ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

NGÀNH: KHOA HỌC MÁY TÍNH

MÔN HỌC: LẬP TRÌNH PYTHON CHO MÁY HỌC

KERNEL SUPPORT VECTOR MACHINE

GVHD: TS.Nguyễn Vĩnh Tiệp
SV: Phạm Trung Hiếu - 19521512
Võ Khoa Nam - 19521877
Trịnh Minh Hoàng - 19521547

Contents

1 Giới thiệu

1.1 SVM là gì ?

Support vector machine (SVM) là một khái niệm trong thống kê và khoa học máy tính cho một tập hợp những phương pháp học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy. SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Do đó SVM là một thuật toán phân loại nhị phân. Với một bộ các ví dụ luyện tập thuộc hai thể loại cho trước, thuật toán luyện tập SVM xây dựng một mô hình SVM để phân loại các ví dụ khác vào hai thể loại đó. Một mô hình SVM là một cách biểu diễn các điểm trong không gian và lựa chọn ranh giới giữa hai thể loại sao cho khoảng cách từ các ví dụ luyện tập tới ranh giới là xa nhất có thể. Các ví dụ mới cũng được biểu diễn trong cùng một không gian và được thuật toán dự đoán thuộc một trong hai thể loại tùy vào ví dụ đó nằm ở phía nào của ranh giới.

1.2 Các thành phần trong SVM

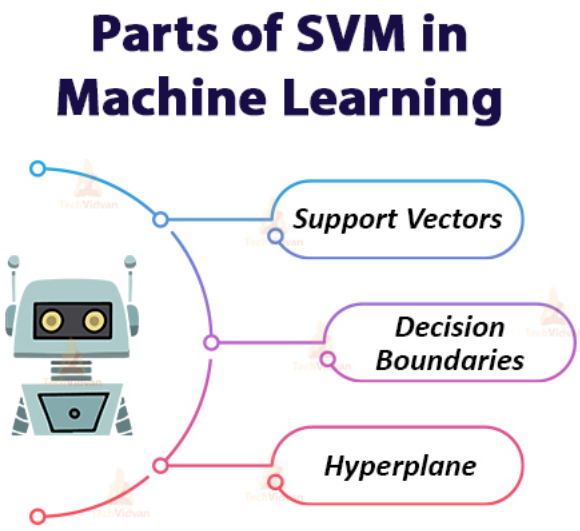

Figure 1: Các thành phần trong SVM

1.2.1 Support Vector

Các vectơ hỗ trợ là các điểm dữ liệu đặc biệt trong tập dữ liệu. Chúng chịu trách nhiệm về việc xây dựng siêu phẳng và là những điểm gần siêu phẳng nhất. Nếu những điểm này bị loại bỏ, vị trí của siêu phẳng sẽ bị thay đổi. Siêu phẳng có các ranh giới quyết định xung quanh nó.

Các vectơ hỗ trợ giúp giảm và tăng kích thước của các ranh giới. Chúng là những thành phần chính trong việc tạo ra một SVM.

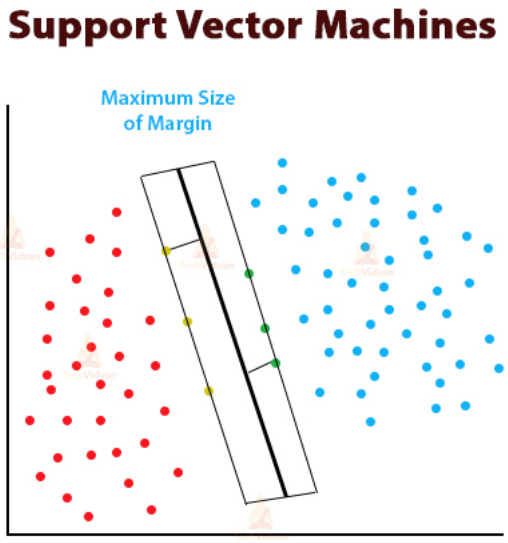

Figure 2: Support vector

Trong figure 2, các điểm màu vàng và màu xanh lá cây ở đây là các vector hỗ trợ. Các điểm màu xanh lam và đỏ là các lớp riêng biệt. Đường tối ở giữa là siêu phẳng trong 2-D và hai đường dọc theo siêu phẳng là ranh giới quyết định. Chúng tạo thành bề mặt quyết định chung.

