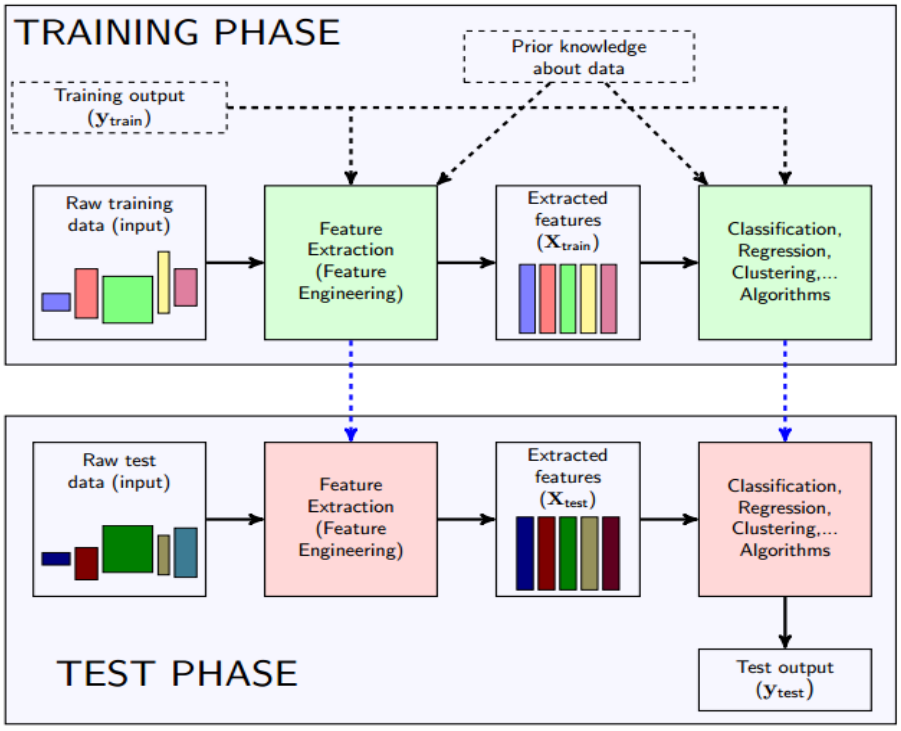
Mô hình chung cho bài toán Machine Learning



Thuật toán Mạng neuron đa tầng và lan truyền ngược, Multi-layer Perceptron and Backpropagation (MLP)

Ví dụ: Sử dụng thuật toán MLP để xây dựng mô hình hóa chất lượng rượu vang trắng dựa trên các thử nghiệm hóa lý

File Dữ liệu:

https://archive.ics.uci.edu/ml/datasets/wine+quality

Mô tả dữ liệu:

Attribute Information: For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity - độ axit cố định
2 - volatile acidity - tính axit dễ bay hơi
3 - citric acid - axit xitric
4 - residual sugar - đường dư
5 - chlorides - clorua
6 - free sulfur dioxide -
7 - total sulfur dioxide - tổng lưu huỳnh điôxit
8 - density - mật độ
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):

12 - quality (score between 0 and 10) - chất lượng được đánh giá bởi điểm nằm trong khoảng từ 0 đến 10

1. Sử dụng các thư viện: numpy, matplotlib, sklearn,… thông qua lệnh import

Import thư viện

```
import math as m
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import seaborn as sns
```

2. **Đọc file chứa dữ liệu và in ra tiêu đề các cột**

```
data = pd.read_csv('datasets/uci-wine-quality/winequality-white.csv')
print(list(data.columns))
#targets - 'quality' column
#features - all other columns
```

```
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',
'density', 'pH', 'sulphates', 'alcohol', 'quality']
```

In ra dữ liệu 5 dòng đầu tiên:

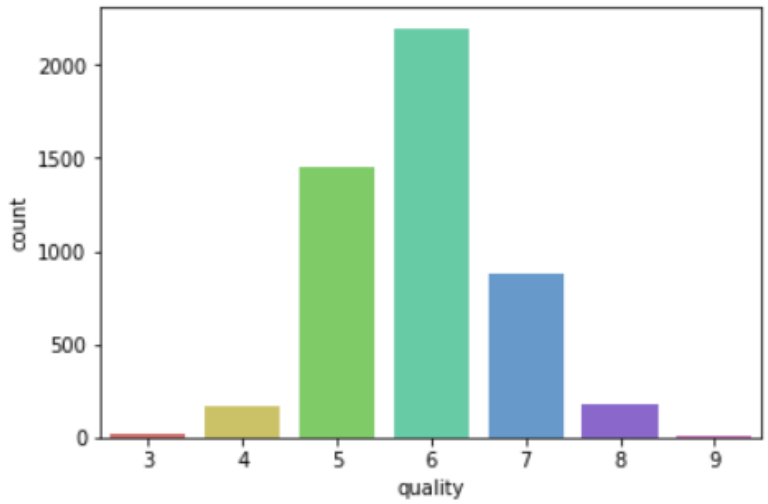
```
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

3. **Vẽ đồ thị phân bổ dữ liệu quality và số lượng mẫu**

```
sns.countplot(x='quality', data=data, palette='hls')
plt.show()
```

Kết quả:



4. **Thêm cột category, với giá trị catergory cụ thể như sau:**
   quality < 6 thì category = 0
   quality > 6 thì category = 2
   quality = 6 thì category = 1

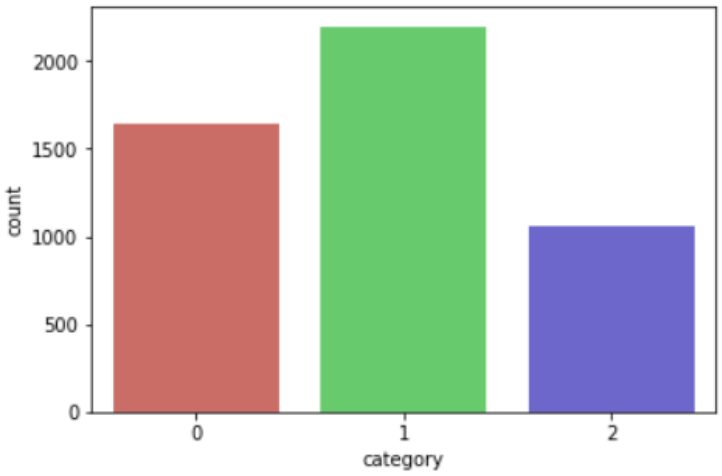
```
c = []
for q in data['quality'].values:
    if q < 6:
        c.append(0)
    elif q > 6:
        c.append(2)
    else:
        c.append(1)

data['category'] = c
data.head()
```

Kết quả:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	category
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6	1
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1

5. **Vẽ đồ thị phân bổ dữ liệu category và số lượng mẫu**

```
sns.countplot(x='category', data=data, palette='hls')
plt.show()
```



6. Giá trị Đặc trưng, gán data_features giá trị từ các cột dữ liệu:

```
data_features = data.columns.values.tolist()
print(data_features)
remove_features = ['quality']
to_keep_features = [i for i in data_features if i not in remove_features]
data_final = data[to_keep_features]
data_final.head()
```

Kết quả:

```
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',
'density', 'pH', 'sulphates', 'alcohol', 'quality', 'category']
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	category
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	1
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1

7. Chuyển dữ liệu thành mảng:

```
X = np.array(data_final.loc[:, data_final.columns != 'category'])
#X.head()
print(X)
```

```
[[ 7.    0.27  0.36 ...  3.    0.45  8.8 ]
 [ 6.3   0.3   0.34 ...  3.3   0.49  9.5 ]
 [ 8.1   0.28  0.4  ...  3.26  0.44 10.1 ]
 ...
 [ 6.5   0.24  0.19 ...  2.99  0.46  9.4 ]
 [ 5.5   0.29  0.3  ...  3.34  0.38 12.8 ]
 [ 6.    0.21  0.38 ...  3.26  0.32 11.8 ]]
```

```
y = np.array(data_final.loc[:, data_final.columns == 'category'])
#y.head()
print(y)
```

```
[[1]
 [1]
 [1]
 ...
 [1]
 [2]
 [1]]
```

