Ozan MÖHÜRCÜ

Data Analyst | Data Scientist

LinkedIn GitHub

📚 Content

📦 Libraries

🔙 Go to Content

In [46]:

```
# Public libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

# Visualization libraries
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
!pip install pywaffle > devnull
from pywaffle import Waffle
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.gridspec as gridspec
import matplotlib.cm as cm

# Plotly
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

# SHAP ve XGBoost
import shap
import xgboost
from sklearn.preprocessing import StandardScaler

# Statistics and other
from scipy import stats
from scipy.stats import kde
from math import pi

# Country converter
!pip install country-converter > devnull
import country_converter as coco

# Bokeh
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
output_notebook()

# Natural Language Toolkit (nltk)
import nltk

# Instant display of graphs for Jupyter
%matplotlib inline
```

🔍 Data Review

🔍 Data Review

🔙 Go to Content

📊 About the Dataset

Column	Description
Pregnancies	Number of times the patient has been pregnant.
Glucose	Plasma glucose concentration from an oral glucose tolerance test.
BloodPressure	Diastolic blood pressure (measured in mm Hg).
SkinThickness	Triceps skinfold thickness (measured in mm).
Insulin	2-hour serum insulin level (mu U/ml).
BMI	Body Mass Index (weight in kg divided by height in m²).
DiabetesPedigreeFunction	A function that scores likelihood of diabetes based on family history.
Age	Age of the patient (in years).
Outcome	Indicates whether the patient has diabetes (1) or not (0).

In [231…

```
df = pd.read_csv('/kaggle/input/diabetes-dataset/diabetes.csv')

def style_df(df):
    return df.style.set_properties(**{
        'background-color': '#e6f2ff',
        'color': '#000000',
        'border': '1px solid #d3d3d3',
        'padding': '8px',
        'text-align': 'center',
        'font-family': 'Arial, sans-serif'
    }).set_table_styles([
        {'selector': 'th', 'props': [('background-color', '#d1e8ff'), ('colo
    ])

style_df(df.head())
```

Out[231…

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes

0	6	148	72	35	0	33.600000
1	1	85	66	29	0	26.600000
2	8	183	64	0	0	23.300000
3	1	89	66	23	94	28.100000
4	0	137	40	35	168	43.100000

🎨 Data Visualization

Go to Content

📊 3D Scatter

🔙 **Go to Content**

In [188…

```
u = np.linspace(0, 2 * np.pi, 50)
v = np.linspace(0, 2 * np.pi, 50)
u, v = np.meshgrid(u, v)
r = 2
x = (r + np.cos(u/2) * np.sin(v) - np.sin(u/2) * np.sin(2*v)) * np.cos(u)
y = (r + np.cos(u/2) * np.sin(v) - np.sin(u/2) * np.sin(2*v)) * np.sin(u)
z = np.sin(u/2) * np.sin(v) + np.cos(u/2) * np.sin(2*v)


data_x = df['Glucose'] / df['Glucose'].max() * 2 * np.pi
data_y = df['BMI'] / df['BMI'].max() * 2 * np.pi
data_z = df['Age'] / df['Age'].max()

fig = go.Figure(data=[
    go.Surface(
        x=x, y=y, z=z,
        colorscale='Inferno', opacity=0.4
    ),
    go.Scatter3d(
        x=data_x, y=data_y, z=data_z,
        mode='markers',
        marker=dict(
            size=5,
            color=df['Outcome'],
            colorscale=['#6B5B95', '#FF6F61'],
            opacity=0.7
        )
    )
])


high_glucose_bmi = df[(df['Glucose'] > df['Glucose'].mean()) & (df['BMI'] >
fig.add_annotation(
    text=f"Diabetes rate in high Glucose and BMI: {high_glucose_bmi:.2%}.\nT
    xref="paper", yref="paper", x=0.05, y=0.98,
```

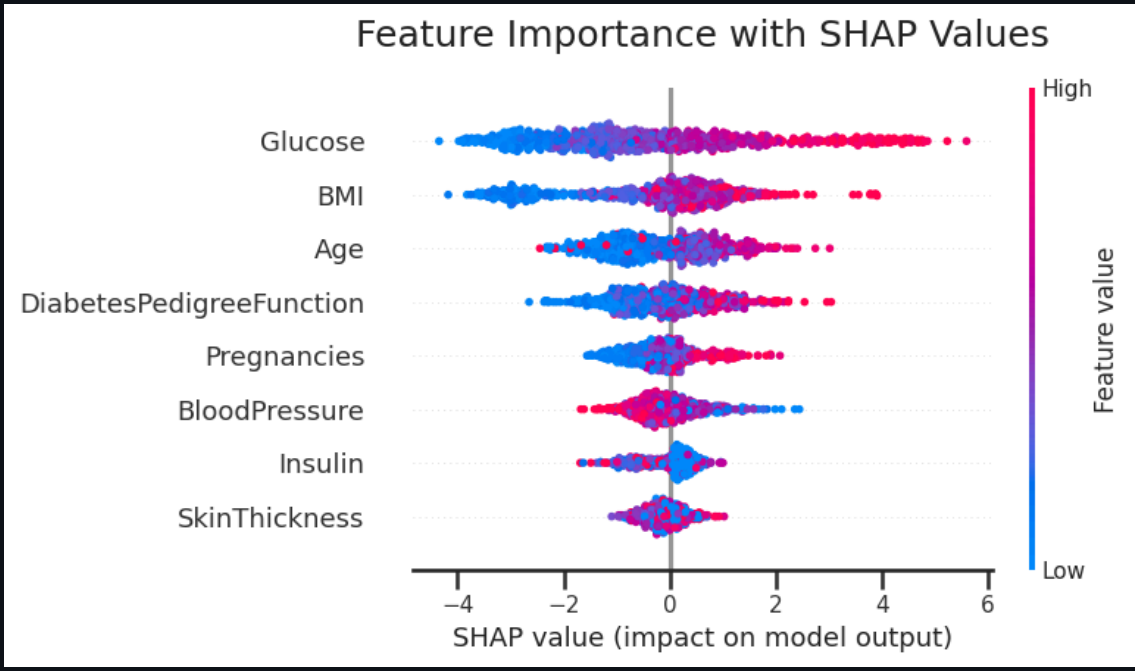
```
    showarrow=False, font=dict(color="white", size=12),
    bgcolor="#FF6F61", opacity=0.7
)

fig.update_layout(
    title="Klein Bottle: Glucose, BMI, Age",
    template='plotly_dark',
    scene=dict(
        xaxis_title="Glucose", yaxis_title="BMI", zaxis_title="Age"
    )
)
fig.show(renderer='iframe_connected')
```

📊 Shap Plots

🔙 **Go to Content**

In [189…

```
model = xgboost.XGBClassifier().fit(df.drop('Outcome', axis=1), df['Outcome'
explainer = shap.Explainer(model)
shap_values = explainer(df.drop('Outcome', axis=1))

plt.figure(figsize=(10, 6))
shap.plots.beeswarm(shap_values, max_display=10, show=False)
plt.title('Feature Importance with SHAP Values', pad=20)
plt.tight_layout()
plt.show()
```

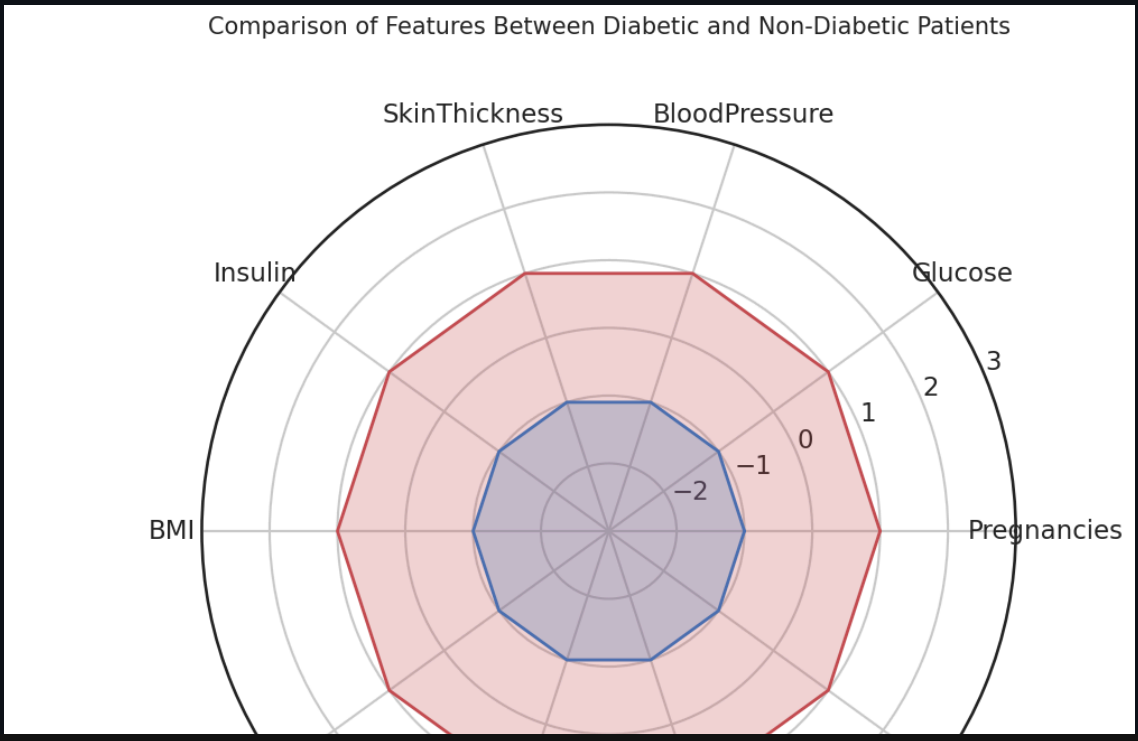


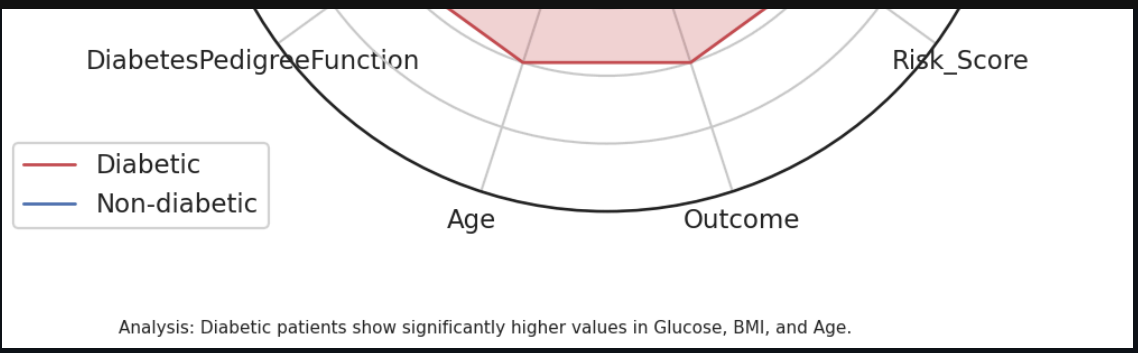
📉 Radar Chart

🔙 **Go to Content**

In [194…

```
numeric_features = df.select_dtypes(include=[np.number]).columns

diabetic = df[df['Outcome'] == 1][numeric_features].mean().values
non_diabetic = df[df['Outcome'] == 0][numeric_features].mean().values


scaler = StandardScaler()
combined = np.vstack((diabetic, non_diabetic))
scaled = scaler.fit_transform(combined)
diabetic_scaled = scaled[0]
non_diabetic_scaled = scaled[1]


angles = np.linspace(0, 2*np.pi, len(numeric_features), endpoint=False).toli
angles += angles[:1]


diabetic_scaled = np.append(diabetic_scaled, diabetic_scaled[0])
non_diabetic_scaled = np.append(non_diabetic_scaled, non_diabetic_scaled[0])


plt.figure(figsize=(12, 10))
ax = plt.subplot(111, polar=True)
ax.plot(angles, diabetic_scaled, 'r', linewidth=2, label='Diabetic')
ax.fill(angles, diabetic_scaled, 'r', alpha=0.25)
ax.plot(angles, non_diabetic_scaled, 'b', linewidth=2, label='Non-diabetic')
ax.fill(angles, non_diabetic_scaled, 'b', alpha=0.25)
ax.set_thetagrids(np.degrees(angles[:-1]), numeric_features)
ax.set_ylim(-3, 3)
ax.grid(True)
plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
plt.title('Comparison of Features Between Diabetic and Non-Diabetic Patients
ax.text(-0.1, -0.15, 'Analysis: Diabetic patients show significantly higher
        transform=ax.transAxes, fontsize=11)
plt.tight_layout()
plt.savefig('radar_plot.png', dpi=300, bbox_inches='tight')
plt.show()
```





Analysis: Diabetic patients show significantly higher values in Glucose, BMI, and Age.

