Mini Project

SỬ DỤNG CÁC MÔ HÌNH MACHINE LEARNING ĐỂ CHẨN ĐOÁN BỆNH TIỂU ĐƯỜNG

Outline

- Giới thiệu
- Phân tích dữ liệu
- Cài đặt mô hình
- Kết luận và hướng phát triển

Giới thiệu: Giới thiệu đề tài

- Bệnh tiểu đường:là thuật ngữ dùng để đề cập tới một nhóm bệnh gây ảnh hưởng đến cách cơ thể, liên quan lượng đường (glucose) trong máu.

- Mô tả đề tài: sử dụng các mô hình học máy… và sử dụng các dữ liệu, 8 thuộc tính về cơ thể như tuổi tác, nồng độ glucose, huyết áp, độ dày của da, tiền sử bệnh, BMI, insulin, số lần mang thai… để dự đoán nguy cơ mắc bệnh của người đó.

Giới thiệu: Dataset

- Bộ dữ liệu sử dụng: lấy từ tập dataset [Diabetes Kaggle]

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

Phân tích dữ liệu

- Thực hiện một số khảo sát về dữ liệu
  - Thống kê dữ liệu
  - Phân tích dữ liệu

  …

Thống kê dữ liệu

Tập dữ liệu chứa 768 hàng bản ghi và 9 cột thuộc tính.

Các loại dữ liệu của các thuộc tính bao gồm 1 số nhị phân rời rạc định lượng, 6 số nguyên rời rạc định lượng và 2 số thực liên tục.

Sử dụng không gian bộ nhớ ít nhất là 54,1 kilobyte (KB).

```
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                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Thống kê dữ liệu

```
data = pd.read_csv('diabetes_data.csv')
data.head()
```

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

Phân tích dữ liệu

Phân tích và tiền xử lý dữ liệu

In [17]:
```
# liệt kê các giá trị null
NumofNA = dict(zip(data.columns.tolist(),data.isna().sum().tolist()))
print(NumofNA)
```

```
{'Pregnancies': 0, 'Glucose': 0, 'BloodPressure': 0, 'SkinThickness': 0, 'Insulin': 0, 'BMI': 0, 'DiabetesPedigreeFunction': 0, 'Age': 0,
'Outcome': 0}
```

In [18]:
```
# Liệt kê các giá trị khác nhau của các cột
data.agg(['nunique'])
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
nunique	17	136	47	51	186	248	517	52	2

Phân tích dữ liệu

Describe() được sử dụng để thu thập số liệu thống kê tóm tắt bao gồm các thước đo về xu hướng trung tâm như giá trị trung bình và giá trị trung bình, và các thước đo độ phân tán như độ lệch chuẩn, rất hữu ích trong việc cung cấp mô tả nhanh chóng và đơn giản về tập dữ liệu và các đặc điểm của nó.

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

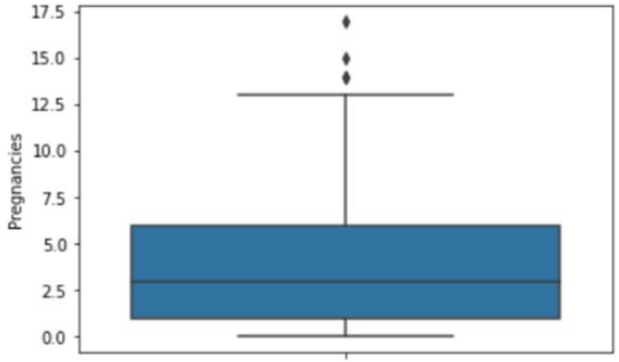
Phân tích dữ liệu

01

Pregnancies: được xử lý bằng cách thay thế các giá trị lớn bằng một giá trị đại diện, với số lần mang thai là 10

```
# kiểm tra ngoại lệ bằng biểu đồ (Pregnancies: số lần mang thai)
sns.boxplot(y = fresh_data['Pregnancies'])
```

