

TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



BÀI BÁO CÁO
**PHÂN TÍCH TÌNH TRẠNG, SỨC KHỎE CỦA NHỮNG NGƯỜI
BỆNH HOẶC KHÔNG MẮC BỆNH TIÊU ĐƯỜNG**

Tên thành viên:

1. Ngô Thanh Lam - 2151050220
2. Nguyễn Như Phong - 2151050326
3. Lê Văn Tân - 2151050391

Người hướng dẫn

Tp.Hồ Chí Minh, 2023

MỤC LỤC

A. PHẦN MỞ ĐẦU.....	5
1. Lý do chọn chủ đề.....	5
2. Bệnh tiểu đường là gì?.....	5
B. PHẦN BÁO CÁO.....	6
I. CHUẨN BỊ DỮ LIỆU- TRIỂN KHAI BẰNG PYTHON.....	6
1. Bộ dữ liệu và mô hình được sử dụng.....	6
2. Mô tả thuộc tính.....	6
3. Nhập thư viện và tải tập dữ liệu.....	7
4. Hiển thị dữ liệu.....	8
5. Đổi tên DiabPedigreeFunction thành DPF để có tính nhất quán tốt hơn.....	8
6. Thông tin tổng quan của dữ liệu.....	9
7. Kiểm tra số lượng giá trị duy nhất trong mỗi cột.....	9
8. Thông kê tóm tắt.....	10
9. Xử lý các cột có giá trị '0'.....	10
10. Giá trị bị mất.....	10
a. Xử lý :Glucose, BloodPressure, BMI.....	11
b. Xử lý: Insulin dựa trên Glucose.....	11
c. Xử lý: Skin Thickness theo BMI.....	12
II. PHÂN TÍCH DỮ LIỆU KHÁM PHÁ.....	13
1. Phân tích đơn biến (0 - không mắc bệnh tiểu đường).....	13
a. Số lượng biến mục tiêu.....	13
2. Phân tích hai biến.....	15
2.1 Correlation matrix.....	15
2.2 Tương quan với bệnh tiểu đường.....	16
2.3 Biểu đồ phân bố tính năng với Outcome.....	18
2.4 Box_Plot.....	18
2.5 Pair plot.....	19
2.6 Biểu đồ phân tán của Blood Pressure với Glucose với Age wrt Kết quả.....	20
2.7 Sơ đồ phân tán của Glucose với Insulin với DPF wrt Kết quả.....	20
III. XỬ LÝ DỮ LIỆU.....	21
1. Loại bỏ các ngoại lệ.....	21
2. Loại bỏ độ lệch.....	21
3. Mẫu tính năng đã sẵn sàng.....	23
IV. MÔ HÌNH & SIÊU ĐIỀU CHỈNH.....	24
1. Cài đặt gói.....	24
2. Đào tạo và phân chia dữ liệu.....	24
3. Huấn luyện dữ liệu với các mô hình cơ sở mà không cần điều chỉnh siêu tham số.....	25
4. Điều chỉnh siêu tham số bằng GridSearchCV.....	26
V. PHÂN CỤM-GOM CỤM BỆNH NHÂN TIỂU ĐƯỜNG.....	27
1. Phân tích phân cụm.....	27
2. Gom cụm.....	30
VI. THUẬT TOÁN PHÂN LOẠI.....	31
1. KNN.....	31
2. Naive Bayes.....	33
3. Cây quyết định.....	34
a. Lập mô hình bằng CART.....	34

b. Đánh giá hiệu suất bằng xác thực chéo.....	35
c. Tối ưu hóa siêu tham số với GridSearchCV.....	37
GridSearchCV là một phương pháp tìm kiếm siêu tham số để tối ưu hóa hiệu suất của mô hình học máy. Nó tìm kiếm qua một lưới các giá trị siêu tham số và đánh giá mô hình trên mỗi tổ hợp, trả về giá trị tối ưu của siêu tham số dựa trên độ đo hiệu suất đã chọn trước đó.....	37
d. Mô hình cuối cùng.....	37
e. Phân tích độ phức tạp của mô hình với các đường cong học tập.....	39
f. Trực quan hóa cây quyết định.....	40
VII. DỰ ĐOÁN [TYPE-I & TYPE-II] HOẶC [T1DM & T2DM]	41
1. Giải thích về dự đoán.....	41
a. Chuẩn đoán bệnh tiểu đường và tiền tiểu đường:.....	41
b. Phân biệt loại bệnh tiểu đường bằng mức độ insulin:.....	41
c. Mô tả chức năng:.....	41
d. Quá trình dự đoán:.....	42
2. Trích xuất Glucose, Insulin và Kết quả trong tệp csv mới.....	42
3. Đọc và hiển thị dữ liệu mới.....	42
4. Mô tả dữ liệu.....	43
5. Huấn luyện và phân chia dữ liệu.....	43
6. Hiển thị hình dạng của dữ liệu được chia tách.....	43
7. Sử dụng phân lớp KNN.....	44
8. Lấy dữ liệu tùy chỉnh để kiểm tra dự đoán.....	44
9. Lưu và hiển thị tất cả các dự đoán của T1DM và T2DM.....	45
10. Đã lưu mô hình bằng Pickle.....	45
VIII. KẾT HỢP DỮ LIỆU VÀ PHÂN TÍCH	45
1. Bộ dữ liệu và mô hình được sử dụng.....	45
2. Mô tả thuộc tính.....	46
3. Xử lý dữ liệu.....	46
4. Kiểm tra số lượng giá trị duy nhất trong mỗi cột.....	47
5. Nhận xét bổ sung cho tập dữ liệu Pima Indians Diabetes.....	48
C. PHẦN TÀI LIỆU THAM KHẢO	51

DANH SÁCH HÌNH ẢNH

A. PHẦN MỞ ĐẦU.....	5
1. Lý do chọn chủ đề.....	5
2. Bệnh tiểu đường là gì?.....	5
B. PHẦN BÁO CÁO.....	6
I. CHUẨN BỊ DỮ LIỆU- TRIỂN KHAI BẰNG PYTHON.....	6
II. PHÂN TÍCH DỮ LIỆU KHÁM PHÁ.....	13
Hình 1.1. Biểu đồ tỉ lệ người mắc bệnh và không mắc bệnh tiểu đường.....	13
Hình 1.2. Biểu đồ thể hiện số lượng người mắc bệnh tiểu đường và không mắc bệnh tiểu đường.....	13
Hình 1.3. Biểu đồ Box Plot.....	14
Hình 1.4. Biểu đồ Histogram.....	14
Hình 2.1. Ma trận tương quan giữa các đặc trưng.....	16
Hình 2.2. Ma trận tương quan của các đặc trưng với ‘Outcome’.....	17
Hình 2.3. Phân phối các tính năng theo biến mục tiêu ‘Outcome’	18
Hình 2.4. Biểu đồ hộp.....	18
Hình 2.5. Pair-plot.....	19
Hình 2.6. Phân tán của Glucose với Insulin với DPF wrt Kết quả.....	20
Hình 2.7. Phân tán của Glucose với Insulin với DPF wrt.....	20
III. XỬ LÝ DỮ LIỆU.....	21
IV. MÔ HÌNH & SIÊU ĐIỀU CHỈNH.....	24
V. PHÂN CỤM-GOM CỤM BỆNH NHÂN TIỂU ĐƯỜNG.....	27
Hình 1.1. Biểu đồ Elbow.....	28
Hình 1.2. Biểu đồ phân tán trực quan hóa mối quan hệ giữa Glucose và BMI.....	29
Hình 2.1. Biểu đồ gom cụm giữa ‘Blood Pressure’ và ‘Pregnancies’	31
VI. THUẬT TOÁN PHÂN LOẠI.....	31
Hình 3.1. Cây quyết định.....	40
VII. DỰ ĐOÁN [TYPE-I & TYPE-II] HOẶC [T1DM & T2DM].....	41
VIII. KẾT HỢP DỮ LIỆU VÀ PHÂN TÍCH.....	45
Hình 5.1: Tỷ lệ mắc bệnh và không mắc bệnh Tiểu đường của Người phụ nữ da đỏ với Người phụ nữ Mỹ.....	48
Hình 5.2. Biểu đồ Huyết áp, BMI và Bệnh tiểu đường.....	49
Hình 5.3. Biểu đồ nhóm tuổi liên quan đến bệnh tiểu đường.....	49
Hình 5.4. Biểu đồ Bệnh đột quy và bệnh tiểu đường.....	50
Hình 5.5. Biểu đồ hoạt động hàng ngày - sử dụng rau - sử dụng trái cây và bệnh tiểu đường.....	50
C. PHẦN TÀI LIỆU THAM KHẢO.....	51

A. PHẦN MỞ ĐẦU

1. Lý do chọn chủ đề

Bệnh tiểu đường đứng trong số những mối nguy hiểm đáng kể trên toàn cầu, không chỉ là một căn bệnh đơn lẻ mà còn là nguyên nhân của nhiều vấn đề sức khỏe nghiêm trọng khác như bệnh tim, mù lòa, và nhiều bệnh lý khác. Thường thì quá trình chuẩn đoán bệnh tiểu đường yêu cầu bệnh nhân phải đến trung tâm y tế, tìm kiếm ý kiến bác sĩ và chờ đợi một khoảng thời gian đáng kể để nhận được bản báo cáo chuẩn đoán.

Dự án này đặt ra mục tiêu quan trọng là xác định liệu bệnh nhân có mắc bệnh tiểu đường hay không, dựa trên các phép đo chuẩn đoán. Thay vì phương pháp truyền thống, dự án này hứa hẹn mang lại sự thuận tiện và nhanh chóng hơn trong việc xác định tình trạng sức khỏe của bệnh nhân thông qua các thông số chuẩn đoán quan trọng. Điều này có thể giúp giảm bớt thời gian chờ đợi và tăng cơ hội phát hiện sớm bệnh tiểu đường, giúp cải thiện chất lượng cuộc sống và dự báo sức khỏe cho bệnh nhân.

2. Bệnh tiểu đường là gì?

Bệnh tiểu đường là một nhóm bệnh lý trong đó cơ thể không sản xuất đủ insulin, không sử dụng insulin một cách hiệu quả, hoặc thậm chí là kết hợp cả hai vấn đề trên. Khi một trong những tình trạng này xảy ra, cơ thể không thể chuyển đường từ máu vào các tế bào, dẫn đến việc tăng đường trong máu.

Glucose, dạng đường tồn tại trong máu, đóng vai trò quan trọng như một nguồn năng lượng chính. Sự thiếu hụt insulin hoặc khả năng chống lại insulin khiến cho đường tích tụ trong máu, gây ra nhiều vấn đề sức khỏe.

Có ba loại chính của bệnh tiểu đường, bao gồm tiểu đường tuýp 1, tiểu đường tuýp 2 và tiểu đường thai kỳ. Mỗi loại bệnh tiểu đường có đặc điểm và ảnh hưởng khác nhau đến cơ thể, đòi hỏi sự quản lý và điều trị đặc biệt.

B. PHẦN BÁO CÁO

I. CHUẨN BỊ DỮ LIỆU- TRIỂN KHAI BẰNG PYTHON

1. Bộ dữ liệu và mô hình được sử dụng

Bộ dữ liệu: Pima Indians Diabetes Database (<https://www.kaggle.com/>) . Bộ dữ liệu này có nguồn gốc từ Viện Tiểu đường, Tiêu hóa và Bệnh thận Quốc gia, chứa thông tin của 768 phụ nữ từ một khu dân cư gần Phoenix, Arizona, Hoa Kỳ. Kết quả xét nghiệm là Bệnh tiểu đường, 258 xét nghiệm dương tính và 500 xét nghiệm âm tính. Mục tiêu của tập dữ liệu là dự đoán chuẩn đoán xem bệnh nhân có mắc bệnh tiểu đường hay không, dựa trên các phép đo chẩn đoán nhất định có trong tập dữ liệu. Một số hạn chế đã được đặt ra đối với việc lựa chọn các trường hợp này từ cơ sở dữ liệu lớn hơn. Đặc biệt, tất cả bệnh nhân ở đây đều là nữ, ít nhất 21 tuổi, gốc Án Độ Pima.

Nội dung bộ dữ liệu: Các bộ dữ liệu bao gồm một số biến dự đoán y tế và một biến mục tiêu ‘Outcome’. Các biến dự đoán bao gồm số lần mang thai mà bệnh nhân đã có, chỉ số BMI, mức insulin, tuổi, v.v.

Mô hình: Decision Tree, Random Forest, Support Vector Machine(Linear, Ploy, Bayesian), Logistic Regression, K-Neighbours Classifiers, AdaBoost Classifier, Gradient Boost Classifier

2. Mô tả thuộc tính

- **Pregnancies:** Số lần mang thai
- **Glucose:** Nồng độ Glucose trong huyết tương (mg/dl)
- **Blood Pressure:** Huyết áp tâm trương (mmHg)
- **Skin Thickness:** Một giá trị được sử dụng để ước tính lượng mỡ trong cơ thể. Độ dày nếp gấp da cơ tam đầu bình thường ở phụ nữ là 23mm. Độ dày cao hơn dẫn đến béo phì và nguy cơ mắc bệnh tiểu đường tăng lên.
- **Insulin:** Insulin huyết thanh 2 giờ (mu U/ml)
- **BMI:** Chỉ số khối cơ thể (cân nặng tính bằng kg/ chiều cao tính bằng m²)
- **Diabetes Pedigree Function:** Nó cung cấp thông tin về tiền sử bệnh tiểu đường ở người thân và mối quan hệ di truyền của người thân đó với bệnh nhân. Chức năng phả hệ cao hơn có nghĩa là bệnh nhân có nhiều khả năng mắc bệnh tiểu đường hơn.
- **Age:** Tuổi (năm)
- **Outcome:** Biến loại (0 hoặc 1) trong đó ‘0’ biểu thị bệnh nhân không mắc bệnh tiểu đường và ‘1’ biểu thị bệnh nhân mắc bệnh tiểu đường.

3. Nhập thư viện và tải tập dữ liệu

```
[ ] !pip install plotly  
!pip install scikit-plot  
!pip install sklearn-porter
```

```
[ ] import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import plotly.express as px  
import seaborn as sns  
import plotly.graph_objects as go  
import matplotlib.cm as cm  
import plotly.offline  
  
from sklearn.linear_model import LinearRegression , LogisticRegression  
from sklearn.model_selection import train_test_split, cross_val_score, cross_validate, GridSearchCV  
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix, classification_report, mean_squared_error  
from scikitplot.metrics import plot_roc  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.preprocessing import StandardScaler, RobustScaler  
  
from mpl_toolkits.mplot3d import Axes3D  
from pandas.plotting import parallel_coordinates, andrews_curves, radviz  
  
from sklearn.cluster import KMeans  
from plotly.subplots import make_subplots  
  
from sklearn.metrics import silhouette_score  
  
import joblib  
import pydotplus  
from matplotlib import pyplot as plt  
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export_text  
from sklearn.metrics import classification_report, roc_auc_score  
from sklearn.model_selection import train_test_split, GridSearchCV, cross_validate, validation_curve  
import graphviz  
  
import warnings  
from termcolor import colored  
warnings.filterwarnings("ignore")
```

```

#Chức năng phát hiện và làm sạch ngoại lệ
def outlier_thresholds(dataframe, col_name, q1=0.05, q3=0.95):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    return low_limit, up_limit

def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if (dataframe[(dataframe[col_name] > up_limit) | (dataframe[col_name] < low_limit)].any(axis=None)):
        return True
    else:
        return False

def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit

df = pd.read_csv('/content/drive/MyDrive/Data/diabetes.csv')

```

4. Hiển thị dữ liệu

Dữ liệu khảo sát từ Viện Tiểu đường, Tiêu hóa và Bệnh thận Quốc gia khi điều tra tất cả bệnh nhân ở đây đều là nữ, ít nhất 21 tuổi, gốc Án Độ Pima. Dữ liệu bao gồm 9 cột và 768 dòng mỗi dòng tương ứng với một bệnh nhân khi bị bệnh tiểu đường hoặc không bị bệnh tiểu đường

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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
...
763	10	101	76	48	180	32.9		0.171	63
764	2	122	70	27	0	36.8		0.340	27
765	5	121	72	23	112	26.2		0.245	30
766	1	126	60	0	0	30.1		0.349	47
767	1	93	70	31	0	30.4		0.315	23

768 rows × 9 columns

Nhận xét: Có một số giá trị trong Insulin không thể bằng 0. Vì vậy, cần phải xử lý chúng bằng cách quy gán. (thành giá trị mean)

5. Đổi tên DiabPedigreeFunction thành DPF để có tính nhất quán tốt hơn

Đổi tên DiabPedigreeFunction thành DPF để có tính nhất quán tốt hơn

```
[12] df = df.rename(columns = {'DiabetesPedigreeFunction':'DPF'})
```

6. Thông tin tổng quan của dữ liệu

```
[ ] 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   DPF            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
```

7. Kiểm tra số lượng giá trị duy nhất trong mỗi cột

```
▶ dict = {}  
for i in list(df.columns):  
    dict[i] = df[i].value_counts().shape[0]  
  
pd.DataFrame(dict,index=["unique count"]).transpose()  
  
[ ] unique count  
Pregnancies      17  
Glucose          136  
BloodPressure     47  
SkinThickness     51  
Insulin           186  
BMI               248  
DPF               517  
Age               52  
Outcome           2
```