1.2.2 Hyperplane

Hyperplane (siêu phẳng) trong không gian mới được định nghĩa là tập hợp các điểm có tích vô hướng với một vectơ cố định trong không gian đó là một

hằng số. Vectơ xác định một siêu phẳng sử dụng trong SVM là một tổ hợp tuyến tính của các vectơ dữ liệu luyện tập trong không gian mới. Siêu phẳng là không gian con có chiều nhỏ hơn một chiều so với không gian xung quanh nó.

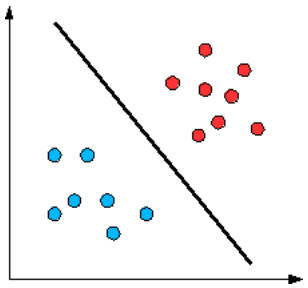

Figure 3: Hyperplane

Figure 3, siêu phẳng của không gian hai chiều là một đường một chiều phân chia các điểm xanh và đỏ.

1.2.3 Decision boudaries

Decision boundaries (ranh giới quyết định) là hai đường thẳng được tạo ra từ các điểm vector hỗ trợ, dọc theo siêu phẳng chính là ranh giới quyết định trong SVM. Khoảng cách giữa hai vạch đậm nhạt gọi là biên độ. Một dạng siêu phẳng tối ưu hoặc tốt nhất khi kích thước ký quỹ là tối đa. Thuật toán SVM điều chỉnh siêu phẳng và lề của nó theo các vectơ hỗ trợ.

1.3 SVM hoạt động như thế nào ?

Cách hoạt động của SVM được hiểu đơn giản là. SVM là thuật toán tìm các điểm gần nhau nhất, sau đó nó sẽ vẽ một đường thẳng giữa chúng (trong không gian 2 chiều). Đường thẳng sẽ là tối ưu nế nó vuông góc với đường nối các điểm dữ liệu. Phần tốt nhất là hai lớp khác nhau được hình thành ở hai bên của đường thẳng. Bất kỳ điểm dữ liệu mới nào đi vào tập dữ liệu nó sẽ không ảnh hưởng đến siêu mặt phẳng này. Các điểm duy nhất sẽ ảnh hưởng đến siêu phẳng là các vectơ hỗ trợ. Siêu phẳng sẽ không cho phép dữ liệu từ cả hai lớp trộn lẫn trong hầu hết các trường hợp. Ngoài ra, siêu phẳng có thể tự điều chỉnh bằng cách tối đa hóa kích thước của lề của nó. Lề là không gian giữa siêu phẳng và các ranh giới quyết định.

1.4 Kernel SVM

Kernel hoặc phương thức hạt nhân (còn gọi là hàm Kernel) là tập hợp các loại thuật toán khác nhau đang được sử dụng để phân tích mẫu. Chúng được sử dụng để giải quyết một vấn đề phi tuyến tính bằng cách sử dụng một bộ phân loại tuyến tính. Phương thức Kernels được sử dụng trong SVM (Máy vectơ hỗ trợ) được sử dụng trong các bài toán phân loại và hồi quy. SVM sử dụng cái được gọi là "Kernel Trick", nơi dữ liệu được chuyển đổi và một ranh giới tối ưu được tìm thấy cho các đầu ra có thể.

Ý tưởng cơ bản của Kernel SVM và các phương pháp kernel nói chung là tìm một phép biến đổi sao cho dữ liệu ban đầu là không phân biệt tuyến tính được biến sang không gian mới. Ở không gian mới này, dữ liệu trở nên phân biệt tuyến tính.

