diabetes_analysis

October 4, 2025

1 Phân tích và tiền xử lý bộ dữ liệu Tiểu Đường (Pima Indians Diabetes)

1.1 1. Định nghĩa vấn đề (Define Problem)

- Mô tả: Bộ dữ liệu Pima Indians Diabetes dùng để dự đoán khả năng mắc bệnh tiểu đường kiểu 2.
- Mục tiêu: Xây dựng pipeline tiền xử lý dữ liệu phục vụ huấn luyện mô hình phân loại biến đầu ra `Outcome`.
- Biến đầu ra (Output): `Outcome` (0: không mắc tiểu đường, 1: mắc tiểu đường)
- Các biến đầu vào (Input Features):
  1. Pregnancies: Số lần mang thai
  2. Glucose: Nồng độ glucose huyết tương sau 2 giờ (mg/dL)
  3. BloodPressure: Huyết áp tâm trương (mm Hg)
  4. SkinThickness: Độ dày nếp gấp da cơ tam đầu (mm)
  5. Insulin: Nồng độ insulin trong huyết thanh (mu U/ml)
  6. BMI: Chỉ số khối cơ thể (kg/mˆ2)
  7. DiabetesPedigreeFunction: Chỉ số phả hệ tiểu đường (khả năng di truyền)
  8. Age: Tuổi (năm)
- Ghi chú: Một số cột có giá trị 0 là không hợp lệ về mặt sinh học (Glucose, BloodPressure, SkinThickness, Insulin, BMI) và sẽ được xem như giá trị thiếu (missing).

1.2 2. Khai báo thư viện (Load Libraries)

[1]:
```
# Load libraries
import os, sys, warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
from IPython import display

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.impute import SimpleImputer

warnings.filterwarnings("ignore")
```

```
%matplotlib inline

plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['axes.grid'] = True

print('Libraries loaded.')
```

```
Libraries loaded.
```

1.3 3. Nạp dữ liệu (Load Dataset)

[2]:
```
# Load dataset
DATA_PATH = 'diabetes.csv'
df_raw = pd.read_csv(DATA_PATH)
df = df_raw.copy()
print(f'+ Shape: {df.shape}')
# Use display.display due to module import
display.display(df.head())
```

```
+ Shape: (768, 9)
```

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

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

1.4 4. Hiển thị thông tin dữ liệu (Data Overview)

- Kiểm tra kích thước, kiểu dữ liệu, vài dòng đầu/cuối, thông tin tổng quát.
- Kiểm tra giá trị duy nhất của Outcome.

[3]:
```
# Data Overview
print(f'+ Shape: {df.shape}')
print('\n+ Dtypes:')
print(df.dtypes)
print('\n+ Head:')
display.display(df.head())
print('\n+ Tail:')
display.display(df.tail())
```

```
print('\n+ Info:')
df.info()
print('\n+ Unique Outcome values:', df['Outcome'].unique())
```

```
+ Shape: (768, 9)

+ Dtypes:
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object

+ Head:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1


+ Tail:
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
```

```
767                      0.315   23        0
```

```
+ 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

+ Unique Outcome values: [1 0]
```

1.5 5. Kiểm tra tính toàn vẹn dữ liệu (Data Integrity Checks)

- Null / NaN
- Duplicates
- Cột có nhiều giá trị 0 bất thường

[4]:
```
# Integrity checks
null_counts = df.isnull().sum()
nan_counts  = df.isna().sum()
duplicated_n = df.duplicated().sum()
print('+ Null counts:\n', null_counts)
print('\n+ NaN counts:\n', nan_counts)
print(f'\n+ Duplicated rows: {duplicated_n}')
if duplicated_n>0:
    display.display(df[df.duplicated()])

# Columns with zeros possibly invalid
invalid_zero_cols = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
zero_summary = {c: int((df[c]==0).sum()) for c in invalid_zero_cols}
print('\n+ Zero value counts (potential missing):')
for k,v in zero_summary.items():
    print(f'  - {k}: {v}')
```

```
+ Null counts:
 Pregnancies                 0
Glucose                     0
```

```
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

+ NaN counts:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

+ Duplicated rows: 0

+ Zero value counts (potential missing):
  - Glucose: 5
  - BloodPressure: 35
  - SkinThickness: 227
  - Insulin: 374
  - BMI: 11
```

Nhận xét dữ liệu: - Bộ dữ liệu gồm 768 dòng và 9 cột. - Không có giá trị null/NaN thực sự, nhưng nhiều cột sinh học có giá trị 0 bất thường (Glucose, BloodPressure, SkinThickness, Insulin, BMI). - Outcome đã được mã hóa nhị phân (0/1), tỷ lệ lớp có sự mất cân bằng nhẹ. - Một số biến có phân phối lệch (skew) và có thể có ngoại lệ (outlier). - Cần xử lý giá trị 0 bất hợp lệ và chuẩn hóa dữ liệu trước khi huấn luyện mô hình.

1.6 6. Thống kê mô tả (Descriptive Statistics)

- Thống kê cơ bản, skew, kurtosis; nhận xét phạm vi và phân phối sơ bộ.

[5]:
```
# Descriptive statistics
desc = df.describe().T
skewness = df.skew().to_frame(name='skew')
kurtosis = df.kurtosis().to_frame(name='kurtosis')
desc_full = desc.join(skewness).join(kurtosis)
display.display(desc_full)

print('\nRange check:')
```

```
for c in df.columns:
    print(f'{c}: min={df[c].min()}, max={df[c].max()}')
```

```
                          count        mean         std     min       25%  \
Pregnancies               768.0    3.845052    3.369578   0.000   1.00000
Glucose                   768.0  120.894531   31.972618   0.000  99.00000
BloodPressure             768.0   69.105469   19.355807   0.000  62.00000
SkinThickness             768.0   20.536458   15.952218   0.000   0.00000
Insulin                   768.0   79.799479  115.244002   0.000   0.00000
BMI                       768.0   31.992578    7.884160   0.000  27.30000
DiabetesPedigreeFunction  768.0    0.471876    0.331329   0.078   0.24375
Age                       768.0   33.240885   11.760232  21.000  24.00000
Outcome                   768.0    0.348958    0.476951   0.000   0.00000

                             50%       75%     max      skew  kurtosis
Pregnancies                3.0000    6.00000   17.00  0.901674  0.159220
Glucose                  117.0000  140.25000  199.00  0.173754  0.640780
BloodPressure             72.0000   80.00000  122.00 -1.843608  5.180157
SkinThickness             23.0000   32.00000   99.00  0.109372 -0.520072
Insulin                   30.5000  127.25000  846.00  2.272251  7.214260
BMI                       32.0000   36.60000   67.10 -0.428982  3.290443
DiabetesPedigreeFunction   0.3725    0.62625    2.42  1.919911  5.594954
Age                       29.0000   41.00000   81.00  1.129597  0.643159
Outcome                    0.0000    1.00000    1.00  0.635017 -1.600930


Range check:
Pregnancies: min=0, max=17
Glucose: min=0, max=199
BloodPressure: min=0, max=122
SkinThickness: min=0, max=99
Insulin: min=0, max=846
BMI: min=0.0, max=67.1
DiabetesPedigreeFunction: min=0.078, max=2.42
Age: min=21, max=81
Outcome: min=0, max=1
```