8. Thống kê tóm tắt

```
df[con_cols].describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DPF	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00

Nhận xét:

Có sự khác biệt lớn về giá trị trung bình và chúng ta có thể thấy không có giá trị nào bị thiếu, nhưng đối với một số cột như Glucose , BP, Skin Thickness, BMI có giá trị tối thiểu là 0, điều này là không thể, do đó chúng ta có thể coi đây là giá trị bị thiếu và quy kết cho phù hợp.

9. Xử lý các cột có giá trị '0'

```
[85] features = df.columns  
cols = (df[features] == 0).sum()  
print(cols)
```

Pregnancies	111
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DPF	0
Age	0
Outcome	500
dtype: int64	

Không thể bỏ qua những giá trị dữ liệu rất nhỏ này. Vì vậy, hãy xử lý chúng.

10. Giá trị bị mất

```
[89] df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']].dropna()

[87] df.isnull().sum()

Pregnancies      0
Glucose          5
BloodPressure    35
SkinThickness   227
Insulin         374
BMI             11
DPF              0
Age              0
Outcome          0
dtype: int64
```



a. Xử lý :Glucose, BloodPressure, BMI

```
[94] #Thay thế các giá trị null bằng giá trị trung bình của cột đó:
df['Glucose'].fillna(df['Glucose'].median(), inplace =True)
df['BloodPressure'].fillna(df['BloodPressure'].median(), inplace =True)
df['BMI'].fillna(df['BMI'].median(), inplace =True)
```

b. Xử lý: Insulin dựa trên Glucose

```
[96] by_Glucose_Age_Insulin_Grp = df.groupby(['Glucose'])

def fill_Insulin(series):
    return series.fillna(series.median())
df['Insulin'] = by_Glucose_Age_Insulin_Grp['Insulin'].transform(fill_Insulin)

[97] df['Insulin'] = df['Insulin'].fillna(df['Insulin'].mean())
```

Nhận xét:

Insulin đóng vai trò quan trọng trong việc điều chỉnh nồng độ glucose trong máu. Insulin giúp tế bào trong cơ thể hấp thụ glucose từ máu và chuyển nó vào các tế bào để sử dụng làm năng lượng. Vì vậy, insulin đóng vai trò quan trọng trong việc điều chỉnh sự hấp thụ và sử dụng glucose, giúp duy trì một mức độ glucose trong máu ổn định và trong khoảng bình thường. Nếu cơ thể không sản xuất đủ insulin hoặc không phản ứng đúng với insulin, có thể dẫn đến tình trạng bất cân đối nồng độ glucose trong máu, như tiểu đường (diabetes).

c. Xử lý: Skin Thickness theo BMI

```
[98] by_BMI_Insulin = df.groupby(['BMI'])

def fill_SkinThickness(series):
    return series.fillna(series.mean())
df['SkinThickness'] = by_BMI_Insulin['SkinThickness'].transform(fill_SkinThickness)

[99] df['SkinThickness'].fillna(df['SkinThickness'].mean(), inplace= True)

[100] df.isnull().sum()

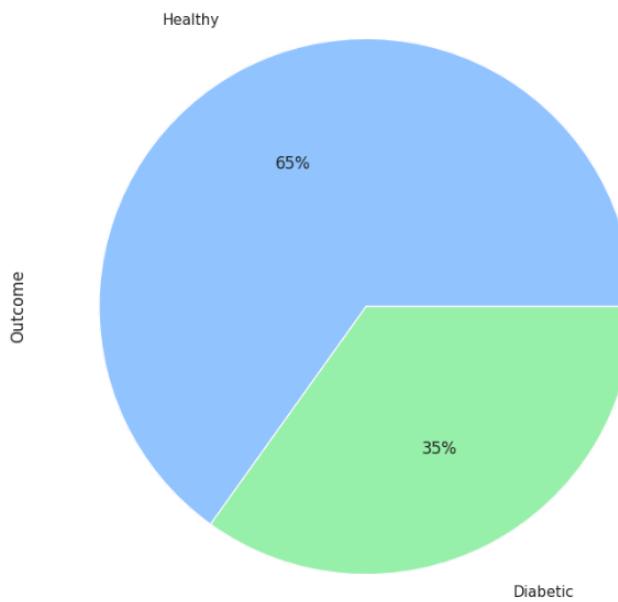
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DPF              0
Age              0
Outcome          0
dtype: int64
```

Nhận xét: Skin thickness và BMI không có liên quan trực tiếp với nhau. Tuy nhiên, cả hai có thể bị ảnh hưởng bởi cùng một yếu tố chung, đó là mức độ mỡ trong cơ thể. Mức độ mỡ trong cơ thể lại liên quan đến bệnh tiểu đường.

II. PHÂN TÍCH DỮ LIỆU KHÁM PHÁ

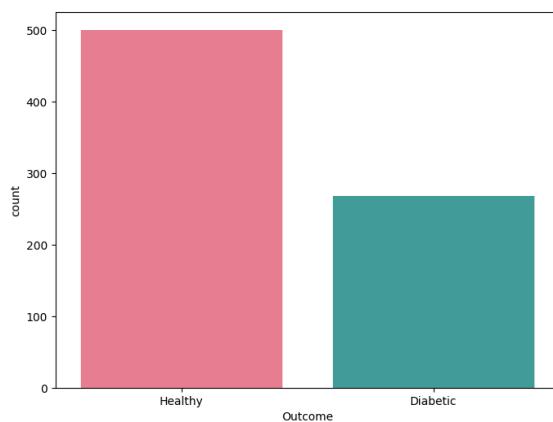
1. Phân tích đơn biến (0 - không mắc bệnh tiểu đường)

a. Số lượng biến mục tiêu



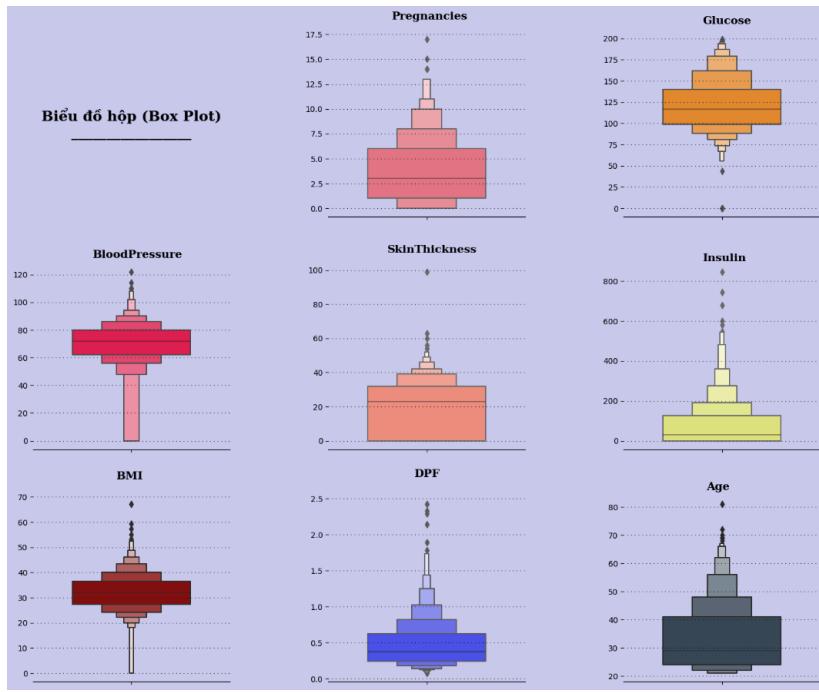
Hình 1.1. Biểu đồ tỉ lệ người mắc bệnh và không mắc bệnh tiểu đường

```
[101] from matplotlib.pyplot import figure, show\n\n    figure(figsize=(8,6))\n    ax = sns.countplot(x=df['Outcome'], data=df, palette="husl")\n    ax.set_xticklabels(["Healthy","Diabetic"])\n    healthy, diabetics = df['Outcome'].value_counts().values\n    print("Samples of diabetic people: ", diabetics)\n    print("Samples of healthy people: ", healthy)\n\nSamples of diabetic people: 268\nSamples of healthy people: 500
```

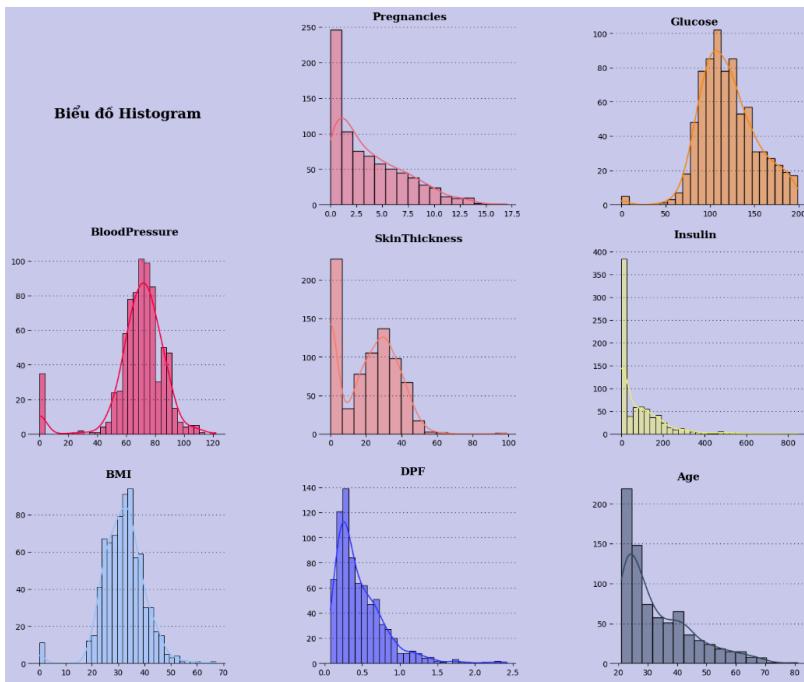


Hình 1.2. Biểu đồ thể hiện số lượng người mắc bệnh tiểu đường và không mắc bệnh tiểu đường

Nhận xét: Số lượng người bị bệnh tiểu đường chiếm thấp hơn số lượng người không bị bệnh tiểu đường.



Hình 1.3. Biểu đồ Box Plot



Hình 1.4. Biểu đồ Histogram

Nhận xét các chỉ số về người ít có khả năng bị bệnh tiểu đường

Pregnancies: Người mà mang thai từ 0-6 lần

- Số lần mang thai có thể ảnh hưởng đến sức khỏe của phụ nữ, và nghiên cứu đã chỉ ra rằng người mang thai nhiều lần có nguy cơ bị bệnh tiểu đường tăng.

Glucose Người có lượng đường huyết từ 100- 150 mg/dl

- Mức đường huyết trong khoảng này thường được coi là bình thường, và người có mức đường huyết ổn định ít khả năng phát triển bệnh tiểu đường.

Blood Pressure: Người có chỉ số huyết áp từ 70-80

- Huyết áp ổn định thường là một dấu hiệu tích cực, và người có huyết áp ổn định ít khả năng phát triển bệnh tiểu đường.

Skin Thickness: Người có chỉ số Skin Thickness từ 0-30

- Chỉ số Skin Thickness có thể liên quan đến mức mỡ dưới da, và người có chỉ số thấp hơn có thể ít khả năng phát triển bệnh tiểu đường.

Insulin: Người có chỉ số Insulin từ 0-200

- Mức Insulin bình thường thường liên quan đến sự cân bằng insulin trong cơ thể, và người có mức độ này ít khả năng gặp vấn đề về đường huyết.

BMI : Người có chỉ số BMI từ 25-35

- BMI trong khoảng này thường được coi là khoảng bình thường hoặc hơi thừa cân, và người có BMI trong khoảng này ít khả năng phát triển bệnh tiểu đường.

DPF: Người có chỉ số dòng họ tiểu đường từ 0.25-0.5

- Chỉ số DPF thường được sử dụng để đánh giá yếu tố di truyền, và người có chỉ số này trong khoảng ổn định thì ít khả năng bị ảnh hưởng bởi yếu tố di truyền.

Age: Người có tuổi từ 25- 45

- Tuổi có thể ảnh hưởng đến rủi ro bệnh tiểu đường, và người trẻ hơn thường ít khả năng phát triển bệnh

2. Phân tích hai biến

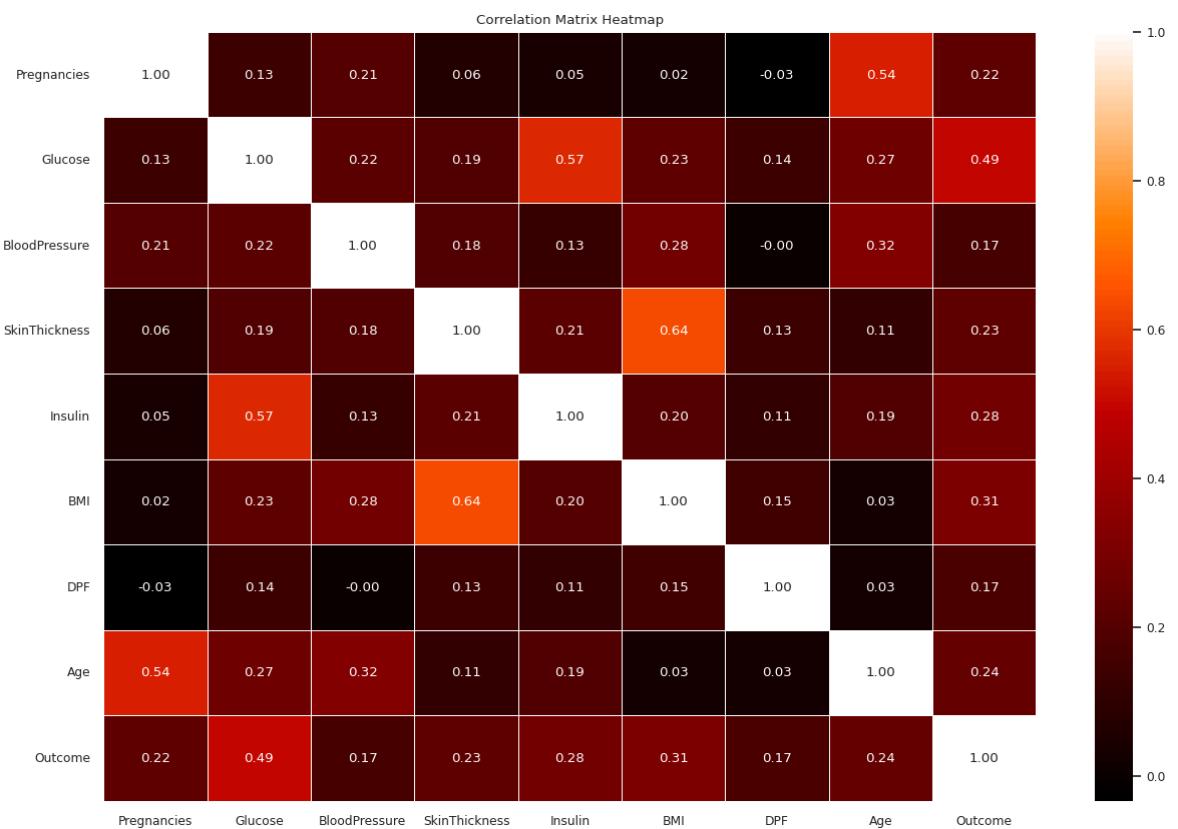
2.1 Correlation matrix

```
df_corr = df.corr().transpose()
df_corr
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age	Outcome
Pregnancies	1.000000	0.128213	0.208615	0.064381	0.046741	0.021559	-0.033523	0.544341	0.221898
Glucose	0.128213	1.000000	0.218937	0.188996	0.566640	0.231049	0.137327	0.266909	0.492782
BloodPressure	0.208615	0.218937	1.000000	0.183123	0.125499	0.281257	-0.002378	0.324915	0.165723
SkinThickness	0.064381	0.188996	0.183123	1.000000	0.210167	0.636708	0.128380	0.108672	0.232150
Insulin	0.046741	0.566640	0.125499	0.210167	1.000000	0.198895	0.114325	0.185146	0.279690
BMI	0.021559	0.231049	0.281257	0.636708	0.198895	1.000000	0.153438	0.025597	0.312038
DPF	-0.033523	0.137327	-0.002378	0.128380	0.114325	0.153438	1.000000	0.033561	0.173844
Age	0.544341	0.266909	0.324915	0.108672	0.185146	0.025597	0.033561	1.000000	0.238356
Outcome	0.221898	0.492782	0.165723	0.232150	0.279690	0.312038	0.173844	0.238356	1.000000

```
[43] #Graph I.
correlation_matrix = df.corr()

plt.figure(figsize=(15, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='gist_heat', linewidths=0.5, fmt=".2f")
plt.title("Correlation Matrix Heatmap")
plt.show()
```



Hình 2.1. Ma trận tương quan giữa các đặc trưng

Nhận xét:

Mỗi tương quan giữa các biến đặc trưng trong tập dữ liệu *thấp*, không có cặp biến nào có mối tương quan cao. Để hiểu rõ hơn về ảnh hưởng của từng đặc trưng đối với mục tiêu ('Outcome'), cần tiến hành phân tích mối tương tác giữa mỗi đặc trưng và 'Outcome'.

