ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

KTDL | BÀI 9: SỰ PHÂN LỚP DỮ LIỆU

Sinh viên thực hiện: Nguyễn Công Hoài Nam
Mã số sinh viên: 21280099

Ngày 9 tháng 6 năm 2024

MỤC LỤC

1. Thông tin về dữ liệu

Tập dữ liệu **Banknote Authentication** gồm các đặc trưng được trích xuất từ các hình ảnh được chụp xác thực các tờ tiền giấy (hợp lệ hay giả mạo)

Chúng bao gồm các đặc trưng:

- **Variance**: độ biến thiên
- **Skewness**: độ lệch
- **Curtosis**: độ nhọn
- **Entrophy**: mức độ hỗn loạn
- **Class**: nhãn đầu ra (1: hợp lệ, 0: giả mạo)

Ta sẽ import data từ **UCI Machine Learning Repository** bằng thư viện **ucimlrepo** của họ

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

from ucimlrepo import fetch_ucirepo

# fetch dataset
banknote_authentication = fetch_ucirepo(id=267)

# data (as pandas dataframes)
X = banknote_authentication.data.features
y = banknote_authentication.data.targets
df = pd.concat([X, y], axis=1)
df.head()
```

	variance	skewness	curtosis	entropy	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

2. Kiểm tra tính tuyến tính của dữ liệu

Để kiểm tra dữ liệu có tuyến tính hay không ta sẽ vẽ ma trận tương quan để kiểm tra sự tương của các biến

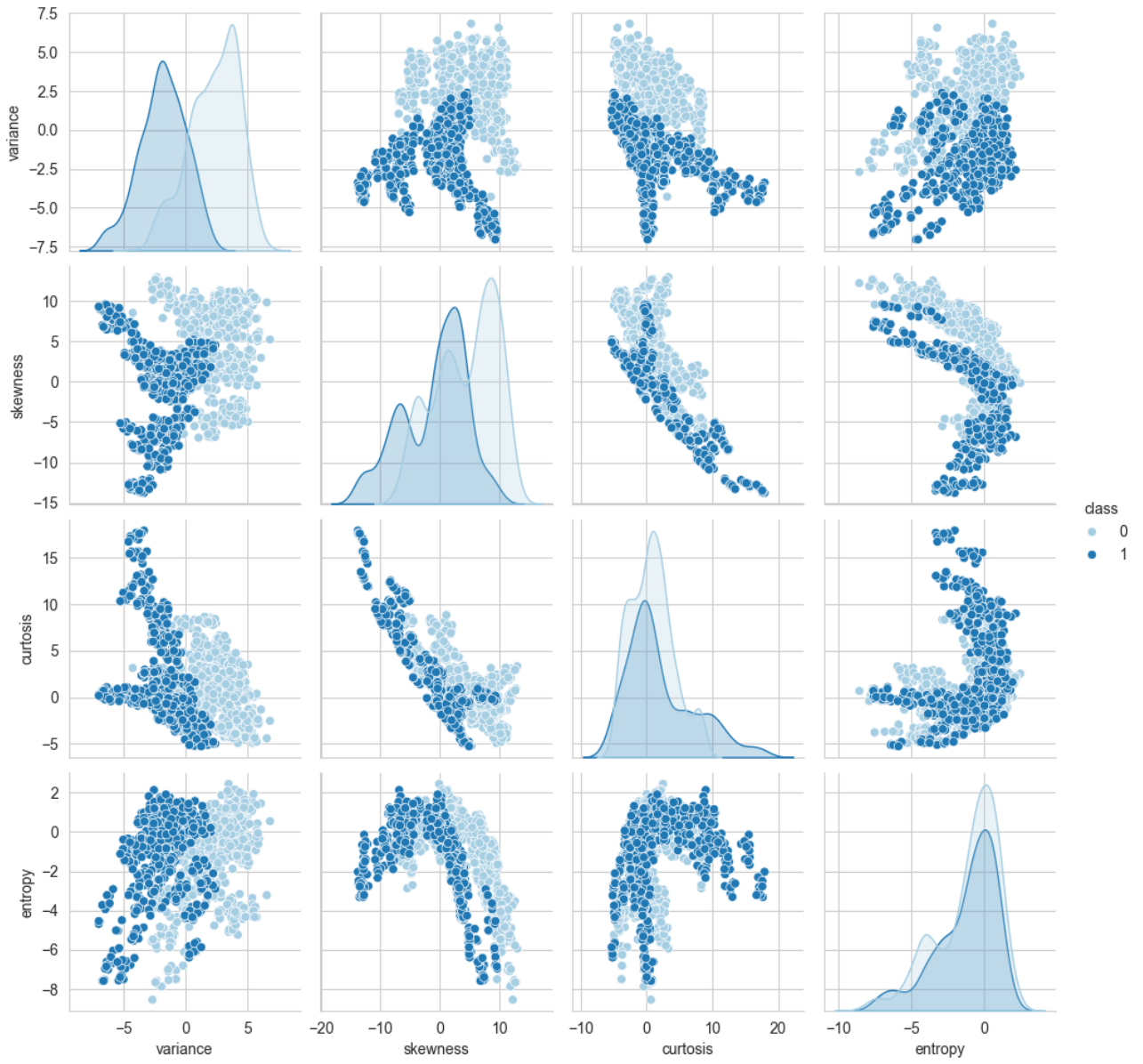
```
plt.figure(figsize=(6,6))
```