1.7 7. Phân bố Outcome & mất cân bằng lớp (Outcome Distribution)

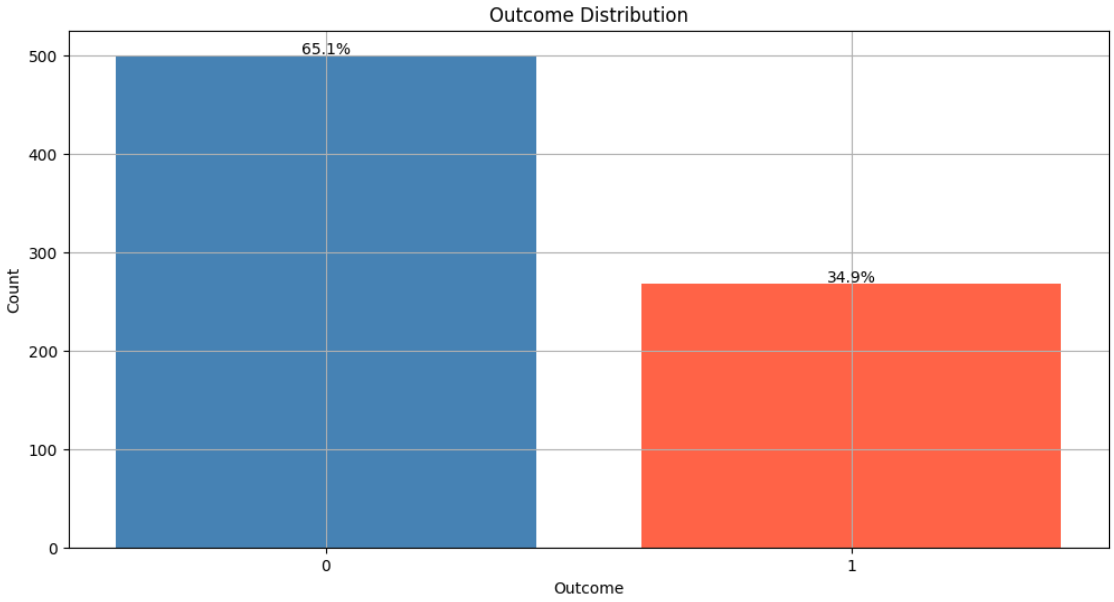
- Đếm số lượng và tỉ lệ phần trăm mỗi lớp.

[6]:
```
# Outcome distribution
counts = df['Outcome'].value_counts().sort_index()
perc = counts / counts.sum() * 100
print('Counts:\n', counts)
print('\nPercent (%):\n', perc.round(2))
fig, ax = plt.subplots()
ax.bar(counts.index.astype(str), counts.values, color=['steelblue','tomato'])
```

```
ax.set_xlabel('Outcome')
ax.set_ylabel('Count')
ax.set_title('Outcome Distribution')
for i,v in enumerate(counts.values):
    ax.text(i, v+2, f'{perc.values[i]:.1f}%', ha='center')
plt.show()
```

```
Counts:
 Outcome
0    500
1    268
Name: count, dtype: int64

Percent (%):
 Outcome
0    65.1
1    34.9
Name: count, dtype: float64
```



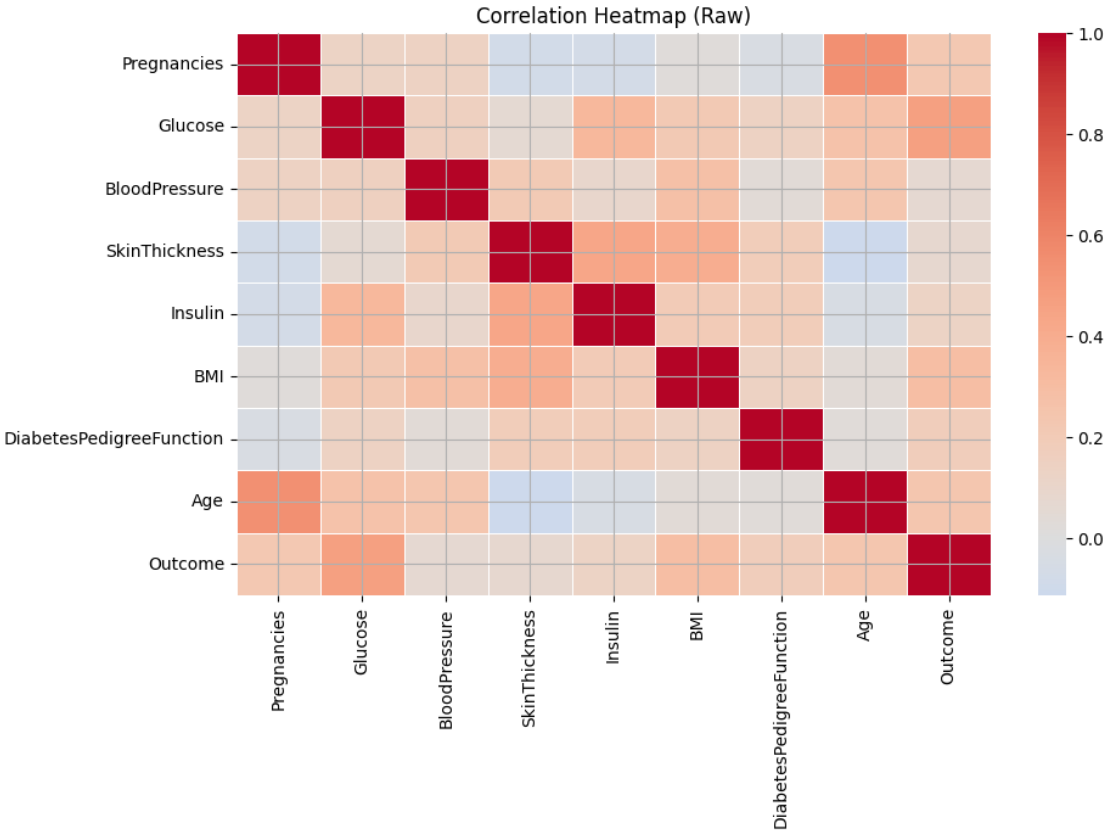
1.8 9. Ma trận tương quan (Correlation Matrix)

- Pearson correlation & heatmap; top thuộc tính tương quan với Outcome.