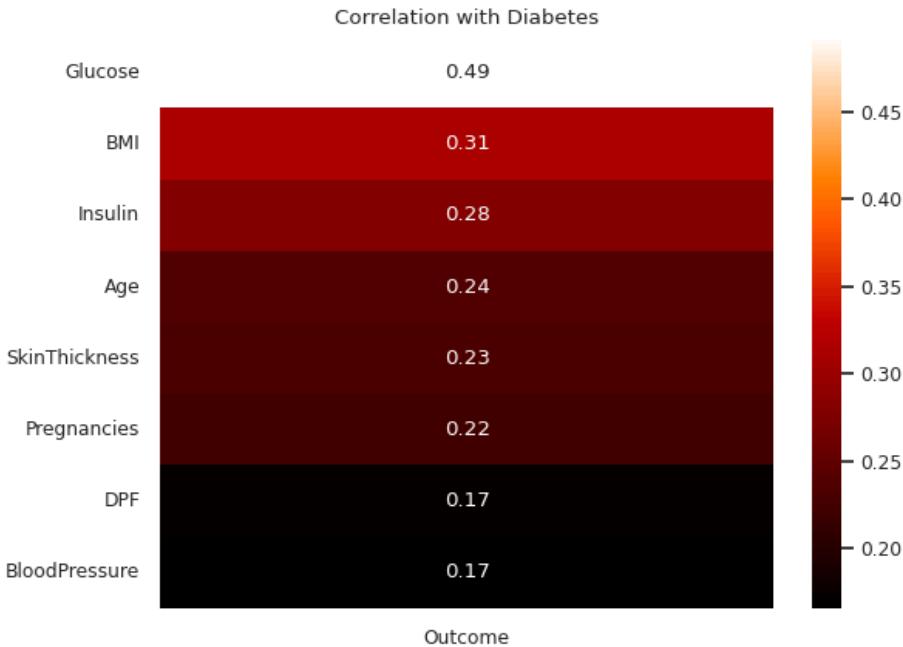
Ngoài ra, mối tương quan trung bình giữa 'Age' và 'Pregnancies' chỉ làm rõ thêm ý tưởng thông thường rằng phụ nữ có xu hướng có cơ hội mang thai cao hơn khi tuổi tăng lên.

2.2 Tương quan với bệnh tiểu đường

```
[50] corr = df.corr()
target_corr = corr['Outcome'].drop('Outcome')

# Sắp xếp các giá trị tương quan theo thứ tự giảm dần
target_corr_sorted = target_corr.sort_values(ascending=False)

# Tạo bản đồ nhiệt về các mối tương quan với cột mục tiêu
sns.set(font_scale=0.8)
sns.set_style("white")
sns.set_palette("PuBuGn_d")
sns.heatmap(target_corr_sorted.to_frame(), cmap="gist_heat", annot=True, fmt='.2f')
plt.title('Correlation with Diabetes')
plt.show()
```

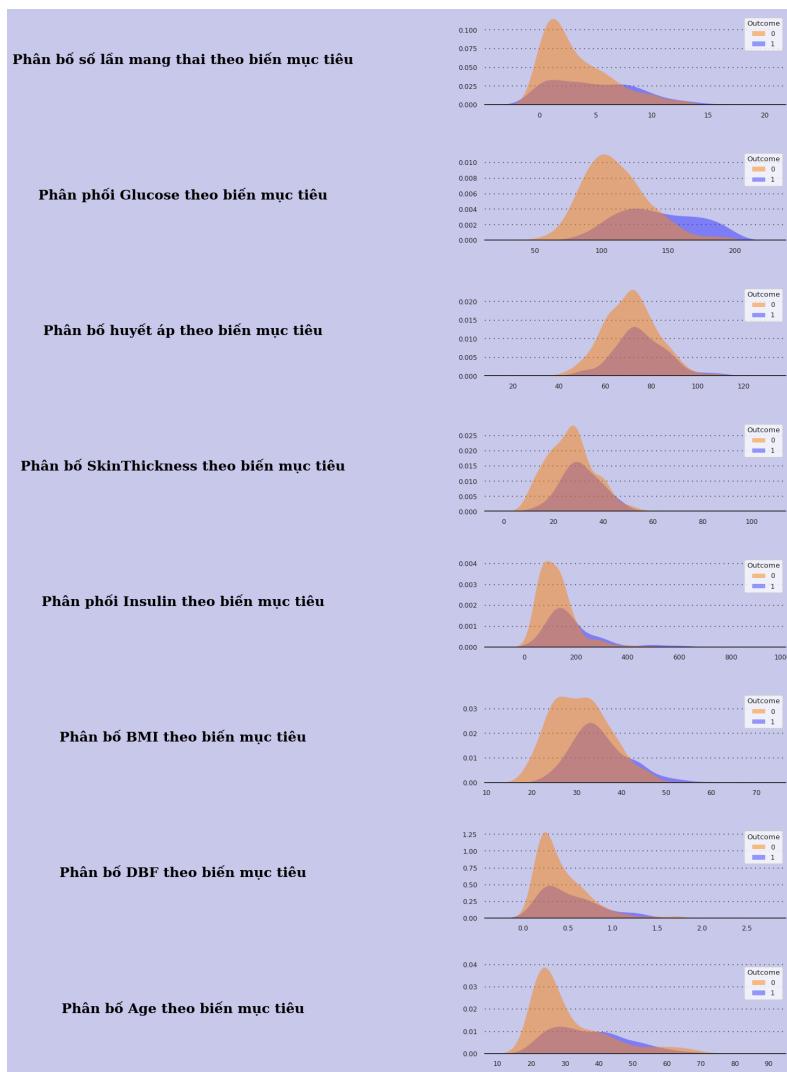


Hình 2.2. Ma trận tương quan của các đặc trưng với ‘Outcome’

Nhìn vào kết quả tương quan, chúng ta có thể thấy rằng các Đặc điểm khác nhau có mức độ tương quan khác nhau với kết quả (bệnh tiểu đường).

- Glucose: Với hệ số tương quan là 0,47, đây là đặc điểm có mối tương quan chặt chẽ nhất với kết quả. Điều này cho thấy mức glucose cao hơn có thể là một dấu hiệu quan trọng của bệnh tiểu đường.
- BMI: Đặc điểm này có mối tương quan 0,29 với kết quả. Mặc dù không mạnh bằng glucose nhưng đây vẫn là mối tương quan vừa phải, cho thấy chỉ số khối cơ thể cũng có thể là một yếu tố quan trọng trong bệnh tiểu đường.
- Age: Tuổi có mối tương quan 0,24 với kết quả. Điều này cho thấy những người lớn tuổi có nhiều khả năng mắc bệnh tiểu đường hơn.
- Pregnancies: Đặc điểm này có mối tương quan 0,22 với kết quả. Điều này có thể gợi ý rằng mang thai nhiều hơn có thể làm tăng nguy cơ mắc bệnh tiểu đường.
- DPF: Với hệ số tương quan là 0,17, đặc điểm này có mối tương quan nhẹ với kết quả. Điều này cho thấy ảnh hưởng di truyền có thể đóng một vai trò trong bệnh tiểu đường.
- Insulin: Tính năng này có mối tương quan 0,13 với kết quả. Đây là mối tương quan tương đối yếu, cho thấy rằng chỉ riêng mức insulin có thể không phải là yếu tố dự báo mạnh mẽ về bệnh tiểu đường.
- Skin Thickness: Với hệ số tương quan là 0,07, đặc điểm này có mối tương quan rất yếu với kết quả. Điều này cho thấy độ dày của da có thể không phải là yếu tố quan trọng gây ra bệnh tiểu đường.
- Blood Pressure: Tính năng này có mối tương quan yếu nhất là 0,07 với kết quả. Điều này cho thấy huyết áp có thể không phải là yếu tố quan trọng gây ra bệnh tiểu đường.

2.3 Biểu đồ phân bố tính năng với Outcome



Hình 2.3. Phân phối các tính năng theo biến mục tiêu ‘Outcome’

2.4 Box_Plot



Hình 2.4. Biểu đồ hộp

Nhận xét: Chỉ số Glucose và BMI càng cao thì càng có nguy cơ mắc bệnh tiểu đường. Các chỉ số khác không ảnh hưởng nhiều đến nguy cơ gây ra bệnh tiểu đường.

2.5 Pair plot

Hệ số tương quan Pearson: giúp bạn tìm ra mối quan hệ giữa hai đại lượng. Nó cung cấp cho bạn thước đo mức độ liên kết giữa hai biến. Giá trị của Hệ số tương quan Pearson có thể nằm trong khoảng từ -1 đến +1. 1 có nghĩa là chúng có mối tương quan cao và 0 có nghĩa là không có mối tương quan.

Bản đồ nhiệt là sự thể hiện thông tin hai chiều với sự trợ giúp của màu sắc. Bản đồ nhiệt có thể giúp người dùng hình dung thông tin đơn giản hoặc phức tạp.

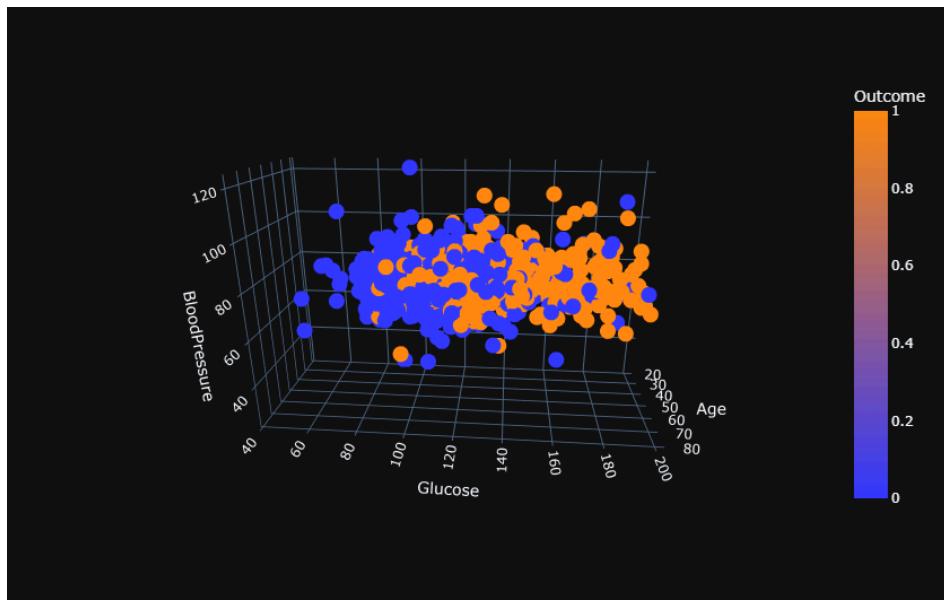


Hình 2.5. Pair-plot

Có thể thấy:

- Đặc điểm Glucose và BMI có sự phân bố gần bình thường
- Các tính năng khác bị lệch nghiêm trọng về phía bên trái của bản phân phối
- Các giá trị riêng biệt trong biểu đồ của các tính năng Glucose, BMI và Huyết áp cho biết các giá trị ngoại lệ trong dữ liệu cần được xử lý trong quy trình xử lý dữ liệu để chuẩn bị dữ liệu cho mô hình ML

2.6 Biểu đồ phân tán của Blood Pressure với Glucose với Age wrt Kết quả



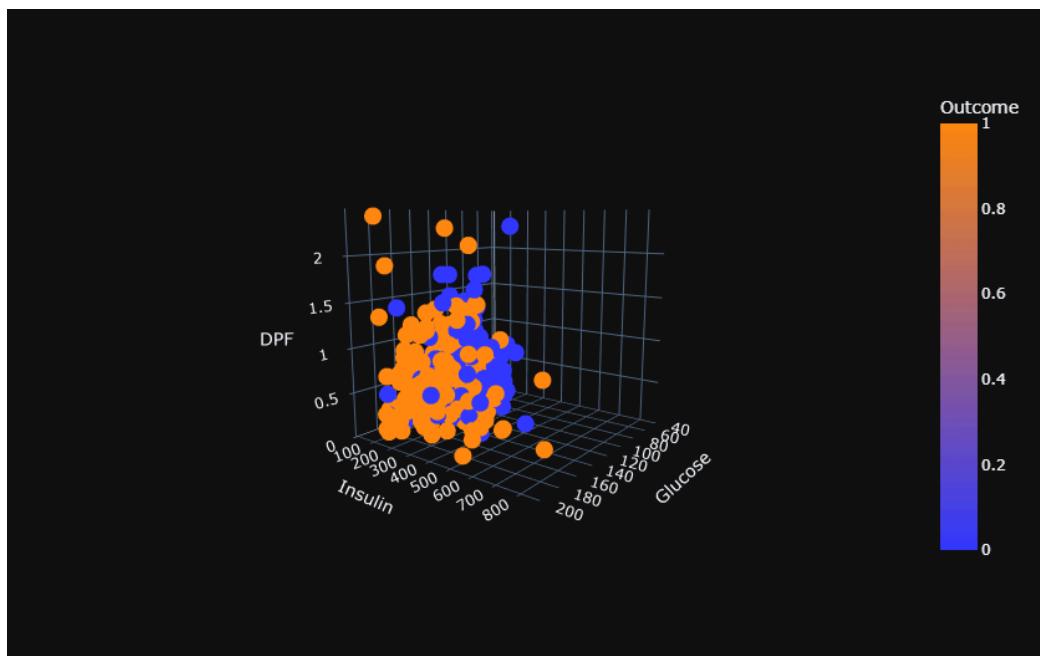
Hình 2.6. Phân tán của Glucose với Insulin với DPF wrt Kết quả

- Nhận xét:

- Blood Pressure với Glucose: Có thể thấy sự phân tán Huyết áp có sự tương đồng với lượng đường huyết
- Blood Pressure với Age: Sự phân tán Huyết áp trái ngược so với độ tuổi
- Glucose với Age: Sự phân tán lượng đường huyết trái ngược so với độ tuổi

=> Cho thấy các mối liên hệ trên không liên quan nhiều đến nguy cơ mắc bệnh tiểu đường

2.7 Sơ đồ phân tán của Glucose với Insulin với DPF wrt Kết quả



Hình 2.7. Phân tán của Glucose với Insulin với DPF wrt Kết quả

- Nhận xét: với DPF

- Glucose với Insulin: Sự phân tán Lượng đường huyết trái ngược so với Insulin
- Glucose với DPF: Sự phân tán Lượng đường huyết trái ngược so với mức độ duy truyền
- Insulin với DPF: Có thể thấy sự phân tán Insulin có sự tương đồng với mức độ duy truyền

=> Cho thấy các mối liên hệ trên có mối liên quan với nhau, nhất là Insulin với DPF. Đó cũng là 1 dấu hiệu để nhận biết có nguy cơ bệnh tiểu đường