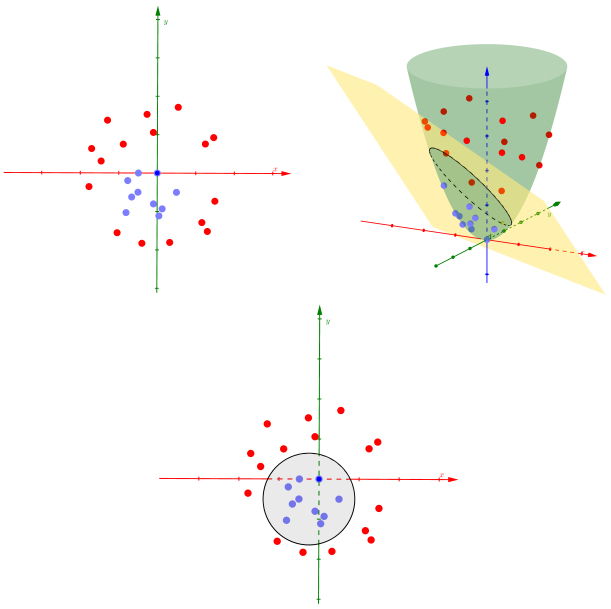

Figure 4: Ý tưởng kernel SVM

Trong hình đầu tiên trên cùng ta thấy được dữ liệu của hai classes là không phân biệt tuyến tính trong không gian hai chiều. Nếu coi thêm chiều thứ ba các điểm dữ liệu sẽ được phân bố trên 1 parabolic và đã trở nên phân biệt tuyến tính . Mặt phẳng màu vàng là mặt phân chia, có thể tìm được bởi Hard/Soft Margin SVM. Giao điểm của mặt phẳng tìm được và mặt

parabolic là một đường ellipse, khi chiếu toàn bộ dữ liệu cũng như đường ellipse này xuống không gian hai chiều ban đầu, ta đã tìm được đường phân chia hai classses.
Nói một cách ngắn gọn, Kernel SVM là việc đi tìm một hàm số biến đổi dữ liệu x từ không gian feature ban đầu thành dữ liệu trong một không gian mới bằng hàm số $\Phi(x)$.
Trong ví dụ này, hàm $\Phi(x)$ đơn giản là giới thiệu thêm một chiều dữ liệu mới (một feature mới) là một hàm số của các features đã biết. Hàm số này cần thỏa mãn mục đích của chúng ta: trong không gian mới, dữ liệu giữa hai classes là phân biệt tuyến tính hoặc gần như phần biệt tuyến tính. Khi đó, ta có thể dùng các bộ phân lớp tuyến tính thông thường như PLA, Logistic Regression, hay Hard/Soft Margin SVM.

1.5 Hàm số kernel

1.5.1 Tính chất của các hàm kernel

Không phải hàm k() bất kỳ nào cũng được sử dụng. Các hàm kerrnel cần có các tính chất:

- Đối xứng: $k(x, z) = k(z, x)$ điều này dễ dàng nhận ra vì tích vô hướng của hai vector có tính đối xứng.
- Về lý thuyết các hàm kernel thoã mãn điều kiện Mercer

1.5.2 Linear kernel

Đây là trường hợp đơn giản với kernel chính tích vô hướng của hai vector:

$$k(x, z) = x^T z$$

Khi sử dụng hàm trong sklearn, kernel này được chọn bằng cách là kernel = 'linear'.

1.5.3 Polynomial kernel

$$k(x, z) = (\Upsilon + \gamma x^T z)^d$$

Với d là một số dương để chỉ bậc của đa thức. d có thể không là số tự nhiên vì mục đích chính của ta không phải là bậc của đa thức mà là cách tính kernel. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.
Khi sử dụng hàm trong sklearn, kernel này được chọn bằng cách kernel = 'poly'.

1.5.4 Radial basic function

Radial Basic Function (RBF) kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế, và là lựa chọn mặc định trong sklearn. Nó được định nghĩa bởi:

$$k(x, z) = exp(-\gamma \|x - z\|_2^2), \gamma > 0$$

Trong sklearn, kernel được chọn bằng cách gọi linear = 'rbf'.