```
sns.heatmap(df.corr(), cmap="Blues_r", annot=True, fmt='.2',linewidths=.5, cbar=False)
# Show the plot
plt.show()
```

- Những cặp feature có tương quan gần cao (gần với -1 hoặc 1) thì chúng tuyến tính, ngược lại chúng phi tuyến.
- Ta có có thể thấy một số cặp feature có sự tuyến tính như (**sknewness** với **curtoisis**) nhưng cũng có nhiều cặp features phi tuyến như (**curtosis** và **class**).
- Ta kết luận bộ dữ liệu không hoàn toàn tuyến tính

Để kiểm chứng lại nhận định của mình ta sẽ vẽ scatter plot của từng cặp biến trong tập dữ liệu

```
sns.pairplot(df, hue = 'class', palette="Paired")
```



Dễ dàng thấy ở nhiều cặp biến các điểm dữ liệu không phân tán theo một đường thẳng nào => dữ liệu không tuyến tính

3. Xây dựng mô hình

3.1. Import các thư viện cần thiết

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.pipeline import make_pipeline
```

3.2. Chia dữ liệu thành tập train và test

```
# Split dataset
X = X.values
y = y.values.ravel()
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.3. Chuẩn hoá dữ liệu

```
# Standardization
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3.4. Hyperparameter tuning

Bởi vì bộ dữ liệu không tuyến tính hoàn toàn nên ta sẽ sử dụng kernel **rbf** dành cho data phi tuyến
Để tìm kiếm bộ siêu tham số tốt nhất cho model ta sẽ sử dụng thuật toán **GridSearchCV** để tìm ra tham số tốt nhất cho bài toán này

```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

# Sử dụng GridSearchCV để tìm best hyperparameters cho model với kernel = 'rbf'
svc = SVC()
best_svc = GridSearchCV(svc, param_grid)
best_svc.fit(X_train, y_train)
print(f"Best param: {best_svc.best_params_}")
```