🔮 3D Scatter

🔙 **Go to Content**

In [197…

```
df['AgeGroup'] = pd.cut(df['Age'], bins=[20, 30, 40, 50, 60, 100], labels=['
age_diabetes = df.groupby('AgeGroup')['Outcome'].mean().reset_index()
theta = np.linspace(0, 4 * np.pi, len(age_diabetes))
r = age_diabetes['Outcome'] * 10
z = np.linspace(0, 1, len(theta))

fig = go.Figure(data=[
    go.Scatter3d(
        x=r * np.cos(theta), y=r * np.sin(theta), z=z,
        mode='lines+markers+text',
        line=dict(color='#FF6F61', width=5),
        marker=dict(size=10, color='#6B5B95'),
        text=age_diabetes['AgeGroup'],
        textposition="middle center"
    )
])

max_age_risk = age_diabetes['Outcome'].max()
max_age_group = age_diabetes.loc[age_diabetes['Outcome'].idxmax(), 'AgeGroup
fig.add_annotation(
    text=f"The risk in the {max_age_group} age group: {max_age_risk:.2%}.\nI
    xref="paper", yref="paper", x=0.05, y=0.98,
    showarrow=False, font=dict(color="white", size=12),
    bgcolor="#FF6F61", opacity=0.7
)

fig.update_layout(
    title="Infinity Mirror: Age and Diabetes Rate",
    template='plotly_dark',
    scene=dict(
        xaxis_title="X", yaxis_title="Y", zaxis_title="Diabetes Rate"
    )
)
fig.show(renderer='iframe_connected')
```

🌌 Galaxy Chart

🔙 **Go to Content**

In [198…

```
fig, ax = plt.subplots(figsize=(12, 8), facecolor='#0d0d0d')
x, y = df['Glucose'], df['BMI']
k = kde.gaussian_kde([x, y])
xi, yi = np.mgrid[x.min():x.max():100j, y.min():y.max():100j]
zi = k(np.vstack([xi.flatten(), yi.flatten()]))

ax.contourf(xi, yi, zi.reshape(xi.shape), cmap='inferno', levels=50, alpha=0
ax.scatter(x, y, c=df['Outcome'].map({0: '#00FFFF', 1: '#FF00FF'}), s=20, al
ax.set_facecolor('#0d0d0d')
ax.set_xlabel('Glikoz', fontsize=12, color='white')
ax.set_ylabel('BMI', fontsize=12, color='white')
ax.set_title('Fractal Density Galaxy', fontsize=16, color='white')
plt.grid(False)
plt.show()
```

🪐 Hologram

🔙 **Go to Content**

In [199…

```
output_notebook()
p = figure(width=800, height=400, title="Holographic Wave: Feature Distribut
           background_fill_color="#0d0d0d", border_fill_color="#0d0d0d")

colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9', '#92A8D1']
features = ['Glucose', 'BMI', 'Age', 'Insulin', 'BloodPressure']

for i, feat in enumerate(features):
    hist, edges = np.histogram(df[feat], bins=50, density=True)
```

```
    hist = hist / hist.max() * 0.4 + i * 0.5
    p.patch(edges[:-1], hist, fill_color=colors[i], fill_alpha=0.5, line_col

p.grid.grid_line_color = None
p.axis.axis_line_color = None
p.axis.major_tick_line_color = None
p.title.text_color = "white"
p.xaxis.axis_label = "Value"
p.yaxis.axis_label = "Density (Layered)"
p.xaxis.axis_label_text_color = "white"
p.yaxis.axis_label_text_color = "white"
show(p)
```

Loading BokehJS ...

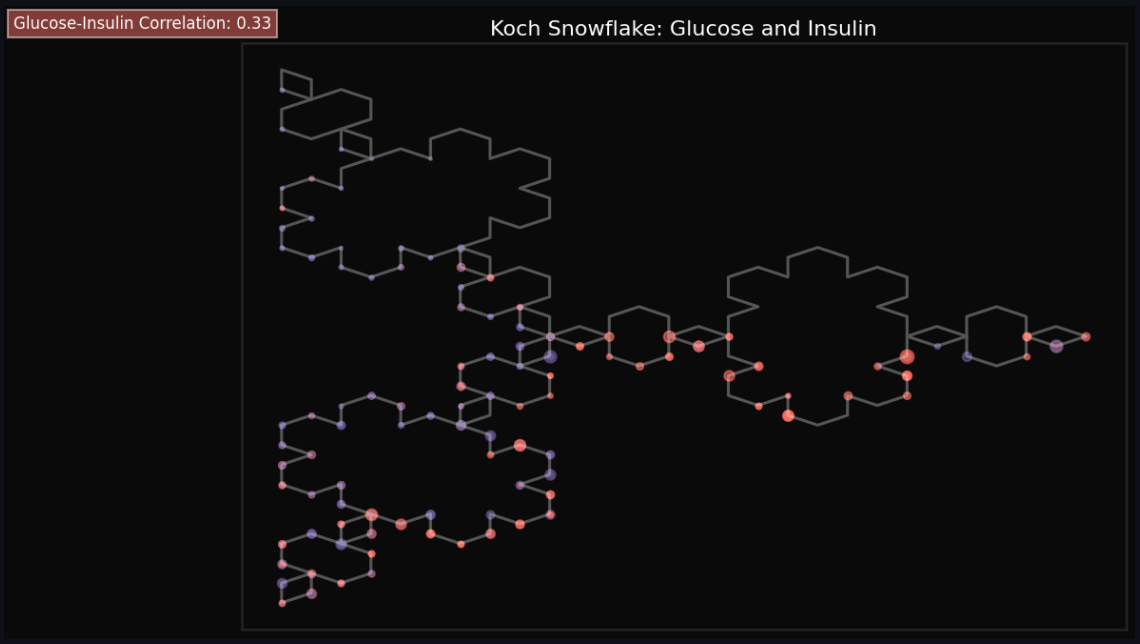
❄️ Koch Snowflake

🔙 **Go to Content**

In [200…

```
def koch_snowflake(n, length):
    def koch_line(start, end, n):
        if n == 0:
            return [start, end]
        delta = end - start
        third = delta / 3
        p1 = start
        p2 = start + third
        p3 = start + third + third * np.exp(1j * np.pi / 3)
        p4 = start + 2 * third
        p5 = end
        return (koch_line(p1, p2, n-1)[:-1] + koch_line(p2, p3, n-1)[:-1] +
                koch_line(p3, p4, n-1)[:-1] + koch_line(p4, p5, n-1))

    points = []
    for i in range(3):
        start = length * np.exp(1j * (2 * np.pi * i / 3))
        end = length * np.exp(1j * (2 * np.pi * (i + 1) / 3))
        points += koch_line(start, end, n)
    return np.array(points)

fig, ax = plt.subplots(figsize=(12, 8), facecolor='#0d0d0d')
koch_points = koch_snowflake(3, 100)
koch_x, koch_y = koch_points.real, koch_points.imag
ax.plot(koch_x, koch_y, color='white', alpha=0.3)


glucose_scaled = (df['Glucose'] / df['Glucose'].max()) * len(koch_points)
insulin_scaled = df['Insulin'] / df['Insulin'].max() * 10
for idx, (g, i, o) in enumerate(zip(glucose_scaled, insulin_scaled, df['Outc
    point_idx = int(g) % len(koch_points)
    x, y = koch_points[point_idx].real, koch_points[point_idx].imag
    ax.scatter(x, y, s=i*10, c='#FF6F61' if o == 1 else '#6B5B95', alpha=0.6

# Analiz metni
insulin_corr = df['Glucose'].corr(df['Insulin'])
ax.text(-100, 100, f"Glucose-Insulin Correlation: {insulin_corr:.2f}", color
        bbox=dict(facecolor='#FF6F61', alpha=0.5))
```

```
ax.set_facecolor('#0d0d0d')
ax.set_title('Koch Snowflake: Glucose and Insulin', fontsize=16, color='whit
ax.set_xticks([]); ax.set_yticks([])
plt.grid(False)
plt.show()
```



🔄 Möbius Strip

🔙 **Go to Content**

In [61]:

```
import plotly.graph_objects as go
import numpy as np


u = np.linspace(0, 2 * np.pi, len(df))
v = np.linspace(-1, 1, 10)
u, v = np.meshgrid(u, v)
x = (1 + v/2 * np.cos(u/2)) * np.cos(u)
y = (1 + v/2 * np.cos(u/2)) * np.sin(u)
z = v/2 * np.sin(u/2)


features = ['Glucose', 'BMI', 'Age']
feature_values = df[features].mean()
u_data = np.linspace(0, 2 * np.pi, len(features))
x_data = (1 + 0.5 * np.cos(u_data/2)) * np.cos(u_data)
y_data = (1 + 0.5 * np.cos(u_data/2)) * np.sin(u_data)
z_data = 0.5 * np.sin(u_data/2)

fig = go.Figure(data=[
    go.Surface(x=x, y=y, z=z, colorscale='Viridis', opacity=0.3),
    go.Scatter3d(
        x=x_data, y=y_data, z=z_data,
        mode='markers+text',
        marker=dict(size=10, color='#FF6F61'),
        text=features
```

```
        text=features,
        textposition="middle center"
    )
])


corr_glucose_bmi = df['Glucose'].corr(df['BMI'])
fig.add_annotation(
    text=f"Glikoz-BMI Korelasyonu: {corr_glucose_bmi:.2f}",
    xref="paper", yref="paper", x=0.05, y=0.95,
    showarrow=False, font=dict(color="white", size=12),
    bgcolor="#FF6F61", opacity=0.7
)

fig.update_layout(
    title="Möbius Strip: Feature Relationships",
    template='plotly_dark',
    scene=dict(
        xaxis_title="X", yaxis_title="Y", zaxis_title="Z"
    )
)
fig.show(renderer='iframe_connected')
```

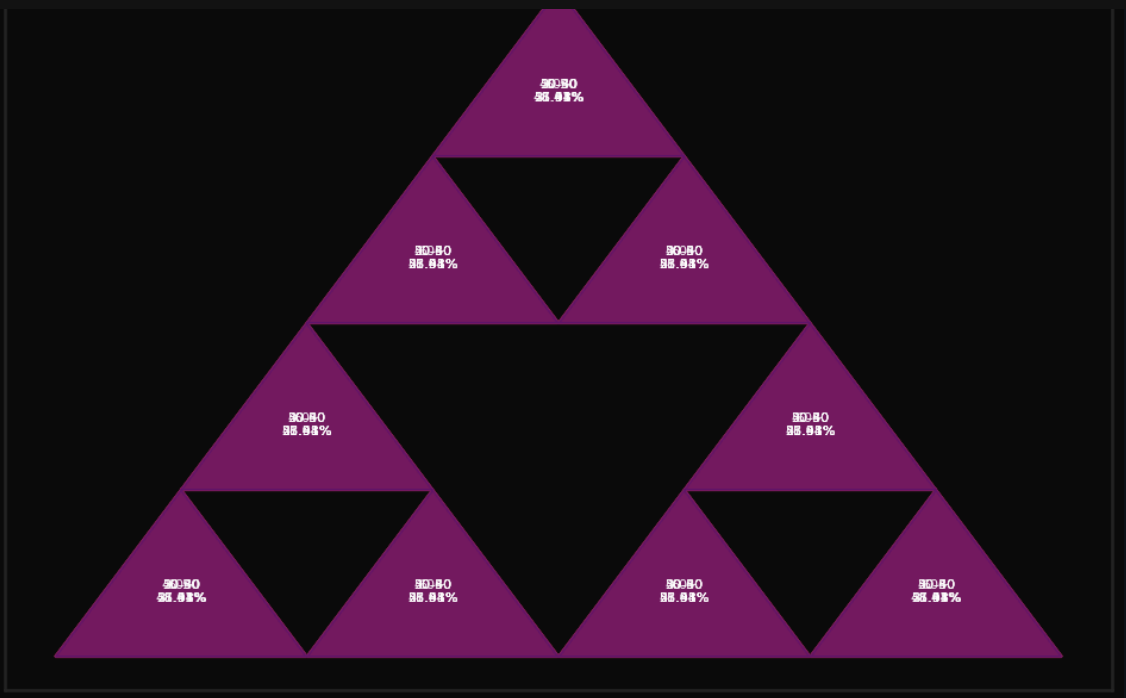
🔺 Sierpinski Triangle

🔙 **Go to Content**

In [62]:

```
def sierpinski_triangle(ax, p1, p2, p3, n, age_group, outcome_ratio):
    if n == 0:
        color = plt.cm.inferno(outcome_ratio)
        ax.fill([p1[0], p2[0], p3[0]], [p1[1], p2[1], p3[1]], color=color, a
        ax.text((p1[0] + p2[0] + p3[0]) / 3, (p1[1] + p2[1] + p3[1]) / 3,
                f"{age_group}\n{outcome_ratio:.2%}", color='white', fontsize
        return
    p12 = (p1 + p2) / 2
    p23 = (p2 + p3) / 2
    p31 = (p3 + p1) / 2
    sierpinski_triangle(ax, p1, p12, p31, n-1, age_group, outcome_ratio)
    sierpinski_triangle(ax, p12, p2, p23, n-1, age_group, outcome_ratio)
    sierpinski_triangle(ax, p31, p23, p3, n-1, age_group, outcome_ratio)

fig, ax = plt.subplots(figsize=(12, 8), facecolor='#0d0d0d')
p1, p2, p3 = np.array([0, 0]), np.array([100, 0]), np.array([50, 86.6])

for age_group in df['AgeGroup'].unique():
    outcome_ratio = df[df['AgeGroup'] == age_group]['Outcome'].mean()
    sierpinski_triangle(ax, p1, p2, p3, 2, age_group, outcome_ratio)

ax.set_facecolor('#0d0d0d')
ax.set_title('Sierpinski Triangle: Diabetes Rate in Age Groups', fontsize=16
ax.set_xticks([]); ax.set_yticks([])
plt.grid(False)
plt.show()
```