1.5.5 Sigmoid

Hàm sigmoid cũng được sử dụng làm một hàm kernel trong SVM:

$$k(x, z) = \tanh(\gamma x^T z + \Upsilon)$$

Trong sklearn, kernel được chọn bằng cách gọi linear = 'sigmoid'.

1.5.6 Bảng tóm tắt các kernel thông dụng

Tên	Công thức	Kernel	Thiết lập hệ số
linear	$k(x, z) = x^T z$	'linear'	không có hệ số
polinomial	$k(x, z) = (\Upsilon + \gamma x^T z)^d$	'poly'	d: degree, γ: gamma, Υ : coef0
rbf	$k(x, z) = exp(-\gamma \|x - z\|_2^2)$	'rbf'	$\gamma > 0$: gamma
sigmoid	$k(x, z) = \tanh(\gamma x^T z + \Upsilon)$	'sigmoid'	γ: gamma, Υ : coef0

Table 1: Bảng các kernel thông dụng

1.6 Các siêu tham số trong kernel SVM

1.6.1 C

C: float, default = 1.0

Tham số C cho biết mức độ tối ưu hoá SVM mà bạn muốn để tránh phân loại sai dữ liệu đào tạo. Đối với các C có giá trị lớn, việc tối ưu hoá chọn một siêu phẳng sẽ có lợi nhuận thấp nếu siêu phẳng đó phân loại chính xác các điểm dữ liệu. Ngược lại C có giá trị nhỏ sẽ tối ưu hoá việc tìm kiếm siêu phẳng phân tách các điểm dữ liệu sẽ có lề lớn hơn, ngay cả khi siêu phẳng đó phân loại sai nhiều điểm hơn, ngay cả khi dữ liệu có thể phân tách tuyến tính.

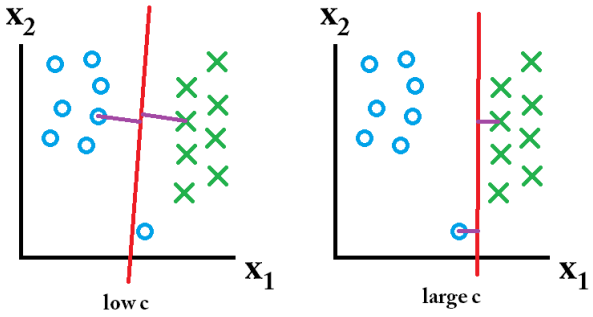

Figure 5: Cách chọn tham số C

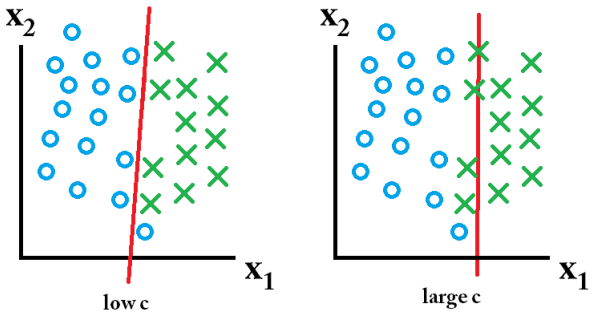

Figure 6: Cách chọn tham số C

Trong figure 5 bên trên. Ở bên trái tham số C có giá trị nhỏ hơn, thì ta có thể thấy đươc lề (đường màu tím) tạo ra giữa siêu phẳng (đường màu đỏ) sẽ có khoảng cách khá lớn và nó bỏ qua một điểm dữ liệu (điểm màu xanh lam) phân loại không chính xác. Ở bên phải tham số C có giá trị lớn hơn, bạn sẽ có được một siêu phẳng với lề nhỏ hơn và đạt độ chính xác cao hơn đối với việc phân loại.
Vậy nên chọn giá trị C thế nào cho phù hợp. Điều đó phụ thuộc vào dữ liệu mà bạn cần dự đoán nó phân bố thế nào. Trong figure 6 có thể thấy là việc chọn giá trị C thấp sẽ cho kết quả tốt hơn C có giá trị lớn.