III. XỬ LÝ DỮ LIỆU

1. Loại bỏ các ngoại lệ

```
[65] print ("---HÌNH DÁNG TRƯỚC KHI LOẠI BỎ CÁC NGOẠI LỆ---")
      print(f"Hình dạng của tập dữ liệu: {colored(df.shape, 'yellow')}")

      df.drop(df[df["Pregnancies"] > 14].index,inplace=True)
      df.drop(df[df["Glucose"] < 50].index,inplace=True)
      df.drop(df[df["BloodPressure"] > 120].index,inplace=True)
      df.drop(df[df["SkinThickness"] > 80].index,inplace=True)
      df.drop(df[df["Insulin"] > 600].index,inplace=True)
      df.drop(df[df["BMI"] > 55].index,inplace=True)
      df.drop(df[df["DPF"] > 2].index,inplace=True)
      df.drop(df[df["Age"] > 70].index,inplace=True)

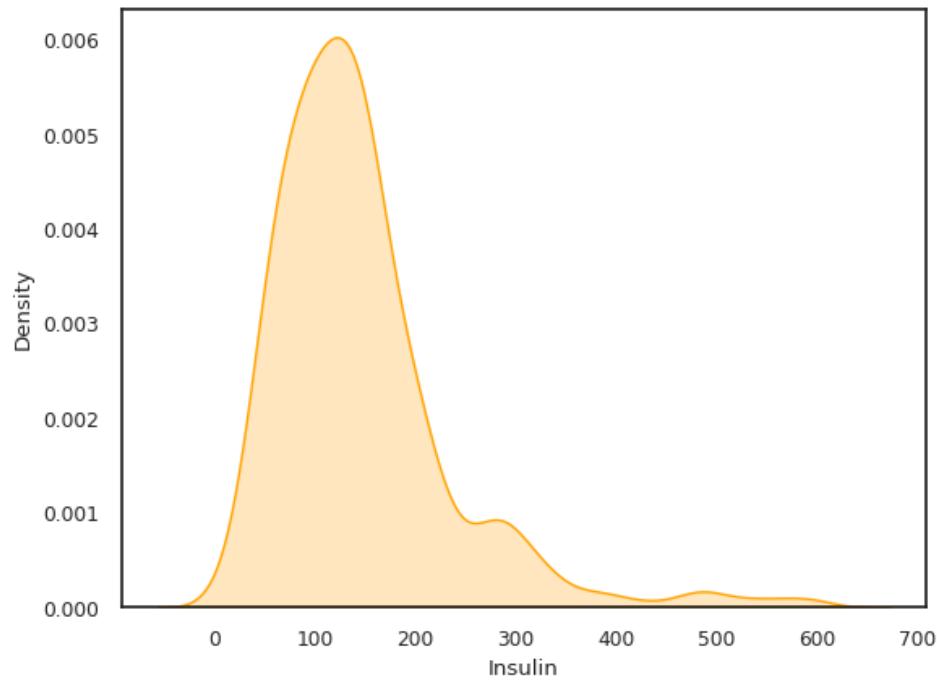
      print("")
      print ("---HÌNH DÁNG SAU KHI LOẠI BỎ CÁC NGOẠI LỆ---")
      print(f"Hình dạng của tập dữ liệu: {colored(df.shape, 'yellow')}")

---HÌNH DÁNG TRƯỚC KHI LOẠI BỎ CÁC NGOẠI LỆ---
Hình dạng của tập dữ liệu: (768, 9)

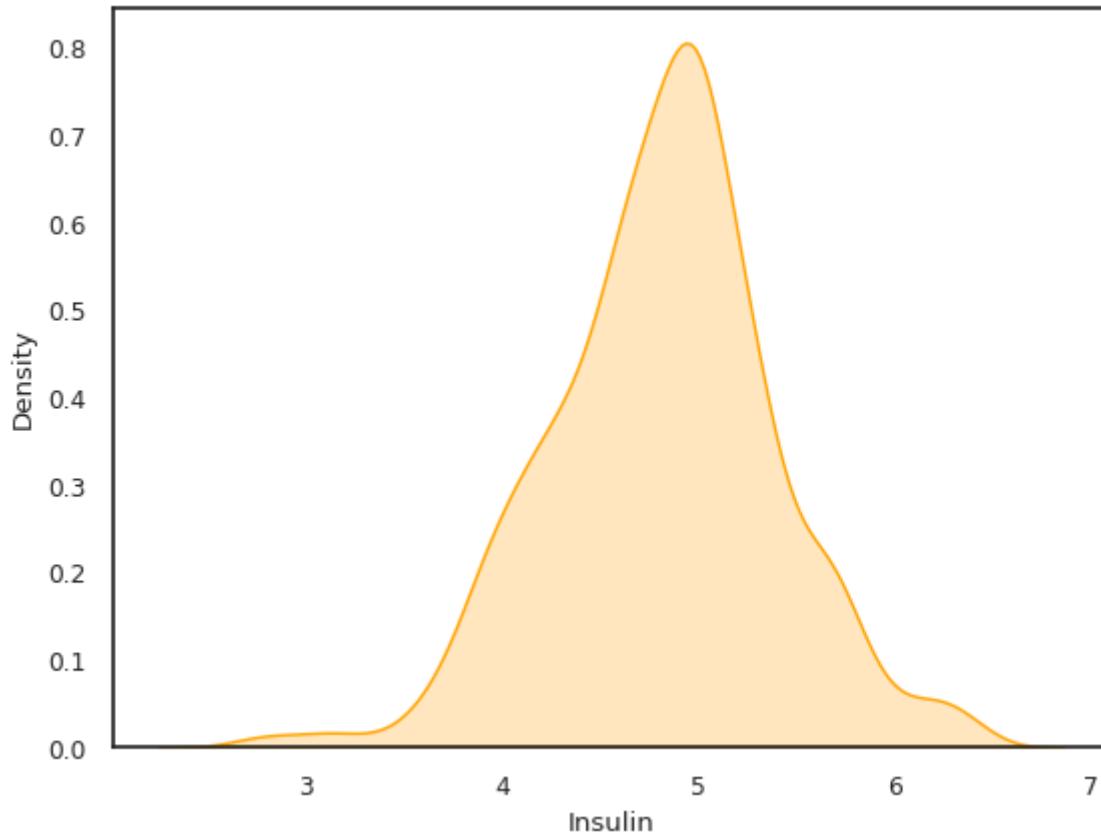
---HÌNH DÁNG SAU KHI LOẠI BỎ CÁC NGOẠI LỆ---
Hình dạng của tập dữ liệu: (752, 9)
```

2. Loại bỏ độ lệch

```
[86] print ("Kiểm tra sự phân phổi Insulin")
      sns.kdeplot(df['Insulin'],color='Orange',fill=True)
```



```
print ("Removing the skewness using a log function and checking the distribution again")
df['Insulin'] = df['Insulin'].map(lambda i : np.log(i) if i > 0 else 0)
sns.kdeplot(df['Insulin'],color='Orange',fill=True)
```



3. Mẫu tính năng đã sẵn sàng

```
[112] # nhập bộ chia tỷ lệ
from sklearn.preprocessing import StandardScaler

# tạo bản sao của khung dữ liệu
df1 = df
col_cols = list(df1.columns)

# xóa biến mục tiêu khỏi danh sách cột
col_cols.pop()

# tách biệt đặc điểm và mục tiêu
X = df1.drop(['Outcome'],axis=1)
y = df1[['Outcome']]

# khởi tạo bộ chia tỷ lệ
scaler = StandardScaler()
X[col_cols] = scaler.fit_transform(X[col_cols])
print("The first 5 rows of X are")
X.head()
```

The first 5 rows of X are

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age
0	0.652738	0.900609	-0.022757	0.715461	1.416074	0.193305	0.546800	1.467078
1	-0.853888	-1.207144	-0.528558	0.062640	-1.622397	-0.869417	-0.360223	-0.180588
2	1.255388	2.071583	-0.697158	0.020742	0.325059	-1.370415	0.694684	-0.093869
3	-0.853888	-1.073319	-0.528558	-0.590181	-0.480825	-0.641691	-0.964904	-1.047781
5	0.351413	-0.169996	0.145844	-1.025394	-0.286907	-1.021235	-0.853170	-0.267308

IV. MÔ HÌNH & SIÊU ĐIỀU CHỈNH

1. Cài đặt gói

```
[ ] # Train test split
from sklearn.model_selection import train_test_split

# Base Models
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

# Ensembling and Boosting
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier

# Metrics
from sklearn.metrics import accuracy_score, classification_report, roc_curve

# Cross Validation
from sklearn.model_selection import cross_val_score

# Hyper-parameter tuning
from functools import partial
from skopt import gp_minimize
from skopt import space
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

2. Đào tạo và phân chia dữ liệu

```
[119] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size = 0.2, random_state = 42)
print(f"Hình dạng của X_train is {colored(X_train.shape,'yellow')}")
print(f"Hình dạng của X_test is {colored(X_test.shape,'yellow')}")
print(f"Hình dạng của y_train is {colored(y_train.shape,'yellow')}")
print(f"Hình dạng của y_test is {colored(y_test.shape,'yellow')}")

Hình dạng của X_train is (601, 8)
Hình dạng của X_test is (151, 8)
Hình dạng của y_train is (601, 1)
Hình dạng của y_test is (151, 1)
```

3. Huấn luyện dữ liệu với các mô hình cơ sở mà không cần điều chỉnh siêu tham số

```
[120] models = [
    ('DecisionTreeClassifier', DecisionTreeClassifier()),
    ('RandomForestClassifier', RandomForestClassifier()),
    ('SVM_Linear', SVC(kernel='linear')),
    ('SVM_Polynomial', SVC(kernel='poly')),
    ('SVM_Gaussian', SVC(kernel='rbf')),
    ('LogisticRegression', LogisticRegression()),
    ('KNeighborsClassifier', KNeighborsClassifier()),
    ('AdaBoostClassifier', AdaBoostClassifier()),
    ('GradientBoostingClassifier', GradientBoostingClassifier())
]

print("Điểm chính xác của các mô hình là :")
for model_name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"{colored(model_name, 'blue')}")
    print(f"{colored(accuracy_score(y_test,y_pred), 'yellow')}\n")
```

➡ Điểm chính xác của các mô hình là :

DecisionTreeClassifier

0.7284768211920529

RandomForestClassifier

0.8079470198675497

SVM_Linear

0.7814569536423841

SVM_Polynomial

0.7748344370860927

SVM_Gaussian

0.7615894039735099

LogisticRegression

0.7947019867549668

KNeighborsClassifier

0.7682119205298014

AdaBoostClassifier

0.7814569536423841

GradientBoostingClassifier

0.8013245033112583

4. Điều chỉnh siêu tham số bằng GridSearchCV

```
[121] # Xác định siêu tham số cho mỗi phân loại
param_grid_decision_tree = {'max_depth': [3, 5, 7, 10]}
param_grid_random_forest = {'n_estimators': [50, 100, 150], 'max_depth': [3, 5, 7]}
param_grid_svc_linear = {'C': [0.1, 1, 10]}
param_grid_svc_poly = {'C': [0.1, 1, 10], 'degree': [2, 3, 4]}
param_grid_svc_gaussian = {'C': [0.1, 1, 10], 'gamma': [0.1, 1, 10]}
param_grid_logistic_regression = {'C': [0.1, 1, 10]}
param_grid_kneighbors = {'n_neighbors': [3, 5, 7]}
param_grid_adaboost = {'n_estimators': [50, 100, 150], 'learning_rate': [0.01, 0.1, 1]}
param_grid_gradientboost = {'n_estimators': [50, 100, 150], 'learning_rate': [0.01, 0.1, 1]}

# Tạo danh sách các bộ chứa tên model và lưới tham số tương ứng của chúng
models_params = [
    ('DecisionTreeClassifier', DecisionTreeClassifier(), param_grid_decision_tree),
    ('RandomForestClassifier', RandomForestClassifier(), param_grid_random_forest),
    ('SVM_Linear', SVC(kernel='linear'), param_grid_svc_linear),
    ('SVM_Polynomial', SVC(kernel='poly'), param_grid_svc_poly),
    ('SVM_Gaussian', SVC(kernel='rbf'), param_grid_svc_gaussian),
    ('LogisticRegression', LogisticRegression(), param_grid_logistic_regression),
    ('KNeighborsClassifier', KNeighborsClassifier(), param_grid_kneighbors),
    ('AdaBoostClassifier', AdaBoostClassifier(), param_grid_adaboost),
    ('GradientBoostingClassifier', GradientBoostingClassifier(), param_grid_gradientboost)
]

best_score = 0
best_model_name = ""

# Thực hiện GridSearchCV cho từng mô hình và in điểm
for name, model, param_grid in models_params:
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(X_train, y_train)
    score = grid_search.best_score_

    print(f"Best parameters for {name}:")
    print(grid_search.best_params_)
    print(f"Best score for {name}: {score}")

# Chọn điểm tốt nhất và mô hình tương ứng
    if score > best_score:
        best_score = score
        best_model_name = name

print(f"\nBest model: {best_model_name} with score: {best_score}")
```

```

Best parameters for DecisionTreeClassifier:
{'max_depth': 5}
Best score for DecisionTreeClassifier: 0.7154958677685951
Best parameters for RandomForestClassifier:
{'max_depth': 5, 'n_estimators': 50}
Best score for RandomForestClassifier: 0.7521487603305785
Best parameters for SVM_Linear:
{'C': 10}
Best score for SVM_Linear: 0.757107438016529
Best parameters for SVM_Polynomial:
{'C': 1, 'degree': 3}
Best score for SVM_Polynomial: 0.7321349862258953
Best parameters for SVM_Gaussian:
{'C': 1, 'gamma': 0.1}
Best score for SVM_Gaussian: 0.7454407713498623
Best parameters for LogisticRegression:
{'C': 1}
Best score for LogisticRegression: 0.7538429752066116
Best parameters for KNeighborsClassifier:
{'n_neighbors': 7}
Best score for KNeighborsClassifier: 0.7188016528925619
Best parameters for AdaBoostClassifier:
{'learning_rate': 0.1, 'n_estimators': 100}
Best score for AdaBoostClassifier: 0.7487741046831956
Best parameters for GradientBoostingClassifier:
{'learning_rate': 0.01, 'n_estimators': 150}
Best score for GradientBoostingClassifier: 0.7338292011019284

Best model: SVM_Linear with score: 0.757107438016529

```

V. PHÂN CỤM-GOM CỤM BỆNH NHÂN TIỀU ĐƯỜNG

1. Phân tích phân cụm

Chuẩn hóa dữ liệu

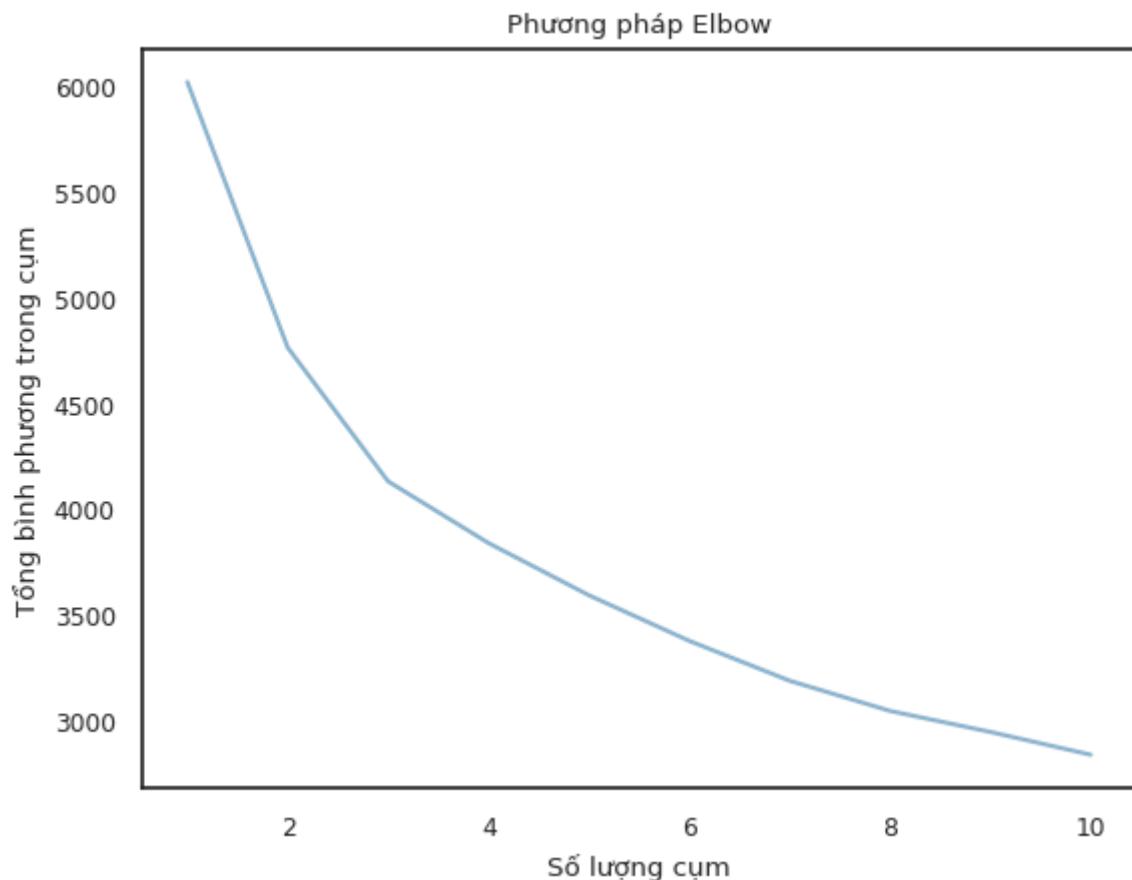
Tiền xử lý là bước quan trọng trước khi huấn luyện mô hình. Ở đây, đặc điểm số được chuẩn hóa và đặc điểm phân loại được mã hóa. Chuẩn hóa không là bắt buộc, nhưng thường là thực hành tốt. StandardScaler trong sklearn giúp biến đổi dữ liệu sao cho giá trị trung bình là 0 và độ lệch chuẩn là 1.

