Bài toán phân loại sử dụng SVM

Mục tiêu:

- Xây dựng được mô hình svm sử dụng thư viện sklearn.
- Ứng dụng, hiểu cách áp dụng mô hình SVM vào giải quyết bài toán thực tế (Ví dụ: phân loại văn bản)
- Sử dụng độ đo Accuracy để làm độ đo đánh giá chất lượng mô hình.

Dữ liệu:

- Có tập các văn bản và nhãn tương ứng của từng văn bản trong một khoảng thời gian
- Tập các nhãn (10 nhãn khác nhau):
    - Giải trí, Khoa học - Công nghệ, Kinh tế, Pháp luật, Sức khỏe, Thể thao, Thời sự, Tin khác, Độc giả, Đời sống - Xã hội
- Ví dụ văn bản nhãn **thể thao**:
    - "Dân_trí Real Madrid đã dẫn trước trong cả trận đấu , nhưng họ vẫn phải chấp_nhận bị Dortmund cầm hòa 2-2 ở Bernabeu . Real Madrid chấp_nhận đứng thứ_hai ở bảng F Champions League ..."

Bài Làm

Mô hình SVM

- ❖ SVM là một mô hình học có giám sát, được sử dụng chủ yếu cho bài toán **phân loại** nhưng cũng có thể áp dụng cho bài toán **hồi quy.**
- ❖ Mục tiêu chính của SVM là tìm một **siêu phẳng** (hyperplane) tối ưu để phân tách các lớp dữ liệu.

Siêu phẳng:

- Siêu phẳng: Siêu phẳng là một khái niệm trong hình học, dùng để chỉ một không gian con có số chiều ít hơn 1 so với không gian chứa nó

Trong SVM siêu phẳng được sử dụng để phân tích các lớp của dữ liệu

- Trong không gian 2D (mặt phẳng), siêu phẳng là một **đường thẳng.**
- Trong không gian 3D, siêu phẳng là một **mặt phẳng.**
- Trong không gian n chiều, siêu phẳng là một không gian con có số chiều n−1.

Phân tách tuyến tính:

- Một tập dữ liệu được gọi là phân tách tuyến tính nếu tồn tại một đường thẳng (hoặc siêu phẳng) có thể phân chia tất cả các điểm dữ liệu thuộc các lớp khác nhau mà không có điểm nào bị phân loại sai.

- **Trong không gian 2D**: Phân tách tuyến tính có nghĩa là bạn có thể vẽ một đường thẳng để phân chia hai lớp dữ liệu.
- **Trong không gian 3D**: Phân tách tuyến tính có nghĩa là bạn có thể vẽ một mặt phẳng để phân chia hai lớp dữ liệu.
- **Trong không gian n*n* chiều**: Phân tách tuyến tính có nghĩa là bạn có thể tìm một siêu phẳng (n−1*n*−1 chiều) để phân chia hai lớp dữ liệu.

1. Giới thiệu về dữ liệu

Dữ liệu sử dụng trong bài toán là các văn bản tiếng Việt đã được gán nhãn thuộc một trong 10 chủ đề khác nhau, bao gồm:

Giải trí

Khoa học - Công nghệ

Kinh tế

Pháp luật

Sức khỏe

Thể thao

Thời sự

Tin khác

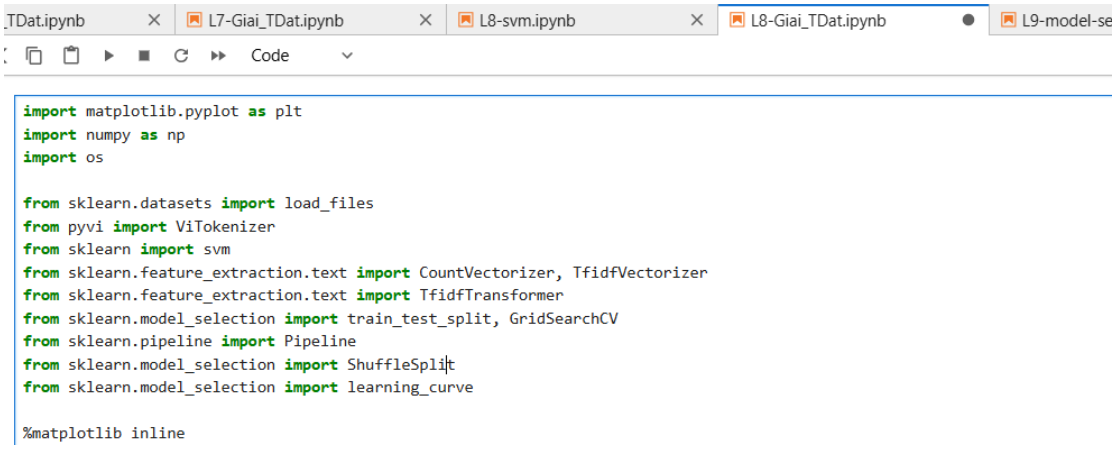
Độc giả

Đời sống - Xã hội

Dữ liệu này sẽ được sử dụng để huấn luyện mô hình Support Vector Machine (SVM) nhằm phân loại văn bản dựa trên nội dung.

2. Tải dữ liệu

Cài thư viện để sử dụng



sklearn.datasets.load_files: Đọc dữ liệu văn bản từ thư mục.

sklearn.feature_extraction.text: Chuyển đổi văn bản thành ma trận số.

sklearn.model_selection.train_test_split: Chia dữ liệu thành train và test.

sklearn.svm.SVC: Mô hình Support Vector Machine.

sklearn.metrics.accuracy_score: Đánh giá mô hình.

Tìm hiểu bộ dữ liệu của bài toán:

```
# Định nghĩa đường dẫn tới dữ liệu
DATA_PATH = "data/news_vnexpress/"

# In thông tin về số lượng văn bản trong từng nhãn
header = "%-20s%-30s" % ("Số lượng văn bản", "Nhãn")
print(header)
print("-" * 50)

total = 0
for label in os.listdir(DATA_PATH):
```

```
    n = len(os.listdir(os.path.join(DATA_PATH, label)))
    total += n
    entry = "%-20d%-30s" % (n, label)
    print(entry)
print("-" * 50)
print(f"Tổng số văn bản: {total}")

# Load dữ liệu văn bản
data_train = load_files(container_path=DATA_PATH, encoding="utf-8")

# Hiển thị danh sách các nhãn
print(data_train.target_names)
```

Kết quả

```
Số lượng văn bản    Nhãn
--------------------------------------------------
120                 doi-song
54                  du-lich
201                 giai-tri
105                 giao-duc
144                 khoa-hoc
262                 kinh-doanh
59                  phap-luat
162                 suc-khoe
173                 the-thao
59                  thoi-su
--------------------------------------------------
Tổng số văn bản: 1339
['doi-song', 'du-lich', 'giai-tri', 'giao-duc', 'khoa-hoc', 'kinh-doanh', 'phap-luat', 'suc-khoe', 'the-thao', 'thoi-su']
```

load_files() tự động gán nhãn dựa trên thư mục chứa văn bản.

data_train.target_names là danh sách nhãn có trong tập dữ liệu.