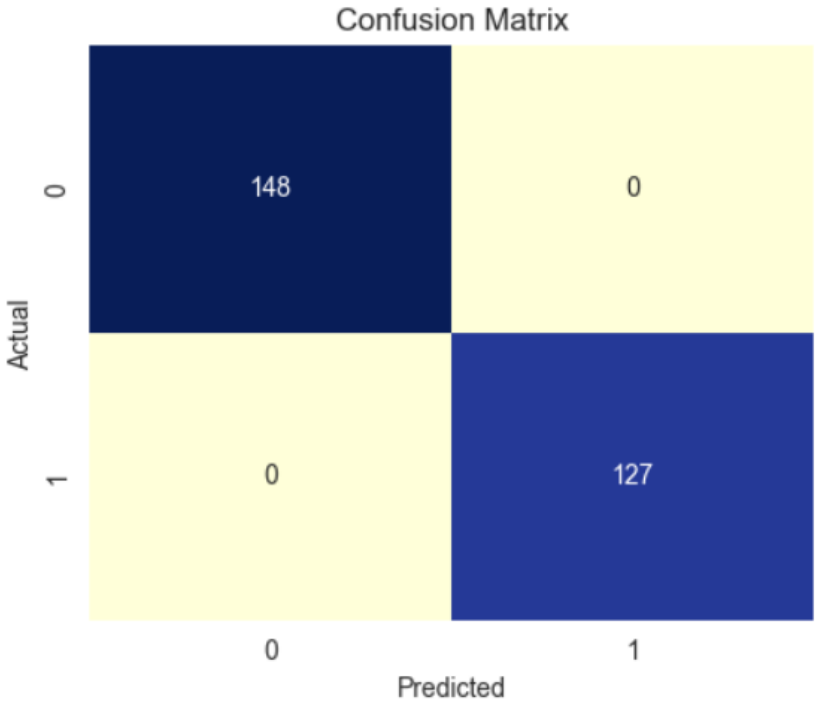
Kết quả bộ tham số tốt nhất là:
Best param: 'C': 10, 'gamma': 0.1, 'kernel': 'rbf'

3.5. Evaluate

Để đánh giá mô hình tốt hay tệ, ta sẽ đánh giá nó qua các metric, cụ thể ở đây là **confusion matrix** và **classification report** thể hiện thông số ở các class khác nhau

```
def evaluate(y_true, y_pred):

    confusion = confusion_matrix(y_true, y_pred)
    report = classification_report(y_true, y_pred)

    print("Classification report:\n", report)
    print("Confusion Matrix")

    plt.figure(figsize=(5, 4))
    sns.heatmap(confusion, annot=True, fmt='d', cmap='YlGnBu', cbar=False)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()

y_pred = best_svc.predict(X_test)
evaluate(y_test,y_pred)
```

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       148
           1       1.00      1.00      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275

Confusion Matrix
```



Kết quả cho thấy model mà ta xây dựng có điểm số tuyệt đối.

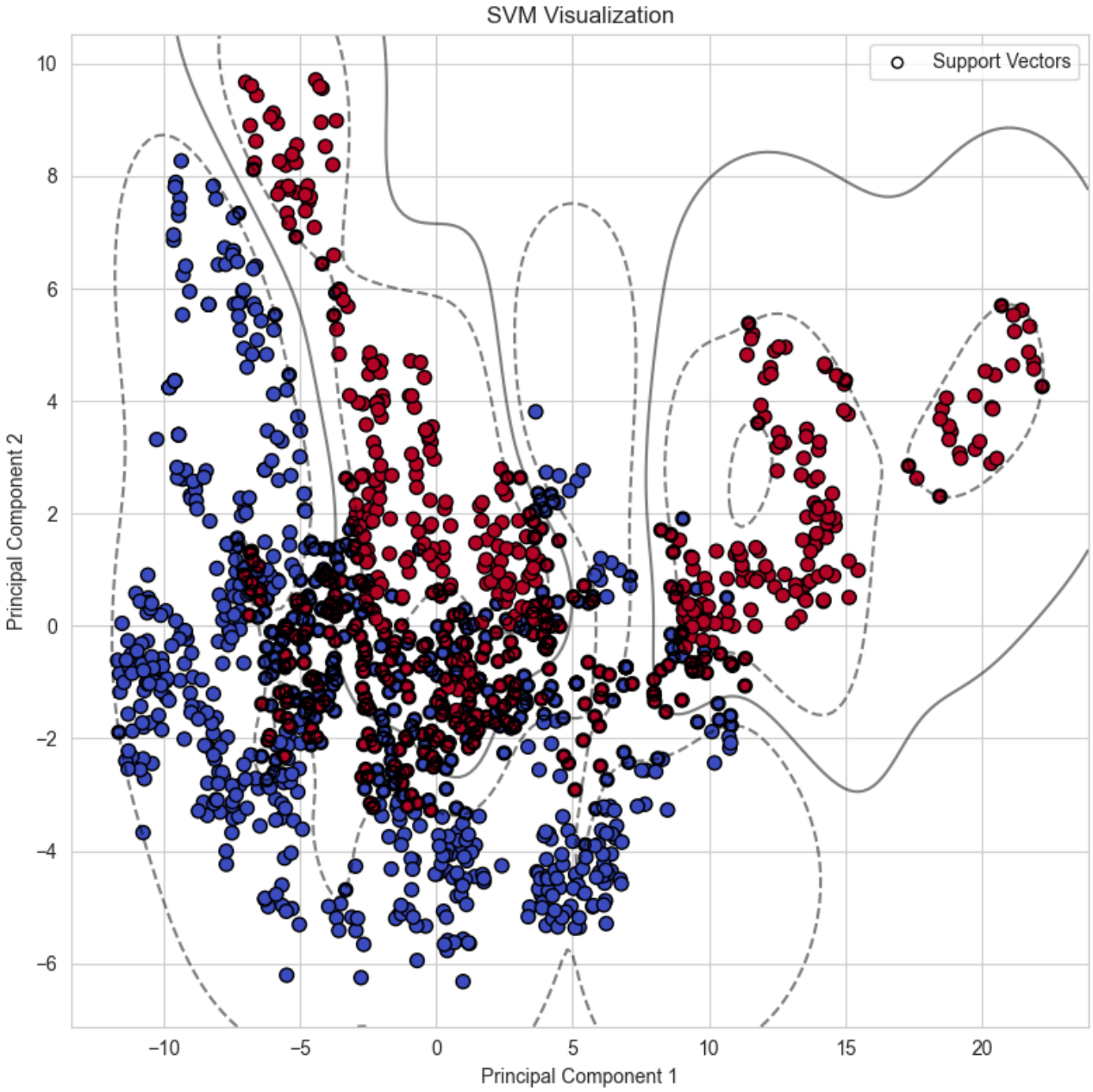
4. Trực quan hoá (vẽ hình)

Bởi vì bộ data này là dữ liệu nhiều chiều nên để có thể vẽ hình, thì ta chỉ có thể chọn 2 biến trong dataset để thể hiện.
Ở đây em chọn giảm chiều dữ liệu xuống còn 2 chiều để có cái nhìn tổng quát về dữ liệu