Sierpinski Üçgeni: Yaş Gruplarında Diyabet Oranı

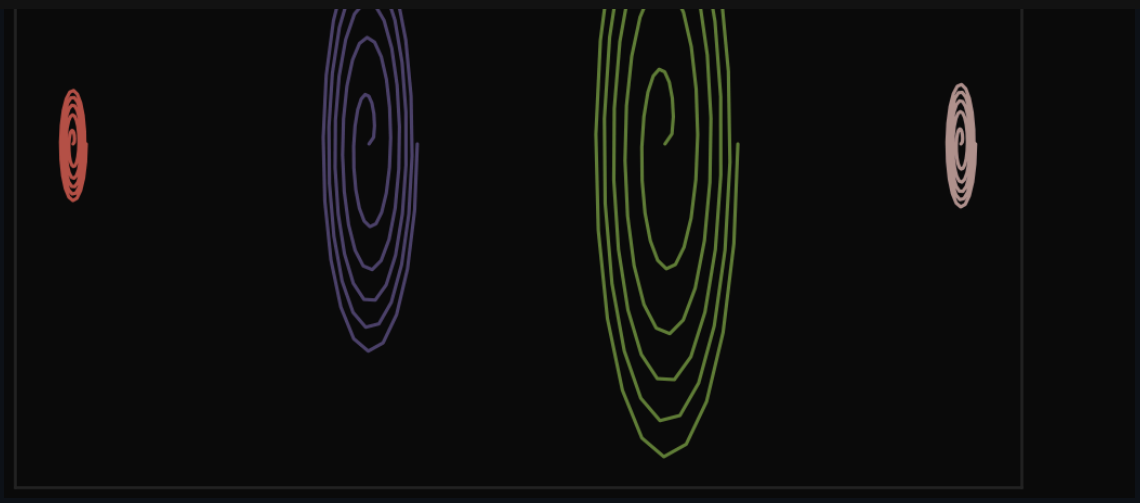


🌐 Fibonacci Spiral Network

🔙 **Go to Content**

In [63]:

```
def fibonacci_spiral(n, scale):
    phi = (1 + np.sqrt(5)) / 2
    theta = np.linspace(0, n * np.pi, 100)
    r = scale * np.sqrt(theta)
    return r * np.cos(theta), r * np.sin(theta)

fig, ax = plt.subplots(figsize=(12, 8), facecolor='#0d0d0d')
features = ['Glucose', 'BMI', 'Age', 'Insulin']
colors = ['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9']

for i, feat in enumerate(features):
    density, bins = np.histogram(df[feat], bins=30, density=True)
    scale = density.max() * 100
    x, y = fibonacci_spiral(10, scale)
    ax.plot(x + i * 200, y, color=colors[i], alpha=0.7)
    ax.text(i * 200, 50, f"{feat} Yoğunluk: {density.max():.2f}", color='whi
            bbox=dict(facecolor=colors[i], alpha=0.5))

ax.set_facecolor('#0d0d0d')
ax.set_title('Fibonacci Spiral Network: Feature Densities', fontsize=16, col
ax.set_xticks([]); ax.set_yticks([])
plt.grid(False)
plt.show()
```



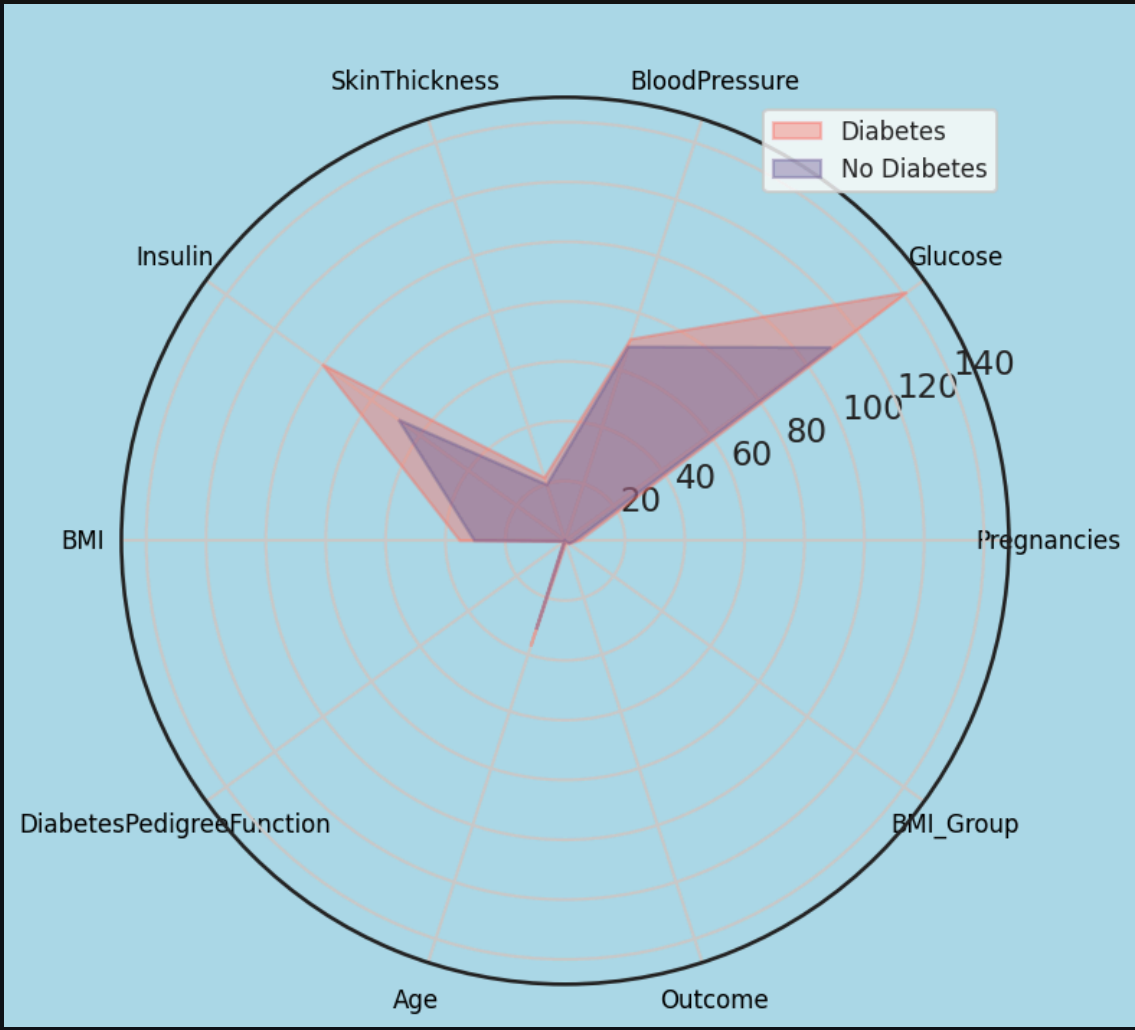
📊 Radar Chart

🔙 **Go to Content**

In [237…

```
for col in df.columns[:-1]:
    if df[col].dtype.name == 'category':
        df[col] = pd.factorize(df[col])[0]

categories = df.columns[:-1]
diabetes = df[df['Outcome'] == 1][categories].mean()
no_diabetes = df[df['Outcome'] == 0][categories].mean()

# Radar chart data preparation
angles = [n / float(len(categories)) * 2 * np.pi for n in range(len(categori
angles += angles[:1]


fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))


ax.fill(angles, diabetes.tolist() + diabetes.tolist()[:1], color='#FF6F61',
ax.fill(angles, no_diabetes.tolist() + no_diabetes.tolist()[:1], color='#6B5


ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories, fontsize=12, color='black')


plt.title('Average Characteristics According to Diabetes Status', size=16, c


plt.legend(loc='upper right', fontsize=12)


fig.patch.set_facecolor('#ADD8E6')
ax.set_facecolor('#ADD8E6')


plt.show()
```

Average Characteristics According to Diabetes Status



🌀 Cosmic Flow

🔙 **Go to Content**

In [74]:

```
df['AgeGroup'] = pd.cut(df['Age'], bins=[20, 30, 40, 50, 60, 100], labels=['
age_groups = df['AgeGroup'].value_counts().index
labels = list(age_groups) + ['no Diabetes', 'Diabetes']
source, target, value = [], [], []

for i, age in enumerate(age_groups):
    no_diabetes = len(df[(df['AgeGroup'] == age) & (df['Outcome'] == 0)])
    yes_diabetes = len(df[(df['AgeGroup'] == age) & (df['Outcome'] == 1)])
    source.extend([i, i])
    target.extend([len(age_groups), len(age_groups) + 1])
    value.extend([no_diabetes, yes_diabetes])

fig = go.Figure(data=[go.Sankey(
    node=dict(
        pad=15,
        thickness=20,
        line=dict(color="white", width=0.5),
        label=labels,
        color=['#FF6F61', '#6B5B95', '#88B04B', '#F7CAC9', '#92A8D1', '#FFCC
    ),
    link=dict(
        source=source,
```

```
        target=target,
        value=value,
        color=[f'rgba({np.random.randint(100, 255)}, {np.random.randint(100,
    )
)])

fig.update_layout(title_text="Cosmic Flow: Age Groups and Diabetes", font_si
fig.show(renderer='iframe_connected')
```