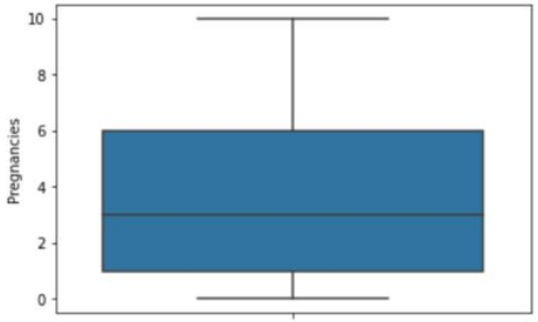
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7901ed6bd0>
```




```
# thay thế số lần mang thai lớn thành một giá trị
fresh_data.loc[(fresh_data['Pregnancies'] > 10), 'Pregnancies'] = 10

sns.boxplot(y = fresh_data['Pregnancies'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f790862ed90>
```



Phân tích dữ liệu

02

Glucose: nồng độ glucose trong huyết tương trong 2 giờ trong một bài kiểm tra dung nạp glucose qua đường miệng, được đo bằng miligam trên decilit (mg/dL). Dữ liệu nằm trong khoảng từ 0 đến 199, biết rằng giá trị không thể bằng 0 nên được làm mịn.

```
count    768.000000
mean     120.894531
std       31.972618
min        0.000000
25%       99.000000
50%      117.000000
75%      140.250000
max      199.000000
Name: Glucose, dtype: float64
Inter Quartile Range  41.25
Lower Limit  37.125
Upper Limit  202.125
```

```
fresh_data['Glucose'] = fresh_data['Glucose'].replace(0,
```

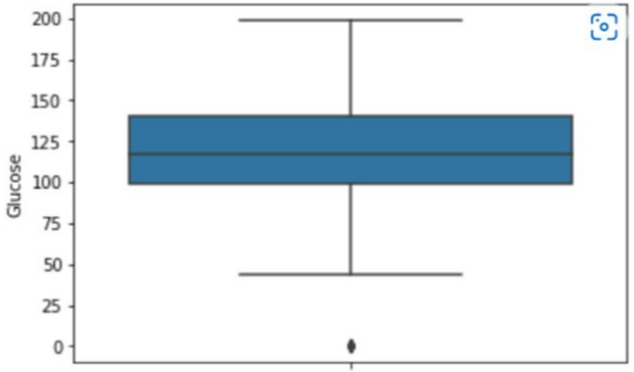
Phân tích dữ liệu

02

Glucose: vì không thể có giá trị bằng 0 nên xử lý bằng cách thay thế giá trị 0 bằng giá trị trung bình. Các giá trị lớn quá hoặc nhỏ quá được đưa về các giá trị đại diện tương ứng (70,160)

```
# boxplot of Glucose to check for outliers
sns.boxplot(y = fresh_data['Glucose'])
```

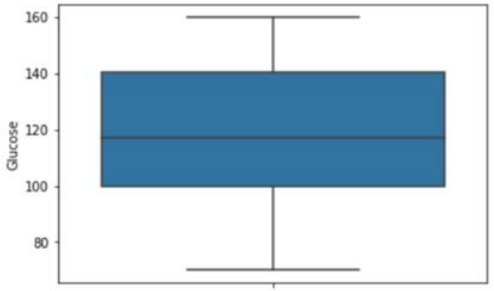
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79085a4150>
```




```
fresh_data['Glucose'] = fresh_data['Glucose'].replace(0, fresh_data.Glucose.mean())
fresh_data.loc[(fresh_data['Glucose'] < 70), 'Glucose'] = 70
fresh_data.loc[(fresh_data['Glucose'] > 160), 'Glucose'] = 160

sns.boxplot(y = fresh_data['Glucose'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79085126d0>
```



Phân tích dữ liệu

03

BloodPressure: huyết áp tâm trương, được đo bằng milimét thủy ngân (mm Hg). Dữ liệu nằm trong khoảng từ 0 đến 122, cho biết rằng giá trị không thể của 0 nên được làm mịn.

