Báo cáo đồ án: Nhóm 28

Huỳnh Lê Tấn Thành
Khoa Khoa học máy tính, Trường Đại học Công nghệ Thông tin
Đại học Quốc gia Thành phố Hồ Chí Minh
19522228@gm.uit.edu.vn

Abstract

Trong chương trình của môn học, em đã được học về các mô hình machine learning khác nhau cũng như các bước khác nhau để xây dựng và đào tạo một mô hình máy học. Đồ án cuối kỳ của môn học với hai tập dữ liệu Compas và Diabetes em có sử dụng mô hình Logistic Regression để xử lý bài toán phân loại theo yêu cầu của đề. Và một số kỹ thuật xử lý dữ liệu như Ordinal Encoding, Standard Scaler. Trong quá trình thực hiện gặp một số khó khăn, kết quả nhận được vẫn chưa tốt và tối ưu, mong nhận được sự góp ý và chấm điểm của thầy.

1 Exploratory Data Analysis và lựa chọn đặc trưng.

1.1 Exploratory Data Analysis (EDA)

- Đọc 2 tập dữ liệu và xem qua một vài giá trị.

```
df = pd.read_json('./compas_data/train.json', lines=True)
df.head()
```

	sex	age	race	juv_fel_count	juv_misd_count	juv_other_count	priors_count	c_charge_degree	two_year_recid
0	Male	38	African-American	1	1	0	9	F	1
1	Male	51	Caucasian	0	0	0	4	M	0
2	Female	38	Hispanic	0	0	0	0	F	0
3	Male	42	Caucasian	0	0	0	0	M	0
4	Male	33	African-American	0	0	0	1	F	1

Figure 1: Tập train của Compas

```
df = pd.read_json('./compas_data/dev.json', lines=True)
df.head()
```

	sex	age	race	juv_fel_count	juv_misd_count	juv_other_count	priors_count	c_charge_degree	two_year_recid
0	Female	35	African-American	0	0	0	0	M	0
1	Male	26	African-American	0	0	0	6	F	1
2	Female	19	Caucasian	0	0	0	2	F	1
3	Male	33	Caucasian	0	0	0	0	F	1
4	Male	32	Caucasian	0	0	0	5	F	1

Figure 2: Tập dev của Compas

```
df = pd.read_json('./diabetes_data/train.json', lines=True)
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	80	55	0	0	19.1	0.258	21	0
1	1	124	74	36	0	27.8	0.100	30	0
2	10	108	66	0	0	32.4	0.272	42	1
3	2	120	76	37	105	39.7	0.215	29	0
4	11	136	84	35	130	28.3	0.260	42	1

Figure 3: Tập train của Diabetes

```
df = pd.read_json('./diabetes_data/dev.json', lines=True)
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	144	82	40	0	41.3	0.607	28	0
1	0	179	50	36	159	37.8	0.455	22	1
2	9	184	85	15	0	30.0	1.213	49	1
3	0	134	58	20	291	26.4	0.352	21	0
4	4	142	86	0	0	44.0	0.645	22	1

Figure 4: Tập dev của Diabetes

Ta có thể quan sát thấy cả 2 tập dữ liệu đều có nhiều đặc trưng đầu vào dùng để dự đoán một cột dữ liệu kiểu nhị phân với 2 giá trị 0 và 1. Ở tập dữ liệu Compas có những giá trị chưa phải dạng số có thể tính toán cần thực hiện kỹ thuật Ordinal Encoding để chuyển về dạng số.

- Kiểm tra tồn tại giá trị null

```
df.isna().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Figure 5: Kiểm tra null trên tập train của Diabetes

Kiểm tra thấy không tồn tại giá trị *null* trên tập dữ liệu train của Diabetes. Thực hiện tương tự với các tập dữ liệu còn lại cũng không phát hiện giá trị *null*.

- Kiểm tra giá trị trùng