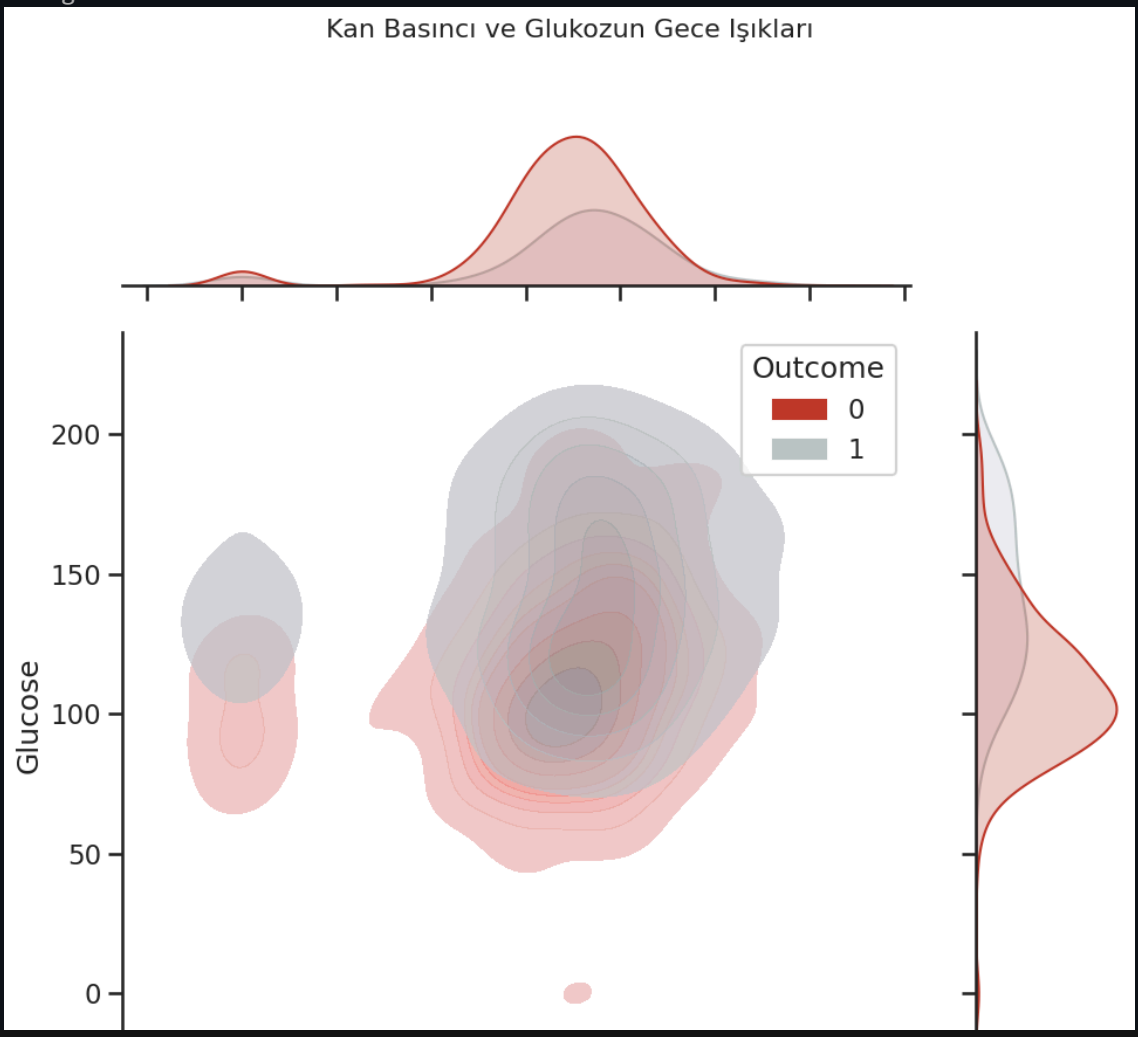
🖇️ Joinplot

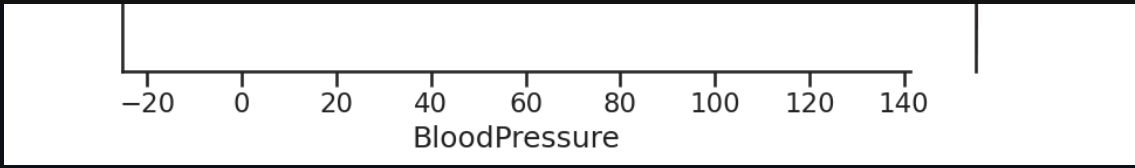
🔙 **Go to Content**

In [207…

```
plt.figure(figsize=(14, 10))
sns.jointplot(data=df, x='BloodPressure', y='Glucose', hue='Outcome',
              palette={0: '#C0392B', 1: '#BDC3C7'}, height=10, ratio=4,
              kind='kde', fill=True, alpha=0.7,
              joint_kws={'linewidths': 1, 'edgecolor': 'w'})

plt.suptitle('Night Lights for Blood Pressure and Glucose', y=1.02, fontsize
plt.tight_layout()
plt.show()
```

```
<Figure size 1400x1000 with 0 Axes>
```



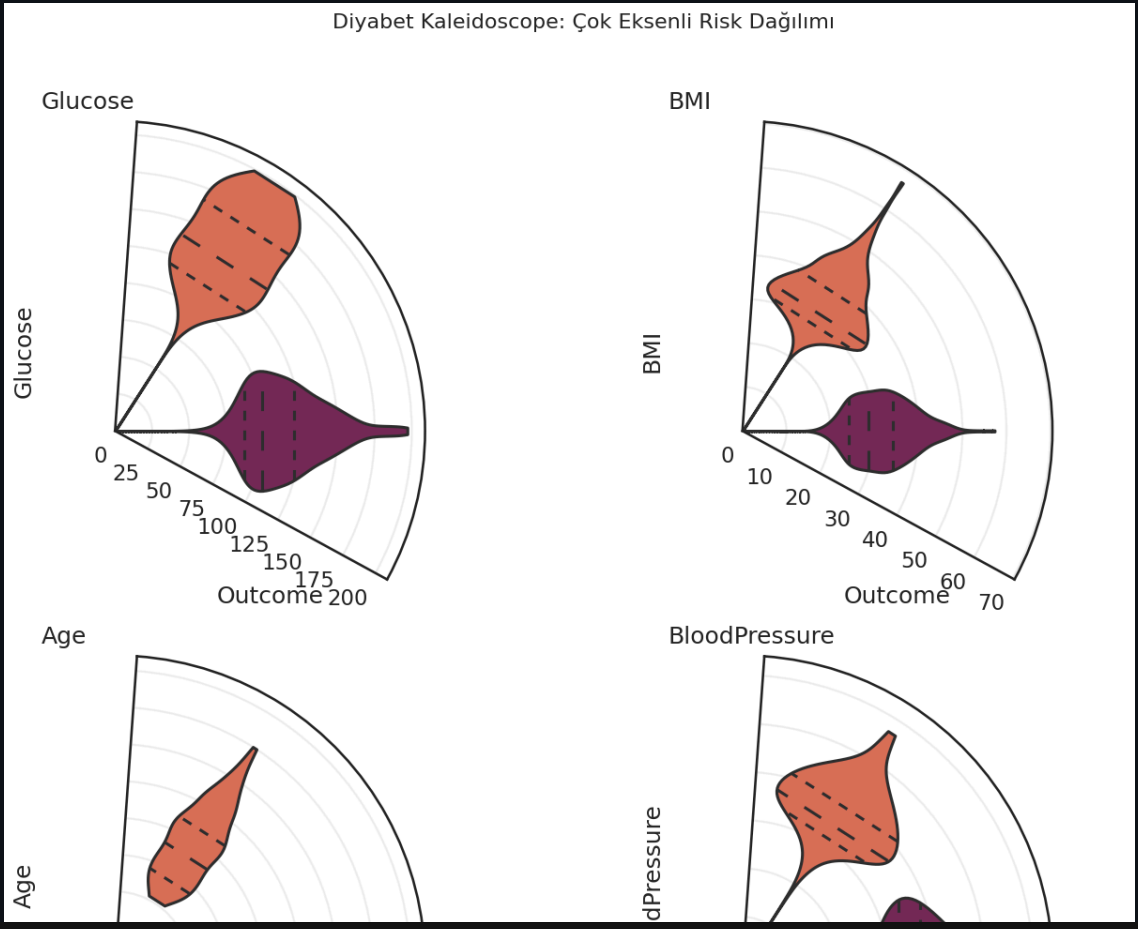


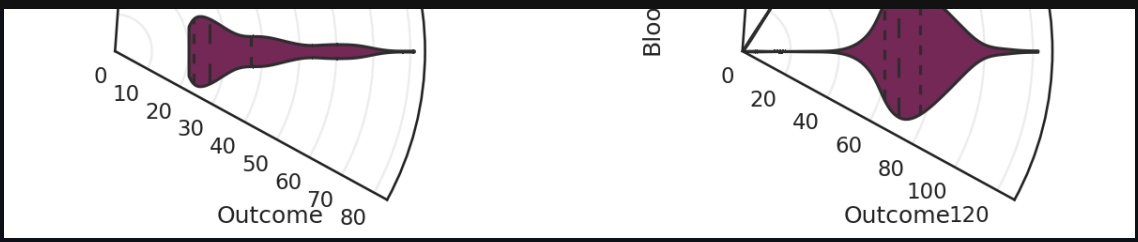
🎻 Violinplot

🔙 **Go to Content**

In [154…

```
features = ['Glucose', 'BMI', 'Age', 'BloodPressure']
n_features = len(features)


fig = plt.figure(figsize=(14, 12))

for i, feature in enumerate(features):
    ax = plt.subplot(2, 2, i+1, polar=True)
    sns.violinplot(x=df['Outcome'], y=df[feature], palette='rocket',
                   inner='quartile', cut=0, ax=ax)
    ax.set_xticks([])
    ax.set_title(feature, loc='left', pad=10)
    ax.grid(alpha=0.3)


plt.suptitle('Diabetes Kaleidoscope: Multiaxial Risk Distribution',
             y=1.02, fontsize=16)
plt.tight_layout()
plt.show()
```