[7]:
```
# Correlation matrix
corr = df.corr(method='pearson')
plt.figure(figsize=(10,6))
sns.heatmap(corr, annot=False, cmap='coolwarm', center=0, linewidths=.5)
plt.title('Correlation Heatmap (Raw)')
```

```
plt.show()

outcome_corr = corr['Outcome'].drop('Outcome').abs().
 ↪sort_values(ascending=False)
print('Top correlations (abs) with Outcome:')
print(outcome_corr)
```




```
Top correlations (abs) with Outcome:
Glucose                     0.466581
BMI                         0.292695
Age                         0.238356
Pregnancies                 0.221898
DiabetesPedigreeFunction    0.173844
Insulin                     0.130548
SkinThickness               0.074752
BloodPressure               0.065068
Name: Outcome, dtype: float64
```

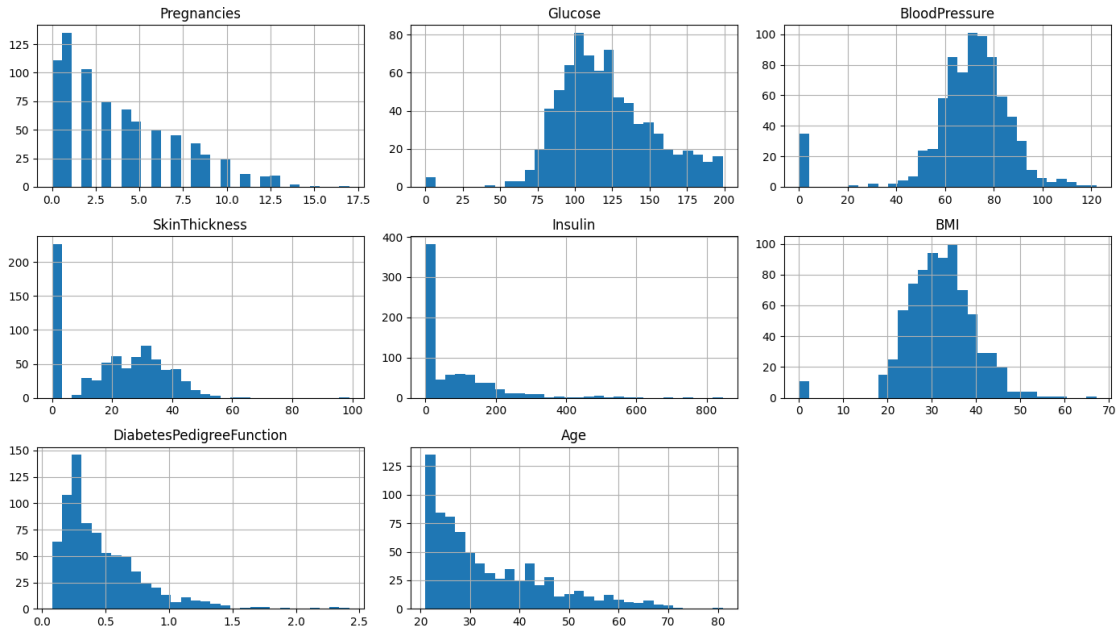
1.9 10. Hiển thị đơn biến (Univariate Plots)

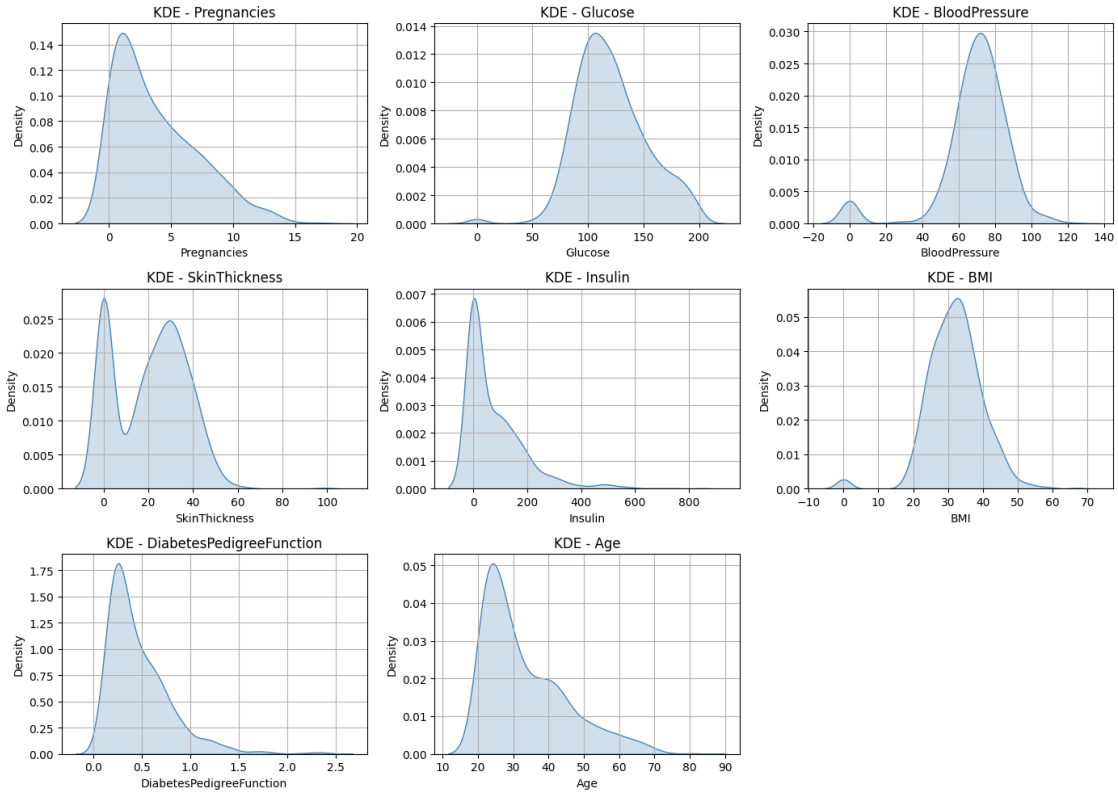
- Histogram, KDE, Boxplot để phát hiện outliers.