1.6.2 Kernel

kernel: 'linear','poly','rbf','sigmoid', default = 'rbf'

Chức năng chính của kernel là lấy không gian đầu vào có chiều thấp và biến nó thành không gian có chiều cao hơn. Nó chủ yếu hữu ích trong vấn đề phân tách phi tuyến tính. Một số hạt nhân(kernel) được sử dụng trong mô hình là hàm cơ sở tuyến tính, đa thức, bán kính (RBF) (còn được gọi là hạt nhân Gaussian) và sigmoid. Trong khi điều ngược lại để tạo siêu phẳng phi tuyến tính là sử dụng RBF cũng như nhân đa thức. Các hạt nhân(kernel) nâng cao hơn được khuyến nghị để tách các lớp bản chất phi tuyến trong các ứng dụng phức tạp. Chúng tôi có thể xây dựng các bộ phân loại chính xác hơn với sự biến đổi này. Loại hàm nhân được sử dụng nhiều nhất là RBF vì phản hồi cục bộ và hữu hạn dọc theo toàn bộ trục x.

1.6.3 Degree

degree: int, default = 3

Là cấp độ của hàm hạt nhân đa thưc 'poly' bị bỏ qua bởi tất cả các hàm hạt nhân (kernel function) khác. Tham số degree thường có giá trị mặc định là 3.

1.6.4 Gamma

gamma'scale', 'auto' or float, default='scale'

Gamma là một siêu tham số được sử dụng với SVM phi tuyến tính.Tham số gamma xác định mức độ ảnh hưởng của một ví dụ đào tạo. Điều này có

nghĩa là gamma cao sẽ xem xét các điểm gần siêu phẳng và gamma thấp sẽ xem xét các điểm ở khoảng cách xa siêu phẳng hơn.
Trong figure 7 trên ta có thể thấy được là khi giảm tham số gamma thì việc tìm siêu phẳng sẽ xem xét các điểm dữ liệu có khoảng cách xa hơn vì vậy sẽ càng nhiều điểm dữ liệu sẽ được sử dụng.

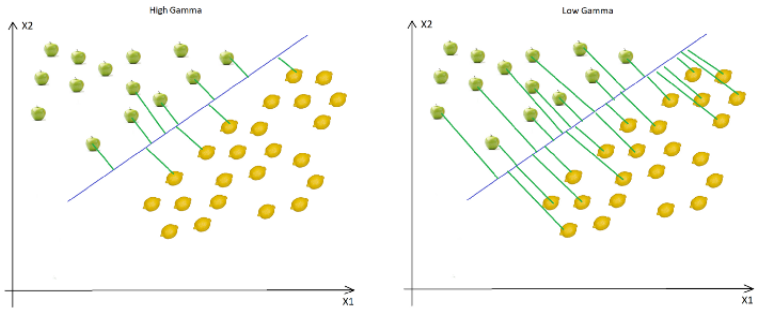

Figure 7: Tham số gamma

1.6.5 coef0

coef0:float, default=0.0

Là một tham số có tính độc lập trong hàm kernel, nó chỉ có ý nghĩa trong 'poly' và 'sigmoid'.

2 Bộ dữ liệu và phương pháp áp dụng

2.1 Bộ dữ liệu

Trong đồ án này thì nhóm chúng em sử dụng tập dữ liệu được gọi là " Cơ sở dữ liệu về bệnh tiểu đường của người da đỏ Pima" được cung cập bởi kaggle. Mục tiêu của bộ dữ liệu này là dự đoán xem một bệnh nhân có mắc bệnh tiểu đường hay là không, dựa trên các thông số và phép đo chuẩn đoán trong tập dữ liệu. Đặc biệt là tất cả các bệnh nhân trong tập dữ liệu này đều là nữ ở độ tuổi 21 trở lên và thuộc tộc người Pima Ấn Độ.
Các trường trong bộ dữ liệu

- Pregnancies: số lần mang thai.
- Glucose: nồng độ glucose trong máu trong 2 giờ thử nghiệm dung nạp glucose qua đường miệng.
- BloodPressure: huyết áp (mm Hg).
- SkinThickness: độ dày nếp gấp da cơ tam đầu (mm).
- Insulin: insulin trong máu (mu U/ml).
- BMI: body mass index (weigh in kg / (height in m)x2).
- DiabetesPedigreeFunction: di truyền bệnh tiểu đường.
- Age: tuổi (years).
- Outcome: đầu ra là biến có giá trị 0 hoặc 1 (268/768 giá trị 1, còn lại là 0).