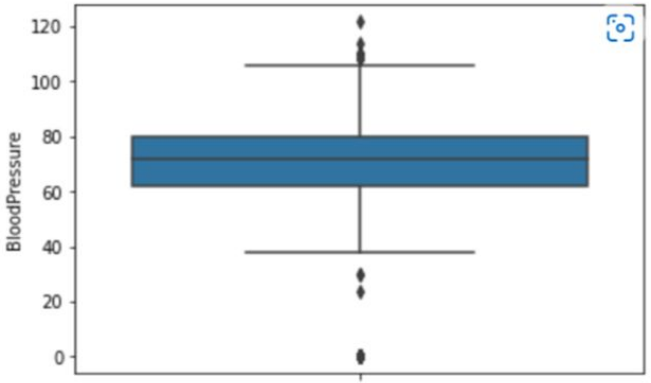
```
count    768.000000
mean      69.105469
std       19.355807
min        0.000000
25%       62.000000
50%       72.000000
75%       80.000000
max      122.000000
Name: BloodPressure, dtype: float64
Inter Quartile Range  18.0
Lower Whisker  35.0
Upper Whisker  107.0
```

Phân tích dữ liệu

03

Huyết áp: không thể có giá trị bằng 0 nên xử lý bằng cách thay thế giá trị 0 bằng giá trị trung bình. Các giá trị lớn quá hoặc nhỏ quá được đưa về các giá trị đại diện tương ứng (50,90)

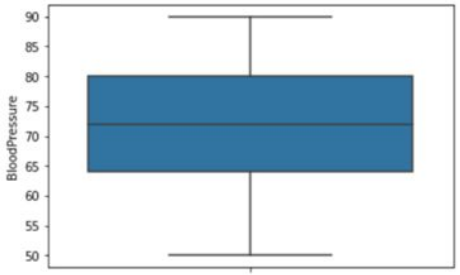
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79085ce790>
```




```
fresh_data['BloodPressure'] = fresh_data['BloodPressure'].replace(0, fresh_data.BloodPressure.mean())
fresh_data.loc[(fresh_data['BloodPressure'] < 50) , 'BloodPressure'] = 50
fresh_data.loc[(fresh_data['BloodPressure'] > 90), 'BloodPressure'] = 90

sns.boxplot(y = fresh_data['BloodPressure'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f790846b750>
```



Phân tích dữ liệu

04

SkinThickness: độ dày nếp gấp da cơ, được tính bằng milimét (mm). Dữ liệu nằm trong khoảng từ 0 đến 99

```
count    768.000000
mean      20.536458
std       15.952218
min        0.000000
25%        0.000000
50%       23.000000
75%       32.000000
max       99.000000
Name: SkinThickness, dtype: float64
Inter Quartile Range  32.0
Lower Limit  -48.0
Upper Limit  80.0
```

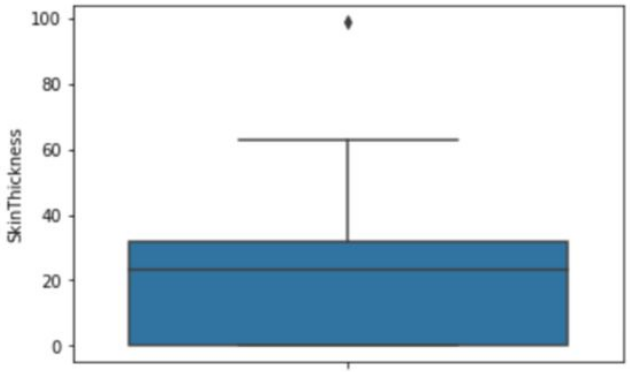
Phân tích dữ liệu

04

SkinThickness: được xử lý bằng cách các giá trị quá lớn hoặc quá nhỏ được đưa về các giá trị đại diện tương ứng (20,40)