```
df.duplicated().sum()
```

```
1704
```

Figure 6: Kiểm tra trên tập train của Compas

```
df.duplicated().sum()
```

```
60
```

Figure 7: Kiểm tra trên tập dev của Compas

Ta thấy khi kiểm tra trên tập train và dev của dữ liệu Conpas đều xuất hiện giá trị trùng nhau cần loại bỏ những dòng giá trị này chỉ giữ lại một dòng. Thực hiện kiểm tra tương tự với dữ liệu Diabetes không phát hiện giá trị trùng nhau.

1.2 Lựa chọn đặc trưng

Ta sẽ dùng Heatmap được cung cấp sẵn trong thư viện của Seaborn để phân tích mối tương quan giữa các đặc trưng.

- Từ biểu đồ Heatmap (Figure 8) ta có thể thấy sự tương quan của cột *'two_year_recid'* cần dự đoán so với 2 cột *'race'* và *'c_charge_degree'* khá thấp nên có thể bỏ 2 cột này.

- Từ biểu đồ Heatmap (Figure 9) ta có thể thấy sự tương quan của cột *'Outcome'* cần dự đoán so với cột *'BloodPressure'* khá thấp nên có thể bỏ cột này.

2 Chuẩn bị dữ liệu

- Xây dựng hàm giúp chuẩn bị dữ liệu.

Compas

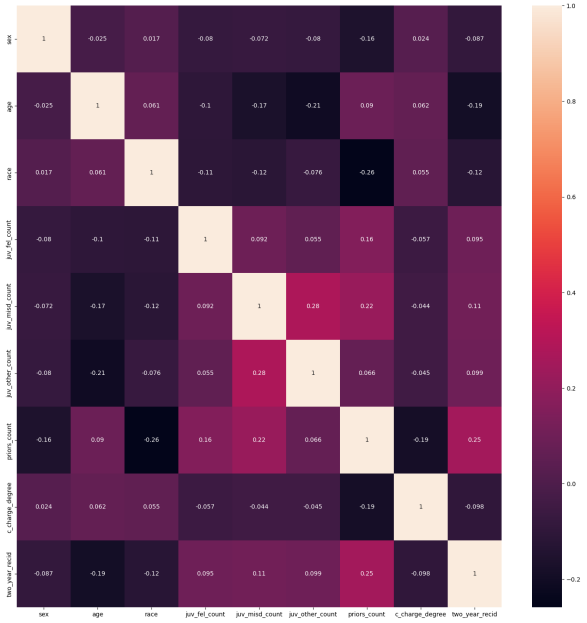

Figure 8: Heatmap tập train của compas

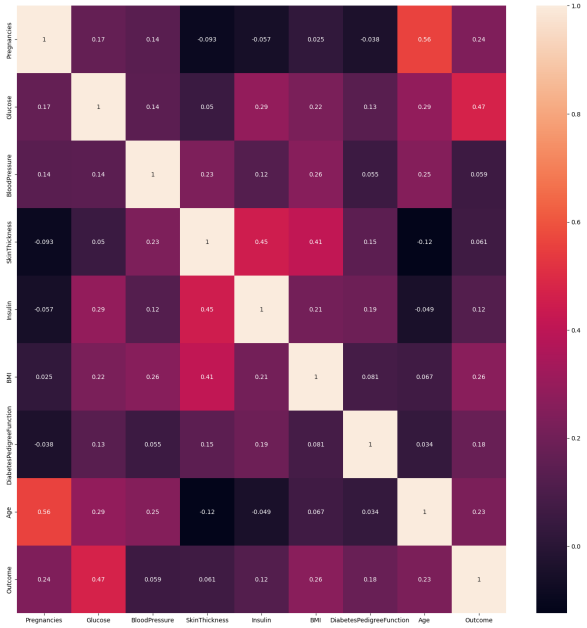

Figure 9: Heatmap tập train của diabetes

```
from sklearn.preprocessing import
    ↪ StandardScaler
def data_preprocessing(df):
    # Loại bỏ những dòng giá trị trùng
    df.drop_duplicates(inplace=True)
    # Đặt lại index cho các dòng dữ liệu
    df.reset_index(drop=True,
        ↪ inplace=True)

```

```
    # Ordinal Encoding cột 'sex'
    df['sex'] = df['sex'].apply(lambda s: (s=='Male' and 1) or (s=='Female' and 2))

    # Xoá cột 'race' và 'c_charge_degree'
    df.drop(['race', 'c_charge_degree'], axis=1, inplace=True)

    # Chia tập dữ liệu đọc vào ban đầu thành tập dữ liệu dùng để dự đoán (X) và tập mục tiêu dự đoán (y)
    X = df.drop('two_year_recid', axis=1)
    y = df['two_year_recid']

    # Standard Scaler các cột dữ liệu dùng dể dự đoán
    std = StandardScaler()
    for col in X.columns:
        X[col] = std.fit_transform(X[col].to_numpy().reshape(-1, 1))

    return df, X, y
```

Diabetes

```
from sklearn.preprocessing import StandardScaler
def data_preprocessing(df):
    # Xoá cột 'BloodPressure'
    df.drop(['BloodPressure'], axis=1, inplace=True)

