

Model_Selection

August 7, 2023

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

Mounted at /content/drive

===== Nguồn <http://users.soict.hust.edu.vn/khoattq/ml-dm-course/> =====

1 Bài toán

- Cần đánh giá hiệu quả của một mô hình phân loại?
- So sánh hiệu quả của 2 mô hình khác nhau?

Nhưng ta chỉ có một tập dữ liệu đã thu thập được. Để trả lời hai câu hỏi trên thì cần thực hiện bước “Lựa chọn tham số” của mô hình đã chọn.

Bài này sẽ hướng dẫn cách thực hiện từng bước chi tiết, từ lựa chọn tham số (sử dụng Cross validation), cho đến đánh giá (sử dụng Holdout) và so sánh hai mô hình khác nhau. Tập dữ liệu sử dụng là tập văn bản đã thu thập được ở Bài học số 2 (tập tin tức từ Vnexpress).

1.1 Mục lục

- Load dữ liệu từ thư mục
- Tiền xử lý dữ liệu
- Lựa chọn tham số các mô hình
- So sánh SVM và Random Forest

```
[ ]: cd /content/drive/MyDrive/ML_course/SVM_practice/data
```

/content/drive/MyDrive/ML_course/SVM_practice/data

```
[ ]: !pip install pyvi
```

```
[ ]: import os
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import learning_curve
from tqdm import tqdm

from sklearn.datasets import load_files
from pyvi import ViTokenizer
```

```

from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, \
    ConfusionMatrixDisplay

%matplotlib inline

```

1.2 Load dữ liệu từ thư mục

Giả sử cấu trúc thư mục như sau

- data/news_vnexpress/
 - Kinh tế:
 - * bài báo 1.txt
 - * bài báo 2.txt
 - Pháp luật
 - * bài báo 3.txt
 - * bài báo 4.txt

```

[ ]: INPUT = '/content/drive/MyDrive/ML_course/SVM_practice/data/news_1135/'
os.makedirs("images",exist_ok=True) # thư mục lưu các hình ảnh kết quả trong
    quá trình huấn luyện và đánh giá

```

```

[ ]: # statistics
print('Các nhãn và số văn bản tương ứng trong dữ liệu')
print('-----')
n = 0
for label in os.listdir(INPUT):
    if 'stop' in label:
        continue
    print(f'{label}: {len(os.listdir(os.path.join(INPUT, label)))}')
    n += len(os.listdir(os.path.join(INPUT, label)))

print('-----')
print(f"Tổng số văn bản: {n}")

```

Các nhãn và số văn bản tương ứng trong dữ liệu

Độc giả: 52
 Đời sống - Xã hội: 91
 Tin khác: 100
 Sức khỏe: 75
 Thể thao: 140

Thời sự: 138
 Pháp luật: 50
 Kinh tế: 186
 Giải trí: 107
 Khoa học - Công nghệ: 196

 Tổng số văn bản: 1135

```

[ ]: # load data
data_train = load_files(container_path=INPUT, encoding="utf-8")

print('mapping:')
for i in range(len(data_train.target_names)):
    print(f'{data_train.target_names[i]} - {i}')

print('-----')
print(data_train filenames[0:1])
# print(data_train.data[0:1])
print(data_train.target[0:1])
print(data_train.data[0:1])

print("\nTổng số văn bản: {}".format( len(data_train.filenames)))
  
```

mapping:
 Giải trí - 0
 Khoa học - Công nghệ - 1
 Kinh tế - 2
 Pháp luật - 3
 Sức khỏe - 4
 Thể thao - 5
 Thời sự - 6
 Tin khác - 7
 Đời sống - Xã hội - 8
 Đọc giả - 9

['/content/drive/MyDrive/ML_course/SVM_practice/data/news_1135/Tin
 khác/0218e1df21ce358b9c6485176a48f1fcaeedef67.txt']