```
sns.boxplot(y = fresh_data['SkinThickness'])
```

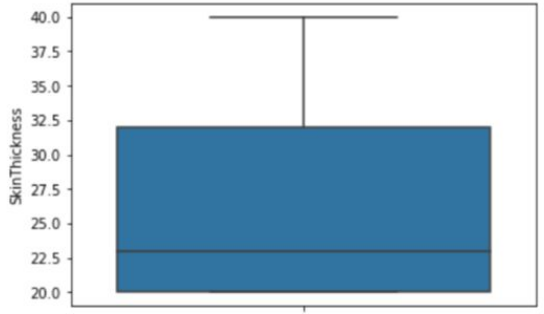
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79084b3450>
```




```
fresh_data.loc[(fresh_data['SkinThickness'] < 20), 'SkinThickness'] = 20
fresh_data.loc[(fresh_data['SkinThickness'] > 40), 'SkinThickness'] = 40

sns.boxplot(y = fresh_data['SkinThickness'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79083b6a50>
```



Phân tích dữ liệu

05

Insulin: là insulin huyết thanh kéo dài 2 giờ, được đo bằng đơn vị micromet trên mililit (muU/ml). Dữ liệu nằm trong khoảng từ 0 đến 846

```
count    768.000000
mean      79.799479
std      115.244002
min        0.000000
25%        0.000000
50%       30.500000
75%      127.250000
max      846.000000
Name: Insulin, dtype: float64
Inter Quartile Range  127.25
Lower Limit  -190.875
Upper Limit  318.125
```

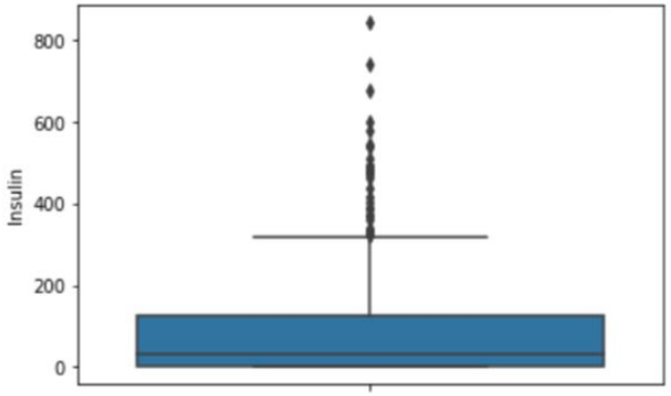
Phân tích dữ liệu

05

Insulin: được xử lý bằng cách thay thế các giá trị quá lớn được đưa về các giá trị đại diện tương ứng: 150

```
# boxplot of Insulin to check for outliers
sns.boxplot(y = fresh_data['Insulin'])
```

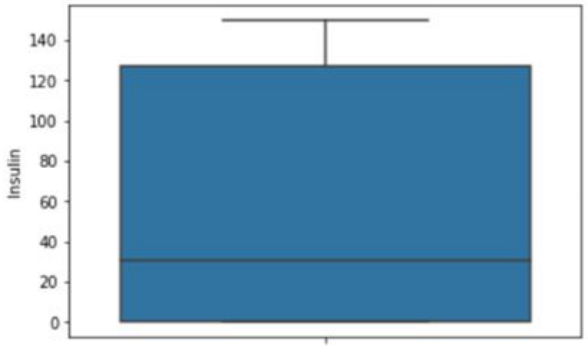
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7908377910>
```




```
# fresh_data['Insulin'] = fresh_data['Insulin'].replace(0, fresh_data.Insulin.mean())
fresh_data.loc[(fresh_data['Insulin'] > 150) , 'Insulin'] = 150

sns.boxplot(y = fresh_data['Insulin'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7908294c90>
```



Phân tích dữ liệu

06

BMI: chỉ số khối cơ thể (BMI) cho cân nặng tính bằng kg và chiều cao tính bằng m (kg/m2). Dữ liệu nằm trong khoảng từ 0 đến 67

