

## Chapter 3 - Ex3: Diabetes

Cho dữ liệu diabetes.csv

**Yêu cầu: Áp dụng thuật toán LogisticRegression 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. Chuẩn hóa dữ liệu (nếu cần)
2. Tạo X\_train, X\_test, y\_train, y\_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
3. Áp dụng thuật toán LogisticRegression
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)
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter3_Logistic_I
```

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
```

```
In [3]: from sklearn.linear_model import LogisticRegression
```

```
In [4]: Diabetes = pd.read_csv("diabetes.csv")
```

```
In [5]: 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 [6]: Diabetes.head()
```

```
Out[6]:
```

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

In [7]: `Diabetes.describe()`

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [8]: `# import seaborn as sns`  
`# sns.pairplot(Diabetes)`  
`# plt.show()`

In [9]: `inputData=Diabetes.iloc[:,8]`  
`outputData=Diabetes.iloc[:,8]`

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

Out[10]:

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

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

Out[11]:

```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [12]: `pos = np.where(outputData == 1) # trong 768 nguoi thi co 268 nguoi duong tinh`  
`len(pos[0])`

Out[12]: 268

```
In [13]: X_train,X_test,Y_train,Y_test = train_test_split(inputData,outputData,test_size=0.2,random_state=0)

In [14]: clf = LogisticRegression(solver='liblinear')

In [15]: clf.fit(X_train,Y_train)

Out[15]: 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)

In [16]: print('Training data/ Score: ', clf.score(X_train,Y_train))

Training data/ Score:  0.7783985102420856

In [17]: print('Testing data/ Score learn: ', clf.score(X_test,Y_test))

Testing data/ Score learn:  0.7619047619047619

In [18]: Y_pred = clf.predict(X_test)

In [19]: y_new = clf.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 [20]: y_new

Out[20]: array([1, 0, 0], dtype=int64)

In [21]: # Nhận xét: R^2 của Training và Testing không chênh lệch nhiều,
# model không bị overfitting.
# Tuy nhiên R^2 không cao

In [22]: # Có giải pháp nào khác không???
# In confusion matrix
```

## Select important features

```
In [23]: # Univariate Selection

In [24]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2
```

```
In [25]: #apply SelectKBest class to extract top all best features
bestfeatures = SelectKBest(score_func=chi2, k='all')
fit = bestfeatures.fit(inputData,outputData)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(inputData.columns)
```

```
In [26]: #concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score'] #naming the dataframe columns
print(featureScores.nlargest(8,'Score')) #print 8 best features
```

	Specs	Score
4	Insulin	2175.565273
1	Glucose	1411.887041
7	Age	181.303689
5	BMI	127.669343
0	Pregnancies	111.519691
3	SkinThickness	53.108040
2	BloodPressure	17.605373
6	DiabetesPedigreeFunction	5.392682

```
In [27]: # Feature Selection
```

```
In [28]: from sklearn.ensemble import ExtraTreesClassifier
```

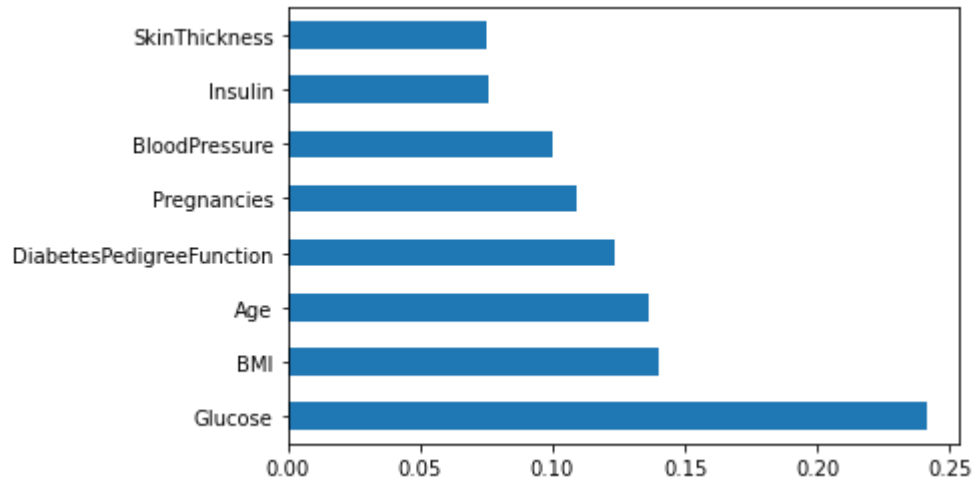
```
In [29]: model = ExtraTreesClassifier()
model.fit(inputData,outputData)
```

c:\program files\python36\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

```
Out[29]: ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
e,
oob_score=False, random_state=None, verbose=0,
warm_start=False)
```

```
In [30]: print(model.feature_importances_)  
# use inbuilt class feature_importances of tree based regressor  
# plot graph of feature importances for better visualization  
feat_importances = pd.Series(model.feature_importances_,  
                             index=inputData.columns)  
feat_importances.nlargest(8).plot(kind='barh')  
plt.show()  
  
[0.10909818 0.24099454 0.10000002 0.07470672 0.07594424 0.13993194  
 0.1232894  0.13603495]
```



```
In [31]: # Giai phap nen lam: Scale du lieu (nho kiem tra phan phoi cua cac cot)  
# Co the dung Log khong???
```