8. Phân chia tập dữ liệu ra thành 2 tập: tập huấn luyện và tập kiểm tra:

```
train_features, test_features, train_targets, test_targets = train_test_split(X, y, test_size=0.3, random_state=0)
print("###### Training and test datasets ######")
print("Training size: ", len(train_targets))
print("Test size: ", len(test_targets))
```

```
###### Training and test datasets ######
Training size:  3428
Test size:  1470
```

9. Biến đổi kích thước ma trận dữ liệu huấn luyện đích, kiểm tra đích:

```
train_targets = train_targets.reshape(train_targets.shape[0])
test_targets = test_targets.reshape(test_targets.shape[0])
print(train_targets)
```

```
[0 1 0 ... 1 1 0]
```

10. Tạo mô hình sử dụng thuật toán MLP, số lớp ẩn là 5, huấn luyện mô hình:

```
mlpClassifier5 = MLPClassifier(solver='lbfgs',
                               alpha=1e-5,
                               hidden_layer_sizes=(5,),
                               random_state=1,
                               max_iter=1000000)
mlpClassifier5.fit(train_features, train_targets)
print(mlpClassifier5.coefs_)
print(mlpClassifier5.intercepts_)
```

Kết quả:

```
[array([[-1.01615303e-01, -6.45139824e-01,  4.43574109e-01,
        -2.67880305e-01, -4.63611723e-01],
       [-4.99224368e-01, -3.75720895e-01,  2.08638682e+01,
        -1.27586871e-01,  4.62029008e-02],
       [-9.89548360e-02,  1.75281068e-01, -2.76234843e-01,
         4.61761202e-01, -5.80031603e-01],
       [ 2.08755438e-01, -6.77525789e-01, -2.37899858e-01,
        -4.48545574e-01, -3.84680267e-01],
       [ 3.68293430e-01,  5.71056666e-01,  5.89032172e+00,
         2.35419235e-01,  4.60687026e-01],
       [ 4.83237461e-01, -5.69540253e-01, -4.64361960e-02,
        -4.44441300e-01, -9.98865319e-01],
       [-4.91866646e-01, -1.60702648e+00,  9.67343712e-03,
        -8.95417353e-01, -2.00431851e+00],
       [-2.25920559e-01,  1.71259341e-01,  2.21670152e+01,
        -5.94602166e-01,  3.01296817e-01],
       [ 5.98661942e-01,  1.69489753e-01, -3.09120120e-01,
         3.37736227e-01, -5.03307751e-01],
       [-6.38098705e-02,  4.53277810e-01, -6.10712932e+00,
        -2.62570240e-01, -4.56296759e-01],
       [-5.88585831e-01, -1.59571520e-01, -3.45233476e+00,
        -3.29907847e-01, -6.30394498e-02]]), array([[-0.68869855, -0.1488424
5,  0.33667261],
       [-0.03543796, -0.59570527, -0.23473311],
       [ 0.59788801,  0.34544818,  0.13609094],
       [ 0.16256951,  0.68413189, -0.62600843],
       [-0.56347563,  0.55610316, -0.26220011]])]
[array([-0.54701693,  0.03369534, 21.34964092,  0.10465444,  0.23958723]), a
rray([-2.53775342,  0.72790817,  1.70701104])]
```

11. Hiển thị tập huấn luyện:

```
train_predictions = mlpClassifier5.predict(train_features)
print("###### Training - Prediction results of MLP Backpropagation ######")
print("Target labels:     ", train_targets)
print("Prediction labels: ", train_predictions)
```

```
###### Training - Prediction results of MLP Backpropagation ######
Target labels:      [0 1 0 ... 1 1 0]
Prediction labels:  [0 1 0 ... 1 1 0]
```