```
count    768.000000
mean      31.992578
std        7.884160
min        0.000000
25%       27.300000
50%       32.000000
75%       36.600000
max       67.100000
Name: BMI, dtype: float64
Inter Quartile Range  9.3
Lower Limit  13.35
Upper Limit  50.550000000000004
```

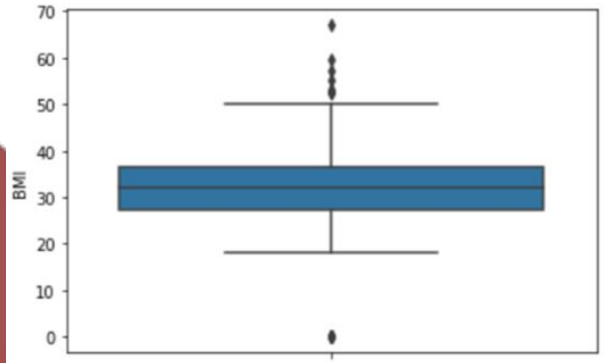
Phân tích dữ liệu

06

BMI: Chỉ số BMI không thể có giá trị bằng 0 nên xử lý bằng cách thay thế giá trị 0 bằng giá trị trung bình. Các giá trị lớn quá hoặc nhỏ quá được đưa về các giá trị đại diện tương ứng (15,40)

```
# kiểm tra ngoại lệ của bmi
sns.boxplot(y = fresh_data['BMI'])
```

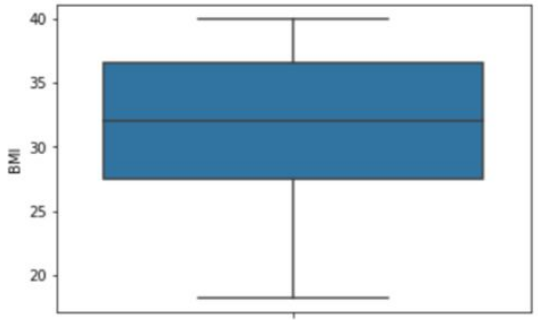
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7908285050>
```




```
fresh_data['BMI'] = fresh_data['BMI'].replace(0, fresh_data.BMI.mean())
fresh_data.loc[(fresh_data['BMI'] < 15), 'BMI'] = 15
fresh_data.loc[(fresh_data['BMI'] > 40), 'BMI'] = 40

sns.boxplot(y = fresh_data['BMI'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79081eaf10>
```



Phân tích dữ liệu

07

DiabestesPedigreeFunction: là một chức năng đánh giá khả năng mắc bệnh tiểu đường dựa trên tiền sử gia đình, với phạm vi thực tế từ 0,08 đến 2,42.

```
count    768.000000
mean       0.471876
std        0.331329
min        0.078000
25%        0.243750
50%        0.372500
75%        0.626250
max        2.420000
Name: DiabetesPedigreeFunction, dtype: float64
Inter Quartile Range  0.38249999999999995
Lower Limit  -0.32999999999999996
Upper Limit  1.2
```

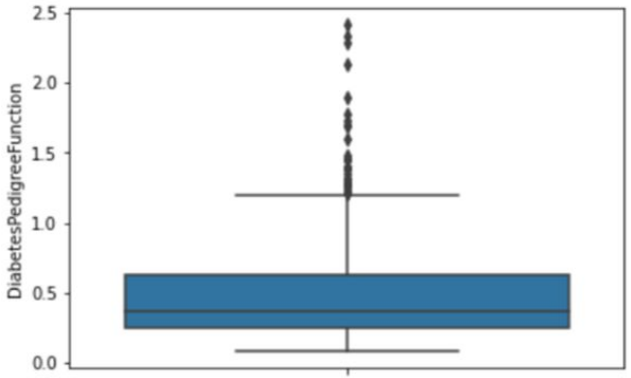
Phân tích dữ liệu

07

DiabestesPedigreeFunction: vì tỷ lệ thì khó có thể quá 1 nên xử lý bằng cách thay các giá trị lớn hơn 1 bằng 1

```
sns.boxplot(y = fresh_data['DiabetesPedigreeFunction'])
```