Bộ dữ liệu là kết quả thu thập của 768 người, gồm 768 hàng và 9 cột. Trong đó 8 cột đầu là các trường dữ liệu để dự đoán bệnh tiểu đường, cột cuối cùng là kết quả người đó có mắc bệnh tiểu đường hay không (1 là có, 0 là không). Trong 768 người thu thập thì có 500 người được gán nhãn là 0 (tức không mắc bệnh tiểu đường) và 268 người được gán nhãn là 1 (mắc bệnh tiểu đường). Không tồn tại các cột có giá trị là null hoặc mất giá trị (missing value).

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Figure 8: 5 hàng đầu tiên của bộ dữ liệu

2.2 Xử lý các trường dữ liệu không mong muốn

Nhìn vào data và biểu đồ phân bố data ta có thể nhận thấy rằng một vài tiêu chí có giá trị không mong muốn.
Các giá trị như bloodpressure(huyết áp), BMI, insulin, glucose có tồn tại những giá trị bằng 0. Vấn đề này là không hợp lý đối với người bình thường.Điều này cho thấy bộ dữ liệu có vấn đề cần phải giải quyết.

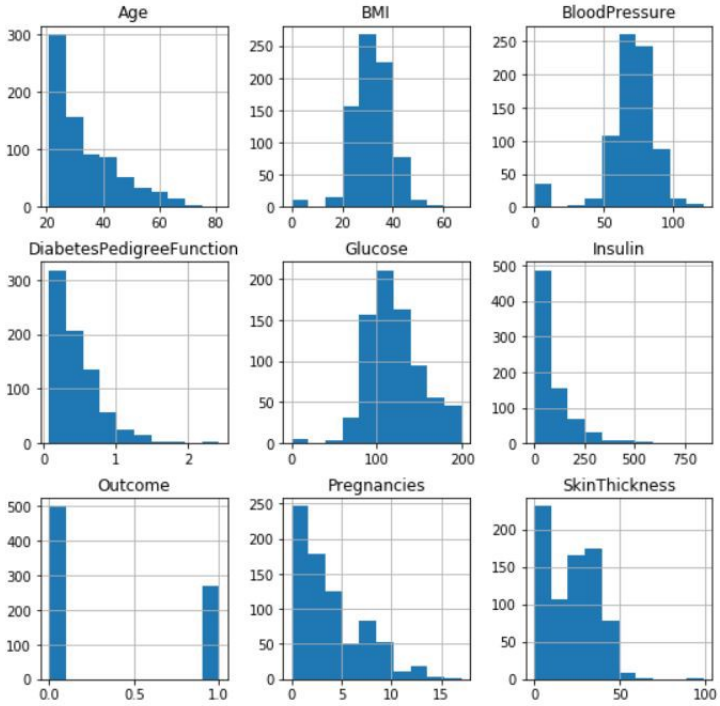

Figure 9: Các biểu đồ phân bố giá trị của bộ dữ liệu

Các đề xuất xử lý các dữ liệu trên:

1. Xoá/loại bỏ các trường hợp trên: tuy nhiên việc xoá hoặc loại bỏ các trường hợp mà có quá nhiều giá trị bằng 0, sẽ không khả thi và sẽ làm mất thông tin như các giá trị của các cột (SkinThickness hay Insulin có lần lượt là 227 và 334 giá trị bằng 0). Tuy nhiên, việc xoá loại bỏ này thích hợp với các cột dữ liệu như là BMI, glucose hay BloodPressure.