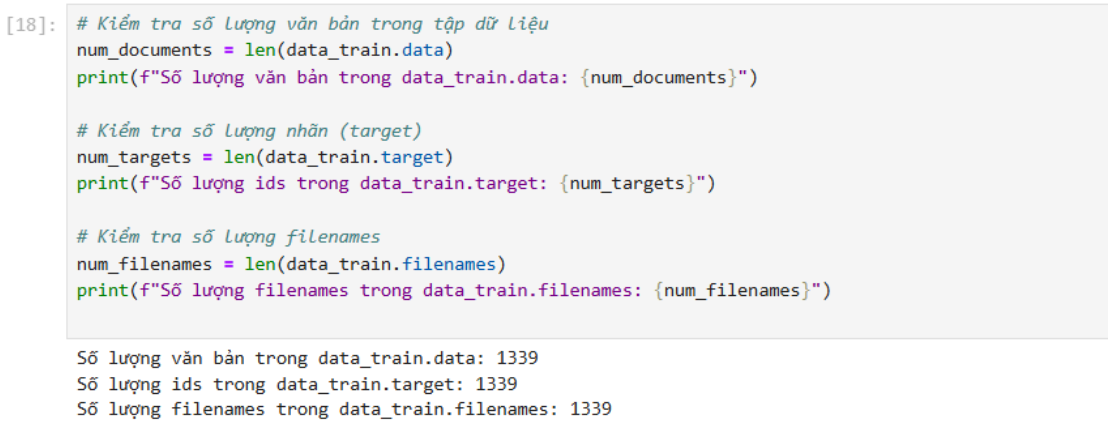
Bài tập

- Kiểm tra các thông tin sau:
    + Số lượng văn bản trong data_train.data
    + Số lượng ids trong data_train.target
    + Số lượng filenames trong data_train.filenames

# Kiểm tra số lượng văn bản trong tập dữ liệu
num_documents = len(data_train.data)
print(f"Số lượng văn bản trong data_train.data: {num_documents}")

# Kiểm tra số lượng nhãn (target)
num_targets = len(data_train.target)
print(f"Số lượng ids trong data_train.target: {num_targets}")

# Kiểm tra số lượng filenames
num_filenames = len(data_train.filenames)
print(f"Số lượng filenames trong data_train.filenames: {num_filenames}")



Giải thích:

- len(data_train.data): Số lượng văn bản có trong tập dữ liệu.
- len(data_train.target): Số lượng nhãn tương ứng với từng văn bản.
- len(data_train.filenames): Số lượng đường dẫn tới các tệp tin chứa văn bản.

3. Tiền xử lý dữ liệu

Tính đầy đủ: Dữ liệu đầy đủ và không bị thiếu bất kỳ một thuộc tính nào.

Tính trung thực: Dữ liệu được sử dụng có nguồn đáng tin cậy.

Tính đồng nhất: Dữ liệu đã đồng nhất.

Cấu trúc: Dữ liệu có cấu trúc rõ ràng.

```
# Load dữ liệu các stopwords
with open("D:/L2-ML-code/data/vietnamese-stopwords.txt", encoding='utf-8') as f:
    stopwords = f.readlines()
stopwords = [x.strip().replace(" ", "_") for x in stopwords]
print(f"Tổng số lượng từ dừng: {len(stopwords)}")
print("Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại): ", stopwords[:10])
print()

"""
Chuyển hoá dữ liệu text về dạng vector tfidf
    - loại bỏ từ dừng
    - sinh từ điển
    - chuyển thành dữ liệu dạng ma trận 2 chiều kích thước n x m với n là số lượng văn bản và m là số lượng từ trong từ điển
"""
module_count_vector = CountVectorizer(stop_words=stopwords)
model_rf_preprocess = Pipeline([('vect', module_count_vector),
                  ('tfidf', TfidfTransformer()),
                  ])
data_preprocessed = model_rf_preprocess.fit_transform(data_train.data, data_train.target)
print("5 từ đầu tiên trong từ điển:\n")
i = 0
for k,v in module_count_vector.vocabulary_.items():
    i+=1
    print(i, ": ", (k, v))
    if i > 5:
        break
print()

# Số chiều của dữ liệu
print(f"Số chiều của dữ liệu: {data_preprocessed.shape}")
print(f"Số từ trong từ điển: {len(module_count_vector.vocabulary_)}")
```

```
Tổng số lượng từ dừng: 2063
Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại):  ['a_lô', 'a_ha', 'ai', 'ai_ai', 'ai_nấy', 'ai_đó', 'alô', 'amen', 'anh', 'anh_ấy']

5 từ đầu tiên trong từ điển:

1 :  ('tổ', 11850)
2 :  ('chức', 2866)
3 :  ('tế', 11819)
4 :  ('giới', 4722)
5 :  ('who', 12513)
6 :  ('ước', 13181)
```