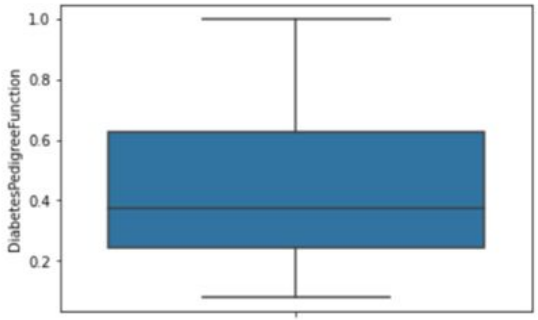
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79081ea310>
```




```
fresh_data.loc[(fresh_data['DiabetesPedigreeFunction'] > 1), 'DiabetesPedigreeFunction'] = 1

sns.boxplot(y = fresh_data['DiabetesPedigreeFunction'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f790814ea50>
```



Phân tích dữ liệu

08

Age (tuổi): tính theo năm có phạm vi thực tế từ 21 đến 81.

```
Mean      33.240885416666664
Median    29.0
count     768.000000
mean       33.240885
std        11.760232
min        21.000000
25%        24.000000
50%        29.000000
75%        41.000000
max        81.000000
Name: Age, dtype: float64
Inter Quartile Range  17.0
Lower Limit  -1.5
Upper Limit  66.5
```

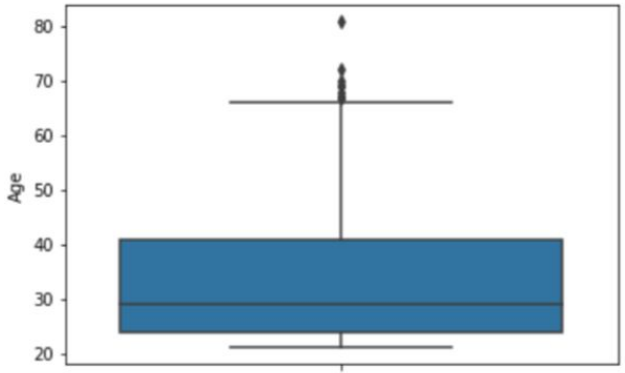
Phân tích dữ liệu

08

Age (tuổi): thay thế giá trị lớn hơn 60 thành 60

```
# kiểm tra ngoại lệ của tuổi
sns.boxplot(y = fresh_data['Age'])
```

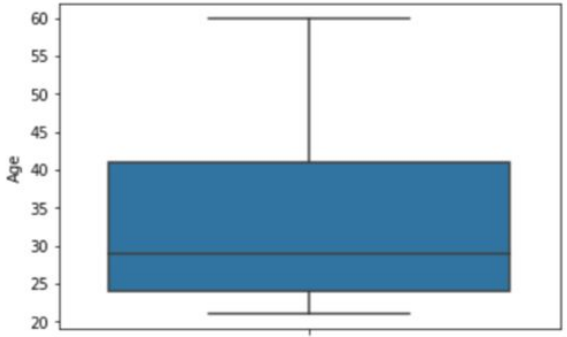
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79080a9e50>
```



