**Đề bài**

Bài tập 3: Phân loại bệnh tiểu đường

Mô tả: Sử dụng tập dữ liệu tiểu đường để xây dựng mô hình phân loại nhằm dự đoán nguy cơ

mắc bệnh tiểu đường dựa trên các chỉ số sức khỏe.

Yêu cầu:

1. Tải tập dữ liệu và chia thành tập huấn luyện (80%) và tập kiểm tra (20%).

2. Tiền xử lý dữ liệu, bao gồm chuẩn hóa các đặc trưng.

3. Áp dụng thuật toán Logistic Regression để xây dựng mô hình phân loại.

4. Đánh giá mô hình bằng các chỉ số: độ chính xác, độ nhạy, độ chính xác, và F1-score.

5. Thử nghiệm với thuật toán SVM và so sánh kết quả.

6. Giải thích sự khác biệt giữa các thuật toán và đưa ra lựa chọn tối ưu.

**Bài làm**

Code :

1. **import** pandas as pd
2. **from** sklearn.model\_selection **import** train\_test\_split
3. **from** sklearn.preprocessing **import** MinMaxScaler
4. **from** sklearn.linear\_model **import** LogisticRegression
5. **from** sklearn.metrics **import** accuracy\_score, recall\_score, f1\_score, classification\_report
6. **from** sklearn.svm **import** SVC
7. **import** numpy as np

1. Tải tập dữ liệu

1. data = pd.read\_csv("diabetes\_data.csv", delimiter=';')
2. **print**(data)

2. Tiền xử lý dữ liệu, bao gồm chuẩn hóa các đặc trưng

1. # Chuyển đổi giới tính: Male thành 1, Female thành 0
2. data['gender'] = data['gender'].apply(**lambda** x: 1 **if** x == 'Male' **else** 0)
4. # Kiểm tra và xử lý dữ liệu bị thiếu nếu có
5. data = data.dropna()  # Loại bỏ các hàng có giá trị bị thiếu
7. # Kiểm tra và xử lý ngoại lai
8. **def** remove\_outliers\_iqr(df, column):
9. Q1 = df[column].quantile(0.25)
10. Q3 = df[column].quantile(0.75)
11. IQR = Q3 - Q1
12. lower\_bound = Q1 - 1.5 \* IQR
13. upper\_bound = Q3 + 1.5 \* IQR
14. **return** df[(df[column] >= lower\_bound) & (df[column] <= upper\_bound)]
16. # Xử lý ngoại lai cho các cột số
17. numerical\_columns = data.select\_dtypes(include=['float64', 'int64']).columns
19. **for** column **in** numerical\_columns:
20. data = remove\_outliers\_iqr(data, column)
22. **print**(data)
24. # Chia dữ liệu thành tập huấn luyện và tập test
25. X = data.drop("class", axis=1)  # Giả sử cột 'class' là nhãn
26. y = data["class"]
28. # Chia dữ liệu thành 80% train và 20% test
29. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
31. # Chuẩn hóa dữ liệu
32. scaler = MinMaxScaler()
33. X\_train = scaler.fit\_transform(X\_train)
34. X\_test = scaler.transform(X\_test)

3. Áp dụng thuật toán Logistic Regression để xây dựng mô hình phân loại

1. logistic\_model = LogisticRegression(max\_iter=1000)
2. logistic\_model.fit(X\_train, y\_train)
4. # Dự đoán trên tập train và test
5. y\_train\_pred\_lr = logistic\_model.predict(X\_train)
6. y\_test\_pred\_lr = logistic\_model.predict(X\_test)

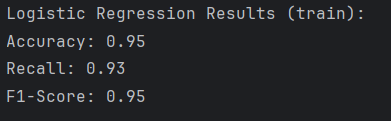
4. Đánh giá mô hình bằng các chỉ số: Độ chính xác, độ nhạy và F1-score