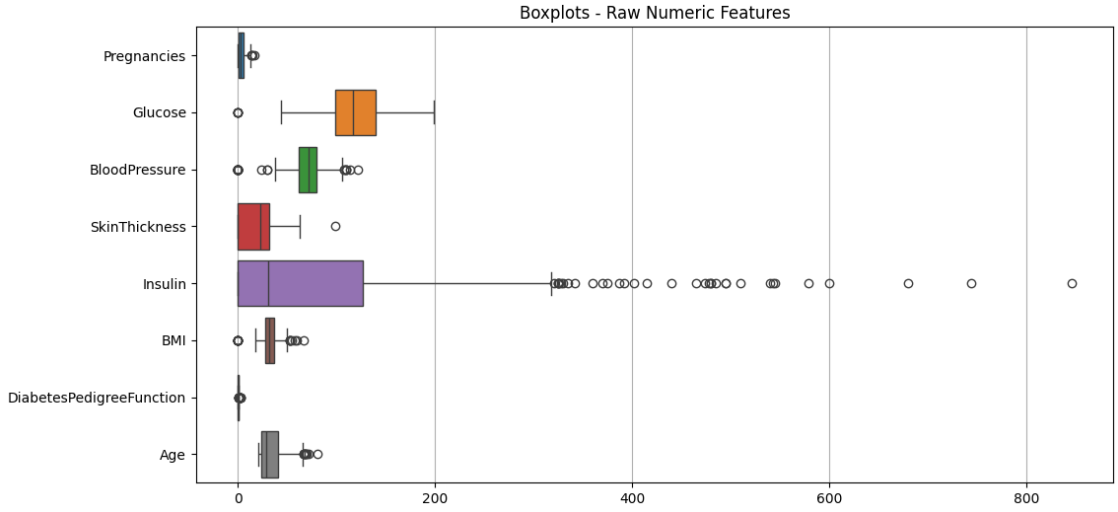
[8]:
```
# Univariate plots
num_cols = [c for c in df.columns if c != 'Outcome']
# Histograms
_ = df[num_cols].hist(bins=30, figsize=(14,8))
plt.tight_layout()
plt.show()

# KDE plots
fig, axes = plt.subplots(len(num_cols)//3 + 1, 3, figsize=(14,10))
axes = axes.flatten()
for i,c in enumerate(num_cols):
    sns.kdeplot(df[c], ax=axes[i], fill=True, color='steelblue')
    axes[i].set_title(f'KDE - {c}')
for j in range(i+1, len(axes)):
    axes[j].axis('off')
plt.tight_layout()
plt.show()

# Boxplots
plt.figure(figsize=(12,6))
sns.boxplot(data=df[num_cols], orient='h')
plt.title('Boxplots - Raw Numeric Features')
plt.show()
```







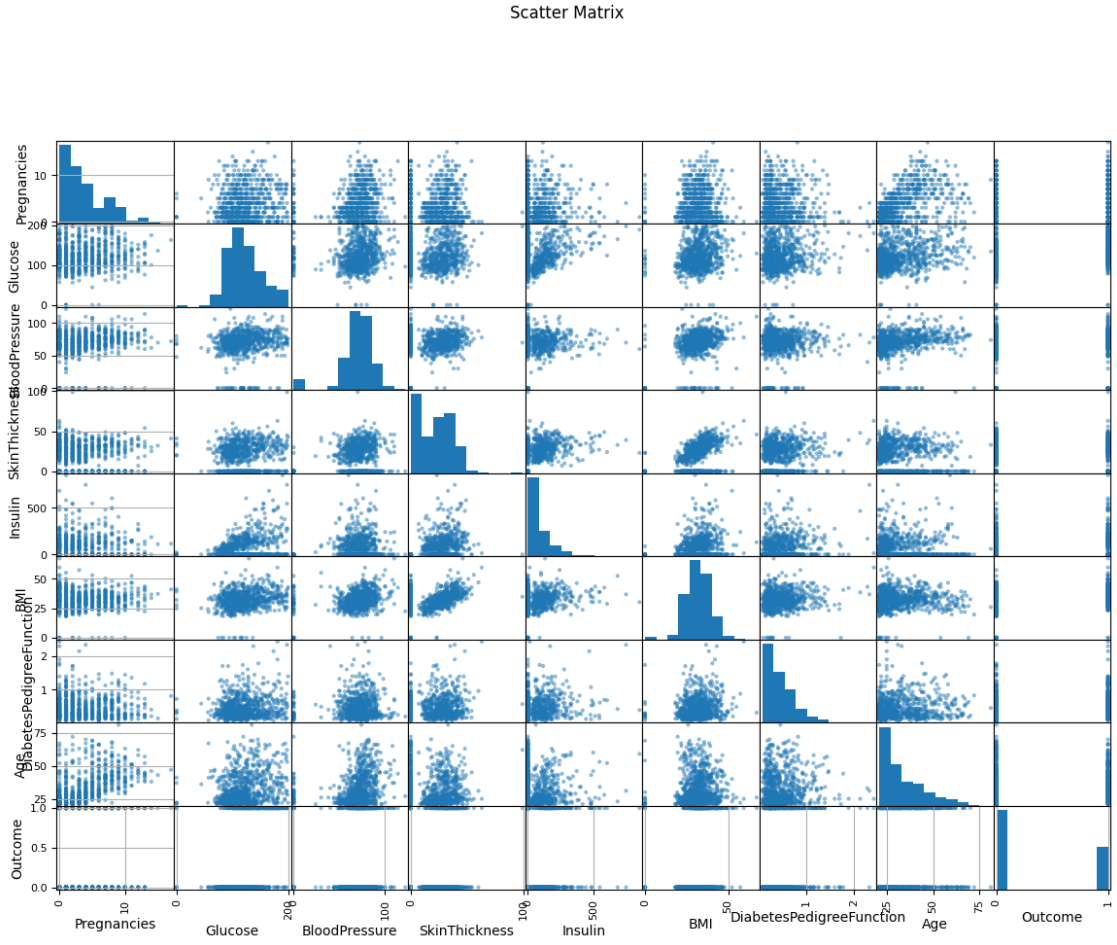
1.10 11. Hiển thị đa biến (Multivariate Plots)

