

İnme (felç)
belirtileri
nelerdir?



Stroke Data Project

Welcome to the data analysis assignment on the Stroke Data Project! In this assignment, we will work with a dataset containing information about various strokes. According to the World Health Organization (WHO), stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get a stroke based on input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient. Through this analysis, you will gain hands-on experience in essential data analysis steps, including data cleaning, visualization, and exploratory data analysis (EDA).

Import Libraries, Loading the Dataset and Initial Exploration

- Load the dataset, display first few rows, check the structure of the dataset.
- Inspect the data types and missing values using `df.info()`
- Get basic statistics for numerical columns with `df.describe()`

In []:

```
import pandas as pd # Veri manipülasyonu ve analiz için pandas
import numpy as np # Sayısal işlemler ve matematiksel fonksiyonlar için
import matplotlib.pyplot as plt # Görselleştirme için matplotlib
import seaborn as sns # İleri düzey görselleştirme için seaborn
import warnings # Uyarı mesajlarını kontrol etmek için warnings
warnings.filterwarnings("ignore") # Uyarıları gizlemek için
warnings.warn("this will not show") # Test amacıyla bir uyarı
import plotly.express as px # İnteraktif görselleştirme için plotly
from plotly.subplots import make_subplots # Plotly ile alt grafikler oluşturmak için
import plotly.graph_objects as go # Plotly için daha özelleştirilmiş grafikler için
import matplotlib.patches as patches # Grafiklere şekil eklemek için matplotlib
from matplotlib.offsetbox import AnnotationBbox, OffsetImage # Görselleştirme için
from matplotlib.font_manager import FontProperties # Özel fontlar eklemek için
```

```
In [2]: data = pd.read_csv("healthcare-dataset-stroke-data.csv")
data.head()
```

```
Out[2]:
```

| | id | gender | age | hypertension | heart_disease | ever_married | work_type |
|---|-------|--------|------|--------------|---------------|--------------|---------------|
| 0 | 9046 | Male | 67.0 | 0 | 1 | Yes | Private |
| 1 | 51676 | Female | 61.0 | 0 | 0 | Yes | Self-employed |
| 2 | 31112 | Male | 80.0 | 0 | 1 | Yes | Private |
| 3 | 60182 | Female | 49.0 | 0 | 0 | Yes | Private |
| 4 | 1665 | Female | 79.0 | 1 | 0 | Yes | Self-employed |

```
In [ ]: df = data.copy() # Veri çerçevesinin bir kopyasını oluşturur, orijinal v
```

```
In [ ]: df.info() # Veri çerçevesinin genel bilgilerini (sütunlar, veri türleri,
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   id                     5110 non-null   int64
1   gender                 5110 non-null   object
2   age                    5110 non-null   float64
3   hypertension            5110 non-null   int64
4   heart_disease           5110 non-null   int64
5   ever_married            5110 non-null   object
6   work_type               5110 non-null   object
7   Residence_type          5110 non-null   object
8   avg_glucose_level       5110 non-null   float64
9   bmi                     4909 non-null   float64
10  smoking_status          5110 non-null   object
11  stroke                  5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

```
In [5]: from skimpy import skim
```

```
In [6]: skim(df[['age', 'bmi', 'avg_glucose_level']])
```

| Data Summary | | Data Types | |
|-------------------|--------|-------------|-------|
| dataframe | Values | Column Type | Count |
| Number of rows | 5110 | float64 | 3 |
| Number of columns | 3 | | |

| number | | | | | |
|-------------------|-----|------|-------|-------|-------|
| column_name | NA | NA % | mean | sd | p0 |
| age | 0 | 0 | 43.23 | 22.61 | 0.08 |
| bmi | 201 | 3.93 | 28.89 | 7.854 | 10.3 |
| avg_glucose_level | 0 | 0 | 106.1 | 45.28 | 55.12 |

In [7]: `df.isnull().sum()`

