

▼ Data Mining - Lab 03


Ho va ten: Nguyen Duc Vu Duc

MSSV: 18110004

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns

1 #Import dataset
2 path='/content/diabetes_null.csv'
3 df=pd.read_csv(path)
```

```
1 df.head()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	5	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	4.0	35.0	168.0	43.1	2.288	33	1

```
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Pregnancies           768 non-null   int64
1   Glucose               763 non-null   float64
2   BloodPressure         733 non-null   float64
3   SkinThickness         541 non-null   float64
4   Insulin               394 non-null   float64
5   BMI                   757 non-null   float64
6   DiabetesPedigreeFunction 768 non-null   float64
7   Age                   768 non-null   int64
8   Outcome               768 non-null   int64
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

▼ Xuất các đặc trưng và kèm theo số lượng dòng NULL/NAN của từng cột đó

```
1 for column in df.columns:
2     print('The number of NaN values in {} columns: {}'.format(column,df[column].isna().sum()))

The number of NaN values in Pregnancies columns: 0
The number of NaN values in Glucose columns: 5
The number of NaN values in BloodPressure columns: 35
The number of NaN values in SkinThickness columns: 227
The number of NaN values in Insulin columns: 374
The number of NaN values in BMI columns: 11
The number of NaN values in DiabetesPedigreeFunction columns: 0
The number of NaN values in Age columns: 0
The number of NaN values in Outcome columns: 0
```

Đặc trưng Age là đặc trưng không có giá trị NULL/NAN nên ta chọn đặc trưng Age là đặc trưng chính. Thực hiện tạo ra 2 bộ data :

▼ Data 1 : gom nhóm Age theo hình thành các nhóm sau đó nếu giá trị NULL/NAN của các đặc trưng khác mà nằm trong nhóm nào thì thay bằng mean của nhóm đó

```
1 df_1=df.copy()
2 df_1['Age group']=np.floor(df_1.loc[:,'Age']/10)
3 df_1=df_1.fillna(df_1.mean())
4 df_1=df_1.drop(['Age'],axis=1)
5 df_1.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	Age group
0	6	148.0	72.0	35.000000	105.659898	33.6	0.627	1	0.0
1	1	85.0	66.0	29.000000	105.659898	26.6	0.351	0	3.0
2	8	183.0	64.0	25.876155	105.659898	23.3	0.672	1	3.0
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	0	2.0
4	0	137.0	4.0	35.000000	168.000000	43.1	2.288	1	3.0

▼ Data 2 : góm nhóm thành 5 nhóm đều nhau sau đó nếu giá trị NULL/NAN của các đặc trưng khác mà nằm trong nhóm nào thì thay bằng mean của nhóm đó

```
1 from sklearn import preprocessing
2 df_2=df.copy()
3 df_2['Age group'] = pd.cut(df_2['Age'], bins=5)
4 df_2=df_2.fillna(df_2.mean())
5 df_2=df_2.drop(['Age'],axis=1)
6 le = preprocessing.LabelEncoder()
7 le.fit(df_2['Age group'])
8 df_2['Age group']=le.transform(df_2['Age group'])
9 df_2=pd.DataFrame(df_2)
10 df_2.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	Age group
0	6	148.0	72.0	35.000000	105.659898	33.6	0.627	1	0
1	1	85.0	66.0	29.000000	105.659898	26.6	0.351	0	1
2	8	183.0	64.0	25.876155	105.659898	23.3	0.672	1	1
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	0	1
4	0	137.0	4.0	35.000000	168.000000	43.1	2.288	1	1

▼ Chạy các thuật toán Classification KNN, Decision Tree, Random Forest, Naïve Bayes

```
1 #Import libraries
2 from sklearn.neighbors import KNeighborsClassifier
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.naive_bayes import GaussianNB
6 from sklearn.model_selection import train_test_split
```

```
1 #Preprocessing dataset
2 data_1=pd.concat([df_1.iloc[:, :7],df_1.iloc[:,8]],axis=1)
3 data_1_label=df_1.iloc[:,7]
4
5 data_2=pd.concat([df_2.iloc[:, :7],df_2.iloc[:,8]],axis=1)
6 data_2_label=df_2.iloc[:,7]
```

```
1 #Train test split 2 dataset
2 X_train_1,X_test_1, y_train_1,y_test_1=train_test_split(data_1,data_1_label,test_size=0.2,random_state=42)
3 X_train_2,X_test_2, y_train_2,y_test_2=train_test_split(data_2,data_2_label,test_size=0.2,random_state=42)
```

```
1 #K Nearest Neighbors Classifier
2 neigh_1 = KNeighborsClassifier(n_neighbors=10)
3 neigh_1.fit(X_train_1, y_train_1)
4 y_1_knn_pred= neigh_1.predict(X_test_1)
5 print('Accuracy of KNN on dataset 1: ',np.sum(y_1_knn_pred == y_test_1)/len(y_test_1))
```