Chuyển dữ liệu từ dạng text về dạng ma trận bằng TF-IDF

```
from sklearn.model_selection import ShuffleSplit

# chia dữ liệu thành 2 phần sử dụng hàm train_test_split.
test_size = 0.2
# cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
X_train, X_test, y_train, y_test = train_test_split( data_preprocessed, data_train.target, test_size=test_size)


# hiển thị một số thông tin về dữ liệu
print("Dữ liệu training = ", X_train.shape, y_train.shape)
print("Dữ liệu testing = ", X_test.shape, y_test.shape)
```

```
Dữ liệu training =  (928, 13227) (928,)
Dữ liệu testing =  (233, 13227) (233,)
```

Chia dữ liệu thành 2 phần dùng để huấn luyện và kiểm tra theo tỉ lệ 80:20

80% dữ liệu dùng để huấn luyện và 20% dữ liệu dùng để kiểm tra

4. Ứng dụng mô hình SVM vào phân loại văn bản

```
print("- Training ...")


# X_train.shape
print("- Train size = {}".format(X_train.shape))
model = svm.SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)

print("- model - train complete")
```

```
- Training ...
- Train size = (928, 13227)
- model - train complete
```

- **svm.SVC(kernel='linear', C=1.0)**: Dùng SVM với kernel tuyến tính.
- **model.fit(X_train, y_train)** để huấn luyện mô hình.

Huấn luyện mô hình SVM với 928 số mẫu huấn luyện và 13227 đặc trưng.

5. Đánh giá chất lượng mô hình bằng độ đo Accuracy

```
# Dự đoán
y_pred = model.predict(X_test)

# Tính độ chính xác
accuracy = accuracy_score(y_test, y_pred)
print(f"Độ chính xác của mô hình: {accuracy:.4f}")
```

```
Độ chính xác của mô hình: 0.9179
```

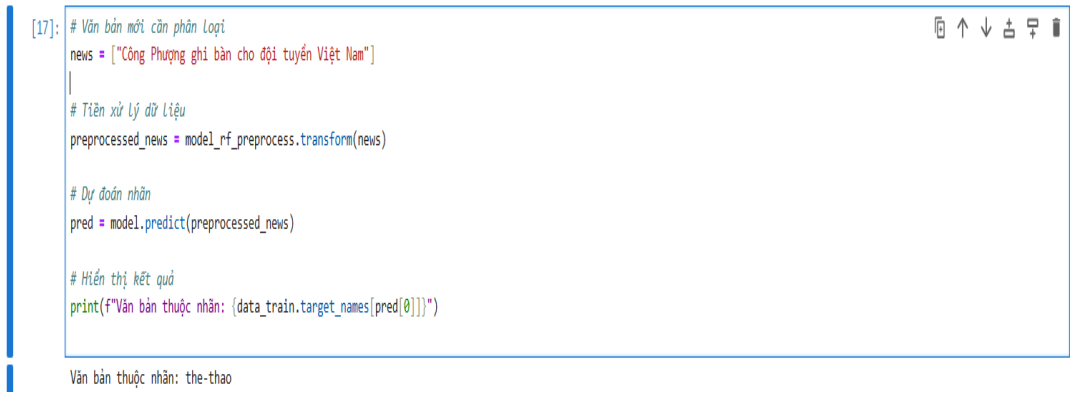
model.predict(X_test) để dự đoán.

accuracy = tổng số văn bản dự đoán đúng / tổng số văn bản có trong tập test_data

```
Độ chính xác của mô hình: 0.9179
```

Mô hình có độ chính xác là 91.8%

6. Dự đoán nhãn cho 1 văn bản mới



```
[17]: # Văn bản mới cần phân loại
news = ["Công Phượng ghi bàn cho đội tuyển Việt Nam"]

# Tiền xử lý dữ liệu
preprocessed_news = model_rf_preprocess.transform(news)

# Dự đoán nhãn
pred = model.predict(preprocessed_news)

# Hiển thị kết quả
print(f"Văn bản thuộc nhãn: {data_train.target_names[pred[0]]}")

Văn bản thuộc nhãn: the-thao
```

Chuyển đổi văn bản mới về dạng vector bằng model_rf_preprocess.transform(news).

Dự đoán nhãn bằng model.predict()`.

Hiển thị kết quả theo danh sách data_train.target_names.