```

[98] # Chuẩn hóa dữ liệu
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df.drop('Outcome', axis=1))

[99] # Xác định số cụm tối ưu bằng phương pháp Elbow
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Phương pháp Elbow')
plt.xlabel('Số lượng cụm')
plt.ylabel('Tổng bình phương trong cụm')
plt.show()

```



Hình 1.1. Biểu đồ Elbow

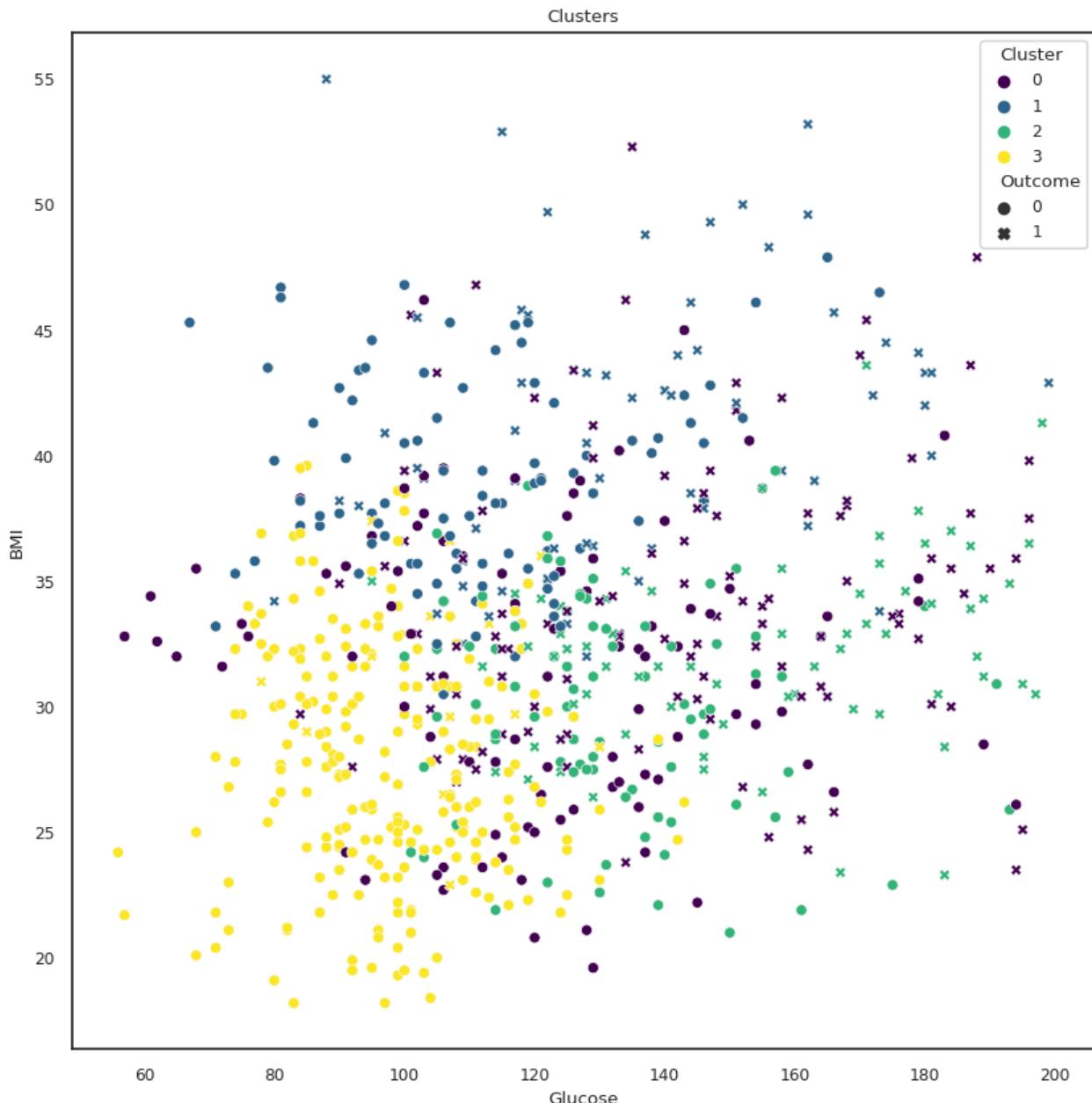
Biểu đồ Elbow là công cụ quan trọng để xác định số lượng cụm tối ưu trong phân cụm K-means. Trên biểu đồ, trục x thể hiện số lượng cụm, trong khi trục y biểu diễn Tổng bình phương trong cụm (WCSS) - một đánh giá về độ nhôm của dữ liệu.

'Elbow' trên biểu đồ là điểm mà thêm cụm không cải thiện đáng kể WCSS. Trong trường hợp này, có vẻ như có khoảng 4 cụm là số lượng tối ưu. Nhìn chung, số cụm tối ưu cho dữ liệu là 4, được xác định dựa trên phương pháp Elbow.

```
[101] # Phân cụm K-means phù hợp (** thay thế 'n_clusters' bằng số cụm mong muốn**)
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
clusters = kmeans.fit_predict(data_scaled)

# Thêm nhãn cụm vào dữ liệu gốc
df['Cluster'] = clusters

# Trực quan hóa các cụm bằng cách sử dụng 'Glucose' và 'BMI'
plt.figure(figsize=(10, 10))
sns.scatterplot(data=df, x='Glucose', y='BMI', hue='Cluster', style='Outcome', palette='viridis')
plt.title('Clusters')
plt.show()
```



Hình 1.2. Biểu đồ phân tán trực quan hóa mối quan hệ giữa Glucose và BMI

Biểu đồ phân tán trực quan hóa mối quan hệ giữa Glucose và BMI, hai tính năng quan trọng trong tập dữ liệu của chúng em. Mỗi điểm trên biểu đồ đại diện cho một cá nhân trong tập dữ liệu và màu của điểm biểu thị cụm mà cá nhân đó thuộc về dựa trên phân tích phân cụm K-means:

- Glucose: Tính năng này được thể hiện trên trục x. Có vẻ như các cá nhân có mức độ glucose khác nhau.
- BMI: Tính năng này được thể hiện trên trục y. Tương tự như glucose, các cá nhân có nhiều giá trị BMI khác nhau.

Clusters: Các màu khác nhau trên biểu đồ đại diện cho các cụm khác nhau. Có vẻ như thuật toán phân cụm K-mean đã nhóm các cá nhân thành các cụm riêng biệt dựa trên Glucose và chỉ số BMI của họ.

2. Gom cùm

```
[125] zero_features = ['Pregnancies','Glucose','BloodPressure','SkinThickness","Insulin",'BMI']
      total_count = df['Glucose'].count()

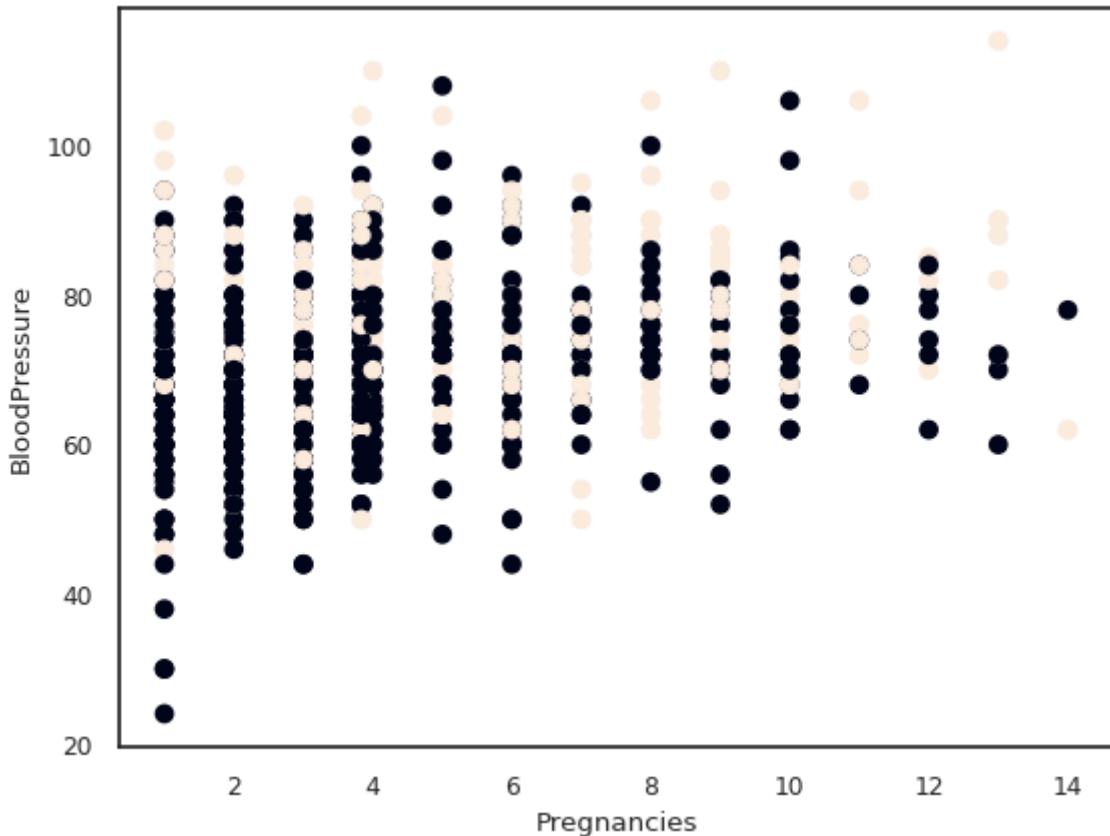
      for feature in zero_features:
          zero_count = df[df[feature]==0][feature].count()
          print('{0} 0 number of cases {1}, percent is {2:.2f} %'.format(feature, zero_count, 100*zero_count/total_count))
# Tiền xử lý dữ liệu
diabetes_mean = df[zero_features].mean()
df[zero_features]=df[zero_features].replace(0, diabetes_mean)

Pregnancies 0 number of cases 107, percent is 14.23 %
Glucose 0 number of cases 0, percent is 0.00 %
BloodPressure 0 number of cases 0, percent is 0.00 %
SkinThickness 0 number of cases 0, percent is 0.00 %
Insulin 0 number of cases 0, percent is 0.00 %
BMI 0 number of cases 0, percent is 0.00 %
```

```
[129] import matplotlib.pyplot as plt
kmeans2 = KMeans(n_clusters=2, random_state=0).fit(df)
y_pred_2 = kmeans2.predict(df) #k=3

plt.xlabel('Pregnancies')
plt.ylabel('BloodPressure')

plt.scatter(df['Pregnancies'], df['BloodPressure'], c=y_pred_2)
plt.show()
```



Hình 2.1. Biểu đồ gom cụm giữa 'Blood Pressure' và 'Pregnancies'

- **Nhận xét:** Có thể thấy huyết áp của phụ nữ người da đỏ sau khi mang thai đa số ở mức bình thường.

VI. THUẬT TOÁN PHÂN LOẠI

1. KNN

Dự đoán được thực hiện dựa trên sự giống nhau của các quan sát với nhau. (Hãy cho tôi biết bạn bè của bạn, tôi sẽ cho bạn biết bạn là ai.)

K-Neighbors đơn vị quan sát gần nhất cho một đơn vị quan sát nhất định được tính toán. Biến phụ thuộc được dự đoán dựa trên k đơn vị quan sát gần nhất. Việc tính toán khoảng cách được thực hiện cho mỗi quan sát bằng cách sử dụng Euclidean hoặc thước đo khoảng cách tương tự. Giá trị trung bình của biến phụ thuộc của k đơn vị quan sát gần nhất được lấy và gán cho quan sát chưa biết của chúng ta.

Nếu chúng ta định sử dụng KNN để phân loại, thay vì giá trị trung bình của k đơn vị quan sát gần nhất, chúng ta sẽ chọn lớp lặp lại thường xuyên nhất và gán nó cho biến đó.

```
[174] ## Tiền xử lý dữ liệu & Kỹ thuật tính năng

y = df["Outcome"]
X = df.drop(["Outcome"], axis=1)
# Chúng tôi áp dụng StandardScaler, trả về một mảng có nhiều mảng
X_scaled = StandardScaler().fit_transform(X)
# Chuyển đổi nó trở lại DataFrame
X = pd.DataFrame(X_scaled, columns=X.columns)

[175] #Mô hình hóa và dự đoán

# Chúng tôi xây dựng mô hình của mình và điều chỉnh X, y cho nó
knn_model = KNeighborsClassifier().fit(X, y)
# Chúng tôi lấy mẫu ngẫu nhiên
random_user = X.sample(1, random_state=45)
# Chúng tôi đưa ra dự đoán
knn_model.predict(random_user)

array([0])

[176] #Đánh giá mô hình

# Đối với ma trận nhầm lẫn:
y_pred = knn_model.predict(X)
# Đối với AUC:
y_prob = knn_model.predict_proba(X)[:, 1]

print(classification_report(y, y_pred))
# acc 0.83
# f1 0.74
# AUC 0.90
roc_auc_score(y, y_prob)
```

	precision	recall	f1-score	support
0	0.85	0.88	0.87	493
1	0.76	0.71	0.74	259
accuracy			0.83	752
macro avg	0.81	0.80	0.80	752
weighted avg	0.82	0.83	0.82	752
0.9004283913006023				

```
[133] # Chúng tôi thực hiện xác thực chéo 5 lần
cv_results = cross_validate(knn_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])

print(cv_results['test_accuracy'].mean())
print(cv_results['test_f1'].mean())
print(cv_results['test_roc_auc'].mean())

# 0.73 Độ chính xác giảm
# 0.59 Điểm F1 giảm
# 0,78 điểm AUC giảm

# Các bước có thể cải thiện:
# 1. Tăng cỡ mẫu.
# 2. Tiền xử lý dữ liệu
# 3. Kỹ thuật tính năng
# 4. Tối ưu hóa thuật toán liên quan.

0.7287417218543046
0.5903149152757672
0.7702414376784124
```

```
[134] # Tối ưu hóa siêu tham số

# Kiểm tra thông số của nó
knn_model.get_params()

{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'uniform'}
```



```
[135] # Chúng ta cố gắng tìm số lượng hàng xóm tối ưu bằng cách thay đổi số lượng hàng xóm
knn_params = {"n_neighbors": range(2, 50)}

# Chúng tôi thực hiện tìm kiếm dạng lưới
knn_gs_best = GridSearchCV(knn_model,
                           knn_params,
                           cv=5,
                           n_jobs=-1,
                           verbose=1).fit(X, y)

# Có 48 ứng cử viên được xét xử và vì gấp 5 lần nên có tổng cộng 240 ca phẫu thuật phù hợp.
# Nó chỉ ra rằng số hàng xóm tốt nhất là 17.
knn_gs_best.best_params_

Fitting 5 folds for each of 48 candidates, totalling 240 fits
{'n_neighbors': 44}
```

```
[136] # Mẫu cuối cùng

# Chúng tôi cập nhật mô hình của mình với các giá trị tốt nhất mà chúng tôi đã trích xuất
# Khi nó ở dạng từ điển dưới dạng khóa-giá trị, chúng tôi sẽ đánh giá nó 2 sao
knn_final = knn_model.set_params(**knn_gs_best.best_params_).fit(X, y)

# Chúng tôi sử dụng xác thực chéo để xem xét lỗi
cv_results = cross_validate(knn_final,
                            X,
                            y,
                            cv=5,
                            scoring=["accuracy", "f1", "roc_auc"])

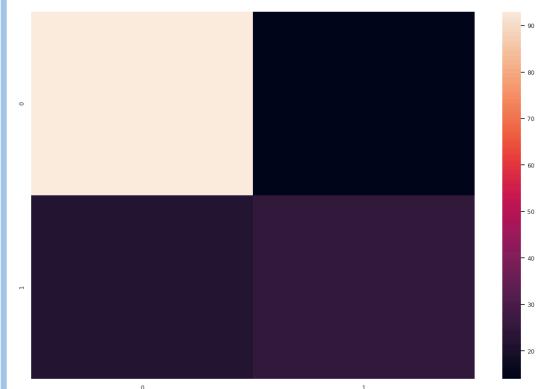
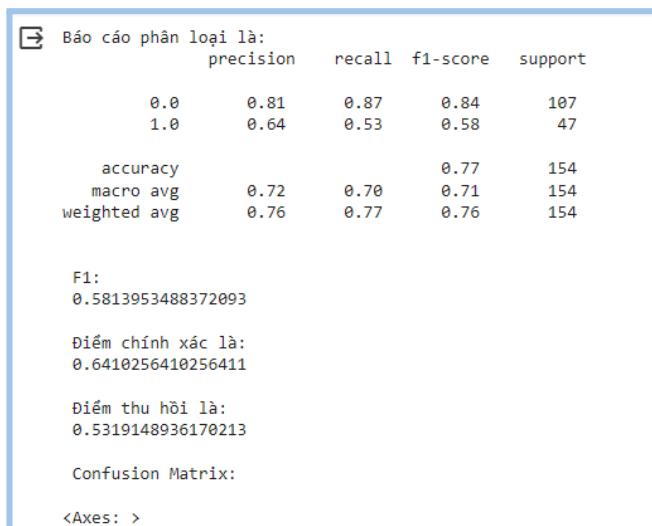
print(cv_results['test_accuracy'].mean())
# Tăng từ 73 lên 76
print(cv_results['test_f1'].mean())
# Tăng từ 59 lên 61
print(cv_results['test_roc_auc'].mean())
# Tăng từ 78 lên 81
```

⇒ 0.7713289183222958
0.6214539927636623
0.8313650401885695



2. Naive Bayes

Naive Bayes là phương pháp phân loại áp dụng nguyên tắc độc lập có điều kiện của lớp với Định lý Bayes. Điều này có nghĩa là sự hiện diện của một đặc điểm không ảnh hưởng đến sự hiện diện của đặc điểm khác trong xác suất của một kết quả nhất định và mỗi yếu tố dự đoán đều có tác động như nhau đối với kết quả đó.



3. Cây quyết định

Cây quyết định là một phương pháp được Leo Breiman đề xuất vào năm 1982. Nó tạo cơ sở cho rừng ngẫu nhiên và nhiều phương pháp thông dụng khác. Mục đích là chuyển đổi các cấu trúc phức tạp trong tập dữ liệu thành các cấu trúc quyết định đơn giản. Các bộ dữ liệu không đồng nhất được chia thành các nhóm con đồng nhất dựa trên một biến mục tiêu được chỉ định.

Nó hoạt động theo quy tắc; ví dụ: nếu số năm kinh nghiệm của một người lớn hơn 4 thì mức lương trên 520 và nếu nhỏ hơn 4 thì mức lương dưới 520. Những mức này có thể có các phân chia khác nhau, ví dụ: nếu số năm kinh nghiệm lớn hơn 4 và mức lương trên 520, nhưng có thể có sự phân chia sâu hơn dựa trên trình độ ngoại ngữ. Ví dụ: lương trên 520 và biết ngoại ngữ trên 3 thì lương trên 800; nếu số lượng ngôn ngữ biết ít hơn 3 thì mức lương dưới 600.

Các điểm phân chia các biến độc lập được gọi là điểm nút bên trong.

Ví dụ: trong trường hợp được đề cập, có hai nút nội bộ - một nút dành cho việc phân chia dựa trên số năm kinh nghiệm và nút còn lại dựa trên số lượng ngôn ngữ đã biết. Có bốn nút cuối (lá) - mức lương lần lượt là 520, 600 và 800.

Các nút bên trong đại diện cho các điểm quyết định, trong khi các nút đầu cuối đại diện cho các giá trị điểm cuối. Trong ví dụ trên, có hai nút bên trong và bốn nút đầu cuối.

a. Lập mô hình bằng CART