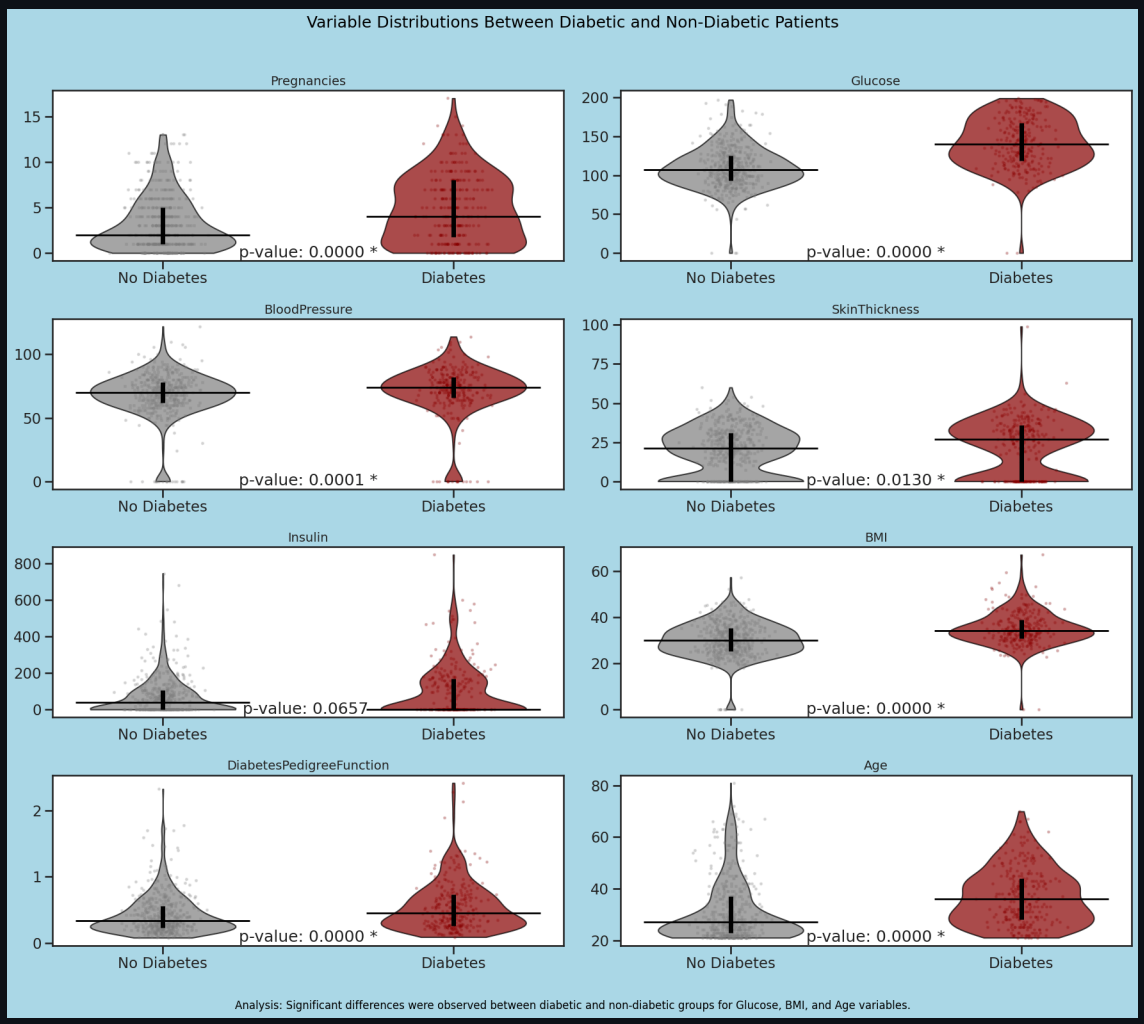
Violinplot

Go Content

In [233…

```
features = df.columns[:-1][:8]
n_features = len(features)

n_cols = 2
n_rows = int(np.ceil(n_features / n_cols))

# Set the background color to light blue
plt.figure(figsize=(18, n_rows * 4), facecolor='#ADD8E6')  # Light blue back

gs = gridspec.GridSpec(n_rows, n_cols)

for i, feature in enumerate(features):
    ax = plt.subplot(gs[i // n_cols, i % n_cols])

    # Violin plot
    parts = ax.violinplot([df[df['Outcome'] == 0][feature], df[df['Outcome']
                           showmeans=False, showmedians=False, showextrema=Fa

    # Colors: Grey for non-diabetic, Dark Red for diabetic
    colors = ['#808080', '#8B0000']  # Gray and Dark Red
    for j, pc in enumerate(parts['bodies']):
        pc.set_facecolor(colors[j])  # Apply gray and dark red colors to the
        pc.set_edgecolor('black')
        pc.set_alpha(0.7)


    quartile1 = [np.percentile(data, 25) for data in [df[df['Outcome'] == 0]
    medians = [np.percentile(data, 50) for data in [df[df['Outcome'] == 0][f
    quartile3 = [np.percentile(data, 75) for data in [df[df['Outcome'] == 0]

    for j, (q1, m, q3) in enumerate(zip(quartile1, medians, quartile3)):
        ax.hlines(m, j + 0.7, j + 1.3, color='black', linestyle='-', lw=2)
        ax.vlines(j + 1, q1, q3, color='black', linestyle='-', lw=5)


    for j, group in enumerate([df[df['Outcome'] == 0][feature], df[df['Outco
        jitter = np.random.normal(j + 1, 0.05, size=len(group))
        ax.scatter(jitter, group, c=colors[j], alpha=0.2, s=5)

    ax.set_xticks([1, 2])
    ax.set_xticklabels(['No Diabetes', 'Diabetes'])
    ax.set_title(f'{feature}', fontsize=14)


    stat, p = stats.mannwhitneyu(df[df['Outcome'] == 0][feature], df[df['Out
    ax.text(0.5, 0.02, f'p-value: {p:.4f} {"*" if p < 0.05 else ""}', transf


plt.suptitle('Variable Distributions Between Diabetic and Non-Diabetic Patie
plt.figtext(0.5, 0.01, 'Analysis: Significant differences were observed betw
```

```
            ha='center', fontsize=12, color='black')

plt.tight_layout(rect=[0, 0.03, 1, 0.97])
plt.savefig('violin_plots.png', dpi=300, bbox_inches='tight')
plt.show()
```



🌹 Nightingale Rose Chart

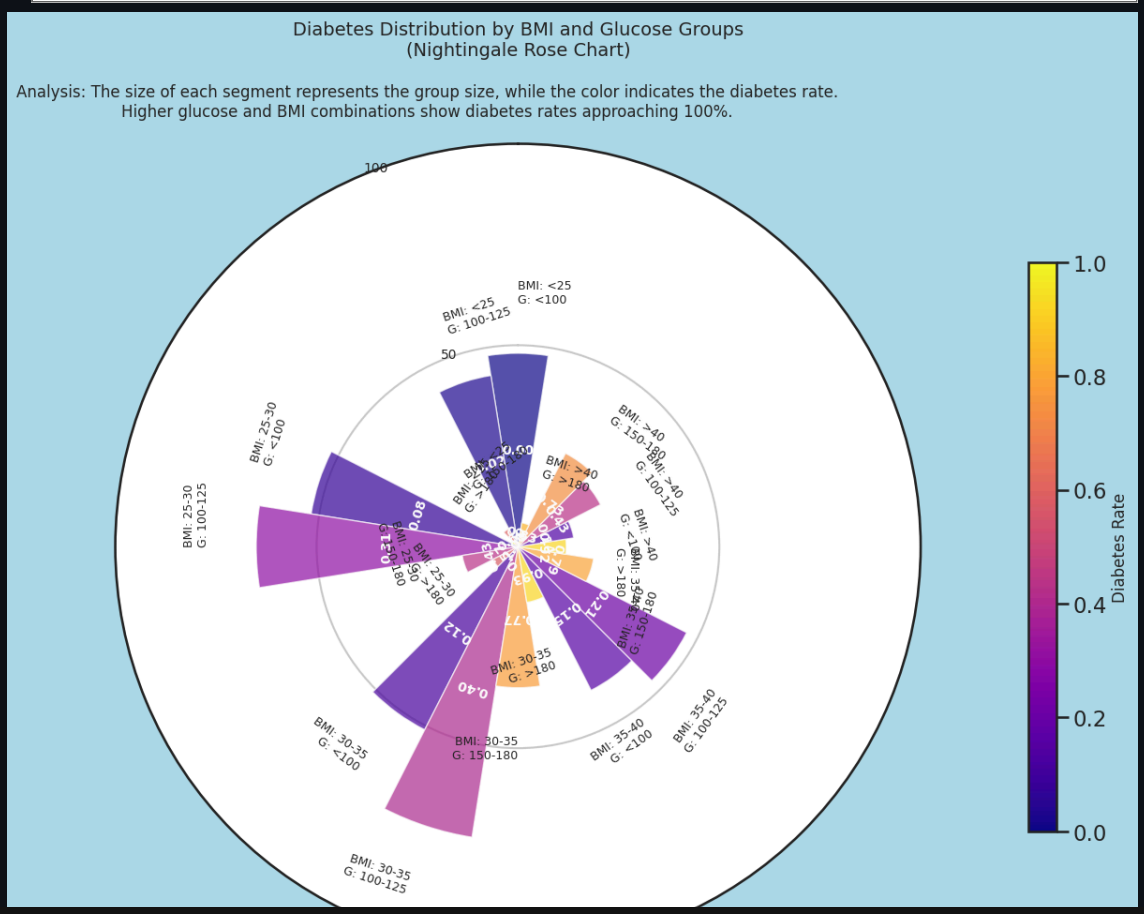
🔙 **Go to Content**

In [234…

```
bmi_bins = [0, 25, 30, 35, 40, 100]
glucose_bins = [0, 100, 125, 150, 180, 300]

bmi_labels = ['<25', '25-30', '30-35', '35-40', '>40']
glucose_labels = ['<100', '100-125', '125-150', '150-180', '>180']

df['BMI_Group'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels)
df['Glucose_Group'] = pd.cut(df['Glucose'], bins=glucose_bins, labels=glucos


cross_tab = pd.crosstab(
    [df['BMI_Group'], df['Glucose_Group']],
    df['Outcome'],
    normalize='index'
)
cross_tab['diabetic_rate'] = cross_tab[1]
```

```
cross_tab['sample_size'] = pd.crosstab([df['BMI_Group'], df['Glucose_Group']
cross_tab = cross_tab.reset_index()


cross_tab = cross_tab[cross_tab['Glucose_Group'] != '125-150']


fig = plt.figure(figsize=(12, 10), facecolor='#ADD8E6')
ax = fig.add_subplot(111, polar=True)

n_groups = len(cross_tab)


angles = np.linspace(0, 2 * np.pi, n_groups, endpoint=False).tolist()
width = 2 * np.pi / n_groups


colors = cm.plasma(cross_tab['diabetic_rate'].values)


bars = ax.bar(
    angles,
    cross_tab['sample_size'],
    width=width,
    color=colors,
    alpha=0.7,
    bottom=0.0,
    linewidth=1,
    edgecolor='white'
)


for bar, angle, rate in zip(bars, angles, cross_tab['diabetic_rate']):
    height = bar.get_height()
    rotation = np.degrees(angle)
    alignment = 'center'

    ax.text(
        angle,
        height * 0.5,
        f"{rate:.2f}",
        ha=alignment,
        va='center',
        rotation=rotation,
        rotation_mode="anchor",
        fontsize=10,
        fontweight='bold',
        color='white'
    )


for i, (bar, angle) in enumerate(zip(bars, angles)):
    height = bar.get_height()
    bmi = cross_tab.iloc[i]['BMI_Group']
    glucose = cross_tab.iloc[i]['Glucose_Group']

    rotation = np.degrees(angle)
    if angle > np.pi/2 and angle < 3*np.pi/2:
        alignment = 'right'
        rotation += 180
    else:
        alignment = 'left'

    ax.text(
```

```
        angle,
        height + 15,
        f"BMI: {bmi}\nG: {glucose}",
        ha=alignment,
        va='center',
        rotation=rotation,
        rotation_mode="anchor",
        fontsize=9
    )


ax.set_theta_zero_location("N")
ax.set_xticks([])


ax.set_yticks([50, 100])
ax.set_yticklabels(['50', '100'], fontsize=10)


sm = plt.cm.ScalarMappable(cmap=cm.plasma, norm=plt.Normalize(vmin=0, vmax=1
sm._A = []
cbar = plt.colorbar(sm, ax=ax, pad=0.1, shrink=0.7)
cbar.set_label('Diabetes Rate', fontsize=12)


plt.title('Diabetes Distribution by BMI and Glucose Groups\n(Nightingale Ros
plt.figtext(0.3, 0.92, 'Analysis: The size of each segment represents the gr
           ha='center', fontsize=12)


plt.tight_layout()


plt.savefig('nightingale_chart.png', dpi=300, bbox_inches='tight')
plt.show()
```



