

Ex 2: Diabetes

Cho dữ liệu diabetes.csv

Yêu cầu: đọc dữ liệu về, chuẩn hóa dữ liệu (nếu cần) và áp dụng thuật toán ADABoosting & LogisticRegression/ XGBoost & để thực hiện việc dự đoán khả năng dương tính với bệnh tiểu đường (positive diabete - outputs) dựa trên các biến lâm sàng khác (clinical variables - inputs)

1. Đọc dữ liệu, 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 0.33
3. Áp dụng thuật toán ADABoosting & LogisticRegression/ XGBoost
4. Tìm kết quả
5. Hãy cho biết với những người có pregnant, glucose, pressure, triceps, insulin, mass, pedigree, age lần lượt như sau thì ai có khả năng dương tính với bệnh tiểu đường, ai không?

8, 176, 90, 34, 300, 33.7, 0.467, 58

1, 100, 66, 15, 56, 23.6, 0.666, 26

12, 88, 74, 40, 54, 35.3, 0.378, 48

Diabetes

Thông tin các cột dữ liệu

1. Pregnancies: số lần mang thai
2. Glucose: Nồng độ glucose huyết tương 2 giờ trong thử nghiệm dung nạp glucose đường uống
3. BloodPressure: Huyết áp tâm trương (mm Hg)
4. SkinThickness: độ dày da gấp Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml). insulin huyết thanh 2-giờ
6. BMI: (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Age (years)
9. Outcome: Class variable (0 or 1)

Chú ý: Tất cả các biến trên liên tục, mục đích là dự đoán ai đó có bị tiểu đường hay không (Outcome=1) dựa trên các biến khác. Các mẫu lấy từ phụ nữ trên 21 years old.

```
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/Chapter8_Boosting/'
```

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

```
In [4]: from sklearn.linear_model import LogisticRegression
```

```
In [5]: Diabetes = pd.read_csv("diabetes.csv")
```

```
In [6]: Diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies      768 non-null int64
Glucose          768 non-null int64
BloodPressure    768 non-null int64
SkinThickness    768 non-null int64
Insulin          768 non-null int64
BMI              768 non-null float64
DiabetesPedigreeFunction  768 non-null float64
Age              768 non-null int64
Outcome          768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
In [7]: #Diabetes.head()
```

```
In [8]: #Diabetes.describe()
```



```
In [9]: # import seaborn as sns
# sns.pairplot(Diabetes)
# plt.show()
```

```
In [10]: inputData=Diabetes.iloc[:,8]
outputData=Diabetes.iloc[:,8]
```

```
In [11]: inputData.head()
```

```
Out[11]:
```

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

```
In [12]: outputData.head()
```

```
Out[12]: 0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

```
In [13]: pos = np.where(outputData == 1)
len(pos[0])
```

```
Out[13]: 268
```

```
In [14]: neg = np.where(outputData == 0)
len(neg[0])
```

```
Out[14]: 500
```

```
In [15]: # creating testing and training set
X_train,X_test,Y_train,Y_test = train_test_split(inputData,
                                                    outputData,
                                                    test_size=0.33)
```

AdaBoost

```
In [16]: from sklearn.ensemble import AdaBoostClassifier
ml = LogisticRegression(solver='liblinear')
boosting = AdaBoostClassifier(n_estimators=100,
                               base_estimator=ml,
                               learning_rate=1)
```

```
In [17]: # Train model
boosting.fit(X_train, Y_train)
```

```
Out[17]: AdaBoostClassifier(algorithm='SAMME.R',
                             base_estimator=LogisticRegression(C=1.0, class_weight=None,
                                                                    dual=False,
                                                                    fit_intercept=True,
                                                                    intercept_scaling=1,
                                                                    l1_ratio=None,
                                                                    max_iter=100,
                                                                    multi_class='warn',
                                                                    n_jobs=None, penalty='l2',
                                                                    random_state=None,
                                                                    solver='liblinear',
                                                                    tol=0.0001, verbose=0,
                                                                    warm_start=False),
                             learning_rate=1, n_estimators=100, random_state=None)
```

```
In [18]: boosting.score(X_train, Y_train)
```

```
Out[18]: 0.7568093385214008
```

```
In [19]: boosting.score(X_test, Y_test)
```

```
Out[19]: 0.7283464566929134
```

```
In [20]: from sklearn.model_selection import cross_val_score
scores = cross_val_score(boosting, inputData, outputData, cv=20)
scores
```

```
Out[20]: array([0.69230769, 0.76923077, 0.79487179, 0.82051282, 0.69230769,
0.66666667, 0.71794872, 0.69230769, 0.68421053, 0.84210526,
0.76315789, 0.73684211, 0.71052632, 0.81578947, 0.76315789,
0.84210526, 0.71052632, 0.68421053, 0.76315789, 0.78947368])
```



```
In [21]: display(np.mean(scores),np.std(scores))
```

```
0.7475708502024291
```

```
0.05500583547455201
```

```
In [24]: y_new = boosting.predict([[8, 176, 90, 34, 300, 33.7, 0.467, 58],  
[1, 100, 66, 15, 56, 23.6, 0.666, 26],  
[12, 88, 74, 40, 54, 35.3, 0.378, 48]])
```

```
In [25]: y_new
```

```
Out[25]: array([1, 0, 0], dtype=int64)
```

```
In [26]: # Nhận xét  
# Thuật toán phối hợp này không tốt hơn thuật toán ban đầu => ...  
# Thử áp dụng với XGBoost  
# Suy nghĩ đến việc chọn một thuật toán khác???
```

XGBoost

```
In [27]: import xgboost as xgb
```

```
In [28]: xgb_model = xgb.XGBClassifier(random_state=42)  
xgb_model.fit(X_train, Y_train)
```

```
Out[28]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,  
colsample_bynode=1, colsample_bytree=1, gamma=0,  
learning_rate=0.1, max_delta_step=0, max_depth=3,  
min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,  
nthread=None, objective='binary:logistic', random_state=42,  
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,  
silent=None, subsample=1, verbosity=1)
```

```
In [29]: xgb_model.score(X_train, Y_train)
```

```
Out[29]: 0.9066147859922179
```

```
In [30]: xgb_model.score(X_test, Y_test)
```

```
Out[30]: 0.7677165354330708
```

```
In [1]: # Model: Overfitting  
# không phù hợp
```

```
In [31]: from sklearn.model_selection import cross_val_score  
scores2 = cross_val_score(xgb_model, inputData, outputData, cv=20)  
scores2
```

```
Out[31]: array([0.64102564, 0.84615385, 0.87179487, 0.79487179, 0.82051282,  
0.64102564, 0.69230769, 0.66666667, 0.71052632, 0.84210526,  
0.81578947, 0.84210526, 0.73684211, 0.84210526, 0.78947368,  
0.92105263, 0.71052632, 0.68421053, 0.73684211, 0.84210526])
```

```
In [32]: display(np.mean(scores2),np.std(scores2))
```

```
0.7724021592442645
```

```
0.08121620180558949
```

```
In [33]: data_new = pd.DataFrame([[8, 176, 90, 34, 300, 33.7, 0.467, 58],  
[1, 100, 66, 15, 56, 23.6, 0.666, 26],  
[12, 88, 74, 40, 54, 35.3, 0.378, 48]],  
columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeF
```

```
In [34]: y_new = xgb_model.predict(data_new)  
y_new
```

```
Out[34]: array([1, 0, 0], dtype=int64)
```