense)	(None, 16)	240
dense_34 (Dense)	(None, 16)	272

=====

Total params: 1294 (5.05 KB)
 Trainable params: 1294 (5.05 KB)
 Non-trainable params: 0 (0.00 Byte)

```
#compiling the model
autoencoder.compile(optimizer='adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
```

```
#Training the model
history = autoencoder.fit(X_train, X_train, epochs=10, batch_size=64, shuffle=False, validation_split=0.2)
```

```
Epoch 1/10
2/2 [=====] - 2s 349ms/step - loss: 0.0938 - accuracy: 0.6146 - val_loss: -0.0728 - val_accuracy: 0.5833
Epoch 2/10
2/2 [=====] - 0s 68ms/step - loss: 0.0693 - accuracy: 0.6146 - val_loss: -0.1054 - val_accuracy: 0.5833
Epoch 3/10
2/2 [=====] - 0s 49ms/step - loss: 0.0446 - accuracy: 0.6146 - val_loss: -0.1388 - val_accuracy: 0.5833
Epoch 4/10
2/2 [=====] - 0s 71ms/step - loss: 0.0196 - accuracy: 0.6146 - val_loss: -0.1733 - val_accuracy: 0.5833
Epoch 5/10
2/2 [=====] - 0s 44ms/step - loss: -0.0060 - accuracy: 0.6146 - val_loss: -0.2090 - val_accuracy: 0.5833
Epoch 6/10
2/2 [=====] - 0s 60ms/step - loss: -0.0322 - accuracy: 0.6146 - val_loss: -0.2457 - val_accuracy: 0.5833
Epoch 7/10
2/2 [=====] - 0s 65ms/step - loss: -0.0591 - accuracy: 0.6146 - val_loss: -0.2836 - val_accuracy: 0.5833
Epoch 8/10
2/2 [=====] - 0s 74ms/step - loss: -0.0868 - accuracy: 0.6146 - val_loss: -0.3229 - val_accuracy: 0.5833
Epoch 9/10
2/2 [=====] - 0s 65ms/step - loss: -0.1154 - accuracy: 0.6146 - val_loss: -0.3636 - val_accuracy: 0.5833
Epoch 10/10
2/2 [=====] - 0s 64ms/step - loss: -0.1449 - accuracy: 0.6146 - val_loss: -0.4062 - val_accuracy: 0.5833
```

```
acc_train_auto = autoencoder.evaluate(X_train, X_train)[1]
```