```
6
7 neigh_2 = KNeighborsClassifier(n_neighbors=5)
8 neigh_2.fit(X_train_2, y_train_2)
9 y_2_knn_pred= neigh_2.predict(X_test_2)
10 print('Accuracy of KNN on dataset 2: ',np.sum(y_2_knn_pred == y_test_2)/len(y_test_2))
11
12 #Decision Tree
13 clf_1 = DecisionTreeClassifier()
14 clf_1.fit(X_train_1, y_train_1)
15 y_1_dt_pred=clf_1.predict(X_test_1)
16 print('Accuracy of Decision Tree on dataset 1: ',np.sum(y_1_dt_pred == y_test_1)/len(y_test_1))
17
18 clf_2 = DecisionTreeClassifier()
19 clf_2.fit(X_train_2, y_train_2)
20 y_2_dt_pred=clf_2.predict(X_test_2)
21 print('Accuracy of Decision Tree on dataset 2: ',np.sum(y_2_dt_pred == y_test_2)/len(y_test_2))
22
23 #Random Forest
24 rf_1 = RandomForestClassifier(max_depth=5, random_state=0)
25 rf_1.fit(X_train_1, y_train_1)
26 y_1_rf_pred=rf_1.predict(X_test_1)
27 print('Accuracy of Random Forest on dataset 1: ',np.sum(y_1_rf_pred == y_test_1)/len(y_test_1))
28
29 rf_2 = RandomForestClassifier(max_depth=5, random_state=0)
30 rf_2.fit(X_train_2, y_train_2)
31 y_2_rf_pred=rf_2.predict(X_test_2)
32 print('Accuracy of Random Forest on dataset 2: ',np.sum(y_2_rf_pred == y_test_2)/len(y_test_2))
33
34 #Naive Bayes
35 nb_1=GaussianNB()
36 nb_1.fit(X_train_1,y_train_1)
37 y_1_nb_pred=nb_1.predict(X_test_1)
38 print('Accuracy of Naive Bayes on dataset 1: ',np.sum(y_1_nb_pred == y_test_1)/len(y_test_1))
39
40 nb_2=GaussianNB()
41 nb_2.fit(X_train_2,y_train_2)
42 y_2_nb_pred=nb_2.predict(X_test_2)
43 print('Accuracy of Naive Bayes on dataset 2: ',np.sum(y_2_nb_pred == y_test_2)/len(y_test_2))
```

```
Accuracy of KNN on dataset 1:  0.7142857142857143
Accuracy of KNN on dataset 2:  0.6818181818181818
Accuracy of Decision Tree on dataset 1:  0.6883116883116883
Accuracy of Decision Tree on dataset 2:  0.6493506493506493
Accuracy of Random Forest on dataset 1:  0.7467532467532467
Accuracy of Random Forest on dataset 2:  0.7532467532467533
Accuracy of Naive Bayes on dataset 1:  0.7402597402597403
Accuracy of Naive Bayes on dataset 2:  0.7532467532467533
```

## Hãy tự phân tích rồi nghĩ ra cách phân nhóm hay thay thế sao cho hiệu quả để có độ

▼

```
1 df_3=df.copy()
2 df_3['Age group'] = pd.qcut(df_3['Age'], q=5)
3 df_3=df_3.fillna(df_3.median())
4 df_3=df_3.drop(['Age'],axis=1)
5 le = preprocessing.LabelEncoder()
6 le.fit(df_3['Age group'])
7 df_3['Age group']=le.transform(df_3['Age group'])
8 df_3=pd.DataFrame(df_3)
9 df_3.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	Age group
0	6	148.0	72.0	35.0	71.0	33.6	0.627	1	0
1	1	85.0	66.0	29.0	71.0	26.6	0.351	0	2
2	8	183.0	64.0	27.0	71.0	23.3	0.672	1	3
3	1	89.0	66.0	23.0	94.0	28.1	0.167	0	0
4	0	137.0	4.0	35.0	168.0	43.1	2.288	1	3

```
1 data_3=pd.concat([df_3.iloc[:, :7],df_3.iloc[:,8]],axis=1)
2 data_3_label=df_3.iloc[:,7]
3 X_train_3,X_test_3, y_train_3,y_test_3=train_test_split(data_3,data_3_label,test_size=0.2,random_state=42)
```

```
1 neigh_3 = KNeighborsClassifier(n_neighbors=5)
2 neigh_3.fit(X_train_3, y_train_3)
3 y_3_knn_pred= neigh_3.predict(X_test_3)
4 print('Accuracy of KNN on dataset 3: ',np.sum(y_3_knn_pred == y_test_3)/len(y_test_3))
5
6 clf_3 = DecisionTreeClassifier(max_depth=3,random_state=0)
7 clf_3.fit(X_train_3, y_train_3)
8 y_3_dt_pred=clf_3.predict(X_test_3)
9 print('Accuracy of Decision Tree on dataset 3: ',np.sum(y_3_dt_pred == y_test_3)/len(y_test_3))
10
11 rf_3 = RandomForestClassifier(max_depth=4, random_state=0)
12 rf_3.fit(X_train_1, y_train_1)
13 y_3_rf_pred=rf_3.predict(X_test_3)
14 print('Accuracy of Random Forest on dataset 3: ',np.sum(y_3_rf_pred == y_test_3)/len(y_test_3))
15
16 nb_3=GaussianNB()
17 nb_3.fit(X_train_3,y_train_3)
18 y_3_nb_pred=nb_3.predict(X_test_3)
19 print('Accuracy of Naive Bayes on dataset 3: ',np.sum(y_3_nb_pred == y_test_3)/len(y_test_3))
```

Accuracy of KNN on dataset 3: 0.6818181818181818

Accuracy of Decision Tree on dataset 3: 0.7337662337662337

Accuracy of Random Forest on dataset 3: 0.7727272727272727  
Accuracy of Naive Baves on dataset 3: 0.7597402597402597

Với cách chọn ở df\_3 thì ta có độ chính xác của dataset trên 3 thuật toán Decision Tree, Random Forest, Naive Bayes cao hơn so với ở df\_1 và df\_2