```
Out[7]: id                0
gender              0
age                0
hypertension       0
heart_disease      0
ever_married       0
work_type          0
Residence_type     0
avg_glucose_level  0
bmi                201
smoking_status     0
stroke             0
dtype: int64
```

```
In [ ]: fig = plt.figure(figsize=(20, 8)) # Grafik boyutunu ayarlamak için

# İlk grafik: Eksik değerler için sütun bazında bar plot
plt.subplot(1, 2, 1)
missing_values = df.isnull().sum() # Her sütundaki eksik değerlerin sayı
ax = sns.barplot(x=missing_values.index, y=missing_values.values, alpha=0.8)
plt.title('Missing Values by Column', pad=20, fontsize=16, fontweight='bold')
plt.xlabel('Columns', fontsize=12) # X eksen etiketi
plt.ylabel('Count of Missing Values', fontsize=12) # Y eksen etiketi
plt.xticks(rotation=45, ha='right') # X eksenindeki etiketleri döndürür

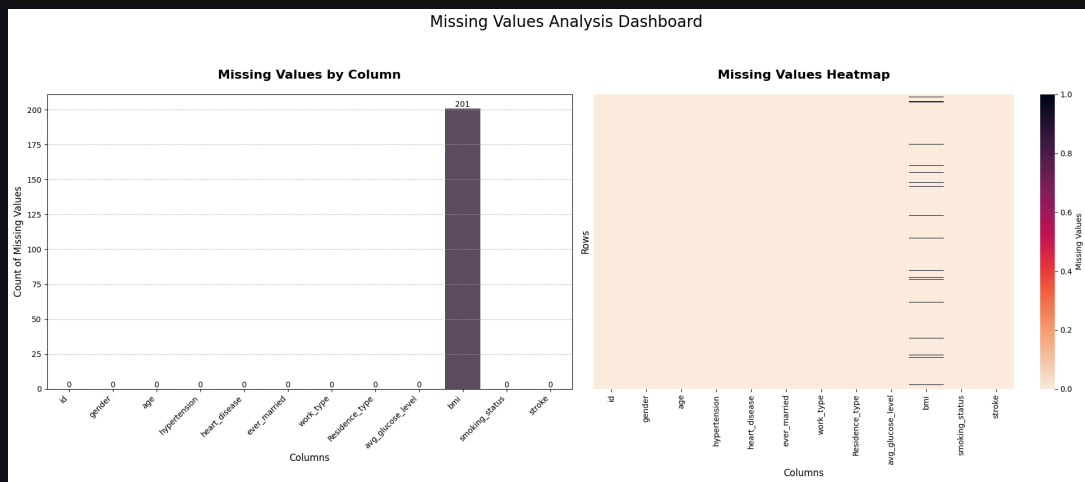
# Barların üstüne eksik değer sayısını yazar
for i, v in enumerate(missing_values.values):
    ax.text(i, v, str(v), ha='center', va='bottom')

ax.grid(axis='y', linestyle='--', alpha=0.7) # Y eksen için grid ekler

# İkinci grafik: Eksik değerler için ısı haritası (heatmap)
plt.subplot(1, 2, 2)
sns.heatmap(df.isnull(),
            cmap='rocket_r', # Renk paleti ayarı
            yticklabels=False, # Y eksen etiketlerini gizler
            cbar_kws={'label': 'Missing Values'}) # Renk skalası etiket
plt.title('Missing Values Heatmap', pad=20, fontsize=16, fontweight='bold')
plt.xlabel('Columns', fontsize=12) # X eksen etiketi
plt.ylabel('Rows', fontsize=12) # Y eksen etiketi

# Genel başlık
plt.suptitle('Missing Values Analysis Dashboard', fontsize=20, y=1.05)
plt.tight_layout() # Alt ve üst alanları sıkıştırır

plt.show() # Grafikleri ekranda gösterir
```



● Eksik Veri Tespiti (Missing Value)

- Tek missing value yalnızca BMI sütununda tespit edilmiştir.
- Bu eksik değerler, heatmap ve msgno ile görselleştirilmiştir.

Data Cleaning:

- Handle missing values.
- Check for duplicates and remove them if found.
- Standardize column names (if necessary) for consistent naming conventions.
- Validate data types and convert columns to appropriate types if needed.

