Ex 1: Bill Authentication

Cho dữ liệu bill_authentication.csv

Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán SVM để thực hiện việc dự đoán Class (1 hay 0) dựa trên thông tin được cung cấp

1. Đọc dữ liệu. Tiền xử lý dữ liệu nếu cần. Trực quan hóa dữ liệu.
2. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.2
3. Áp dụng thuật toán SVM
4. Tìm kết quả
5. Kiểm tra độ chính xác
6. X_new = [[3.2, -2.1, 1.7, 0.1], [-2.9297, -5.0816, 9.0958, -1.0]], hãy cho biết y_new

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/Practice_2023/Chapter6_SVM/'
```

In [3]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

In [4]:
```
bankdata = pd.read_csv("bill_authentication.csv")
```

In [5]:
```
bankdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
Variance    1372 non-null float64
Skewness    1372 non-null float64
Curtosis    1372 non-null float64
Entropy     1372 non-null float64
Class       1372 non-null int64
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
#sns.pairplot(bankdata)
```

In [8]:
```
# Class: có giá trị là 0 và 1
X = bankdata[["Variance", "Skewness","Curtosis","Entropy"]]
y = bankdata["Class"]
```

In [9]:
```
X.head()
```

Out[9]:

	Variance	Skewness	Curtosis	Entropy
0	3.62160	8.6661	-2.8073	-0.44699
1	4.54590	8.1674	-2.4586	-1.46210
2	3.86600	-2.6383	1.9242	0.10645
3	3.45660	9.5228	-4.0112	-3.59440
4	0.32924	-4.4552	4.5718	-0.98880

In [10]:
```
y.head()
```

Out[10]:
```
0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int64
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

In [12]:
```
clf = svm.SVC(kernel='linear')  #...
clf.fit(X_train, y_train)
```

Out[12]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [13]:
```
y_pred = clf.predict(X_test)
```

In [14]:
```
#y_pred
```

In [15]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  98.9090909090909 %
```

In [16]:
```
# Kiểm tra độ chính xác
print("The Train/ Score is: ",
      clf.score(X_train,y_train)*100,"%")
print("The Test/ Score accuracy is: ",
      clf.score(X_test,y_test)*100,"%")
```

```
The Train/ Score is:  98.90610756608933 %
The Test/ Score accuracy is:  98.9090909090909 %
```

In [17]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[152   2]
 [  1 120]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       154
           1       0.98      0.99      0.99       121

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

Kết quả:

- R^2 của cả train và test đề cao và tương đối như nhau
- Precision và recall đều cao
- => Model phù hợp

In [18]:
```
X_new = [[3.2, -2.1, 1.7, 0.1], [-2.9297, -5.0816, 9.0958, -1.0]]
```

In [19]:
```
y_new = clf.predict(X_new)
y_new
```

Out[19]:
```
array([0, 1], dtype=int64)
```