    # Standard Scaler các cột dữ liệu dùng dể dự đoán
    cols = ['Pregnancies', 'Glucose',
            'SkinThickness', 'Insulin',
            'BMI', 'Age',
            'DiabetesPedigreeFunction']
    std = StandardScaler()
    for c in cols:
        df[c] = std.fit_transform(df[c].to_numpy().reshape(-1, 1))


    # Chia tập dữ liệu đọc vào ban đầu thành tập dữ liệu dùng để dự đoán (X) và tập mục tiêu dự đoán (y)
    X = df.drop('Outcome', axis=1)
    y = df['Outcome']

    return df, X, y
```

- Đọc dữ liệu huấn luyện và phát triển

```
import pandas as pd
train_data = pd.read_json(train_file, lines=True)
dev_data = pd.read_json(dev_file, lines=True)
```

- Chuẩn bị dữ liệu cho quá trình huấn luyện

```
train_data, X_train, y_train = data_preprocessing(train_data)
dev_data, X_dev, y_dev = data_preprocessing(dev_data)
```

3 Xác định bài toán, lựa chọn và xây dựng mô hình

3.1 Xác định bài toán và lựa chọn mô hình

- Qua quan sát cả 2 tập dữ liệu ta có thể thấy bài toán dự đoán thuộc loại bài toán phân loại (Classification). Sử dụng nhiều biến đầu vào x để dự đoán giá trị y theo kiểu nhị phân 0 hoặc 1.

- Dựa vào đó em đã lựa chọn thuật toán Logistic Regression. Vì thuật toán này có thể giải quyết tốt các bài toán phân loại. Logistic Regression có ưu điểm là dễ thực hiện, huấn luyện hiệu quả, độ chính xác cao trên một số tập dữ liệu. Mô hình có tốc độ huấn luyện nhanh. Có thể áp dụng các kỹ thuật chính quy hoá (L1 và L2) để tránh hiện tượng quá khớp.(AmiyaRanjanRout, 2023)

3.2 Xây dựng mô hình

3.2.1 Lý thuyết (Tiep, 2017)

Hàm Sigmoid .

$$f(s) = \frac{1}{1 + e^{-s}}$$

Lý do chọn hàm này vì:

- Là hàm số liên tục nhận giá trị thực, bị chặn trong khoảng $(0, 1)$
- Nếu coi điểm có tung độ là $1/2$ làm điểm phân chia thì các điểm càng xa điểm này về phía bên trái có giá trị càng gần 0. Ngược lại, các điểm càng xa điểm này về phía phải có giá trị càng gần 1.
- Mượt (smooth) nên có đạo hàm mọi nơi, có thể được lợi trong việc tối ưu.

Mỗi khi dự toán, kết quả sẽ được được tính theo công thức sau:

$$\widehat{y} = \begin{cases} 0 & f(s) < 0.5 \\ 1 & f(s) \geq 0.5 \end{cases}$$

với $\widehat{y}$ là kết quả dự đoán.

Hàm mất mát (Loss Function) .

Mục tiêu của hàm loss nhằm phản ánh độ tối ưu của bộ tham số. Bộ tham số càng tối ưu thì độ chính xác càng lớn và giá trị loss càng nhỏ.

Theo yêu cầu của bài toán, ta có thể giả sử xác suất của một điểm dữ liệu x rơi vào class 1 là $f(\mathbf{w}^T\mathbf{x})$ và rơi vào class 0 là $1 - f(\mathbf{w}^T\mathbf{x})$. Với y là các điểm dữ liệu kết quả được dự đoán. Ta có công thức:

$$P(y_i = 1|x_i; w) = f(w^T x_i) \quad (1)$$

$$P(y_i = 0|x_i; w) = 1 - f(w^T x_i) \quad (2)$$

trong đó $P(y_i = 1|x_i; w)$ là xác suất xảy ra sự kiện $y_i = 1$ khi biết tham số mô hình w và dữ liệu đầu vào x_i.

Ký hiệu $z_i = f(w^T x_i)$, ta có thể viết gộp biểu thức (1) và (2) thành:

$$P(y_i|x_i; w) = z_i^{y_i}(1 - z_i)^{1-y_i}$$

vì khi $y_i = 0$ thì $P(y_i|x_i; w) = 1 - z_i$ và khi $y_i = 1$ thì $P(y_i|x_i; w) = z_i$

Xét toàn bộ tập dữ liệu huấn luyện $X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{d \times N}$ và $y = [y_1, y_2, \ldots, y_N]$. Chúng ta cần tìm w để kết quả dự đoán càng gần 0 hoặc 1 càng tốt hay nói cách khác tìm w để $P(y|X; w)$ đạt giá trị lớn nhất.

$$w = \arg\max_w P(y|X; w)$$

$$= \arg\max_w \prod_{i=1}^{N} P(y_i|x_i; w)$$

Trực tiếp tối ưu hàm số $P(y_i|x_i; w)$ theo w nhìn qua không đơn giản! Hơn nữa, khi N lớn, tích của N số nhỏ hơn 1 có thể dẫn tới sai số trong tính toán (numerial error) vì tích là một số quá nhỏ. Một phương pháp thường được sử dụng đó là lấy logarit tự nhiên (cơ số e) của likelihood function biến phép nhân thành phép cộng và để tránh việc số quá nhỏ. Sau đó lấy ngược dấu để được một hàm và coi nó là hàm mất mát. Lúc này bài toán tìm giá trị lớn nhất (maximum likelihood) trở thành bài toán tìm giá trị nhỏ nhất của hàm mất mát (hàm này còn được gọi là negative log likelihood):

$$J(w) = -\log P(y|X; w)$$

$$= -\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i)\log(1 - z_i))$$

Gradient Decent .

Đây là một phương thức đơn giản có thể được sử dụng bởi nhiều thuật toán trong Machine Learning. Nó hoạt động bằng cách sử dụng mô hình để tính toán dự đoán cho mỗi trường hợp trong tập huấn luyện và tính toán sai số (Loss) cho mỗi dự đoán.

Ta có thể tìm ra bộ hệ số ứng với hàm Loss đạt giá trị nhỏ nhất cho bài toán tìm hệ số cho mô hình logistic Regression như sau:

- Dự đoán đầu ra với model hiện tại
- Tính giá trị mới cho bộ hệ số dựa vào hàm loss (chênh lệch giữa giá trị dự đoán và kết quả thực tế)
- Lặp lại 2 bước trên đến khi nào thấy hàm loss có giá trị là nhỏ nhất

Quá trình lặp lại kết thúc khi những vòng lặp sau giá trị của hàm Loss không còn giảm nữa (để làm như vậy tham khảo phương thức Early Stopping) hoặc đã chạy hết số vòng lặp đã chỉ định.(Ha, 2021)

Công thức cập nhật (theo thuật toán SGD) cho Logistic Regression là:

$$w = w + \eta(y_i - z_i)x_i$$

3.2.2 Xây dựng mô hình bằng Python

- Từ lý thuyết trên ta xây dựng class triển khai thuật toán Logistic Regression.(Dubba, 2018)