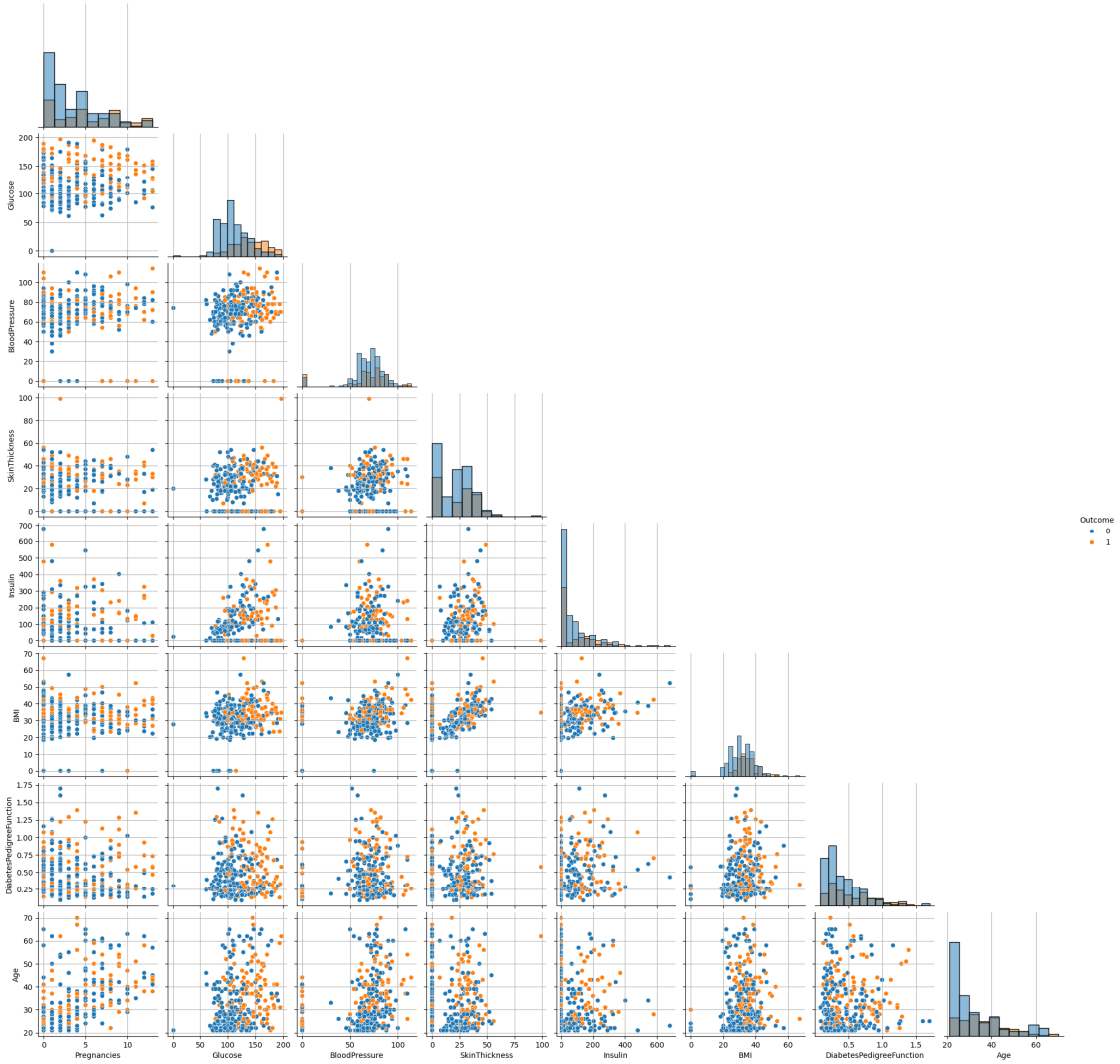
- Scatter matrix, pairplot, heatmap.

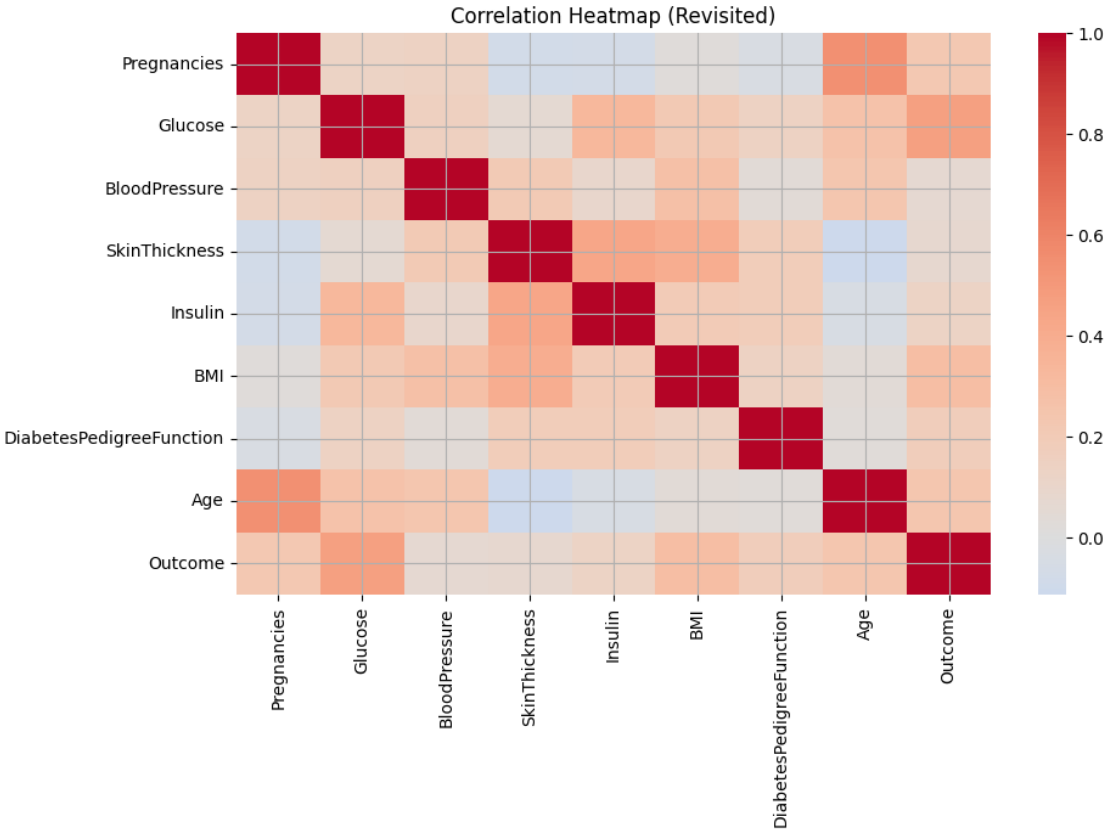
[9]:
```
# Multivariate plots
# Scatter matrix (may be large)
pd.plotting.scatter_matrix(df[num_cols + ['Outcome']], figsize=(14,10),␣
 ↪diagonal='hist')
plt.suptitle('Scatter Matrix', y=1.02)
plt.show()

# Pairplot (sample if needed)
sample_df = df.sample(min(300, len(df)), random_state=42)
sns.pairplot(sample_df, hue='Outcome', diag_kind='hist', corner=True)
plt.show()

# Heatmap again (focused)
plt.figure(figsize=(10,6))
sns.heatmap(corr, cmap='coolwarm', annot=False, center=0)
plt.title('Correlation Heatmap (Revisited)')
plt.show()
```




Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0
1



1.11 12. Tạo bản sao làm sạch (Create Clean DataFrame)

- Sao chép dữ liệu gốc để xử lý: df_clean = df.copy()

[10]:
```
# Create clean copy
df_clean = df.copy()
print('Clean copy created. Shape:', df_clean.shape)
```

```
Clean copy created. Shape: (768, 9)
```

1.12 13. Xử lý giá trị 0 -> NaN rồi nội suy (Impute Missing Values)

- Thay giá trị 0 ở các cột sinh học bằng NaN rồi dùng median để điền.