```
[206] y = df["Outcome"]
X = df.drop(["Outcome"], axis=1)
#Xây dựng mô hình của chúng tôi
cart_model = DecisionTreeClassifier(random_state=1).fit(X, y)

# Đổi với ma trận nhầm lẫn, tạo y_pred
y_pred = cart_model.predict(X)

[139] # Để có điểm Auc, làm y_prob
y_prob = cart_model.predict_proba(X)[:, 1]

# Ma trận hỗn loạn
print(classification_report(y, y_pred))

precision    recall   f1-score   support
          0       1.00      1.00      1.00      493
          1       1.00      1.00      1.00      259

accuracy                           1.00      752
macro avg       1.00      1.00      1.00      752
weighted avg    1.00      1.00      1.00      752
```

Nhân xét:

Chỉ số AUC (Area Under the Curve) là một phép đo đánh giá hiệu suất của mô hình phân loại (classification model). Nó đo lường khả năng của mô hình phân loại để phân biệt giữa các mẫu thuộc các nhóm khác nhau.

Để hiểu chỉ số AUC, trước tiên ta cần biết về đường cong ROC (Receiver Operating Characteristic Curve). Đường cong ROC biểu thị tỷ lệ giữa tỷ lệ true positive (TPR) và tỷ lệ false positive (FPR) của mô hình phân loại với các ngưỡng quyết định khác nhau. TPR là tỷ lệ các mẫu positive được phân loại đúng, còn FPR là tỷ lệ các mẫu negative bị phân loại nhầm thành positive.

Chỉ số AUC tính diện tích nằm dưới đường cong ROC. Giá trị AUC nằm trong khoảng từ 0 đến 1, và một mô hình phân loại tốt có giá trị AUC gần với 1, trong khi một mô hình ngẫu nhiên có giá trị AUC gần với 0.

```
[140] # AUC
roc_auc_score(y, y_prob)

[141] # Chia tập dữ liệu thành 70% huấn luyện và 30% kiểm tra.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=45)

# Tạo mô hình cây quyết định và khớp nó với dữ liệu huấn luyện.
cart_model = DecisionTreeClassifier(random_state=17).fit(X_train, y_train)

[142] # Lỗi đào tạo
y_pred_train = cart_model.predict(X_train)
y_prob_train = cart_model.predict_proba(X_train)[:, 1]
print(classification_report(y_train, y_pred_train))
roc_auc_train = roc_auc_score(y_train, y_prob_train)
print(f"Train AUC Score: {roc_auc_train}")

precision    recall   f1-score   support
          0       1.00      1.00      1.00      351
          1       1.00      1.00      1.00      175

accuracy                           1.00      526
macro avg       1.00      1.00      1.00      526
weighted avg    1.00      1.00      1.00      526

Train AUC Score: 1.0
```

```
[143] # Lỗi kiểm tra
y_pred_test = cart_model.predict(X_test)
y_prob_test = cart_model.predict_proba(X_test)[:, 1]
print(classification_report(y_test, y_pred_test))
roc_auc_test = roc_auc_score(y_test, y_prob_test)
print(f"Test AUC Score: {roc_auc_test}")

precision    recall   f1-score   support
          0       0.75      0.79      0.77      142
          1       0.61      0.56      0.58       84

accuracy                           0.70      226
macro avg       0.68      0.67      0.68      226
weighted avg    0.70      0.70      0.70      226

Test AUC Score: 0.6741281019450035
```

Nhận xét:

Chúng em quan sát thấy rằng không có lỗi trong dữ liệu huấn luyện. Tuy nhiên, khi chúng em kiểm tra mô hình với dữ liệu chưa được xem (bộ kiểm tra)

Chúng em thấy rằng hiệu suất gần như giảm đi một nửa. Điều này chỉ ra rằng mô hình, mặc dù thể hiện hiệu suất cao trên dữ liệu được huấn luyện nhưng lại không khái quát hóa tốt đối với dữ liệu không nhìn thấy, cho thấy quá khớp.

b. Đánh giá hiệu suất bằng xác thực chéo

Lợi ích của xác thực chéo là nó giúp đánh giá mô hình một cách khách quan và tránh tình trạng overfitting (quá khớp) hoặc underfitting (quá khớp chưa đủ) trên dữ liệu. Nó cũng cho phép chúng ta kiểm tra hiệu suất của mô hình trên các tập dữ liệu khác nhau và đảm bảo tính ổn định của mô hình.

Xác thực chéo là một kỹ thuật quan trọng để đánh giá hiệu suất của mô hình học máy trên dữ liệu. Nó giúp chia dữ liệu thành các tập huấn luyện và kiểm tra, và đánh giá khả năng dự đoán của mô hình trên các tập kiểm tra không được sử dụng trong quá trình huấn luyện.

```
[144] # Tạo và huấn luyện mô hình cây quyết định.  
cart_model = DecisionTreeClassifier(random_state=17).fit(X, y)  
  
# Đánh giá hiệu suất bằng cách sử dụng xác thực chéo.  
cv_results = cross_validate(cart_model,  
                           X, y,  
                           cv=5,  
                           scoring=["accuracy", "f1", "roc_auc"])  
  
[145] # Tính độ chính xác trung bình, điểm F1 và điểm ROC AUC trên các nếp gấp.  
cv_accuracy = cv_results['test_accuracy'].mean()  
cv_f1 = cv_results['test_f1'].mean()  
cv_roc_auc = cv_results['test_roc_auc'].mean()  
  
print(f"Mean Cross-Validated Accuracy: {cv_accuracy}")  
# Độ chính xác được xác thực chéo trung bình: 0,7058568882098294  
  
print(f"Mean Cross-Validated F1 Score: {cv_f1}")  
# Điểm F1 được xác thực chéo trung bình: 0,5710621194523633  
  
print(f"Mean Cross-Validated ROC AUC Score: {cv_roc_auc}")  
# Điểm AUC ROC được xác thực chéo trung bình: 0,6719440950384347  
  
Mean Cross-Validated Accuracy: 0.6968035320088299  
Mean Cross-Validated F1 Score: 0.5611412971716384  
Mean Cross-Validated ROC AUC Score: 0.6644274026626967
```

Nhân xét:

Khi kiểm tra mô hình trên toàn bộ tập dữ liệu, nó phù hợp. Sau đó, chia dữ liệu thành train và test, lỗi kiểm tra đã giảm đáng kể.

Thay đổi tính ngẫu nhiên của train và test, mang lại kết quả tốt hơn.

c. Tối ưu hóa siêu tham số với GridSearchCV

GridSearchCV là một phương pháp tìm kiếm siêu tham số để tối ưu hóa hiệu suất của mô hình học máy. Nó tìm kiếm qua một lối rẽ các giá trị siêu tham số và đánh giá mô hình trên mỗi tổ hợp, trả về giá trị tối ưu của siêu tham số dựa trên độ đo hiệu suất đã chọn trước đó.

Ví dụ, giả sử chúng ta muốn tối ưu các siêu tham số của một mô hình máy học thuộc loại Random Forest. Chúng ta có thể sử dụng GridSearchCV để xác định giá trị tối ưu cho số lượng cây (n_estimators) và độ sâu tối đa của cây (max_depth). GridSearchCV sẽ xây dựng và đánh giá mô hình với mọi tổ hợp của các giá trị số lượng cây và độ sâu cây đã xác định trước, sau đó trả về kết quả tốt nhất dựa trên độ đo hiệu suất đã chọn.

```
[146] # Tham số quan trọng đối với chúng ta: min_sample_split, ví dụ, giá trị mặc định của nó là 2, nghĩa là nó sẽ tách ra khi có #có thể dẫn đến việc trang bị quá mức. Một tham số đáng quan tâm khác là max_depth, có thể giúp ngăn chặn việc trang bị quá m

# Hiển thị các tham số hiện tại của mô hình cây quyết định.
cart_model.get_params()

{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 17,
 'splitter': 'best'}
```



```
[148] # Xác định lưới tham số cho tìm kiếm lưới. Chúng tôi xem xét max_depth từ 1 đến 11 và min_samples_split từ 2 đến 20.
y cart_params = {'max_depth': range(1, 11),
 "min_samples_split": range(2, 20)}

# Nếu chúng tôi muốn xem điểm ROC AUC, điểm AUC có thể được cung cấp dưới dạng tham số.
cart_best_grid = GridSearchCV(cart_model,
                                cart_params,
                                cv=5,
                                n_jobs=-1,
                                verbose=1).fit(X, y)

Fitting 5 folds for each of 180 candidates, totalling 900 fits
```



```
[149] # Hiển thị các tham số tốt nhất được tìm thấy bởi GridSearchCV.
y cart_best_grid.best_params_
# {'max_depth': 5, 'min_samples_split': 4}

{'max_depth': 2, 'min_samples_split': 2}
```



```
[150] # Hiển thị điểm xác thực chéo tốt nhất đạt được với các thông số tốt nhất.
y cart_best_grid.best_score_
# 0.75

0.738066225165563
```



```
[151] # Kiểm tra mô hình với mẫu ngẫu nhiên.
y random_sample = X.sample(1, random_state=45)
cart_best_grid.predict(random_sample)

array([0])
```

d. Mô hình cuối cùng

Đặc điểm quan trọng của một mô hình cuối cùng là nó đã được tinh chỉnh hoặc chọn lựa để tối ưu hóa hiệu suất và đáp ứng yêu cầu của bài toán phân tích dữ liệu cụ thể.

Quá trình xây dựng cây quyết định bao gồm việc chọn các đặc trưng quan trọng để phân chia dữ liệu thành các nhánh dựa trên các quy tắc quyết định. Mục tiêu là tạo ra một cây quyết định hiệu quả, có khả năng dự đoán giá trị đầu ra tốt cho dữ liệu mới.

Các cây quyết định có thể được tinh chỉnh và cải thiện bằng cách sử dụng các kỹ thuật như cắt tỉa (pruning) và tăng cường (boosting). Kỹ thuật cắt tỉa giảm kích thước của cây bằng cách loại bỏ các nhánh không cần thiết hoặc không cung cấp giá trị dự đoán thêm. Kỹ thuật tăng cường kết hợp nhiều cây quyết định yếu thành một mô hình mạnh hơn.

Mô hình cuối cùng cây quyết định có thể được sử dụng để dự đoán giá trị đầu ra cho dữ liệu mới bằng cách đi qua các quy tắc quyết định trên cây từ gốc đến lá. Điều này cho phép dự đoán giá trị liên tục dựa trên các quyết định được đưa ra trên các biến đầu vào.

```
[153] # Tạo mô hình cuối cùng với các thông số tốt nhất.
y
cart_final = DecisionTreeClassifier(**cart_best_grid.best_params_, random_state=17).fit(X, y)

# Hiển thị các tham số của mô hình cuối cùng.
cart_final.get_params()

{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 2,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 17,
 'splitter': 'best'}
```

```
[154] # Ngoài ra, sử dụng phương thức set_params trên mô hình đã tạo trước đó để đặt tham số tốt nhất.
cart_final = cart_model.set_params(**cart_best_grid.best_params_).fit(X, y)

# Đánh giá mô hình cuối cùng bằng xác thực chéo.
cv_results_final = cross_validate(cart_final,
                                    X, y,
                                    cv=5,
                                    scoring=["accuracy", "f1", "roc_auc"])

# Hiển thị độ chính xác trung bình, điểm F1 và điểm ROC AUC trên các nếp gấp cho mô hình cuối cùng.
print(cv_results_final['test_accuracy'].mean())
# 0.75
print(cv_results_final['test_f1'].mean())
# 0.61
print(cv_results_final['test_roc_auc'].mean())
# 0.79

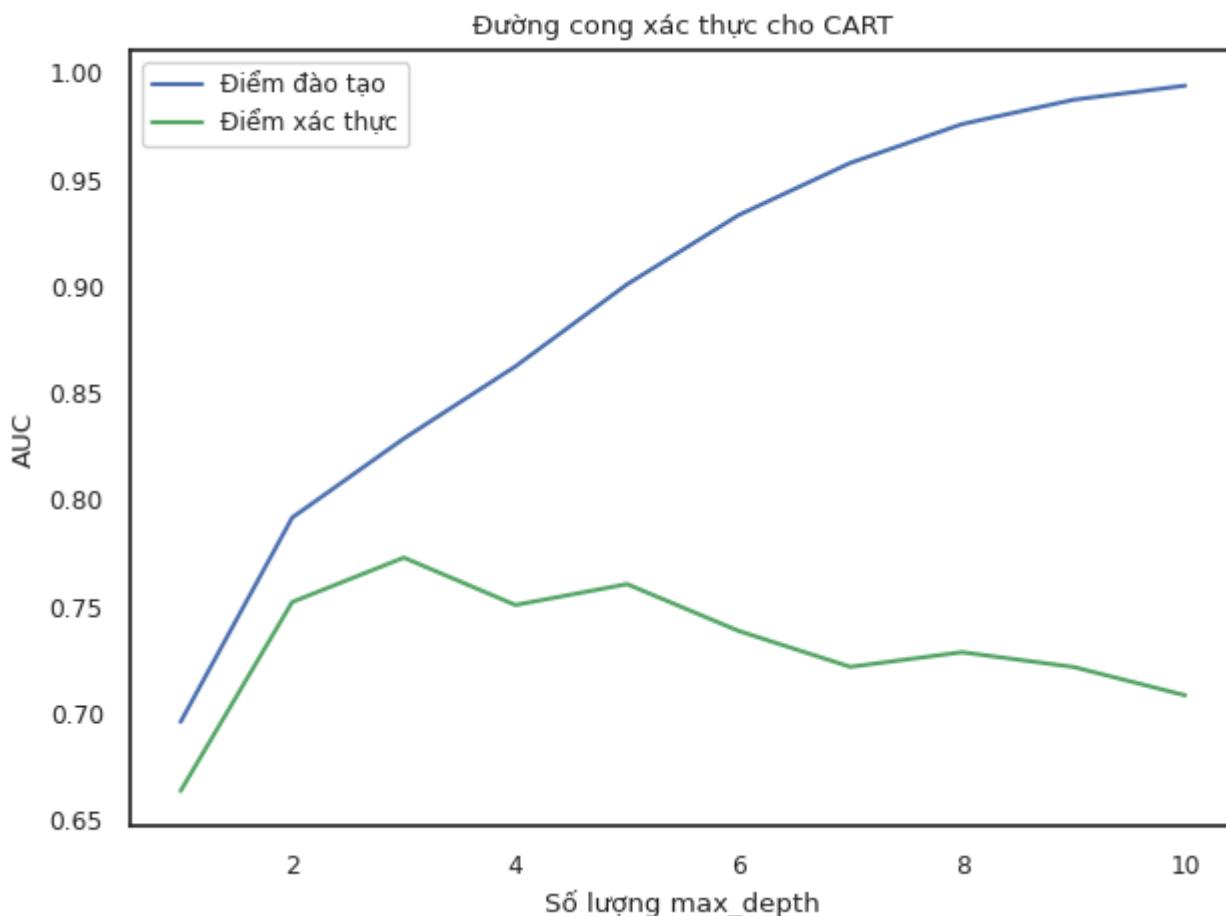
# Đã hoàn thành thành công việc tối ưu hóa siêu tham số cho mô hình cuối cùng.
```

```
0.738066225165563
0.5238250485147036
0.7647368387914606
```

e. Phân tích độ phức tạp của mô hình với các đường cong học tập

Đường cong học tập biểu thị mối quan hệ giữa kích thước tập huấn luyện và hiệu suất của mô hình trên tập huấn luyện và tập kiểm tra. Đường cong học tập cung cấp thông tin về sự học tập của mô hình khi kích thước dữ liệu tăng lên. Nếu đường cong học tập của mô hình hội tụ và đạt độ phức tạp thấp trên cả tập huấn luyện và tập kiểm tra, có thể cho thấy mô hình không bị quá khớp (overfitting) hoặc bị thiếu khớp (underfitting).