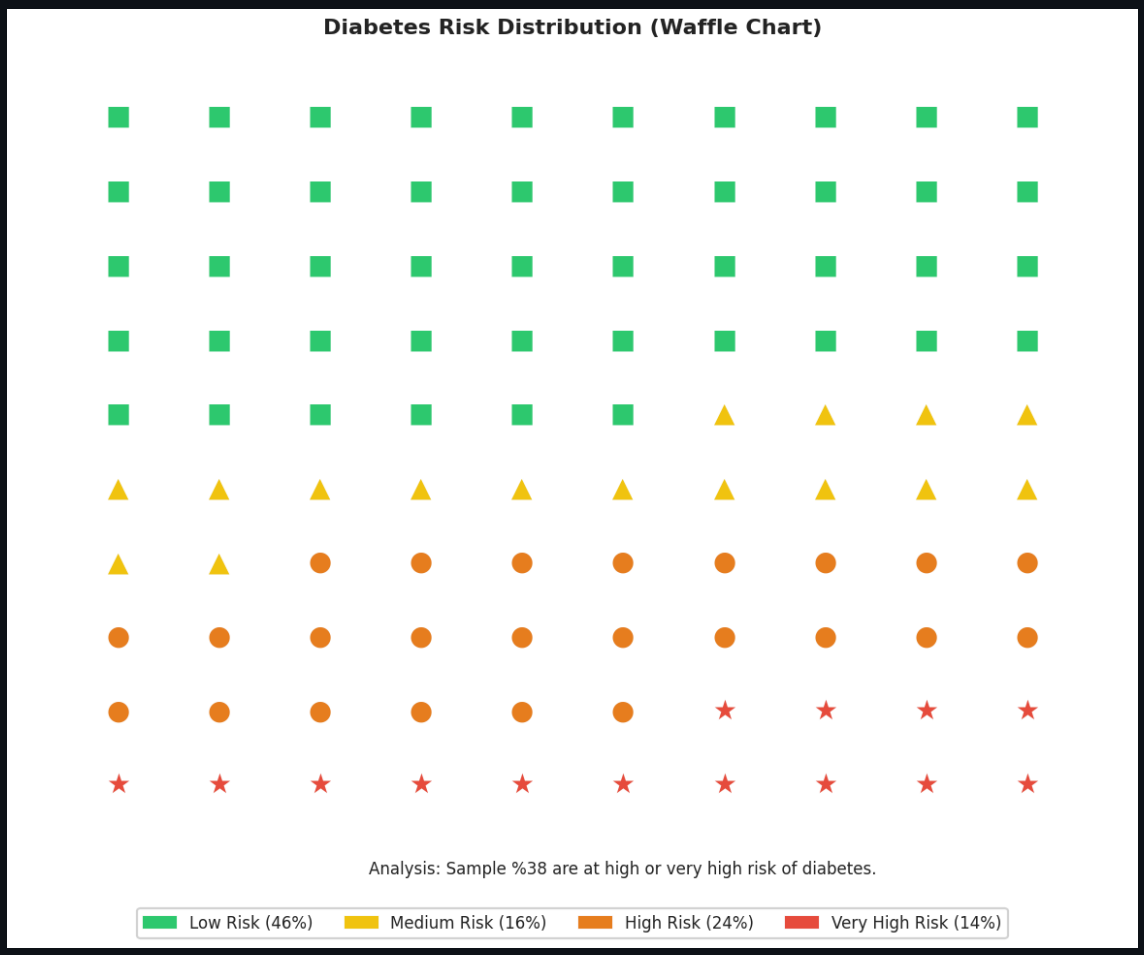
🧇 Waffle Chart

🔙 **Go to Content**

In [218…

```
def calculate_risk_score(row):
    score = 0
    if row['Glucose'] > 125: score += 30
    if row['BMI'] > 30: score += 20
    if row['Age'] > 40: score += 15
    if row['DiabetesPedigreeFunction'] > 0.8: score += 15
    if row['BloodPressure'] > 90: score += 10
    if row['Pregnancies'] > 6: score += 10
    return score

df['Risk_Score'] = df.apply(calculate_risk_score, axis=1)


df['Risk_Level'] = pd.cut(df['Risk_Score'],
                          bins=[0, 20, 40, 60, 100],
                          labels=['Low', 'Medium', 'High', 'Very High'])


risk_counts = df['Risk_Level'].value_counts().sort_index()


total = len(df)
categories = risk_counts.index.tolist()
values = risk_counts.values


cell_counts = [int(round(value / total * 100)) for value in values]


remainder = 100 - sum(cell_counts)
if remainder > 0:
    max_idx = cell_counts.index(max(cell_counts))
    cell_counts[max_idx] += remainder

colors = ['#2ecc71', '#f1c40f', '#e67e22', '#e74c3c']
symbols = ['■', '▲', '●', '★']


x = [i % 10 for i in range(100)]
y = [9 - (i // 10) for i in range(100)]


plt.figure(figsize=(12, 10))
start = 0
for i, (category, count) in enumerate(zip(categories, cell_counts)):
    end = start + count
    category_x = x[start:end]
    category_y = y[start:end]

    for x_pos, y_pos in zip(category_x, category_y):
        plt.text(x_pos, y_pos, symbols[i], fontsize=20, ha='center', va='cer
    start = end
```

```
legend_elements = [
    mpatches.Patch(color=colors[i], label=f'{cat} Risk ({cell_counts[i]}%)')
    for i, cat in enumerate(categories)
]

plt.legend(handles=legend_elements, loc='upper center', bbox_to_anchor=(0.5,
           frameon=True, ncol=4, fontsize=12)


plt.title('Diabetes Risk Distribution (Waffle Chart)', fontsize=16, fontweig
plt.text(5, -1.2, f"Analysis: Sample %{cell_counts[2] + cell_counts[3]} are
         ha='center', fontsize=12)


plt.xlim(-1, 10)
plt.ylim(-1, 10)
plt.axis('off')
plt.tight_layout()
plt.savefig('risk_waffle_chart.png', dpi=300, bbox_inches='tight')
plt.show()
```



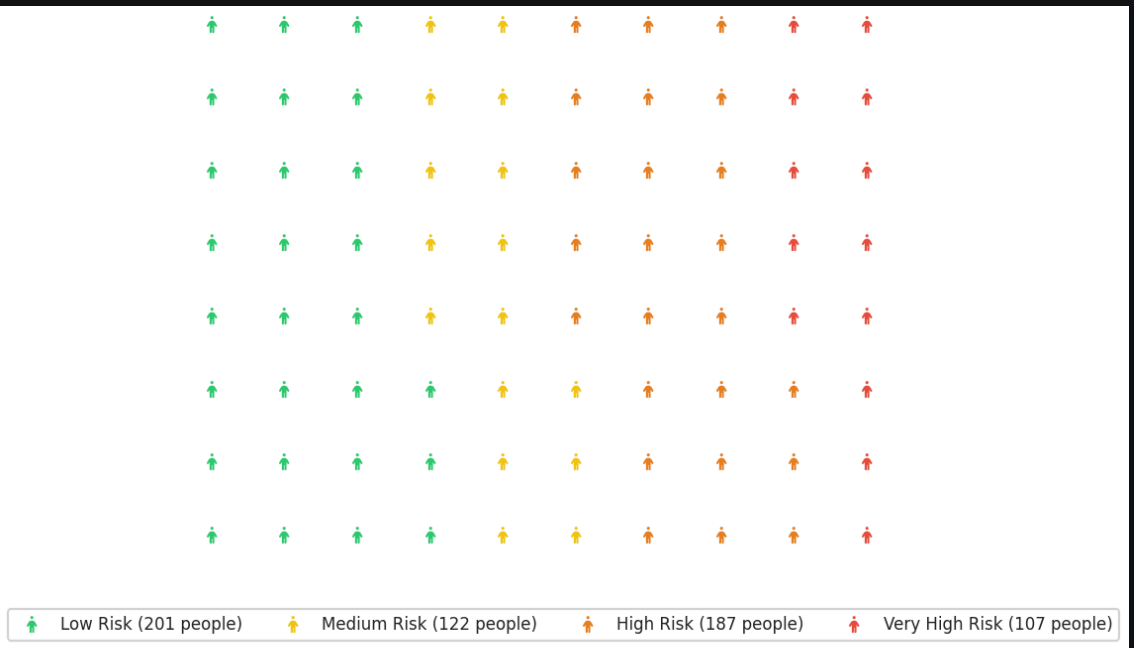
PyWaffle

Go Content

In [209…

```
def calculate_risk_score(row):
    score = 0
    if row['Glucose'] > 125: score += 30
    if row['BMI'] > 30: score += 20
```

```
    if row['Age'] > 40: score += 15
    if row['DiabetesPedigreeFunction'] > 0.8: score += 15
    if row['BloodPressure'] > 90: score += 10
    if row['Pregnancies'] > 6: score += 10
    return score

df['Risk_Score'] = df.apply(calculate_risk_score, axis=1)
df['Risk_Level'] = pd.cut(df['Risk_Score'],
                          bins=[0, 20, 40, 60, 100],
                          labels=['Low', 'Medium', 'High', 'Very High'])


risk_counts = df['Risk_Level'].value_counts().sort_index()
risk_data = {category: value for category, value in zip(risk_counts.index, r


colors = {
    'Low': '#2ecc71',
    'Medium': '#f1c40f',
    'High': '#e67e22',
    'Very High': '#e74c3c'
}

fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=10,
    values=risk_data,
    colors=[colors[k] for k in risk_data.keys()],
    title={
        'label': 'Diabetes Risk Distribution (PyWaffle)',
        'loc': 'center',
        'fontsize': 18
    },
    legend={
        'labels': [f"{k} Risk ({v} people)" for k, v in risk_data.items()],
        'loc': 'lower center',
        'bbox_to_anchor': (0.5, -0.15),
        'ncol': 4,
        'framealpha': 0.8,
        'fontsize': 12
    },
    icons='person',
    icon_size=12,
    icon_legend=True,
    figsize=(12, 9)
)

# Analysis text
plt.figtext(0.5, -0.05,
           f"Analysis: In the sample group, each square represents a patient
           ha='center', fontsize=12, bbox=dict(facecolor='white', alpha=0.8,

plt.tight_layout()
plt.savefig('pywaffle_chart.png', dpi=300, bbox_inches='tight')
plt.show()
```



Analysis: In the sample group, each square represents a patient. 294 patients (%47.6) are in the high or very high-risk category.

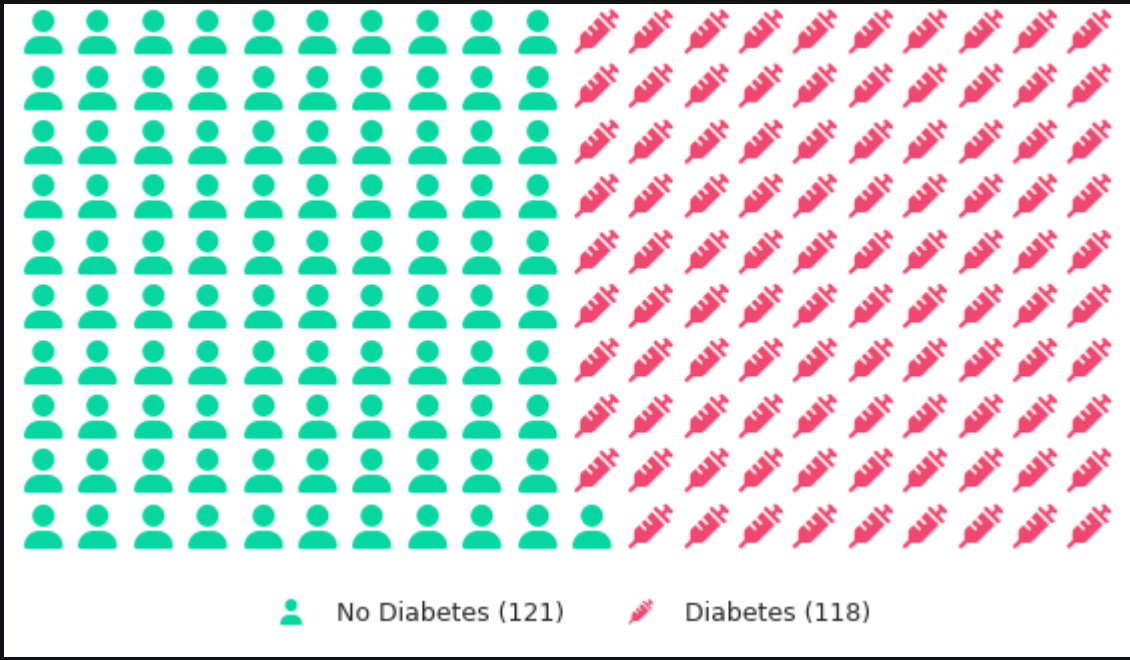
PyWaffle

Go Content

In [211…

```
bins = [20, 30, 45, 60, df['Age'].max() + 1]
labels = ['20-30', '31-45', '46-60', '60+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# 31-45 yaş grubu
group1 = '31-45'
data1 = df[df['Age_Group'] == group1]['Outcome'].value_counts().sort_index()

# 31-45 yaş grubu için Waffle grafiği
fig1 = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=20,
    figsize=(6, 5),
    values=data1.tolist(),
    title={'label': f'{group1} Age group', 'loc': 'center'},
    labels=[f"No Diabetes ({data1.get(0, 0)})", f"Diabetes ({data1.get(1, 0)
    colors=['#06D6A0', '#EF476F'],
    icons=['user', 'syringe'],
    icon_style='solid',
    font_size=16,
    icon_legend=True,
    legend={
        'loc': 'lower center',
        'bbox_to_anchor': (0.5, -0.2),
        'ncol': 2,
        'framealpha': 0,
        'fontsize': 9
    }
)
plt.show()
```