2. Đặt các giá trị trung bình: điều này sẽ có hiệu quả đối với các bộ dữ liệu khác. Tuy nhiên đối với bộ dữ liệu này việc đặt các giá trị trung bình đối với cột BloodPressure (huyết áp) sẽ không cho kết quả tốt đối với mô hình

3. Không sử dụng các cột này làm thông số đánh giá: việc này là không khả quan bởi vì các yếu tố như Insulin, glucose hay BloodPressure,... Là những yếu tố trực tiếp gây ra bệnh tiểu đường.

Trong quá trình áp dụng model SVM trên tập dữ liệu thì nhóm tiến hành phương pháp xử lý thứ nhất đối với bộ dữ liệu này. Là sẽ thực hiện thao tác xoá các hàng BloodPressure, BMI và glucose có giá trị bằng 0.

2.3 Cài đặt và tunning các siêu tham số

2.3.1 Cài đặt

Nhóm sẽ tiến hành cài đặt thuật toán SVM với bộ dữ liệu về bệnh tiểu đường bằng python với công cụ hỗ trợ là thư viện scikit-learn và googlecolab.

Nhóm sử dụng các hàm có sẵn trong thư viện sklearn, pandas như: SVC, Kfold, Gridsearch,...

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, \
    classification_report
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.svm import SVC
from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

Figure 10: Một số thư viện sử dụng

Nhóm thực hiện thống kê các hàng, cột dữ liệu có giá trị là 0 không phù hợp với thực tế.

Sau đó loại bỏ các hàng mà số lượng các giá trị bằng 0 ít như là BMI, glucose

hay bloodpressure. Tiến hành chia tập dữ liệu thành các tập train và test với kích thước là 8-2.

```
print("Total BloodPtrssure : ", df[df.BloodPressure == 0].shape[0])
print("Total Glucose : ", df[df.Glucose == 0].shape[0])
print("Total SkinThickness : ", df[df.SkinThickness == 0].shape[0])
print("Total BMI : ", df[df.BMI == 0].shape[0])
print("Total Insulin : ", df[df.Insulin == 0].shape[0])
```

```
Total BloodPtrssure :  35
Total Glucose :  5
Total SkinThickness :  227
Total BMI :  11
Total Insulin :  374
```

Figure 11: Kết quả thống kê các giá trị bằng 0

Áp dụng model SVM đối với tập dữ liệu và đánh giá độ chính xác của model bằng cách sử dụng thư viện sklearn.
Kết quả của mô hình SVM sử dụng trên tập dữ liệu bệnh tiểu đường cho kết quả đạt accuracy = 0.77.

```
# Creating the SVM model.
clf = SVC()
clf.fit(X_train_scaler,y_train)
y_pred = clf.predict(X_test_scaler)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.86      0.81      0.83       102
           1       0.60      0.67      0.64        43

    accuracy                           0.77       145
   macro avg       0.73      0.74      0.74       145
weighted avg       0.78      0.77      0.78       145
```

Figure 12: Accuracy sử dụng SVM

Kết quả mô hình đánh giá bằng Kfold với k = 5 cho kết quả thấp hơn SVM

với các thông số mặc định. Độ chính xác chỉ đạt accuracy trung bình là 0.7527

```
kfold = KFold(n_splits=5, shuffle=True, random_state=1)
accuracy_list = []
X_data = X
y_data = y
for train, test in kfold.split(X_data, y_data):
    clf = SVC()
    X_train, X_test = X_data.iloc[train], X_data.iloc[test];
    y_train, y_test = y_data.iloc[train], y_data.iloc[test]

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy_list.append(accuracy_score(y_test, y_pred))

print("Accuracy: ", np.mean(accuracy_list) * 100)
```

```
Accuracy:  75.27586206896551
```

Figure 13: Accuracy sử dụng kfold với k = 5

2.3.2 Tunning các siêu tham số

1. Grid Search
   Grid search là một phương pháp hiệu quả để điều chỉnh các tham số trong học có giám sát và cải thiện hiệu suất tổng quát hóa của một mô hình. Với Tìm kiếm theo phương pháp vét cạn, grid search thử tất cả các kết hợp có thể có của các tham số quan tâm và tìm những tham số tốt nhất.
   Sau khi có được các tham số tốt nhất, nó sẽ chạy khớp trở lại trên tất cả các dữ liệu được truyền cho vừa khớp để tạo ra một mô hình mới với các thông số tốt nhất.