```python
import numpy as np

class LogisticRegression:
    # Khởi tạo các tham số cần thiết
    def __init__(self, learning_rate, max_iter, fit_intercept):
        self.learning_rate = learning_rate
        self.max_iter = max_iter
        self.fit_intercept = fit_intercept
        self.weights = []

    # Hàm sigmoid
    def sigmoid(self, s):
        return 1.0/(1 + np.exp(-s))

    # Hàm tính kết quả dự đoán khi biết X và w
    def compute_prediction(self, X, weights):
        s = np.dot(X, weights)
        return self.sigmoid(s)

    # Hàm tính kết quả dự đoán sau khi train xong mô hình
    def predict(self, X):
        predict = []

        if self.fit_intercept:
            X['intercept'] = np.ones((X.shape[0], 1))

        predict = np.round(
            self.compute_prediction(X, self.weights))
        return predict

    # Hàm mất mát
    def compute_cost(self, X, y, weights):
        predictions = self.compute_prediction(X, weights)
        return np.mean(-y * np.log(predictions) - (1 - y) * np.log(1 - predictions))

    # Hàm cập nhật tham số w của mô hình
    def update_weights(self, X_train, y_train, weights, learning_rate):
        for i in range (X_train.shape[0]):
            X = X_train.iloc[i]
            y = y_train.iloc[i]
            predictions = self.compute_prediction(X, weights)
            delta_weights = X.T * (y - predictions)
            weights += learning_rate * delta_weights
        return weights

    # Hàm huấn luyện mô hình
    def train_logistic_regression(self, X_train, y_train):
        if self.fit_intercept:
            intercept = np.ones((X_train.shape[0], 1))
            X_train['intercept'] = intercept

        self.weights = np.zeros(X_train.shape[1])

        before_cost = 1
        for _ in range(self.max_iter):
            self.weights = self.update_weights(
                X_train, y_train,
                self.weights, self.learning_rate)
            now_cost = self.compute_cost(X_train, y_train, self.weights)
            # print(now_cost)
            # Quá trình đào tạo dừng khi loss gần như không giảm nữa
            if (before_cost-now_cost) < 1e-8:
                break
            before_cost = now_cost
```

- Lựa chọn siêu tham số .