31-45 Age group



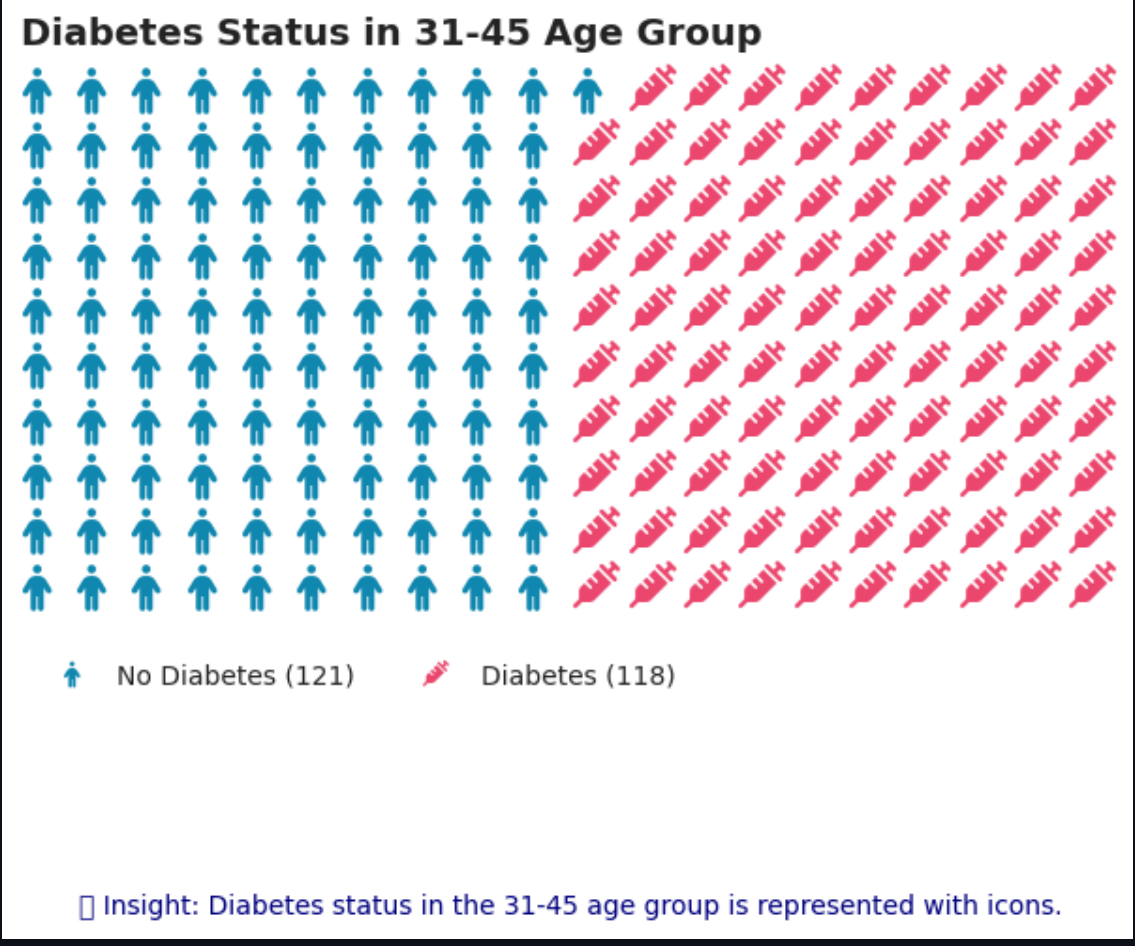
PyWaffle

Go Content

In [212…

```
bins = [20, 30, 45, 60, df['Age'].max()]
labels = ['20-30', '31-45', '46-60', '60+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)


age_group = '31-45'
age_group_data = df[df['Age_Group'] == age_group]
values = age_group_data['Outcome'].value_counts().sort_index()


fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=20,
    values=values.tolist(),
    colors=('#118AB2', '#EF476F'),
    title={
        'label': f'Diabetes Status in {age_group} Age Group',
        'loc': 'left',
        'fontdict': {'fontsize': 14, 'weight': 'bold'}
    },
    icons=['person', 'syringe'],
    icon_style='solid',
    font_size=18,
    icon_legend=True,
    labels=[f"No Diabetes ({values.get(0, 0)})", f"Diabetes ({values.get(1,
    legend={
        'loc': 'lower left',
        'bbox_to_anchor': (0, -0.2),
        'ncol': 2,
        'framealpha': 0,
        'fontsize': 10
    },
    starting_location='NW',
    block_arranging_style='snake',
)
```

```
fig.text(
    0.5, -0.12,
    f"💡 Insight: Diabetes status in the {age_group} age group is represente
    ha='center', fontsize=10, color='navy'
)

plt.tight_layout(rect=[0, 0.05, 1, 1])
plt.show()
```



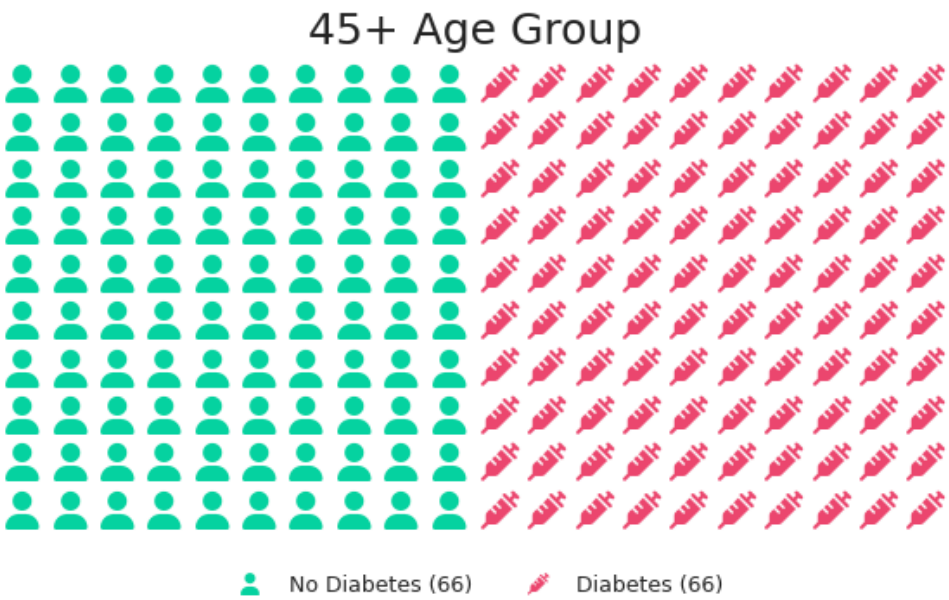
PyWaffle

Go Content

In [213…

```
data2 = df[df['Age_Group'].isin(['46-60', '60+'])]['Outcome'].value_counts()

fig2 = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=20,
    figsize=(6, 5),
    values=data2.tolist(),
    title={'label': '45+ Age Group', 'loc': 'center'},
    labels=[f"No Diabetes ({data2.get(0, 0)})", f"Diabetes ({data2.get(1, 0)
    colors=['#06D6A0', '#EF476F'],
    icons=['user', 'syringe'],
    icon_style='solid',
    font_size=16,
    icon_legend=True,
    legend={
        'loc': 'lower center',
        'bbox_to_anchor': (0.5, -0.2)
```

```
        'ncol': 2,
        'framealpha': 0,
        'fontsize': 9
    }
)

plt.figtext(
    0.5, 0.01,
    "💡 Insight: Each icon represents an individual. The colors and icons in
    ha='center', fontsize=10, color='navy'
)
plt.show()
```



💡 Insight: Each icon represents an individual. The colors and icons indicate diabetes status.

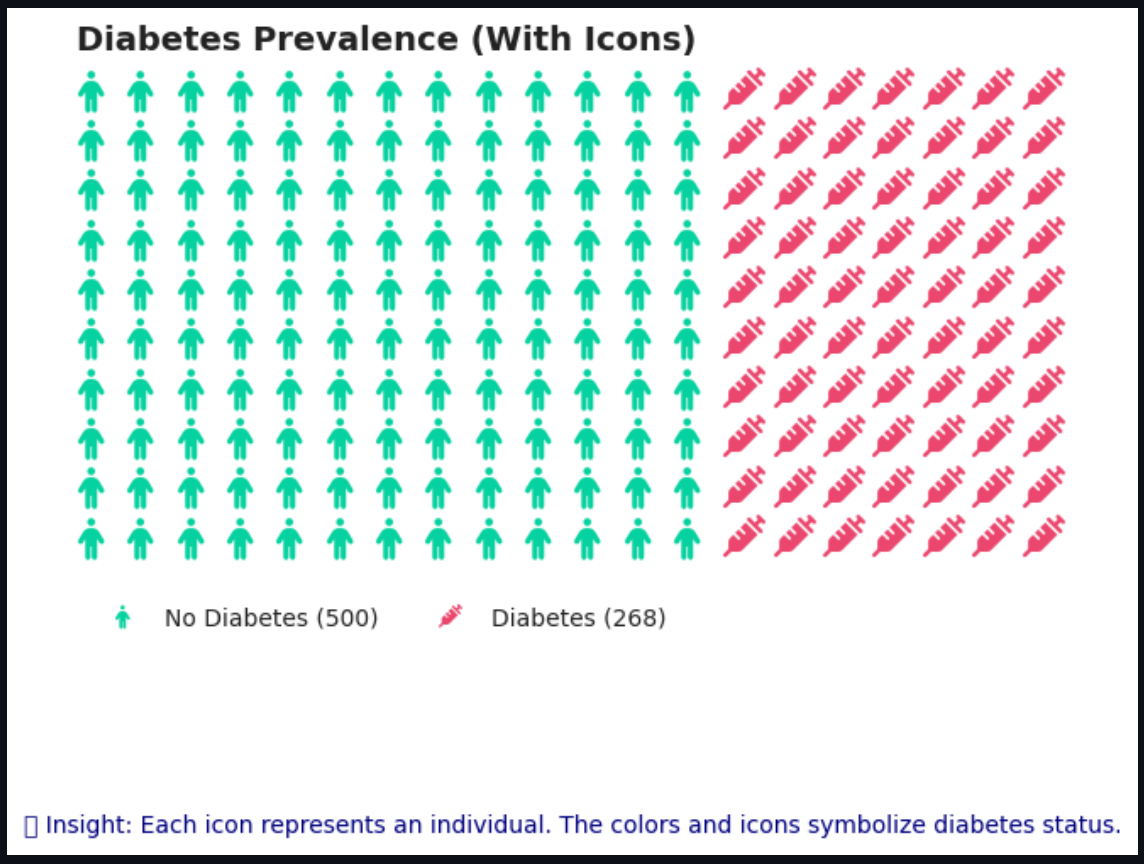
PyWaffle

Go Content

In [214…

```
outcome_counts = [500, 268]
labels = ['No Diabetes', 'Diabetes']

fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=20,
    values=outcome_counts,
    colors=('#06D6A0', '#EF476F'),
    title={
        'label': 'Diabetes Prevalence (With Icons)',
        'loc': 'left',
        'fontdict': {'fontsize': 14, 'weight': 'bold'}
    },
    icons=['person', 'syringe'],
    icon_style='solid',
    font_size=18,
    icon_legend=True,
    labels=[f"{label} ({count})" for label, count in zip(labels, outcome_cou
    legend={
```

```
        'loc': 'lower left',
        'bbox_to_anchor': (0, -0.2),
        'ncol': 2,
        'framealpha': 0,
        'fontsize': 10
    },
    starting_location='NW',
    block_arranging_style='snake',
)


fig.text(
    0.5, -0.12,
    "💡 Insight: Each icon represents an individual. The colors and icons sy
    ha='center', fontsize=10, color='navy'
)

plt.tight_layout(rect=[0, 0.05, 1, 1])
plt.show()
```



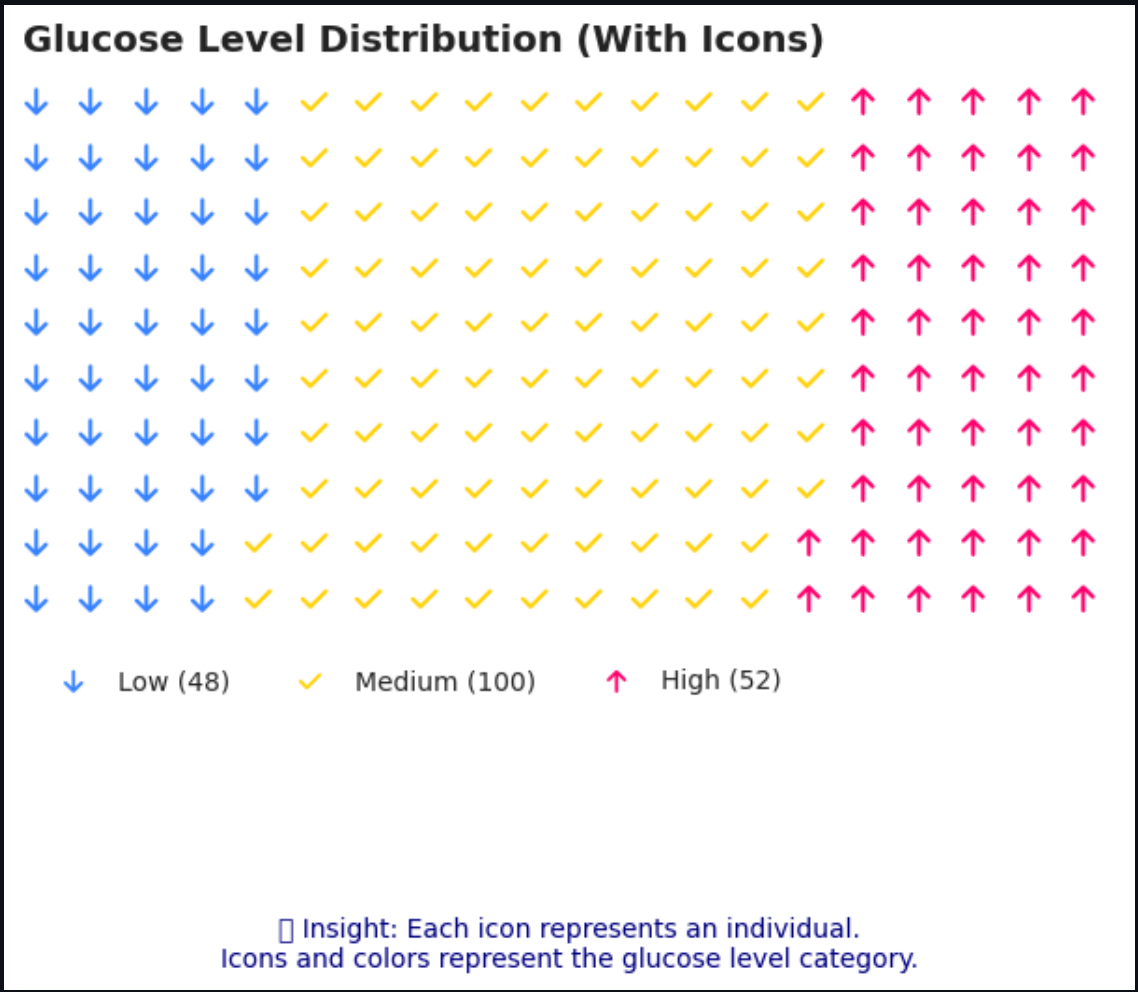
PyWaffle

Go Content

In [215…

```
plot_values = [48, 100, 52]
cat_order = ['Low', 'Medium', 'High']

fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=20,
    values=plot_values,
    colors=('#3A86FF', '#FFD60A', '#FF006E'),
    title={
```