2. RandomizedSearchCV
   RandomizedSearchCV rất hữu ích khi chúng ta có nhiều tham số để thử và thời gian đào tạo rất lâu.
   Thay vì sử dụng phương pháp vét cạn như grid search thì random search sẽ kết hợp các tham số ngẫu nhiên để chọn ra bộ tham số tốt nhất.

3. Tunning tham số
   Nhóm đã thực nghiệm tunning các tham số bằng phương pháp grid search và random search trên bộ dữ liệu ban đầu. Thu được kết quả của hai cái phương pháp như sau:

Phương pháp	**Kernel**	**C**	**gamma**	**Time(s)**	**Accuracy**
Grid search	Linear	100		9.2	0.800
Random search	Linear	100		7.6	0.800
Grid search	rbf	100	0.0001	1.2	0.7724
Random search	rbf	100	0.0001	0.0	0.7724
Grid search	poly	1000	0.01	4143.523	0.7517
Random search	poly	100	1	720	0.7103
Grid search	sigmoid	1	0.01	0.0	0.7655
Random search	sigmoid	1	0.01	0.0	0.7655

Table 2: Bảng kết quả của hai phương pháp tunning tham số

Nhìn vào bảng kết quả có thể thấy được cả hai phương pháp grid search và random search đều chọn được các tham số giống nhau. Tuy nhiên, có thể thấy rằng thời gian tìm các tham số của phương pháp random search nhanh hơn grid search.

3 Ưu điểm và nhược điểm của SVM

3.1 Ưu điểm

SVM xử lý hiệu quả các bộ dữ liệu nhiều chiều và không phân biệt tuyến tính.
SVM hoạt động tốt với cả ba loại dữ liệu là (có cấu trúc, bán cấu trúc và phi cấu trúc).
Nó sử dụng một tập hợp con các điểm huấn luyện trong hàm quyết định (được gọi là vectơ hỗ trợ), vì vậy nó cũng hiệu quả về bộ nhớ.
Có nhiều hàm hạt nhân (function kernel) cho nhiều quyết định khác nhau.

3.2 Nhược điểm

Với nhiều hàm hạt nhân việc chọn hàm hạt nhân phù hợp với từng loại dữ liệu là điều không dễ dàng.
SVM mất nhiều thời gian đào tạo hơn đối với các tập dữ liệu lớn.
Bộ dữ liệu có quá nhiều tính năng. SVM sẽ cho kết quả kém hơn bình thường.

4 Kết luận

Từ lý thuyết và thực nghiệm của nhóm. Nhóm đã rút ra các nhận định về Kernel SVM. Đối với bộ dữ liệu nhóm thực hiện (phân lớp nhị phân) thì SVM có thời gian thực hiện khá nhanh và cho độ chính xác tương đối (accuracy > 0.7).
Tuy nhiên nhóm chỉ mới đánh giá trên tập dữ liệu nhị phân nên chưa thể đánh giá được độ chính xác và tốc độ của SVM đối với bộ dữ liệu phân loại nhiều lớp.

5 Tài liệu tham khảo

1. Machine Learning Workflow on Diabetes Data, 12/12/2021, từ https://towardsdatascience.com/machine-learning-workflow-on-diabetesdata-part-01-5
2. SVM in Machine learning, 12/12/2021, từ https://techvidvan.com/tutorials/svm-in-machine-learning/?fbclid=IwAR3Xcni4xLHUwrQZVxYg4Buwszd1-pksC VaPJwQzqJy12FfdzOdZ0
3. Support Vector Machine, 12/12/2021, từ https://machinelearningcoban.com/2017/04/09/smv/
4. Diabetes Prediction Using Classification Models, 12/12/2021, từ https://www.kaggle.com/simgeerek/diabetes-prediction-using-classificationmodels/data?fbclid=IwAR17dRosVAcAlgPDVB1x_JDSONggRMn2rH4frepudwcB7GmBEwFUjV0tQg