12. Hiển thị kết quả đích và kết quả dự đoán của mô hình đối với tập huấn luyện:

```
accuracy = 100 * accuracy_score(train_targets, train_predictions)
print("###### Training - Prediction accuracy of MLP Backpropagation ######")
print("Accuracy of Logistic Regression:     ", accuracy)
print(classification_report(train_targets, train_predictions))
```

```
###### Training - Prediction accuracy of MLP Backpropagation ######
Accuracy of Logistic Regression:      56.884480746791134
              precision    recall  f1-score   support

           0       0.64      0.56      0.60      1113
           1       0.54      0.70      0.61      1568
           2       0.57      0.29      0.39       747

    accuracy                           0.57      3428
   macro avg       0.58      0.52      0.53      3428
weighted avg       0.58      0.57      0.56      3428
```

13. Hiển thị độ chính xác của mô hình đối với tập kiểm tra:

```
test_predictions = mlpClassifier5.predict(test_features)
print("##### Testing - Prediction results of MLP Backpropagation #####")
print("Target labels:     ", test_targets)
print("Prediction labels: ", test_predictions)
```

```
##### Testing - Prediction results of MLP Backpropagation #####
Target labels:      [0 1 2 ... 1 1 1]
Prediction labels:  [0 0 1 ... 1 1 1]
```

```
accuracy = 100 * accuracy_score(test_targets, test_predictions)
print("##### Testing - Prediction accuracy of MLP Backpropagation #####")
print("Accuracy of Logistic Regression:     ", accuracy)
print(classification_report(test_targets, test_predictions))
```

```
##### Testing - Prediction accuracy of MLP Backpropagation #####
Accuracy of Logistic Regression:      54.21768707482993
              precision    recall  f1-score   support

           0       0.65      0.54      0.59       527
           1       0.49      0.69      0.58       630
           2       0.50      0.25      0.33       313

    accuracy                           0.54      1470
   macro avg       0.55      0.49      0.50      1470
weighted avg       0.55      0.54      0.53      1470
```

14. Thay đổi số lớp ẩn thành 50:

```
mlpClassifier50 = MLPClassifier(solver='lbfgs',
                                alpha=1e-5,
                                hidden_layer_sizes=(50,),
                                random_state=1,
                                 max_iter=1000000)
mlpClassifier50.fit(train_features, train_targets)
```

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(50,), max_iter=1000000,
              random_state=1, solver='lbfgs')
```

15. Hiển thị độ chính xác của mô hình đối với tập huấn luyện:

```
train_predictions = mlpClassifier50.predict(train_features)
```

```
accuracy = 100 * accuracy_score(train_targets, train_predictions)
print("##### Training - Prediction accuracy of MLP Backpropagation #####")
print("Accuracy of Logistic Regression:     ", accuracy)
print(classification_report(train_targets, train_predictions))
```

```
##### Training - Prediction accuracy of MLP Backpropagation #####
Accuracy of Logistic Regression:      46.12018669778296
              precision    recall  f1-score   support

           0       0.45      0.42      0.44      1113
           1       0.47      0.71      0.56      1568
           2       0.00      0.00      0.00       747

    accuracy                           0.46      3428
   macro avg       0.31      0.38      0.33      3428
weighted avg       0.36      0.46      0.40      3428
```

16. Hiển thị độ chính xác của mô hình đối với tập kiểm tra:

```
test_predictions = mlpClassifier50.predict(test_features)
```

```
# accuracy = 100 * accuracy_score(test_targets, test_predictions)
print("##### Testing - Prediction accuracy of MLP Backpropagation #####")
print("Accuracy of Logistic Regression:     ", accuracy)
print(classification_report(test_targets, test_predictions))
```

```
##### Testing - Prediction accuracy of MLP Backpropagation #####
Accuracy of Logistic Regression:      46.12018669778296
              precision    recall  f1-score   support

           0       0.51      0.44      0.48       527
           1       0.44      0.72      0.55       630
           2       0.00      0.00      0.00       313

    accuracy                           0.47      1470
   macro avg       0.32      0.39      0.34      1470
weighted avg       0.37      0.47      0.41      1470
```

Bài tập 1: Sinh viên nhận xét về độ chính xác của thuật toán khi số lớp ẩn thay đổi.

Bài tập 2. Với dữ liệu này sinh viên sử dụng thuật toán khác như KNN, Softmax để so sánh với mô hình MLP ở trên.