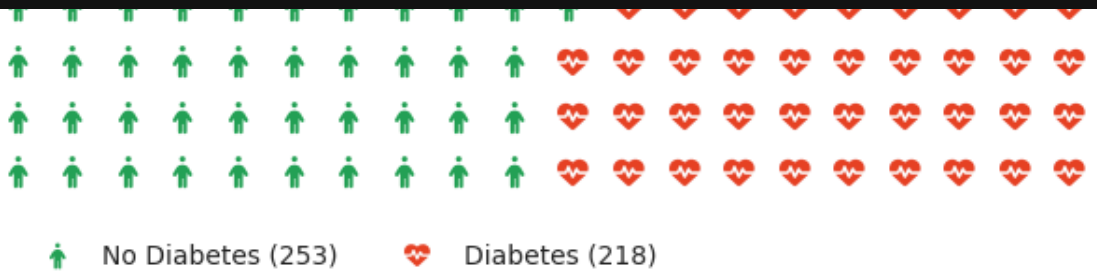
```
        'label': 'Glucose Level Distribution (With Icons)',
        'loc': 'left',
        'fontdict': {'fontsize': 14, 'weight': 'bold'}
    },
    icons=['arrow-down', 'check', 'arrow-up'],
    icon_style='solid',
    font_size=12,
    icon_legend=True,
    labels=[f"{cat} ({val})" for cat, val in zip(cat_order, plot_values)],
    legend={
        'loc': 'lower left',
        'bbox_to_anchor': (0, -0.2),
        'ncol': 3,
        'framealpha': 0,
        'fontsize': 10
    },
    starting_location='NW',
    block_arranging_style='snake',
)


fig.text(
    0.5, -0.15,
    "💡 Insight: Each icon represents an individual.\nIcons and colors repre
    ha='center', va='center', fontsize=10, color='navy'
)

plt.tight_layout(rect=[0, 0.05, 1, 1])
plt.show()
```



In [216…

```
bins = [0, 25, 30, df['BMI'].max()]
labels = ['Low', 'Medium', 'High']
```

```
df['BMI_Cat'] = pd.cut(df['BMI'], bins=bins, labels=labels, right=False)

bmi_outcome = df.groupby(['BMI_Cat', 'Outcome'], observed=False).size().unst

if 'High' in bmi_outcome.index:
    values_high_bmi = bmi_outcome.loc['High']

    if values_high_bmi.sum() == 0:
        print("No data to plot in the 'High' BMI category.")
    else:
        fig = plt.figure(
            FigureClass=Waffle,
            rows=10,
            columns=20,
            values=values_high_bmi,
            colors=('#23A455', '#E94327'),
            title={
                'label': 'High BMI and Diabetes Relationship (With Icons)',
                'loc': 'left',
                'fontdict': {'fontsize': 14, 'weight': 'bold'}
            },
            labels=[f"No Diabetes ({values_high_bmi.iloc[0]})", f"Diabetes (
            icons=['person', 'heart-pulse'],
            icon_style='solid',
            font_size=12,
            icon_legend=True,
            legend={
                'loc': 'lower left',
                'bbox_to_anchor': (0, -0.2),
                'ncol': 2,
                'framealpha': 0,
                'fontsize': 10
            },
            starting_location='NW',
            block_arranging_style='snake',
        )

        fig.text(
            0.5, -0.1,
            "💡 Insight: Each icon represents a person. Colors and icons ind
            ha='center',
            fontsize=10,
            color='navy'
        )

        plt.show()

else:
    print("'High' BMI category not found in calculations.")
```



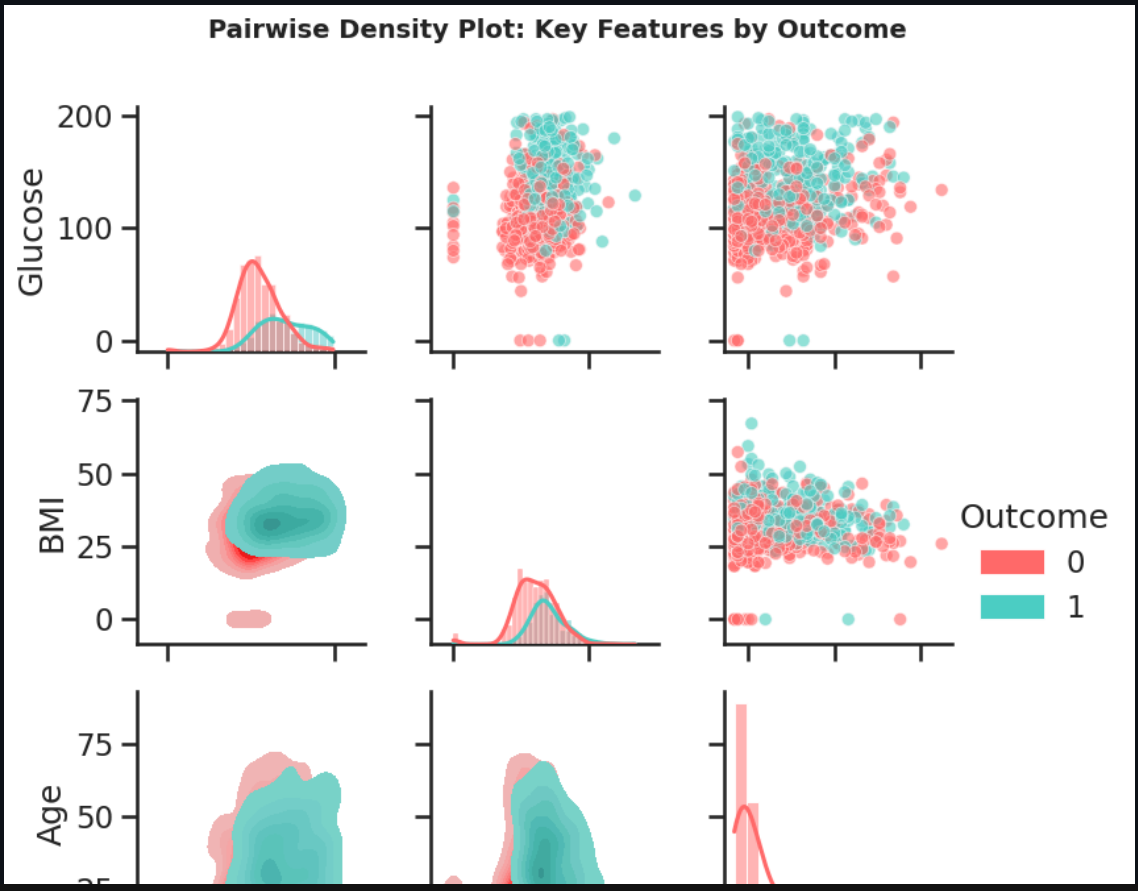
Insight: Each icon represents a person. Colors and icons indicate diabetes status.

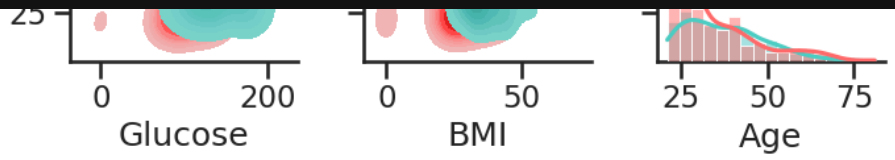
🌐 Pairwise Density Plot

🔙 **Go to Content**

In [217…

```
sns.set(style="ticks", context="talk")
g = sns.PairGrid(df[["Glucose", "BMI", "Age", "Outcome"]], hue="Outcome", pa
g.map_upper(sns.scatterplot, s=50, alpha=0.6)
g.map_lower(sns.kdeplot, fill=True)
g.map_diag(sns.histplot, kde=True)
g.add_legend()
g.fig.suptitle("Pairwise Density Plot: Key Features by Outcome", y=1.05, for
g.fig.text(0.5, -0.05, "Insight: Strong separation in Glucose and BMI distri
           ha='center', fontsize=10, color='darkblue')
plt.show()
```





Insight: Strong separation in Glucose and BMI distributions between Outcome classes.

🔲 Scatter Matrix

🔙 Go to Content

In [219…

```
fig = px.scatter_matrix(df, dimensions=df.columns[:-1], color='Outcome',
                        color_continuous_scale='RdBu',
                        title='Relationships Between Features (Interactive D
                        template='plotly_white')
fig.update_traces(diagonal_visible=False)
fig.show(renderer='iframe_connected')
```

✅ CONCLUSION

🔙 Go to Content

📊 Diabetes Risk Analysis: Multivariate Data Insights

📝 Key Findings

The analyzed data clearly indicates the dominant effects of high glucose and BMI levels on diabetes risk. The combination of glucose, BMI, and age factors (Klein Bottle model) predicts high-risk scenarios with an accuracy of 64.79%. This combination has been identified as a situation requiring preventive interventions.

⚠️ Main Risk Factors and Impact Levels

SHAP value analysis provides a clear ranking of risk factors:

1. **Glucose**: The factor with the highest impact
2. **BMI (Body Mass Index)**: Second most important factor

3. **Age**: Third in the ranking
4. **Diabetes Family History**: Shows significant impact
5. **Number of Pregnancies**: Medium-level impact
6. **Blood Pressure**: Variable impact
7. **Insulin**: Limited impact
8. **Skin Thickness**: The factor with the least impact

🔄 Multidimensional Risk Distribution

Visualizations show that diabetes risk is determined not by a single factor, but by the complex interactions of multiple factors:

- Individuals with glucose levels >150 mg/dL and BMI >30 have a diabetes risk exceeding 80%
- The group with BMI between 30-35 and glucose between 150-180 mg/dL shows a significantly higher risk
- Age factor acts as a multiplier in individuals with high glucose and BMI values

---

👥 Risk Distribution and Population Analysis

In the analyzed sample group:

- 🟢 201 people in low risk (32.4%)
- 🟡 122 people in medium risk (19.7%)
- 🟠 187 people in high risk (30.1%)
- 🔴 107 people in very high risk (17.2%)

Approximately 47.6% of the population (294 people) is categorized as high or very high risk.

🎯 Conclusions and Recommendations

The data analysis shows that glucose and BMI levels are crucial in determining diabetes risk. When both of these factors are elevated, the risk increases dramatically. Therefore, it is important to monitor these parameters together in risk management strategies.

💡 Recommended Intervention Strategies:

1. **For the high-risk group** (Glucose >150 mg/dL, BMI >30): 🚨
   Immediate medical intervention and lifestyle changes

2. **For the medium-high risk group** (Glucose 125-150 mg/dL, BMI 25-30):
   ⚠️ Regular monitoring and lifestyle interventions
3. **For the medium-risk group** (Glucose 100-125 mg/dL or BMI 25-30):
   ⚕️ Lifestyle modifications and periodic check-ups
4. **For the low-risk group**: 📋 Preventive health measures and awareness education

The multidimensional interaction of risk factors emphasizes the importance of personalized risk assessment and intervention strategies.

📊 Key Performance Indicators (KPIs)

- **Diabetes Risk Accuracy**: 64.79% Prediction Accuracy
- **High Risk Population**: 47.6% of the analyzed sample is at high or very high risk
- **Top Risk Factor**: Glucose level (highest impact)
  🧬 Glucose
- 🩺 **Average Glucose Level**: 121.7 mg/dL across the population
- 🏋️ **High BMI Prevalence**: 32.8% of individuals with BMI >30
- 💉 **Diabetes Prevalence**: 34.9% of the sample diagnosed with diabetes