[7]
 ['Dân_trí Sở GD & ĐT tỉnh Gia_Lai vừa ra văn_bản số 2258 / SGDĐT - VP , về việc
 chấn_chỉnh việc tiếp_thị sách và các vật_dụng khác trong các cơ_sở giáo_dục .
 Văn_bản chỉ_dạo , tuyệt_đối không cho phép các cá_nhân , tập_thể đến trường
 tiếp_thị , quảng_cáo mua_bán sách , dụng_cụ học_tập ... cho giáo_viên và học_sinh
 trong nhà_trường . Các tổ_chức , cá_nhân trong ngành giáo_dục tuyệt_đối không
 được thực_hiện hoặc tham_gia giới_thiệu , quảng_bá , vận_động mua , phát_hành
 sách tham_khảo tới học_sinh hoặc phụ_huynh dưới hình_thức nào . Nhà_trường
 tuyệt_đối không được lưu_hành , sử_dụng sách có nội_dung không lành_mạnh , không
 phù_hợp với nội_dung chương_trình phổ_thông . Trường_hợp phát_hiện sách có
 sai_sốt , các đơn_vị cần báo_cáo với cấp trên để có hướng_xử_lý . Các sơ_sở

giáo_dục đề_cao cảnh_giác đối_với trường_hợp mạo_danh cán_bộ , chuyên_viên sở trong ngành đi giới_thiệu sách , đồ_dùng học_sinh ; công_khai phổ_biến các quy_định trên đến cán_bộ , giáo_viên , học_sinh để cùng phòng tránh và ngăn_chặn ... Trước đó , báo Dân_trí đã thông_tin về việc học_sinh của Trường Tiểu_học số 2 xã Hòa Phú (Chu_Păh , Gia_Lai) đã mang 1 tờ giấy thông_báo về việc mua sách tham_khảo mang về cho phụ_huynh và xin tiền để mua sách , khiến nhiều phụ_huynh bức_xúc . Sự_việc được bà Dương Thị Nga - Hiệu_trưởng nhà_trường cho biết , do hôm xảy ra sự_việc , bà đi_vắng nên không hay_biết . Tuệ Mẫn']

Tổng số văn bản: 1135

1.3 Tiền xử lý dữ liệu:

1.3.1 Chuyển dữ liệu dạng text về dạng số

Chuyển dữ liệu dạng text về ma trận (n x m) bằng TF. Chuyển nhãn các văn bản về dạng số

```
[ ]: # load dữ liệu các stopwords
with open("vietnamese-stopwords.txt", encoding="utf-8") as f:
    stopwords = f.readlines()
stopwords = [x.strip().replace(" ", "_") for x in stopwords]
print(f"Số lượng stopwords: {len(stopwords)}")
print(stopwords[:10])

# Chuyển hoá dữ liệu text về dạng vector TF
# - loại bỏ từ dừng
# - sinh từ điển
module_count_vector = CountVectorizer(stop_words=stopwords)
model_rf_preprocess = Pipeline([('vect', module_count_vector),
                                ('tf', TfidfTransformer()),
                                ])

# Hàm thực hiện chuyển đổi dữ liệu text thành dữ liệu số dạng ma trận
# Input: Dữ liệu 2 chiều dạng numpy.array, mảng nhãn id dạng numpy.array
data_preprocessed = model_rf_preprocess.fit_transform(data_train.data,
↪data_train.target)

print(f"\nSố lượng từ trong từ điển: {len(module_count_vector.vocabulary_)}")
print(f"Kích thước dữ liệu sau khi xử lý: {data_preprocessed.shape}")
print(f"Kích thước nhãn tương ứng: {data_train.target.shape}")
```

1.3.2 Chia dữ liệu thành tập train và test

(Nghĩa là ta sẽ dùng Holdout để đánh giá hiệu quả của một mô hình)

```
[ ]: p = 0.2
pivot = int(data_preprocessed.shape[0] * (1-0.2))
X_train, X_test = data_preprocessed[0:pivot], data_preprocessed[pivot:]
Y_train, Y_test = data_train.target[0:pivot], data_train.target[pivot:]
```

2 Lựa chọn (tối ưu) tham số

Chỉ dùng tập train để thực hiện lựa chọn tham số. - SVM: kernel, C - Random Forest: criteria, N
Ta sẽ dùng chiến lược Cross Validation trong bước này.

```
[ ]: def cross_validation(estimator):
    _, train_scores, test_scores = learning_curve(estimator, X_train, Y_train,
    cv=5, n_jobs=-1, train_sizes=[1.0, ], scoring="accuracy")
    test_scores = test_scores[0]
    mean, std = test_scores.mean(), test_scores.std()
    return mean, std

def plot(title, xlabel, X, Y, error, ylabel = "Accuracy"):
    plt.xlabel(xlabel)
    plt.title(title)
    plt.grid()
    plt.ylabel(ylabel)

    plt.errorbar(X, Y, error, linestyle='None', marker='o')
```