[11]:
```
# Impute zeros -> NaN -> median
impute_cols = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
for c in impute_cols:
    zero_mask = df_clean[c] == 0
    n_zero = int(zero_mask.sum())
    if n_zero>0:
        df_clean.loc[zero_mask, c] = np.nan
```

```
        median_val = df_clean[c].median()
        df_clean[c].fillna(median_val, inplace=True)
        print(f'{c}: replaced {n_zero} zeros with median={median_val:.2f}')

print('\nRemaining NaNs after imputation:')
print(df_clean[impute_cols].isna().sum())
```

```
Glucose: replaced 5 zeros with median=117.00
BloodPressure: replaced 35 zeros with median=72.00
SkinThickness: replaced 227 zeros with median=29.00
Insulin: replaced 374 zeros with median=125.00
BMI: replaced 11 zeros with median=32.30

Remaining NaNs after imputation:
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
dtype: int64
```

1.13 14. Loại bỏ trùng lặp (Remove Duplicates)

[12]:
```
# Remove duplicates
before = len(df_clean)
dup_n = df_clean.duplicated().sum()
print(f'Duplicated rows: {dup_n}')
if dup_n>0:
    df_clean.drop_duplicates(ignore_index=True, inplace=True)
    print(f'Removed {dup_n} duplicates. New shape: {df_clean.shape}')
else:
    print('No duplicates removed.')
```

```
Duplicated rows: 0
No duplicates removed.
```

1.14 15. Mã hóa nhãn Output (Ensure Label Encoding)

- Outcome đã là 0/1; ép kiểu int và xác nhận.

[13]:
```
# Ensure label encoding
df_clean['Outcome'] = df_clean['Outcome'].astype(int)
print('Unique Outcome values:', np.unique(df_clean['Outcome']))
```

```
Unique Outcome values: [0 1]
```

1.15 16. Chuẩn hóa dữ liệu Min-Max (MinMax Scaling)

- $z = (x - min)/(max - min)$

[14]:
```
# MinMax Scaling
scale_cols = [c for c in df_clean.columns if c != 'Outcome']
minmax_scaler = MinMaxScaler()
minmax_arr = minmax_scaler.fit_transform(df_clean[scale_cols])
df_minmax = df_clean.copy()
df_minmax[scale_cols] = minmax_arr
print('MinMax scaled. Range per column:')
for i,c in enumerate(scale_cols):
    print(c, f'min={df_minmax[c].min():.3f} max={df_minmax[c].max():.3f}')
```

```
MinMax scaled. Range per column:
Pregnancies min=0.000 max=1.000
Glucose min=0.000 max=1.000
BloodPressure min=0.000 max=1.000
SkinThickness min=0.000 max=1.000
Insulin min=0.000 max=1.000
BMI min=0.000 max=1.000
DiabetesPedigreeFunction min=0.000 max=1.000
Age min=0.000 max=1.000
```

1.16 17. Chuẩn hóa dữ liệu Standard (Standard Scaling)

- $z = (x - \mu)/\sigma$

[15]:
```
# Standard Scaling
standard_scaler = StandardScaler()
standard_arr = standard_scaler.fit_transform(df_clean[scale_cols])
df_standard = df_clean.copy()
df_standard[scale_cols] = standard_arr
print('Standard scaled. Mean/std per column:')
for c in scale_cols:
    print(c, f'mean={df_standard[c].mean():.3f} std={df_standard[c].std():.3f}')
```

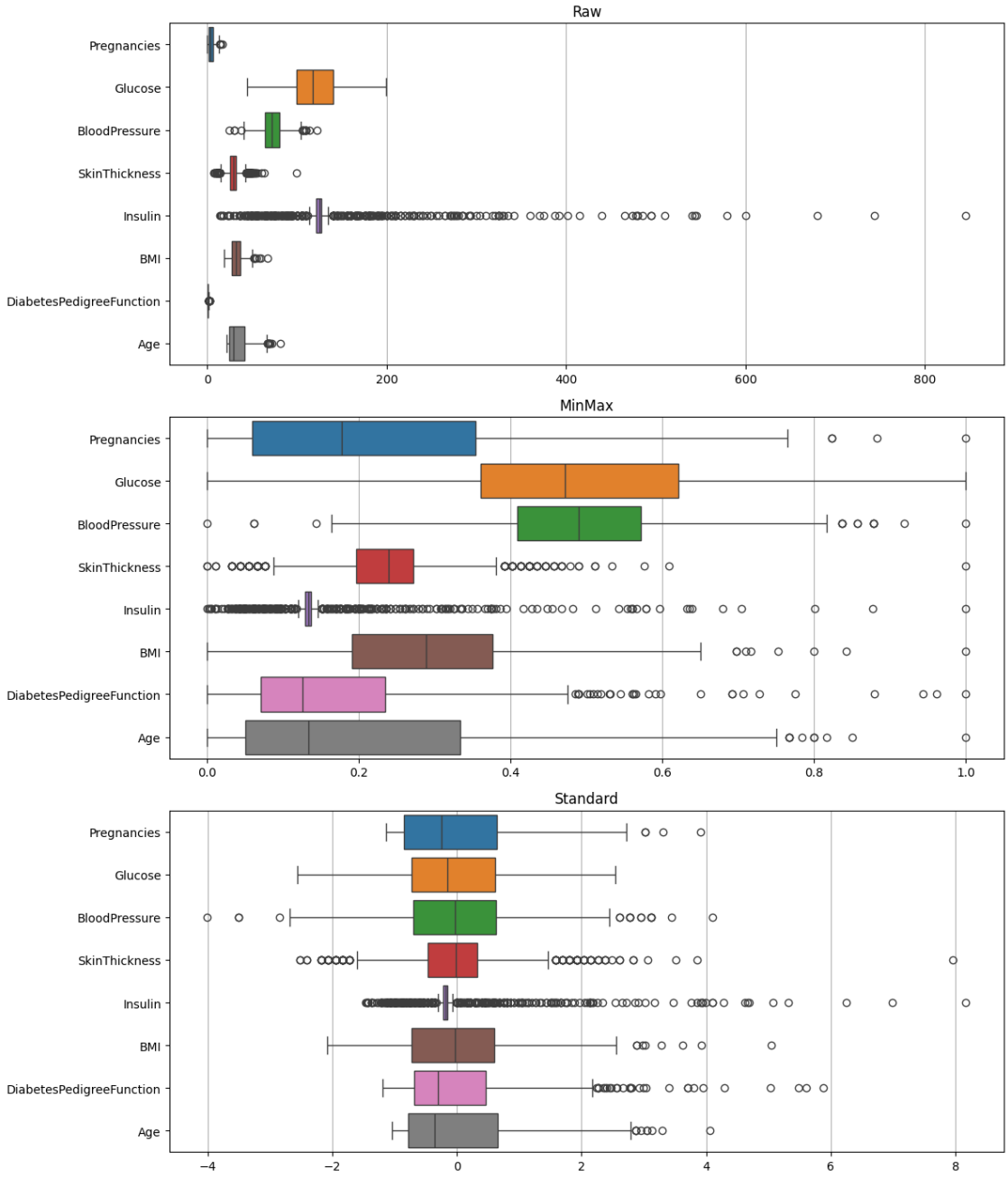
```
Standard scaled. Mean/std per column:
Pregnancies mean=-0.000 std=1.001
Glucose mean=0.000 std=1.001
BloodPressure mean=0.000 std=1.001
SkinThickness mean=-0.000 std=1.001
Insulin mean=0.000 std=1.001
BMI mean=0.000 std=1.001
DiabetesPedigreeFunction mean=0.000 std=1.001
Age mean=0.000 std=1.001
```