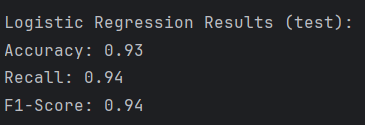
1. accuracy\_lr\_train = accuracy\_score(y\_train, y\_train\_pred\_lr)
2. recall\_lr\_train = recall\_score(y\_train, y\_train\_pred\_lr)
3. f1\_lr\_train = f1\_score(y\_train, y\_train\_pred\_lr)
5. accuracy\_lr\_test = accuracy\_score(y\_test, y\_test\_pred\_lr)
6. recall\_lr\_test = recall\_score(y\_test, y\_test\_pred\_lr)
7. f1\_lr\_test = f1\_score(y\_test, y\_test\_pred\_lr)
9. **print**("Logistic Regression Results (train):")
10. **print**(f"Accuracy: {accuracy\_lr\_train:.2f}")
11. **print**(f"Recall: {recall\_lr\_train:.2f}")
12. **print**(f"F1-Score: {f1\_lr\_train:.2f}")
13. **print**("\nClassification Report:")
14. **print**(classification\_report(y\_train, y\_train\_pred\_lr))
16. **print**("Logistic Regression Results (test):")
17. **print**(f"Accuracy: {accuracy\_lr\_test:.2f}")
18. **print**(f"Recall: {recall\_lr\_test:.2f}")
19. **print**(f"F1-Score: {f1\_lr\_test:.2f}")
20. **print**("\nClassification Report:")
21. **print**(classification\_report(y\_test, y\_test\_pred\_lr))

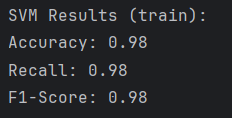
5. Thử nghiệm với thuật toán SVM

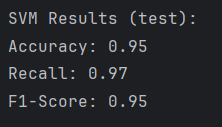
1. svm\_model = SVC()
2. svm\_model.fit(X\_train, y\_train)
4. # Dự đoán trên tập train và test
5. y\_train\_pred\_svm = svm\_model.predict(X\_train)
6. y\_test\_pred\_svm = svm\_model.predict(X\_test)
8. # Đánh giá mô hình SVM
9. accuracy\_svm\_train = accuracy\_score(y\_train, y\_train\_pred\_svm)
10. recall\_svm\_train = recall\_score(y\_train, y\_train\_pred\_svm)
11. f1\_svm\_train = f1\_score(y\_train, y\_train\_pred\_svm)
13. accuracy\_svm\_test = accuracy\_score(y\_test, y\_test\_pred\_svm)
14. recall\_svm\_test = recall\_score(y\_test, y\_test\_pred\_svm)
15. f1\_svm\_test = f1\_score(y\_test, y\_test\_pred\_svm)
17. **print**("SVM Results (train):")
18. **print**(f"Accuracy: {accuracy\_svm\_train:.2f}")
19. **print**(f"Recall: {recall\_svm\_train:.2f}")
20. **print**(f"F1-Score: {f1\_svm\_train:.2f}")
21. **print**("\nClassification Report:")
22. **print**(classification\_report(y\_train, y\_train\_pred\_svm))
24. **print**("SVM Results (test):")
25. **print**(f"Accuracy: {accuracy\_svm\_test:.2f}")
26. **print**(f"Recall: {recall\_svm\_test:.2f}")
27. **print**(f"F1-Score: {f1\_svm\_test:.2f}")
28. **print**("\nClassification Report:")
29. **print**(classification\_report(y\_test, y\_test\_pred\_svm))

6.Giải thích sự khác biệt giữa các thuật toán và đưa ra lựa chọn tối ưu









Từ kết quả cho thấy:

**Logistic Regression**: Là mô hình tuyến tính, có thể không hiệu quả trong các tình huống mà các mối quan hệ giữa đặc trưng và nhãn không phải là tuyến tính. Nếu dữ liệu có cấu trúc phức tạp hoặc không rõ ràng, Logistic Regression có thể không tìm ra quyết định tốt.

**SVM**: Có khả năng xử lý các mối quan hệ phi tuyến tính tốt hơn nhờ vào việc sử dụng kernel để biến đổi không gian đặc trưng. Điều này có thể là lý do khiến SVM hoạt động tốt hơn trong tập dữ liệu này.

SVM có độ chính xác cao hơn trên cả 2 tập train và tập test.

SVM có độ nhạy cao hơn trên cả 2 tập train và tập test.

SVM có F1-Score cao hơn trên cả 2 tập train và tập test.

Vậy lựa chọn tối ưu hơn ở đây là sử dụng thuật toán SVM.