```
[160] # Sử dụng đường cong xác nhận để phân tích độ phức tạp của mô hình.  
#Trong trường hợp này, chúng tôi thay đổi tham số độ sâu tối đa từ 1 đến 10 và quan sát xem nó ảnh hưởng như thế nào  
# điểm ROC AUC cho cả tập huấn luyện và tập kiểm tra.  
  
train_score, test_score = validation_curve(cart_final, X, y,  
                                         param_name="max_depth",  
                                         param_range=range(1, 11),  
                                         scoring="roc_auc",  
                                         cv=10)  
  
mean_train_score = np.mean(train_score, axis=1)  
mean_test_score = np.mean(test_score, axis=1)  
  
# Vẽ đường cong học tập.  
plt.plot(range(1, 11), mean_train_score,  
         label="Điểm đào tạo", color='b')  
plt.plot(range(1, 11), mean_test_score,  
         label="Điểm xác thực", color='g')  
  
plt.title("Đường cong xác thực cho CART")  
plt.xlabel("Số lượng max_depth")  
plt.ylabel("AUC")  
plt.tight_layout()  
plt.legend(loc='best')  
plt.show()
```



Nhận xét:

Nếu hiệu suất trên tập huấn luyện tăng dần và hội tụ tới một giá trị cố định khi kích thước dữ liệu tăng lên, điều này cho thấy mô hình đang học tốt từ dữ liệu và có khả năng khớp tốt với tập huấn luyện.

Nếu hiệu suất trên tập kiểm tra tăng dần khi kích thước dữ liệu tăng lên và hội tụ tới một giá trị cố định, điều này cho thấy mô hình đang hoạt động tốt trên dữ liệu mới và không bị quá khóp.

Nếu khoảng cách giữa hai đường cong nhỏ và hai đường hội tụ về cùng một giá trị, điều này cho thấy mô hình không bị quá khóp hoặc thiếu khóp. Nếu khoảng cách lớn và đường cong kiểm tra không hội tụ, có thể cho thấy mô hình đang bị quá khóp và cần điều chỉnh độ phức tạp.

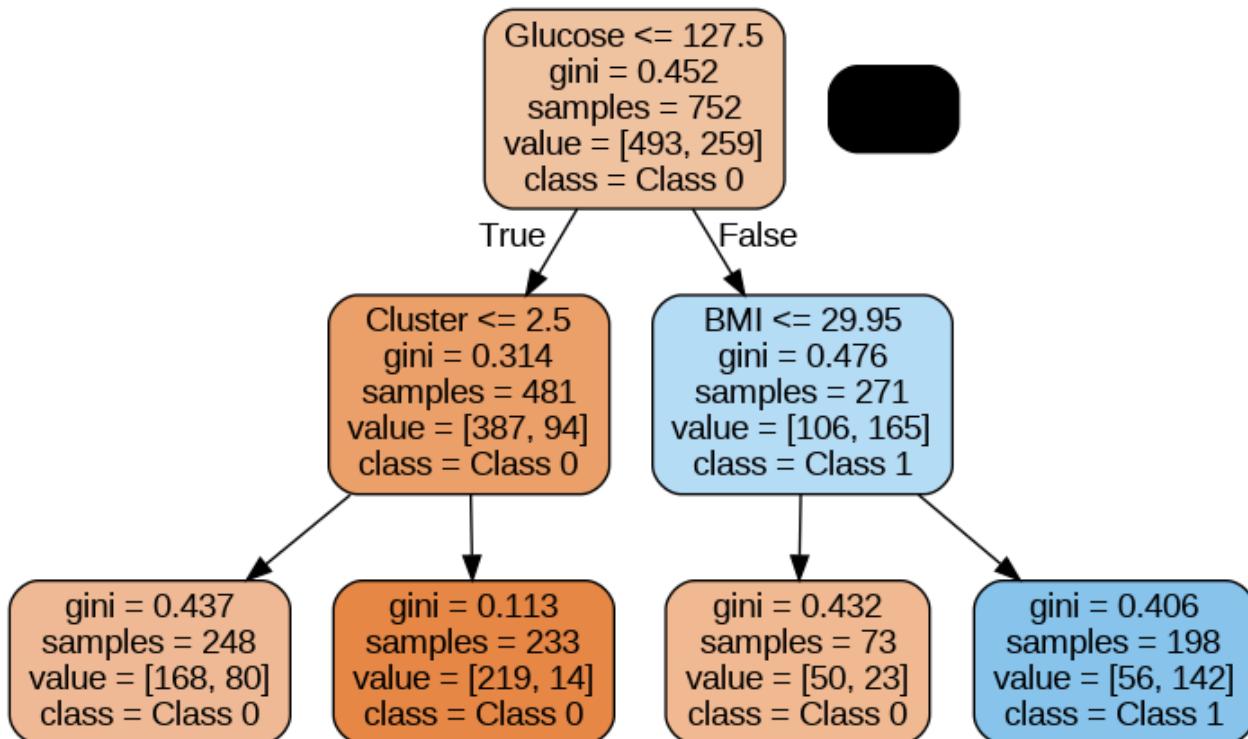
Nếu mô hình có hiệu suất tốt trên tập huấn luyện nhưng hiệu suất trên tập kiểm tra không tăng lên khi kích thước tập huấn luyện tăng, có thể cho thấy mô hình đang bị thiếu khóp và cần một mô hình phức tạp hơn hoặc bộ dữ liệu lớn hơn.

f. Trực quan hóa cây quyết định

```
[131] from sklearn.tree import export_graphviz
      import pydotplus
      from IPython.display import Image

      def visualize_decision_tree(model, feature_names, class_names, file_name):
          dot_data = export_graphviz(model, out_file=None, feature_names=feature_names, class_names=class_names, filled=True, rounded=True)
          graph = pydotplus.graph_from_dot_data(dot_data)
          graph.write_png(file_name)
          return Image(graph.create_png())

      visualize_decision_tree(cart_final, X.columns, ["Class 0", "Class 1"], "decision_tree.png")
```



Hình 3.1. Cây quyết định

Nhận xét:

Quyết định dựa trên Glucose:

- Nếu mức đường huyết (Glucose) của một người thử nghiệm là nhỏ hơn hoặc bằng 127.50, cây quyết định tiếp tục kiểm tra một đặc trưng khác.
- Nếu mức đường huyết ≤ 127.50 và Cluster ≤ 2.50 , mô hình dự đoán là class 0.
- Nếu mức đường huyết ≤ 127.50 và Cluster > 2.50 , mô hình cũng dự đoán là class 0.

Quyết định dựa trên Glucose (phần khác):

- Nếu mức đường huyết > 127.50 , cây quyết định kiểm tra một đặc trưng khác, lần lượt là BMI.
- Nếu BMI ≤ 29.95 , mô hình dự đoán là class 0.
- Nếu BMI > 29.95 , mô hình dự đoán là class 1.

Tóm lại, cây quyết định đưa ra dự đoán dựa trên mức đường huyết và sau đó kiểm tra BMI để tạo ra quyết định cuối cùng. Cây này có vẻ đơn giản, và quyết định chủ yếu tập trung vào mức đường huyết và BMI để phân loại các trường hợp thành hai lớp: class 0 và class 1.

VII. DỰ ĐOÁN [TYPE-I & TYPE-II] HOẶC [T1DM & T2DM]

1. Giải thích về dự đoán

a. Chuẩn đoán bệnh tiểu đường và tiền tiểu đường:

- Các xét nghiệm thường được sử dụng để chẩn đoán bệnh tiểu đường và tiền tiểu đường bao gồm xét nghiệm Đường huyết lúc đói (FPG) và xét nghiệm A1C.
- Xét nghiệm FPG đo lượng đường trong máu sau 8 giờ nhịn ăn
 - Bình thường: $<99 \text{ mg/dL}$
 - Tiền tiểu đường: $100-125 \text{ mg/dL}$
 - Bệnh tiểu đường: $\geq 126 \text{ mg/dL}$

b. Phân biệt loại bệnh tiểu đường bằng mức độ insulin:

1. Mức Insulin là dấu hiệu của các loại bệnh tiểu đường khác nhau.
2. Mức Insulin bình thường: $30-320 \mu\text{U/mL}$
3. Bệnh tiểu đường loại 1: Nồng độ insulin thấp hơn ($<30 \mu\text{U/mL}$) do sự phá hủy các tế bào sản xuất insulin.
4. Bệnh tiểu đường loại 2: Nồng độ insulin cao hơn ($>90 \mu\text{U/mL}$) so với bình thường.

Mục tiêu: Dự đoán các loại bệnh tiểu đường dựa trên kết quả xét nghiệm máu và nồng độ insulin.

c. Mô tả chức năng:

- myfunc(x, y):
 - Đầu vào: x (Giá trị insulin), y (Dự đoán loại bệnh tiểu đường)
- Hợp lý:
 - Nếu $x \leq 30$ và $y == 2$ (Tiểu đường tuýp 2), trả về y.
 - Nếu $x > 30$ và $y == 2$ (Tiểu đường Loại 2), trả về $y + 1$ (biết thị một loại khác).
 - Ngược lại, trả về y.

d. Quá trình dự đoán:

- Chuẩn đoán: Sử dụng xét nghiệm AFP và xét nghiệm A1C để chẩn đoán bệnh tiểu đường hoặc tiền tiểu đường dựa trên mức đường huyết.
- Dự đoán loại: Xác định loại bệnh tiểu đường bằng cách sử dụng nồng độ insulin:
 - Nếu 'Outcome' == 2 (Bệnh tiểu đường Loại 2):
 - Nếu Insulin > 30, điều chỉnh dự đoán sang loại khác.
 - Nếu 'Outcome' != 2 (Các loại bệnh tiểu đường khác), giữ nguyên dự đoán ban đầu

2. Trích xuất Glucose, Insulin và Kết quả trong tệp csv mới

```
[ ] # Load the Pima Indian dataset
file_path = '/content/drive/MyDrive/Data/diabetes.csv' # Replace with your file path
pima_data = pd.read_csv(file_path)

print(pima_data.columns)

# Selecting specific columns: 'glucose', 'insulin', 'outcome'
selected_columns = ['Glucose', 'Insulin', 'Outcome']
selected_data = pima_data[selected_columns]

# Save the selected data to a new CSV file
output_file_path = '/content/drive/MyDrive/Data/diabetes_new.csv' # Replace with your desired output file path
selected_data.to_csv(output_file_path, index=False)

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

3. Đọc và hiển thị dữ liệu mới

```
[ ] data = pd.read_csv('/content/drive/MyDrive/Data/diabetes_new.csv')
data.head()
```

	Glucose	Insulin	Outcome
0	148	0	1
1	85	0	0
2	183	0	1
3	89	94	0
4	137	168	1

4. Mô tả dữ liệu

```
[7] data.describe()
```

	Glucose	Insulin	Outcome	
count	768.000000	768.000000	768.000000	
mean	120.894531	79.799479	0.348958	
std	31.972618	115.244002	0.476951	
min	0.000000	0.000000	0.000000	
25%	99.000000	0.000000	0.000000	
50%	117.000000	30.500000	0.000000	
75%	140.250000	127.250000	1.000000	
max	199.000000	846.000000	1.000000	

5. Huấn luyện và phân chia dữ liệu

```
[ ] from sklearn.model_selection import train_test_split  
splitRatio = 0.2  
  
train , test = train_test_split(data,test_size = splitRatio,random_state = 123)  
  
X_train = train[[x for x in train.columns if x not in ["Outcome"]]]  
y_train = train[["Outcome"]]  
  
X_test = test[[x for x in test.columns if x not in ["Outcome"]]]  
y_test = test[["Outcome"]]
```

6. Hiển thị hình dạng của dữ liệu được chia tách

```
[ ] from sklearn.model_selection import train_test_split  
#X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)  
  
print(X_train.shape)  
print(y_train.shape)  
print(X_test.shape)  
print(y_test.shape)  
  
()
```

(614, 2)
(614, 1)
(154, 2)
(154, 1)

7. Sử dụng phân lớp KNN

```
[ ] from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy_score  
  
knn = KNeighborsClassifier()  
  
knn.fit(X_train,y_train)  
prediction = knn.predict(X_test)  
accuracy_score(y_test,prediction)
```

0.7402597402597403

Độ chính xác của mô hình là 74% tức là tỉ lệ đoán đúng trên tổng số mẫu.

8. Lấy dữ liệu tùy chỉnh để kiểm tra dự đoán

```
[ ] new_df = pd.DataFrame([[160, 30]])  
new_df.columns = ["Glucose", "Insulin"]  
prediction = knn.predict(new_df)  
  
if prediction == 1:  
    print('Type-2')  
else:  
    print('Type-1')
```

Type-2

9. Lưu và hiển thị tất cả các dự đoán của T1DM và T2DM

```
[ ] # Make predictions for the entire dataset using the KNN classifier
predictions = knn.predict(data[['Glucose', 'Insulin']])

# Map predictions to type labels ('Type-1' or 'Type-2')
prediction_labels = np.where(predictions == 1, 'Type-2', 'Type-1')

# Add the predicted labels to the DataFrame
data['Prediction'] = prediction_labels

# Display the DataFrame with the predictions
print(data)

# Save the DataFrame with predictions to an Excel file
data.to_excel('/content/drive/MyDrive/Data/diabetes.xlsx', index=False)
```

	Glucose	Insulin	Outcome	Prediction
0	148	0	1	Type-1
1	85	0	0	Type-1
2	183	0	1	Type-2
3	89	94	0	Type-1
4	137	168	1	Type-2
..
763	101	180	0	Type-1
764	122	0	0	Type-1
765	121	112	0	Type-1
766	126	0	1	Type-1
767	93	0	0	Type-1

[768 rows x 4 columns]

10. Đã lưu mô hình bằng Pickle

```
[ ] import pickle

# Assuming 'knn' is your trained KNN classifier
# Save the trained KNN model to a file using pickle
with open('/content/drive/MyDrive/Data/diabetes.pkl', 'wb') as file:
    pickle.dump(knn, file)
```

VIII. KẾT HỢP DỮ LIỆU VÀ PHÂN TÍCH

1. Bộ dữ liệu và mô hình được sử dụng

Bộ dữ liệu: Là một cuộc khảo sát qua điện thoại liên quan đến sức khỏe được CDC thu thập hàng năm. Cuộc khảo sát thu thập phản hồi từ hơn 400.000 người Mỹ về các hành vi nguy cơ liên quan đến sức khỏe, tình trạng sức khỏe mãn tính và việc sử dụng các dịch vụ phòng ngừa. Các tính năng này là các câu hỏi được đặt ra trực tiếp cho người tham gia hoặc các biến được tính toán dựa trên phản hồi của từng người tham gia. Bộ dữ liệu này là một tập dữ liệu sạch gồm 253.680 câu trả lời khảo sát theo BRFSS 2015 của CDC. (<https://www.kaggle.com/>)

Nội dung bộ dữ liệu: Các bộ dữ liệu bao gồm biến mục tiêu Diabetes_binary , không mắc bệnh tiểu đường hoặc chỉ trong thời kỳ mang thai, 1 là bệnh tiểu đường. Tập dữ liệu này có 21 biến đặc trưng

2. Mô tả thuộc tính

- **Diabetes Binary** - Cho biết người này có bị bệnh tiểu đường hay không(0 = không bị, 1 = tiểu đường)
- **High BP** - Cao huyết áp(0 = không, 1 = có)
- **HighsChool** - Lượng cholesterol trong máu cao(0 = không, 1 = có)
- **CholCheck** - Có kiểm tra cholesterol trong vòng 5 năm(0 = không, 1 = có)
- **BMI** - chỉ số khối cơ thể
- **Smoker** - Đã từng hút ít nhất 100 điếu thuốc lá(0 = không, 1 = có)
- **Troke** - Từng bị đột quỵ(0 = không, 1 = có)
- **HeartDiseaseorAttack** - Bị tim bẩm sinh hoặc đau tim(0 = không, 1 = có)
- **Phys Activity** - Có thực hiện các hoạt động thể chất, không bao gồm công việc(0 = không, 1 = có)
- **Fruits** - Có ăn trái cây ít nhất 1 lần mỗi ngày(0 = không, 1 = có)
- **Veggies** - Có ăn rau củ ít nhất 1 lần mỗi ngày(0 = không, 1 = có)
- **Hvy Alcohol Consump** - Nghiện rượu(0 = không, 1 = có)
- **Any Healthcare** - Có bảo hiểm sức khỏe(0 = không, 1 = có)
- **NoDocbcCost** - Cần đi khám bệnh trong 12 tháng qua nhưng không thẻ vì lý do chi phí(0 = không, 1 = có)
- **GenTle** - Tự đánh giá sức khỏe của bản thân trên thang điểm 1-5(1 = Cực kì tốt, 2 = rất tốt, 3 = tốt, 4 = ổn, 5 = tệ)
- **Men's Health** - số lần gặp các vấn đề tâm lý trong 30 ngày gần nhất(1-30)
- **PhysHltn** - Số lần gặp các vấn đề sức khỏe vật lý trong 30 ngày gần nhất(1-30)
- **Diff Walk** - Khó khăn khi đi bộ hoặc leo cầu thang(0 = không, 1 = có)
- **Sex** - Giới tính(0 = nữ, 1 = nam)
- **Age** - Tuổi(Scale độ tuổi từ 18-80 thành 1-13)
- **Education** - Trình độ học vấn(1-6)
- **Income** - Thu nhập(1-8)