2.1 Đánh giá hiệu quả của các kernel trong SVM

```
[ ]: title = "thay đổi kernel, C = 1"
    xlabel = "kernel"
    X = []
    Y = []
    error = []

    for kernel in tqdm(['linear', 'poly', 'rbf', 'sigmoid']):
        # Với mỗi kernel được chọn,
        # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
        text_clf = svm.SVC(kernel=kernel, C=1.0)
        mean, std = cross_validation(text_clf)
        X.append(kernel)
        Y.append(mean)
        error.append(std)

    # lưu kết quả ra file ảnh
    plot(title, xlabel, X, Y, error)
    plt.savefig('images/svm_change_kernel.png', bbox_inches='tight')
    plt.show()
```

2.2 Đánh giá ảnh hưởng của tham số C trong SVM

```
[ ]: title = "thay đổi C, kernel = linear"
xlabel = "C"
X = []
Y = []
error = []

for C in tqdm([.1, 1.0, 2.0, 5.0, 10.0]):
    # Với từng giá trị C nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = svm.SVC(kernel='linear', C=C)
    mean, std = cross_validation(text_clf)
    X.append(str(C))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig('images/svm_change_C.png', bbox_inches='tight')
plt.show()
```

2.3 Đánh giá ảnh hưởng của độ đo trong Random Forest

```
[ ]: title = "thay đổi criterion, n_estimators = 50"
xlabel = "criterion"
X = []
Y = []
error = []

for criterion in tqdm(["gini", "entropy"]):
    # Với mỗi criterion nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = RandomForestClassifier(criterion=criterion, n_estimators=50)
    mean, std = cross_validation(text_clf)
    X.append(str(criterion))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig('images/RF_change_criterion.png', bbox_inches='tight')
plt.show()
```

2.4 Đánh giá ảnh hưởng của số cây trong Random Forest

```
[ ]: title = "thay đổi n_estimators, criterion = gini"
xlabel = "n_estimators"
X = []
Y = []
error = []

for n_estimators in tqdm([10, 50, 100, 300]):
    # Với từng giá trị n_estimators nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = RandomForestClassifier(criterion='gini',
    ↪n_estimators=n_estimators)
    mean, std = cross_validation(text_clf)
    X.append(str(n_estimators))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig('images/RF_change_N.png', bbox_inches='tight')
plt.show()
```

2.5 Đánh giá ảnh tham số K trong mô hình KNN

```
[ ]: from sklearn.neighbors import KNeighborsClassifier

title = "thay đổi K"
xlabel = "K"
X = []
Y = []
error = []

for k in tqdm([1, 3, 5, 20, 50]):
    # Với từng giá trị k nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = KNeighborsClassifier(n_neighbors=k)
    mean, std = cross_validation(text_clf)
    X.append(str(k))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig('images/KNN_change_K.png', bbox_inches='tight')
plt.show()
```

3 So sánh các mô hình

- Sau khi chọn được các bộ tham số tốt nhất cho mỗi mô hình, ta huấn luyện lại trên toàn bộ tập Train.
- Dùng các mô hình mới huấn luyện để phán đoán cho các dữ liệu trong tập Test
- Đo đạc Độ chính xác (Accuracy) của chúng và so sánh kết quả.

```
[ ]: svm_ = svm.SVC(kernel='linear', C=1.0)
rf = RandomForestClassifier(criterion='gini', n_estimators=300)
knn = KNeighborsClassifier(n_neighbors=5)

# Huấn luyện các mô hình trên tập dữ liệu train đầy đủ
svm_.fit(X_train, Y_train)
rf.fit(X_train, Y_train)
knn.fit(X_train, Y_train)
```

```
[ ]: KNeighborsClassifier()
```

```
[ ]: # Kết quả dự đoán trên tập test
print(f'SVM: {accuracy_score(Y_test, svm_.predict(X_test))}')
print(f'RF: {accuracy_score(Y_test, rf.predict(X_test))}')
print(f'KNN: {accuracy_score(Y_test, knn.predict(X_test))}')
```