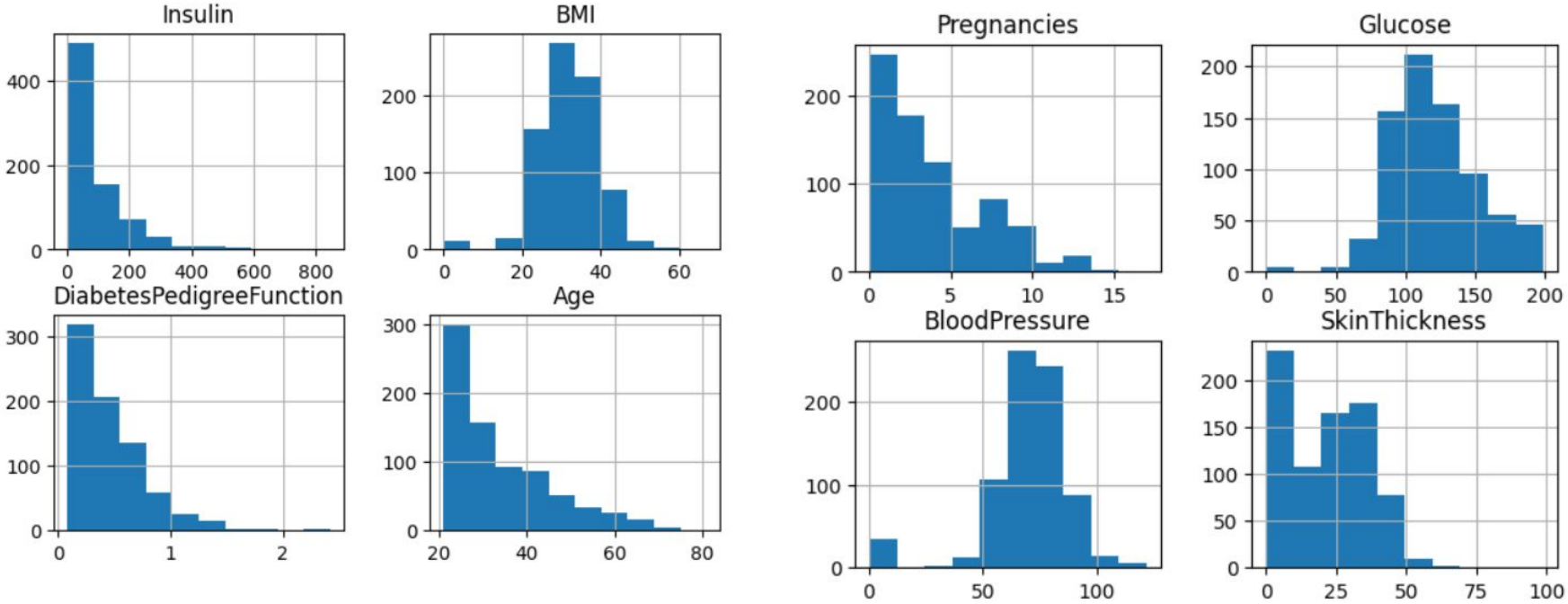

```
fresh_data.loc[(fresh_data['Age'] > 60) , 'Age'] = 60

sns.boxplot(y = fresh_data['Age'])
```

