Tut 5: Sử dụng Naive Bayes Classifier với thư viện Scikit-Learn

- **Tác giả: Cao Chánh Dương.**
- **Nhóm nghiên cứu về AI của Trường Đại học Bách Khoa Thành phố Hồ Chí Minh.**

- **Tut này đề cập đến việc áp dụng Naive Bayes Classifier với 3 phân phối khác nhau:**
  - **Phân phối Gauss.**
  - **Phân phối Multinomial.**
  - **Phân phối Bernoulli.**
- **Mở rộng thêm, chúng ta sẽ tiến hành so sánh việc phân loại dữ liệu text giữa hai phương pháp đã được tiếp cận là k-NN(xem lại lab 1) và phương pháp trong lab 3 này là Naive Bayes Classifier.**

1. Cài đặt thư viện Scikit-Learn

- Tham khảo Tut-0.

2. Sử dụng Naive Bayes Classifier với phân phối Gauss

- Đoạn code sau đây sẽ import những thư viện cần thiết:

```
>>> import numpy as np
>>> from sklearn.naive_bayes import GaussianNB
```

- Trong đó dòng đầu là ta thực hiện import thư viện xử lý mảng quen thuộc(numpy), ở dòng số 2 ta import thư viện GaussianNB từ module sklearn.naive_bayes.
- Sau đó ta tạo một tập dữ liệu đầu vào cơ bản

```
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> Y = np.array([1, 1, 1, 2, 2, 2])
```

Trong đó, X chính là mảng dữ liệu đầu vào với 2 features, còn Y chính là các class tương ứng với từng dữ liệu trên (ở đây ta có 2 loại là class 1 và 2).

- Tiếp đến ta tạo model GaussianNB và fit model trên vào để training:

```
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
```

- Việc tạo model đã xong, giờ chúng ta thử predict một dữ liệu ngoài bất kì:

```
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

Ở đây ta thấy dữ liệu vừa nhập vào rơi vào class 1.

- Học viên tham khảo thêm tại link sau (nhằm thay đổi các thông số nếu muốn):

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

3. Sử dụng Naive Bayes Classifier với phân phối Multinomial/Bernoulli (trên tập dữ liệu Bắc Nam)

- Tập dữ liệu **Bắc Nam** và code mục này được tham khảo tại link sau:

  https://machinelearningcoban.com/2017/08/08/nbc/

- Học viên tham khảo mục 3.1 ở link trên để biết về tập dữ liệu **Bắc Nam**.

- Sau đó, ta tiến hành import các thư viện cơ bản (tương tự mục 2)

```
from __future__ import print_function
from sklearn.naive_bayes import MultinomialNB #or BernoulliNB
import numpy as np
```

- Ta tạo train data và label của chúng:

```
# train data
d1 = [2, 1, 1, 0, 0, 0, 0, 0, 0]
d2 = [1, 1, 0, 1, 1, 0, 0, 0, 0]
d3 = [0, 1, 0, 0, 1, 1, 0, 0, 0]
d4 = [0, 1, 0, 0, 0, 0, 1, 1, 1]

train_data = np.array([d1, d2, d3, d4])
label = np.array(['B', 'B', 'B', 'N'])
```

- Tạo test data:

```
# test data
d5 = np.array([[2, 0, 0, 1, 0, 0, 0, 1, 0]])
d6 = np.array([[0, 1, 0, 0, 0, 0, 0, 1, 1]])
```

- Tạo mô hình Multinomial và tiến hành train:

```
## call MultinomialNB or BernoulliNB
clf = MultinomialNB()
# training
clf.fit(train_data, label)
```

- Khi việc train đã xong, ta thử predict để test kết quả:

```
# test
print('Predicting class of d5:', str(clf.predict(d5)[0]))
print('Probability of d6 in each class:', clf.predict_proba(d6))
```

- Kết quả:

```
Predicting class of d5: B
Probability of d6 in each class: [[ 0.29175335  0.70824665]]
```

- Ở đây ta có thể thấy mô hình đã dự đoán dữ liệu d5 là thuộc class B, và ta cũng tiến hành kiểm tra dữ liệu d6 với cách tương tự, ngoài ra ta còn có thể biết được xác suất tương ứng mà dữ liệu đó rơi vào mỗi class. Chúng ta thấy thư viện Scikit-Learn thật là "vi diệu" đúng không nào ?

4. Ứng dụng Naive Bayes vào việc phân loại text dựa trên vector tf-idf

- Ở đây, học viên sẽ được yêu cầu sử dụng NB vào tập dữ liệu đã crawl về được từ Lab 1 và so sánh kết quả giữa 2 phương pháp để thấy được sự khác biệt giữa hai phương pháp.
- Trong mục này sẽ hướng dẫn thực hiện NB phân loại text trên một ví dụ mẫu, ví dụ được tham khảo tại link sau: http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/applications/plot_topic_classication.html#balancing-the-class-before-classification
- Đầu tiên, chúng ta tiến hành import các thư viện cần thiết:

```
from collections import Counter

from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import make_pipeline as make_pipeline_imb
from imblearn.metrics import classification_report_imbalanced
```

- Trong đó, **fetch_20newsgroups** chính là tập dữ liệu ví dụ mà ta sẽ thực hiện, TfidfVectorizer là thư viện để tạo vector tf-idf mà ta đã làm quen trong Lab 1.
- Thư viện MultinomialNB đã đề cập ở mục trên, ngoài ra ta cần sử dụng Scikit-Learn make_pineline để feed các vector tf-idf vào mô hình phân phối Naive Bayes tương ứng.
- RandomUnderSampler, make_pineline_imb là các thư viện dùng để cân bằng lại các dữ liệu bị mất cân bằng (imbalanced data).
- Sau đó ta tiến hành load dữ liệu, chia dữ liệu train, test, tạo vector class với 4 phần tử tương ứng với 4 chủ đề:

```
categories = ['alt.atheism', 'talk.religion.misc',
              'comp.graphics', 'sci.space']
newsgroups_train = fetch_20newsgroups(subset='train',
                                      categories=categories)
newsgroups_test = fetch_20newsgroups(subset='test',
                                     categories=categories)

X_train = newsgroups_train.data
X_test = newsgroups_test.data

y_train = newsgroups_train.target
y_test = newsgroups_test.target
```

- Đầu tiên chúng ta tính toán với dữ liệu mà không cân bằng lại chúng (cách thông thường):

```
pipe = make_pipeline(TfidfVectorizer(), MultinomialNB())
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)

print(classification_report_imbalanced(y_test, y_pred))
```

- Kết quả cho ta xác suất của các bài báo rơi vào chủ đề tương ứng (tương ứng với **class 0,1,2,3**).

pre	rec	spe	f1	geo	iba	sup
0	0.67	0.94	0.86	0.79	0.81	0.64	319
1	0.96	0.92	0.99	0.94	0.97	0.93	389
2	0.87	0.98	0.94	0.92	0.93	0.85	394
3	0.97	0.36	1.00	0.52	0.92	0.85	251
avg / total	0.87	0.84	0.94	0.82	0.91	0.82	1353

- Tuy nhiên ở đây ta thấy rằng class 3 recall của nó không được ổn định (các bạn tham khảo định nghĩa recall tại đây: https://en.wikipedia.org/wiki/Precision_and_recall).
- Cách làm số 2,ta sẽ sử dụng các công cụ đã nhắc đến ở trên để cân bằng lại các imbalanced data:

```
pipe = make_pipeline_imb(TfidfVectorizer(),
                         RandomUnderSampler(),
                         MultinomialNB())

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
```

- Kết quả ở cách này cho recall tốt hơn cách 1:

pre	rec	spe	f1	geo	iba	sup
0	0.70	0.92	0.88	0.79	0.82	0.66	319
1	0.98	0.85	0.99	0.91	0.96	0.92	389
2	0.95	0.91	0.98	0.93	0.95	0.91	394
3	0.83	0.72	0.97	0.77	0.88	0.77	251
avg / total	0.87	0.86	0.96	0.86	0.91	0.83	1353

- Học viên tiến hành so sánh kết quả này với kết quả đã thực hiện ở Lab 1.