```
# Hàm giúp lựa chọn siêu tham số
def hyperparameter_tuning(X_train, y_train, X_dev, y_dev, param_grid):
    result = pd.DataFrame(columns=['param', 'f1_score', 'time'])
    for l in param_grid['learning_rate']:
        for m in param_grid['max_iter']:
            for f in param_grid['fit_intercept']:
                start = time.time()
                model = LogisticRegression(learning_rate=l, max_iter=m, fit_intercept=f)
                model.train_logistic_regression(X_train, y_train)
                end = time.time()
                y_pred = model.predict(X_dev)
                result.loc[len(result.index)] = [[l, m, f], f1_score(y_dev, y_pred), end-start]

    print(result)
    best_param = result.iloc[result['f1_score'].idxmax(), 0]
    return best_param
```

1. Từ việc kết hợp các giá trị của các phần tử trong tham số *param_grid* ta thu được các bộ siêu tham số khác nhau cho mô hình

2. Huấn luyện mô hình bằng lần lượt từng bộ tham số được tạo ra.

3. Dùng tập dev để đánh giá các mô hình qua quan sát $F1\_score$ và thời gian huấn luyện để lựa chọn bộ siêu tham số tốt nhất.

- Tạo và huấn luyện mô hình

```
# Tìm siêu tham số param tốt nhất
param_grid = {'learning_rate' : [0.001, 0.005, 0.01, 0.05, 0.1, 0.2],
    'max_iter' : [200],
    'fit_intercept' : [False]}
param = hyperparameter_tuning(X_train, y_train, X_dev, y_dev, param_grid)

# Huấn luyện mô hình với siêu tham số đã tìm được
model = LogisticRegression(param[0], param[1], param[2])
model.train_logistic_regression(X_train, y_train)
```

- Lưu mô hình đã huấn luyện

Sử dụng thư viện joblib để lưu