SVM: 0.8590308370044053

RF: 0.801762114537445

KNN: 0.8414096916299559

```
[ ]: Y_pred_svm = svm_.predict(X_test)
cm_svm = confusion_matrix(Y_test, Y_pred_svm, labels=svm_.classes_)
disp_svm = ConfusionMatrixDisplay(confusion_matrix=cm_svm, display_labels=svm_.
    ↪classes_)
disp_svm.plot()
plt.show()
```

```
[ ]: Y_pred_rf = rf.predict(X_test)
cm_rf = confusion_matrix(Y_test, Y_pred_rf, labels=rf.classes_)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf, display_labels=rf.
    ↪classes_)
disp_rf.plot()
plt.show()
```

```
[ ]: Y_pred_knn = knn.predict(X_test)
cm_knn = confusion_matrix(Y_test, Y_pred_knn, labels=knn.classes_)
disp_knn = ConfusionMatrixDisplay(confusion_matrix=cm_knn, display_labels=knn.
    ↪classes_)
disp_knn.plot()
plt.show()
```


4 Bài tập

- Sử dụng dữ liệu đánh giá tín dụng cá nhân
- Sử dụng độ đo đánh giá negative cost
- Lựa chọn tham số cho các mô hình SVM, Random Forest và KNN
- So sánh các mô hình với siêu tham số tốt nhất

```
[ ]: data = np.genfromtxt('german.data-numeric')
X_data = data[:, :24]
Y_data = data[:, -1]
print(X_data.shape)
print(Y_data.shape)
```

```
(1000, 24)
(1000,)
```

```
[ ]: from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.
↪2, random_state=42)
print("Dữ liệu training = ", X_train.shape, Y_train.shape)
print("Dữ liệu testing = ", X_test.shape, Y_test.shape)
```

```
Dữ liệu training = (800, 24) (800,)
Dữ liệu testing = (200, 24) (200,)
```

```
[ ]: # Hàm tính neg_cost, dùng để truyền vào scoring của learning_curve
def neg_cost(estimator, X, y):
    y_true = y
    y_pred = estimator.predict(X)
    true_pos = ((y_true==y_pred)&(y_true==1.0))*0.0
    true_ne = ((y_true==y_pred)&(y_true==2.0))*0.0
    false_ne = ((y_true!=y_pred)&(y_true==1.0))*1.0
    false_pos = ((y_true!=y_pred)&(y_true==2.0))*5.0
    return -sum(true_pos + true_ne + false_pos + false_ne)/len(y_true)
```

```
[ ]: def cross_validation(estimator):
    _, train_scores, test_scores = learning_curve(estimator, X_train, Y_train,
↪cv=10, n_jobs=-1, train_sizes=[0.8, ], scoring=neg_cost)
    test_scores = test_scores[0]
    mean, std = test_scores.mean(), test_scores.std()
    return mean, std

def plot(title, xlabel, X, Y, error, ylabel = "neg cost"):
    plt.xlabel(xlabel)
    plt.title(title)
    plt.grid()
    plt.ylabel(ylabel)
```

```
plt.errorbar(X, Y, error, linestyle='None', marker='o')
```

4.1 SVM

4.2 Random Forest

4.3 KNN

4.4 So sánh

```
[ ]: svm_ = svm.SVC(kernel='poly', C=10)
      rf = RandomForestClassifier(criterion='entropy', n_estimators=50)
      knn = KNeighborsClassifier(n_neighbors=1)

      # Huấn luyện các mô hình trên tập dữ liệu train đầy đủ
      svm_.fit(X_train, Y_train)
      rf.fit(X_train, Y_train)
      knn.fit(X_train, Y_train)
```

```
[ ]: KNeighborsClassifier(n_neighbors=1)
```

```
[ ]: # Kết quả dự đoán trên tập test
      print(f'SVM: {neg_cost(svm_, X_test, Y_test)}')
      print(f'RF: {neg_cost(rf, X_test, Y_test)}')
      print(f'KNN: {neg_cost(knn, X_test, Y_test)}')
```

SVM: -1.105

RF: -0.795

KNN: -1.065