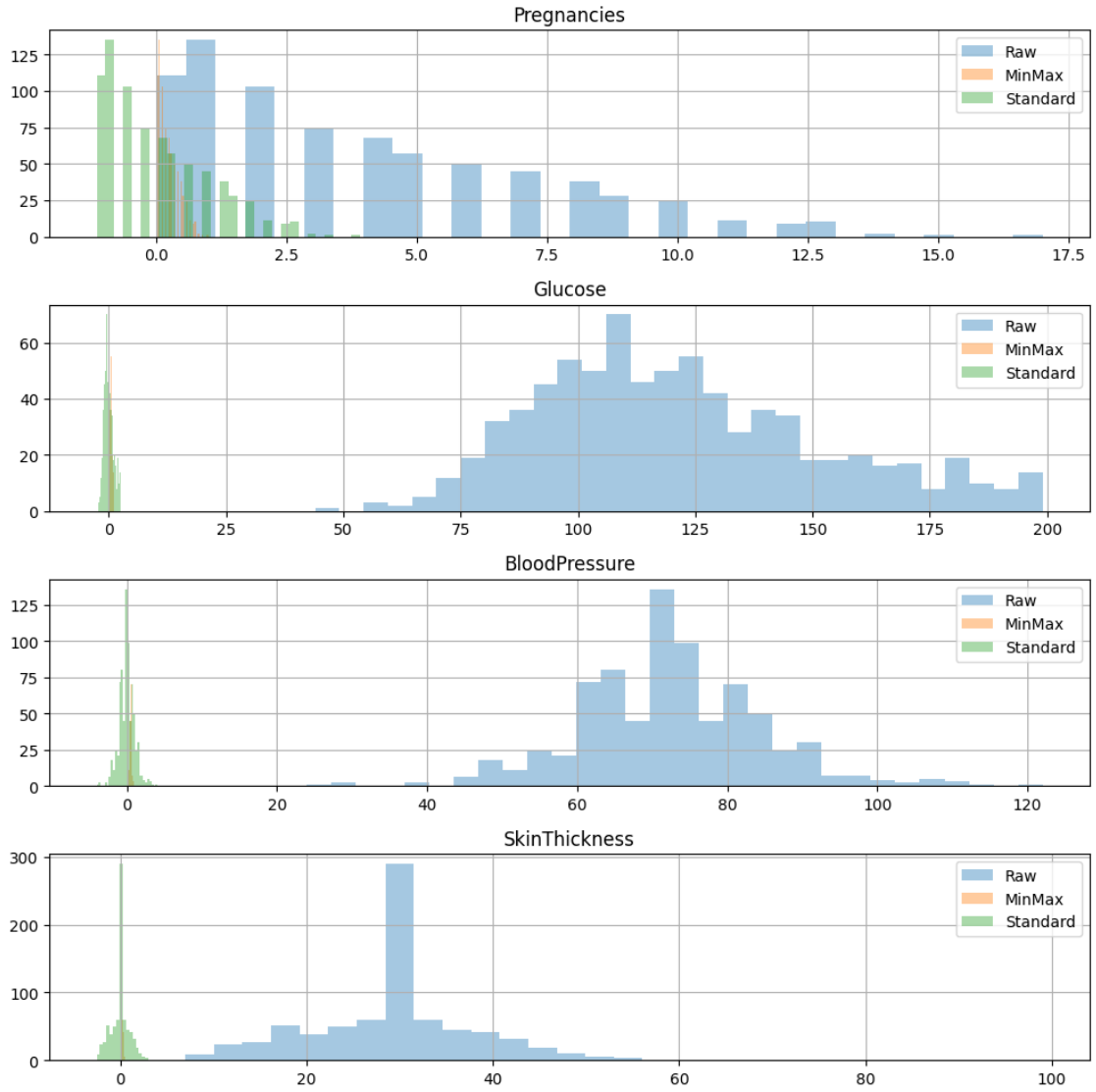
1.17 18. So sánh phân phối sau chuẩn hóa (Compare Scaled Distributions)

- Boxplot và histogram so sánh Raw vs MinMax vs Standard.

[16]:
```
# Compare distributions
fig, axes = plt.subplots(3,1, figsize=(12,14))
sns.boxplot(data=df_clean[scale_cols], ax=axes[0], orient='h')
axes[0].set_title('Raw')
sns.boxplot(data=df_minmax[scale_cols], ax=axes[1], orient='h')
axes[1].set_title('MinMax')
sns.boxplot(data=df_standard[scale_cols], ax=axes[2], orient='h')
axes[2].set_title('Standard')
plt.tight_layout()
plt.show()

# Overlay hist for a few selected features
sel = scale_cols[:4]
fig, axes = plt.subplots(len(sel), 1, figsize=(10, 10))
for i,c in enumerate(sel):
    axes[i].hist(df_clean[c], bins=30, alpha=0.4, label='Raw')
    axes[i].hist(df_minmax[c], bins=30, alpha=0.4, label='MinMax')
    axes[i].hist(df_standard[c], bins=30, alpha=0.4, label='Standard')
    axes[i].set_title(c)
    axes[i].legend()
plt.tight_layout()
plt.show()
```


Raw
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
0
200
400
600
800
MinMax
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
0.0
0.2
0.4
0.6
0.8
1.0
Standard
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
−4
−2
0
2
4
6
8



1.18 19. Tạo đặc trưng mới (Feature Engineering Optional)

- Ví dụ tạo nhóm tuổi, nhóm BMI, tương tác Glucose*BMI.

[17]:
```
# Feature engineering (optional)
df_fe = df_clean.copy()
# Age group
age_bins = [0,30,40,50,60,100]
age_labels = ['<30','30-39','40-49','50-59','60+']
df_fe['AgeGroup'] = pd.cut(df_fe['Age'], bins=age_bins, labels=age_labels,␣
 ↪right=False)
# BMI category
bmi_bins = [0,18.5,25,30,100]
```

```
bmi_labels = ['Under','Normal','Over','Obese']
df_fe['BMICat'] = pd.cut(df_fe['BMI'], bins=bmi_bins, labels=bmi_labels,␣
 ↪right=False)
# Interaction
df_fe['Glucose_BMI'] = df_fe['Glucose'] * df_fe['BMI']
print('New columns added: AgeGroup, BMICat, Glucose_BMI')
display.
 ↪display(df_fe[['Age','AgeGroup','BMI','BMICat','Glucose','Glucose_BMI']].
 ↪head())
new_features = ['AgeGroup','BMICat','Glucose_BMI']
```

```
New columns added: AgeGroup, BMICat, Glucose_BMI
```

	Age	AgeGroup	BMI	BMICat	Glucose	Glucose_BMI
0	50	50-59	33.6	Obese	148.0	4972.8
1	31	30-39	26.6	Over	85.0	2261.0
2	32	30-39	23.3	Normal	183.0	4263.9
3	21	<30	28.1	Over	89.0	2500.9
4	33	30-39	43.1	Obese	137.0	5904.7

1.19 20. Tách Input/Output & ép kiểu (Split X / y)

- X_data, y_data.