Các thuộc tính cần thiết cho bài toán: 'Diabetes_binary', 'HighBP', 'BMI', 'Smoker', 'Sex', 'Age', 'Veggies', 'Fruits', 'PhysActivity', 'Stroke'

3. Xử lý dữ liệu

- diabetes_binary_5050split_health_indicators_BRFSS2015

```
[34] #Lấy những cột cần thiết cho bài toán
df_BRFSS = df_BRFSS[['Diabetes_binary', 'HighBP', 'BMI', 'Smoker', 'Sex', 'Age', 'Veggies', 'Fruits', 'PhysActivity', 'Stroke']]

# Tìm và hiển thị số lượng dòng có giá trị 1 (nam) trong cột "Sex"
rows_with_zero_sex = df_BRFSS[df_BRFSS['Sex'] == 1]
print("Số lượng dòng có giá trị 1 trong cột 'Sex':", len(rows_with_zero_sex))

# Xóa dòng có giá trị 1 trong cột "Sex" từ dataframe
df_BRFSS = df_BRFSS[df_BRFSS['Sex'] != 1]

#Sửa lại tên cột
df_BRFSS = df_BRFSS.rename(columns = {'Diabetes_binary':'Outcome', 'HighBP':'BloodPressure'})
```

Sau đó loại bỏ cột 'Sex'

- Pima Indians Diabetes Database
Xử lý số tuổi thành các nhóm tuổi như dataset trên

```
[129] def assign_age_group(age):
    if 18 <= age <= 24:
        return '1'
    elif 25 <= age <= 29:
        return '2'
    elif 30 <= age <= 34:
        return '3'
    elif 35 <= age <= 39:
        return '4'
    elif 40 <= age <= 44:
        return '5'
    elif 45 <= age <= 49:
        return '6'
    elif 50 <= age <= 54:
        return '7'
    elif 55 <= age <= 59:
        return '8'
    elif 60 <= age <= 64:
        return '9'
    elif 65 <= age <= 69:
        return '10'
    elif 70 <= age <= 74:
        return '11'
    elif 75 <= age <= 79:
        return '12'
    else:
        return '13'
df_Indian['Age'] = df_Indian['Age'].apply(assign_age_group)
```

- Xử lý chỉ số huyết áp (thấp - 0, cao - 1)

```
[131] def assign_BloodPressure_group(BloodPressure):
    if BloodPressure < 80:
        return '0'
    else:
        return '1'

df_Indian['BloodPressure'] = df_Indian['BloodPressure'].apply(assign_BloodPressure_group)
```

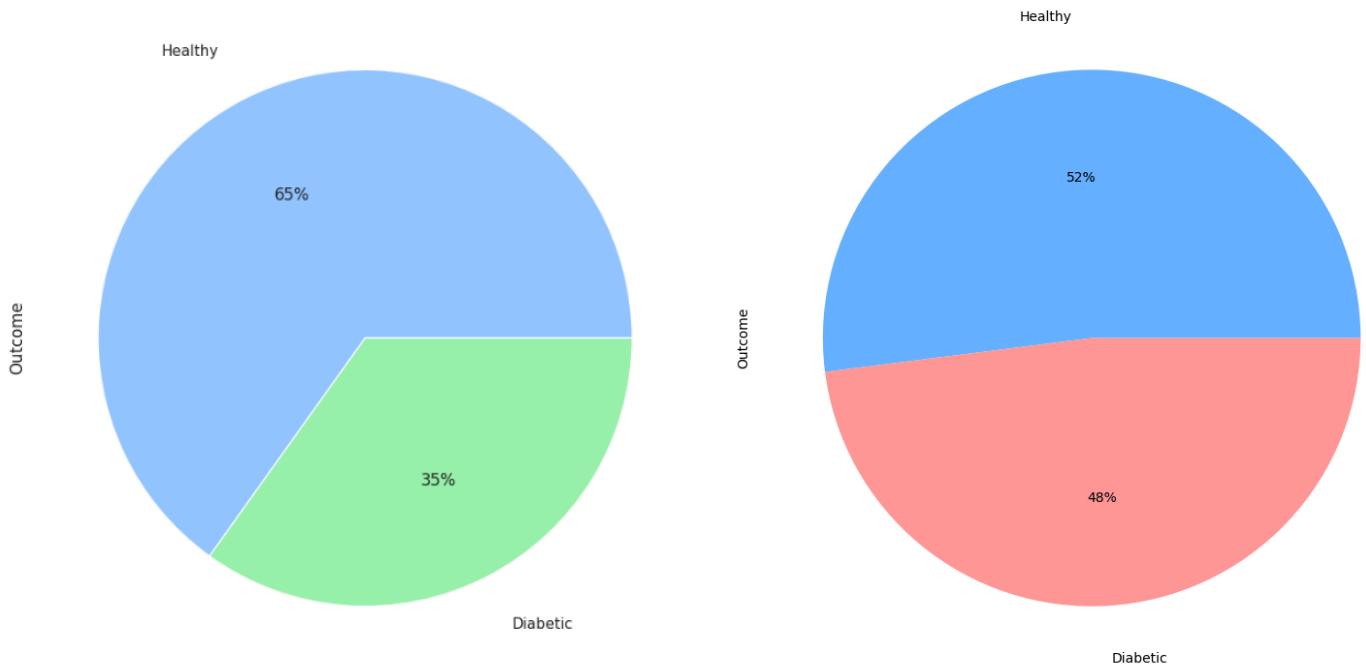
4. Kiểm tra số lượng giá trị duy nhất trong mỗi cột

```
▶ dict = {}
for i in list(df_BRFSS.columns):
    dict[i] = df_BRFSS[i].value_counts().shape[0]

pd.DataFrame(dict, index=["unique count"]).transpose()
```

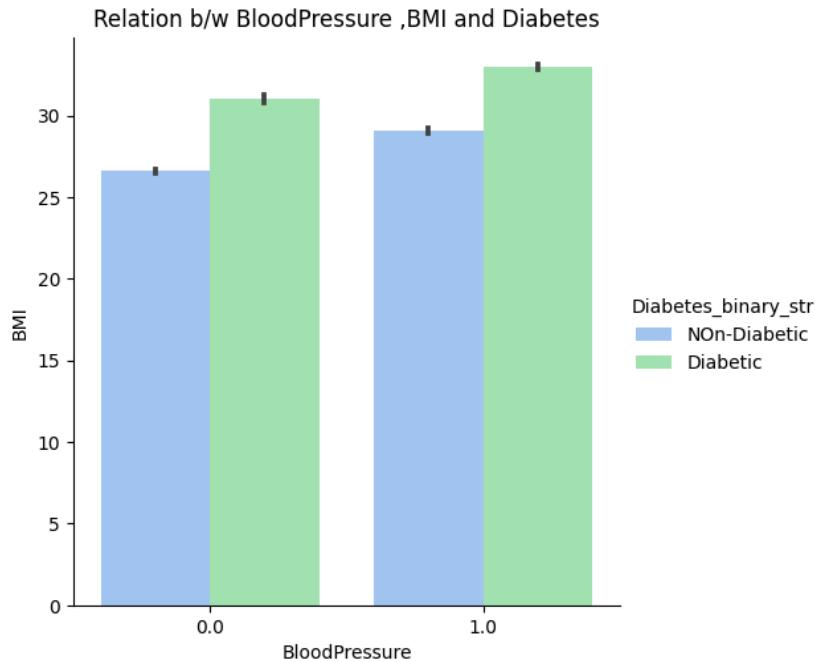
	unique count	grid
Outcome	2	grid
BloodPressure	2	grid
BMI	79	grid
Smoker	2	grid
Age	13	grid
Veggies	2	grid
Fruits	2	grid
PhysActivity	2	grid
Stroke	2	grid

5. Nhận xét bổ sung cho tập dữ liệu Pima Indians Diabetes



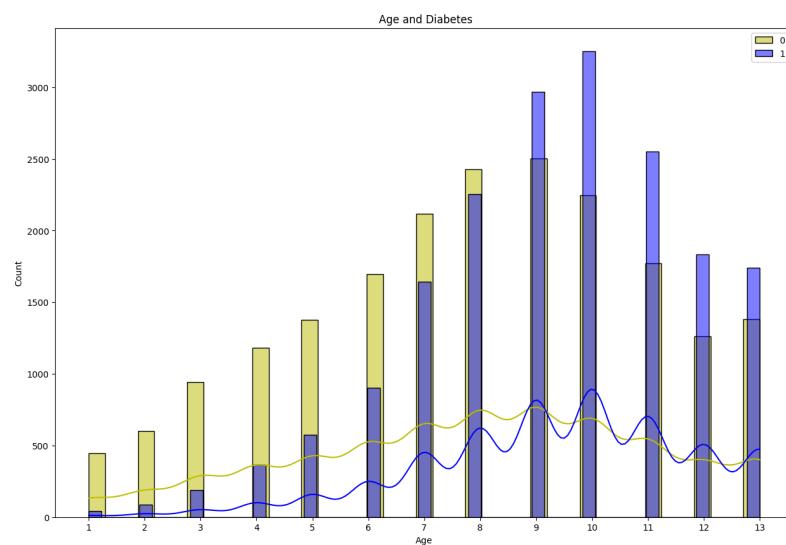
Hình 5.1: Tỷ lệ mắc bệnh và không mắc bệnh Tiểu đường của Người phụ nữ da đỏ với Người phụ nữ Mỹ

- Nhận xét: Theo dữ liệu này cho thấy: Tỷ lệ mắc bệnh của mắc bệnh tiểu đường của Người phụ nữ da đỏ thấp hơn Người phụ nữ Mỹ, cụ thể 35% so với 48% chênh lệch nhau 13%



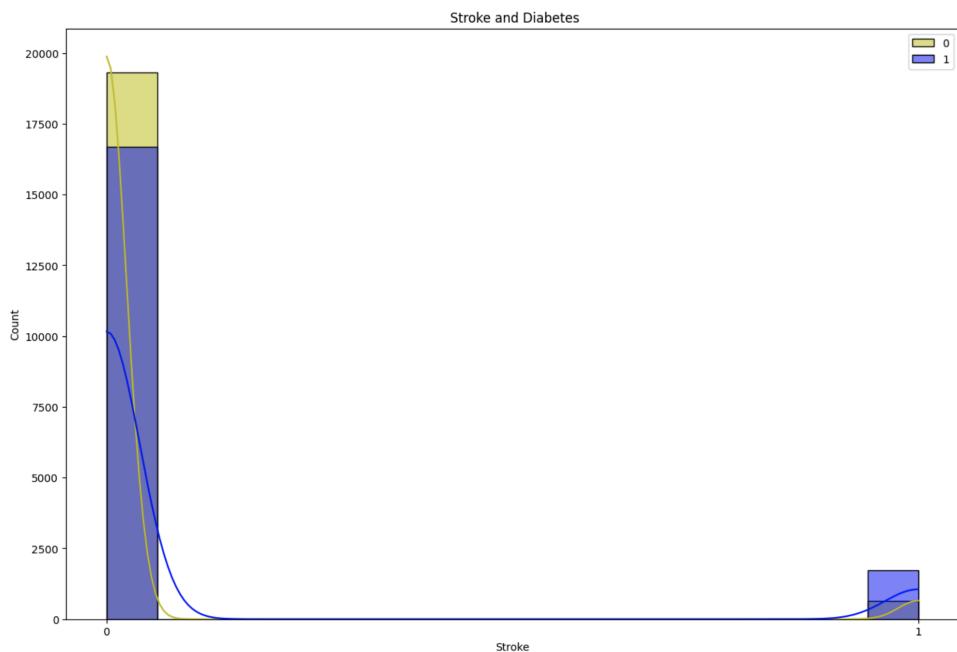
Hình 5.2. Biểu đồ Huyết áp, BMI và Bệnh tiểu đường

- Nhận xét: Huyết áp cao, BMI cao có nguy cơ cao dẫn đến bệnh tiểu đường



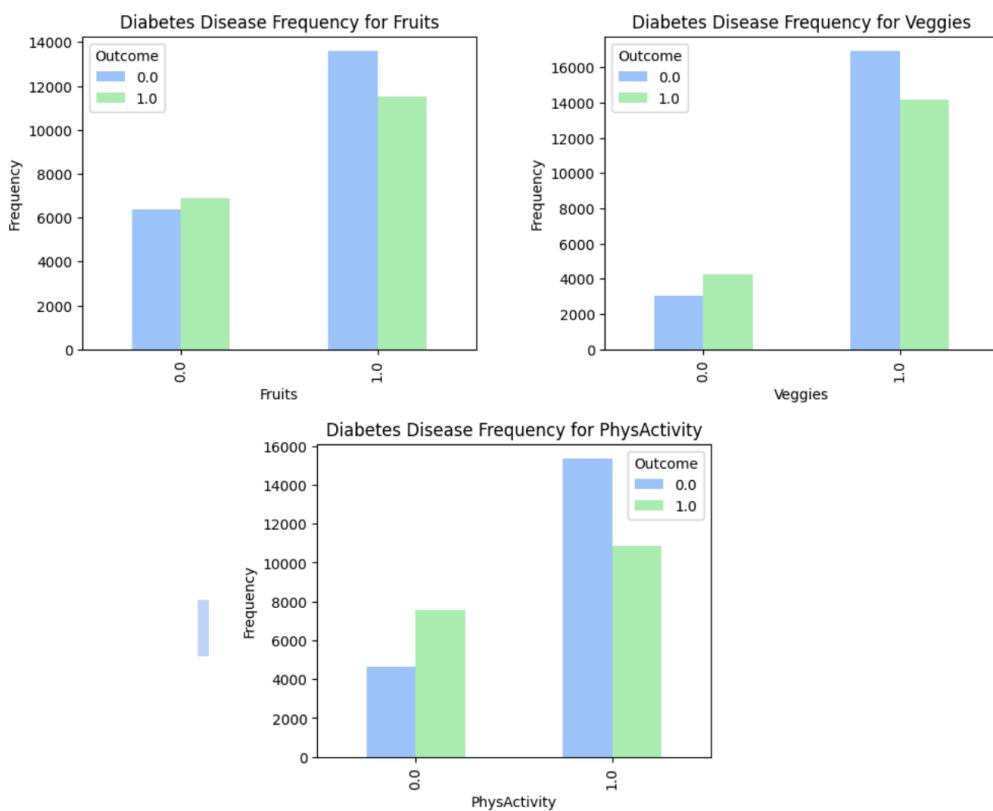
Hình 5.3. Biểu đồ nhóm tuổi liên quan đến bệnh tiểu đường

- Nhận xét: Nhóm tuổi cũng có ảnh hưởng đến bệnh tiểu đường. Biểu đồ trên cho ta biết nhóm tuổi có nguy cơ cao mắc bệnh tiểu đường là từ nhóm 9 đến nhóm 13 (cụ thể là 60 tuổi trở lên)



Hình 5.4. Biểu đồ Bệnh đột quy và bệnh tiểu đường

- Nhận xét: Theo dữ liệu này cho thấy, mắc bệnh đột quy sẽ có nguy cơ cao mắc bệnh Tiểu đường (tỷ lệ thuận).



Hình 5.5. Biểu đồ hoạt động hằng ngày - sử dụng rau - sử dụng trái cây và bệnh tiểu đường

Kết quả :

1. Hoạt động thể chất, ăn ít nhất một loại trái cây, một loại rau mỗi ngày làm giảm nguy cơ mắc bệnh tiểu đường.
2. Lối sống lành mạnh là chìa khóa của cơ thể khỏe mạnh.

Phần kết luận :

- Tập dữ liệu không bị thiếu giá trị (giá trị null).
- Các đặc điểm chính của Bệnh tiểu đường là: BloodPressure , BMI , Stroke, Age, Insulin, Glucose
- Các đặc điểm làm tăng nguy cơ mắc bệnh Tiểu đường là: Smoker, Stroke, BloodPressure
- Biến tính năng ít hiệu quả nhất đối với Bệnh tiểu đường, nhưng chúng có thể giúp giảm nguy cơ Bệnh tiểu đường là: PhysActivity, Fruits, Veggies.

C. PHẦN TÀI LIỆU THAM KHẢO

1. Tài liệu môn học Data Mining của TS. Nguyễn Tiên Đạt.
2. Diabetes.csv được truy lục từ Kaggle:
<https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset>
3. Diabetes_binary_5050split_health_indicators_BRFSS2015.csv được truy lục từ Kaggle:
https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset/data?select=diabetes_binary_5050split_health_indicators_BRFSS2015.csv
4. BEHAVIORAL RISK FACTOR SURVEILLANCE SYSTEM CODEBOOK REPORT, 2015 được truy lục từ:
https://www.cdc.gov/brfss/annual_data/2015/pdf/codebook15_llcp.pdf