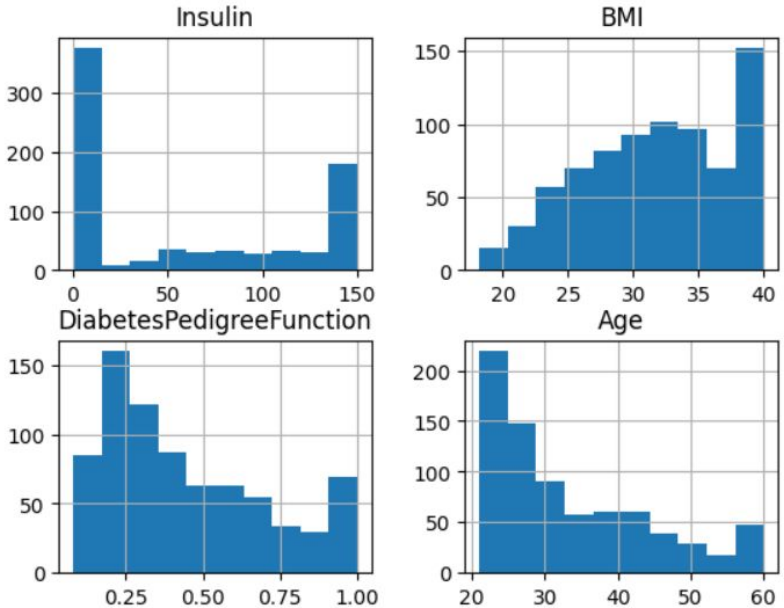
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f790801c410>
```

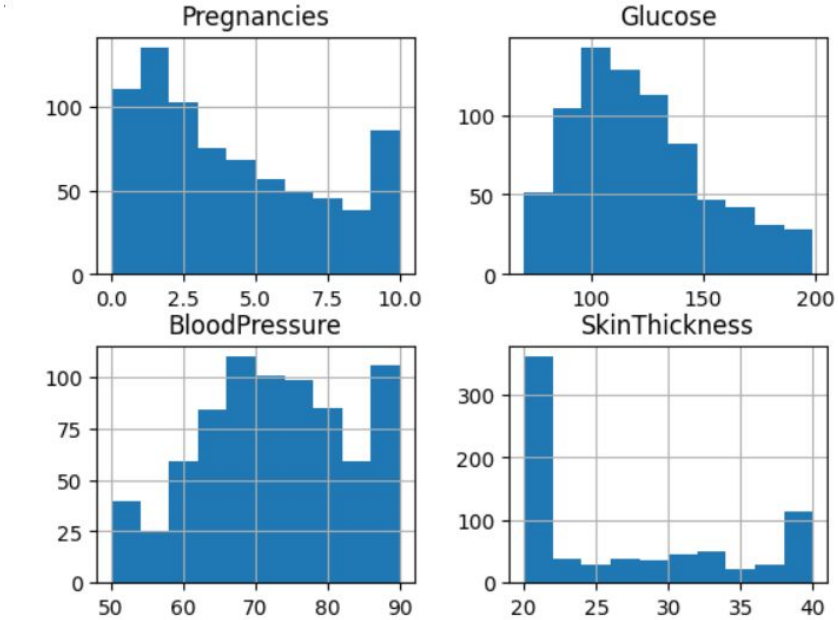


Phân tích dữ liệu



Phân tích dữ liệu





3. Cài đặt mô hình

Bộ dữ liệu bao gồm: 768 hàng

- **80% bộ dữ liệu được sử dụng làm tập train: 614 hàng**
- **20% bộ dữ liệu được sử dụng làm tập test: 154 hàng**

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(final,fresh_data['Outcome'], test_size=0.2, random_state=101)
```

3. Mô hình Decision Tree

Độ chính xác: 76.62%

```
[217] from sklearn import metrics
      import matplotlib.pyplot as plt
      print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
DecisionTrees's Accuracy:  0.7662337662337663
```

3. Mô hình KNN

Độ chính xác: 77.27% (với k = 5)

```
[234] from sklearn import metrics
      print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
      print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))

      Train set Accuracy:  0.8224755700325733
      Test set Accuracy:  0.7727272727272727
```

3. Mô hình KNN

Độ chính xác cao nhất: 80.52% (với k = 10)

```
# Your code here
Ks = 15
mean_acc = np.zeros((Ks-1))

# Try out different k for your kNN algorithm
for k in range(1,Ks):
  neigh = KNeighborsClassifier(n_neighbors=k)
  neigh.fit(X_train,y_train)
  mean_acc[k-1] = metrics.accuracy_score(y_test,neigh.predict(X_test))

# Print the accuracy of different k
mean_acc
```

```
array([0.7012987 , 0.72077922, 0.77272727, 0.77922078, 0.77272727,
       0.77272727, 0.76623377, 0.77922078, 0.79220779, 0.80519481,
       0.78571429, 0.78571429, 0.77272727, 0.75974026])
```

+ Mã + Văn bản

```
[323] print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.8051948051948052 with k= 10
```

3. Mô hình Naive Bayes

Độ chính xác: 75.32%

```
[350] #Import scikit-learn metrics module for accuracy calculation
      from sklearn import metrics

      # Model Accuracy, how often is the classifier correct?
      print("Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
Accuracy: 0.7532467532467533
```

Hướng phát triển

- Bổ sung một số kĩ thuật xử lý dữ liệu
- Mô hình Decision Tree & Random Forest: custom cây
- Cài đặt thử nghiệm các mô hình khác

Thank you