[18]:
```
# Split X/y
X_data = df_clean.drop('Outcome', axis=1).values
y_data = df_clean['Outcome'].values.astype(int)
print('Shapes:', X_data.shape, y_data.shape)
print('First 5 rows X:\n', X_data[:5])
print('First 20 y:', y_data[:20])
```

```
Shapes: (768, 8) (768,)
First 5 rows X:
 [[6.000e+00 1.480e+02 7.200e+01 3.500e+01 1.250e+02 3.360e+01 6.270e-01
  5.000e+01]
 [1.000e+00 8.500e+01 6.600e+01 2.900e+01 1.250e+02 2.660e+01 3.510e-01
  3.100e+01]
 [8.000e+00 1.830e+02 6.400e+01 2.900e+01 1.250e+02 2.330e+01 6.720e-01
  3.200e+01]
 [1.000e+00 8.900e+01 6.600e+01 2.300e+01 9.400e+01 2.810e+01 1.670e-01
  2.100e+01]
 [0.000e+00 1.370e+02 4.000e+01 3.500e+01 1.680e+02 4.310e+01 2.288e+00
  3.300e+01]]
First 20 y: [1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1]
```

1.20 21. Chia tập Train/Test (Train/Test Split)

- Stratify theo Outcome, test_size=0.3, random_state=42.

[19]:
```
# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(
    X_data, y_data, test_size=0.3, stratify=y_data, random_state=42
)
print(f'Train size: {X_train.shape}, Test size: {X_test.shape}')
print('Train class distribution:', np.bincount(y_train))
print('Test class distribution:', np.bincount(y_test))
```

```
Train size: (537, 8), Test size: (231, 8)
Train class distribution: [350 187]
Test class distribution: [150  81]
```

1.21 22. Lưu dữ liệu & đối tượng tiền xử lý (Persist Artifacts)

- Lưu: data.npz, df_clean.csv, scaler objects.
- Thư mục: exps/diabetes

[20]:
```
# Persist artifacts
save_dir = 'exps/diabetes'
os.makedirs(save_dir, exist_ok=True)

# Save numpy data
np.savez(f'{save_dir}/data.npz', X_train=X_train, X_test=X_test,␣
 ↪y_train=y_train, y_test=y_test)
# Save cleaned dataframe
clean_path = f'{save_dir}/df_clean.csv'
df_clean.to_csv(clean_path, index=False)
# Save scalers
joblib.dump(minmax_scaler, f'{save_dir}/minmax_scaler.joblib')
joblib.dump(standard_scaler, f'{save_dir}/standard_scaler.joblib')

print('Saved files:')
print(os.listdir(save_dir))
```

```
Saved files:
['data.npz', 'df_clean.csv', 'standard_scaler.joblib', 'minmax_scaler.joblib']
```

2 Kết thúc

[24]:
```
!jupyter nbconvert diabetes_analysis.ipynb --to latex
```

```
[NbConvertApp] Converting notebook diabetes_analysis.ipynb to latex
[NbConvertApp] ERROR | Error while converting 'diabetes_analysis.ipynb'
Traceback (most recent call last):
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/nbconvertapp.py", line 487, in export_single_notebook
    output, resources = self.exporter.from_filename(
                        ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

```
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/templateexporter.py", line 390, in from_filename
    return super().from_filename(filename, resources, **kw)  #
type:ignore[return-value]
           ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/exporter.py", line 201, in from_filename
    return self.from_file(f, resources=resources, **kw)
           ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/templateexporter.py", line 396, in from_file
    return super().from_file(file_stream, resources, **kw)  #
type:ignore[return-value]
           ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/exporter.py", line 220, in from_file
    return self.from_notebook_node(
           ■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/latex.py", line 92, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
           ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/exporters/templateexporter.py", line 429, in
from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
             ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/jinja2/environment.py", line 1295, in render
    self.environment.handle_exception()
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/jinja2/environment.py", line 942, in handle_exception
    raise rewrite_traceback_stack(source=source)
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/inde
x.tex.j2", line 8, in top-level template code
    ((* extends cell_style *))
    ■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/styl
e_jupyter.tex.j2", line 176, in top-level template code
    \prompt{(((prompt)))}{(((prompt_color)))}{(((execution_count)))}{(((extra_sp
ace)))}
    ■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/base
.tex.j2", line 7, in top-level template code
    ((*- extends 'document_contents.tex.j2' -*))
    ■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/docu
ment_contents.tex.j2", line 51, in top-level template code
```

```
    ((*- block figure scoped -*))
    ■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/disp
lay_priority.j2", line 5, in top-level template code
    ((*- extends 'null.j2' -*))
    ■■■■■■■■■■■■■■■■■■■■■■■■■
  File
"/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/null.j2",
line 30, in top-level template code
    ((*- block body -*))
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/base
.tex.j2", line 241, in block 'body'
    ((( super() )))
  File
"/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/null.j2",
line 32, in block 'body'
    ((*- block any_cell scoped -*))
    ■■■■■■■■■■■■■■■■■■■■■■■■■
  File
"/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/null.j2",
line 85, in block 'any_cell'
    ((*- block markdowncell scoped-*)) ((*- endblock markdowncell -*))
    ■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/share/jupyter/nbconvert/templates/latex/docu
ment_contents.tex.j2", line 68, in block 'markdowncell'
    ((( cell.source | citation2latex | strip_files_prefix |
convert_pandoc('markdown+tex_math_double_backslash', 'json',extra_args=[]) |
resolve_references | convert_explicitly_relative_paths |
convert_pandoc('json','latex'))))
    ■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/filters/pandoc.py", line 36, in convert_pandoc
    return pandoc(source, from_format, to_format, extra_args=extra_args)
           ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/utils/pandoc.py", line 50, in pandoc
    check_pandoc_version()
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/utils/pandoc.py", line 98, in check_pandoc_version
    v = get_pandoc_version()
        ■■■■■■■■■■■■■■■■■■■■
  File "/home/tp_ubuntu/colab/.venv/lib/python3.12/site-
packages/nbconvert/utils/pandoc.py", line 75, in get_pandoc_version
    raise PandocMissing()
nbconvert.utils.pandoc.PandocMissing: Pandoc wasn't found.
Please check that pandoc is installed:
https://pandoc.org/installing.html
```