```
model_path = os.path.join(model_dir, 'trained_model.joblib')
dump(model, model_path)
```

4 Kết quả và nhận xét

Thông thường mô hình sẽ được đánh giá qua tập test nhưng vì không có nên em đã quyết định đánh giá qua tập dev.

4.1 Diabetes

Thực hiện tính Accuracy Score, Precision Score, Recall Score, F1 Score trên tập dev để đánh giá mô hình.

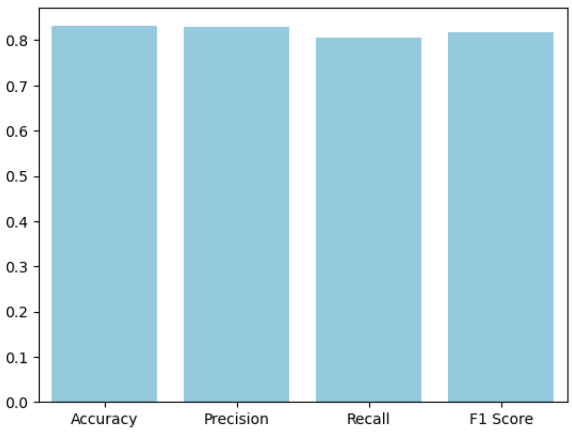

Figure 10: Biểu đồ điểm của mô hình đánh giá trên tập dev_diabetes

Nhìn chung các điểm ở mức khá tốt, dao động quanh 0.82, chỉ có Recall Score (phản ánh tỉ lệ dự đoán bị tiểu đường chính xác trong tất cả các trường hợp bị tiểu đường) là thấp hơn những điểm còn lại nhưng không đáng kể.

4.2 Compas

Thực hiện tính Accuracy Score, Precision Score, Recall Score, F1 Score trên tập dev để đánh gia mô hình.

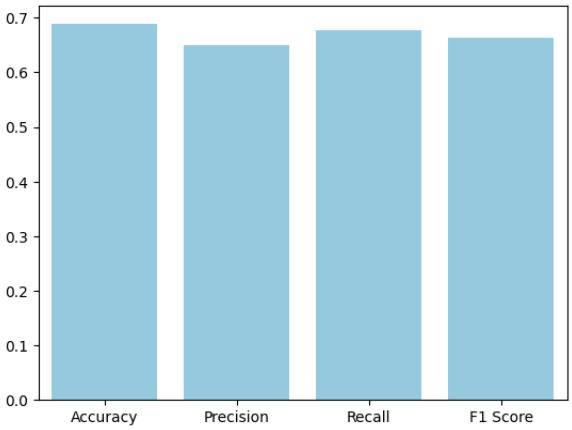

Figure 11: Biểu đồ điểm của mô hình đánh giá trên tập dev_compas

Nhìn chung các điểm đều ở mức trung bình, dao động quanh 0.66, chỉ có Precision Score (phản ánh tỉ lệ dự doán chính xác có phạm tội trong tất cả trường hợp có phạm tội được đưa ra dự đoán) là thấp hơn những điểm còn lại nhưng không đáng kể.

4.3 Confusion Matrix

Confusion Matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class.

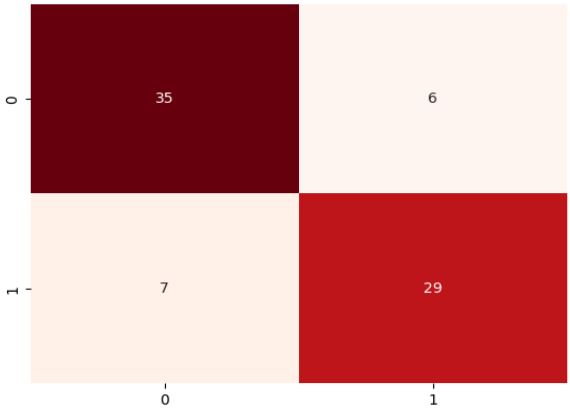

Figure 12: Confusion Matrix đánh giá trên tập dev_diabetes

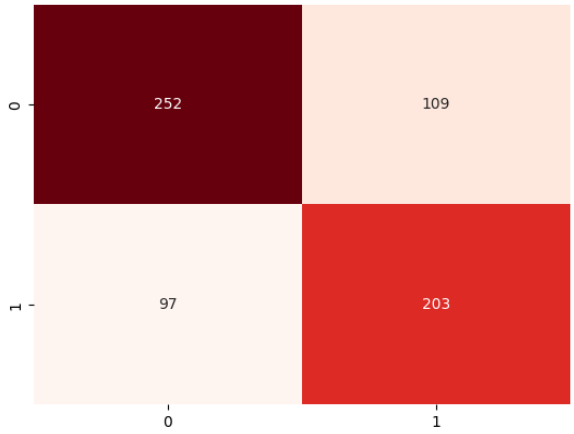

Figure 13: Confusion Matrix đánh giá trên tập dev_compas

Xem qua biểu đồ 2 Confusion Matrix khi đánh giá 2 mô hình trên 2 tập dev Diabets và Compas. Ta thấy các ô thuộc đường chéo chính (True Positive và True Negative) có màu đậm hơn, điểm cao hơn những ô còn lại. Điều này có nghĩa là mô hình có hiệu suất tốt.

5 Kết luận

Qua đánh giá ở trên ta thấy kết quả nhận được của mô hình dự đoán trên tập Diabetes là khá tốt có thể ứng dụng nhưng trên tập Compas chỉ ở mức trung bình cần cải tiến, chỉnh sửa.

Nguyên nhân hiệu suất của mô hình trên tập Compas chưa tốt có thể do còn thiếu bước xử lý giá trị gây nhiễu ở bước EDA hoặc thuật toán chưa phù hợp, không thể khái quát hết các đặc trưng của dữ liệu.

Trong tương lai có thể áp dụng thêm các kỹ thuật Bagging và Boosting như Gradient Boosting để tăng hiệu suất của mô hình, thay đổi thuật toán dự đoán trên tập dữ liệu Compas để có độ chính xác tốt hơn.

References

AmiyaRanjanRout. 2023. Advantages and disadvantages of logistic regression. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/.

Rakend Dubba. 2018. Multivariate logistic regression from scratch. https://www.kaggle.com/code/rakend/multivariate-logistic-regression-from-scratch.

Quan Ha. 2021. [machine learning] logistic regression và bài toán phân loại cảm xúc âm nhạc. http://tutorials.aiclub.cs.uit.edu.vn/index.php/2021/05/12/logistic-regression/.

Vu Huu Tiep. 2017. Logistic regression. https://machinelearningcoban.com/2017/01/27/logisticregression/.