```python
# Hàm tạo mô hình SVC với PCA
def make_2D_SVC(X, y):
    pipeline = make_pipeline(
        PCA(n_components=2),  # PCA data xuống còn 2 chiều
        SVC(C=10, gamma=0.1, kernel='rbf')  # SVC với best hyperparameter tuning
    )
    pipeline.fit(X, y)
    X_2D = pipeline.named_steps['pca'].transform(X)
    model = pipeline.named_steps['svc']
    return X_2D, model
```

Hàm vẽ hình gồm scatterplot, các đường biên, ...

```python
def visualize_2d_svm(X, y):
    X_2D, model = make_2D_SVC(X,y)
    plt.figure(figsize=(8, 8))
    plt.scatter(X_2D[:, 0], X_2D[:, 1], c=y, s=50, cmap='coolwarm', edgecolor='k')

    # Lấy giới hạn của biểu đồ
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Tạo lưới để đánh giá mô hình
    x = np.linspace(xlim[0], xlim[1], 200)
    y = np.linspace(ylim[0], ylim[1], 200)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # Vẽ đường biên quyết định và lề
    ax.contour(X, Y, P, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])

    # Vẽ các support vectors
    ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1], s=30, linewidth=1, edgecolors='black', facecolors='none', label='Support Vectors')

    # Thêm nhãn trục và tiêu đề
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.title('SVM Visualization')
    plt.legend()
    plt.tight_layout()
    plt.show()

visualize_2d_svm(X,y)
```



Hình vẽ cho thấy sự phân hoá khá rõ rệt

5. Kết luận

Trong bài tập này, em đã áp dụng thuật toán **`SVM`** lên bộ dữ liệu **`Banknote Authentication`**. Các bước thực hiện bao gồm kiểm tra tính tuyến tính của dữ liệu, tuning siêu tham số, xây dựng mô hình, và trực quan hóa kết quả. Qua bài tập, em đã hiểu rõ hơn về thuật toán **`SVM`** và cách áp dụng nó vào thực tế.