```
In [ ]: df.columns = df.columns.str.title() # Sütun isimlerini baş harfleri büyük
df.columns # Düzenlenmiş sütun isimlerini gösterir
```

```
Out[ ]: Index(['Id', 'Gender', 'Age', 'Hypertension', 'Heart_Disease', 'Ever_Married',
              'Work_Type', 'Residence_Type', 'Avg_Glucose_Level', 'Bmi',
              'Smoking_Status', 'Stroke'],
              dtype='object')
```

```
In [ ]: df.rename(columns={'Bmi': 'BMI', 'Id': 'ID'}, inplace=True) # 'Bmi' ve 'Id'
df.columns # Düzenlenmiş sütun isimlerini gösterir
```

```
Out[ ]: Index(['ID', 'Gender', 'Age', 'Hypertension', 'Heart_Disease', 'Ever_Married',
              'Work_Type', 'Residence_Type', 'Avg_Glucose_Level', 'BMI',
              'Smoking_Status', 'Stroke'],
              dtype='object')
```

```
In [ ]:
```

```
df.dtypes # Veri çerçevesindeki her sütunun veri türünü gösterir
```

```
Out[ ]: ID                int64
Gender                object
Age                  float64
Hypertension          int64
Heart_Disease         int64
Ever_Married          object
Work_Type             object
Residence_Type        object
Avg_Glucose_Level     float64
BMI                   float64
Smoking_Status        object
Stroke                int64
dtype: object
```

```
In [ ]: def categorize_age(age): # Yaşa göre kategorize eden fonksiyon
        if age >= 0 and age <= 1:
            return 'Infant' # 0-1 yaş arası bebek
        elif age >1 and age <= 3:
            return 'Toddler' # 1-3 yaş arası çocuk
        elif age >3 and age <= 6:
            return 'Preschooler' # 3-6 yaş arası okul öncesi çocuk
        elif age > 6 and age <= 12:
            return 'School Age' # 6-12 yaş arası okul çağı
        elif age > 12 and age < 20:
            return 'Teenager' # 12-20 yaş arası genç
        elif age >= 20 and age <= 24:
            return 'Adolescence' # 20-24 yaş arası ergenlik dönemi
        elif age > 24 and age <= 39:
            return 'Adult' # 24-39 yaş arası yetişkin
        elif age > 39 and age <= 59:
            return 'Middle Aged' # 39-59 yaş arası orta yaş
        else:
            return 'Senior' # 60 ve üzeri yaş arası yaşlı
```

```
In [ ]: df['Age_Group'] = df['Age'].apply(categorize_age) # 'Age' sütununa göre
plt.style.use('seaborn-v0_8-darkgrid') # Seaborn'un koyu ızgara stilini
plt.figure(figsize=(12, 6)) # Grafik boyutunu ayarlar

age_counts = df['Age_Group'].value_counts() # Yaş gruplarının sayısını
ax = sns.barplot(x=age_counts.index,
                 y=age_counts.values,
                 palette='rocket', # Renk paleti
                 alpha=0.8) # Şeffaflık ayarı

plt.title('Distribution of Age Groups', pad=20, fontsize=16, fontweight='bold')
plt.xlabel('Age Group', fontsize=12) # X eksen etiketi
plt.ylabel('Count', fontsize=12) # Y eksen etiketi

# Her barın üstüne değer yazdırır
for i, v in enumerate(age_counts.values):
    ax.text(i, v, str(v), ha='center', va='bottom', fontsize=10)

ax.grid(axis='y', linestyle='--', alpha=0.7) # Y eksen için grid ekle

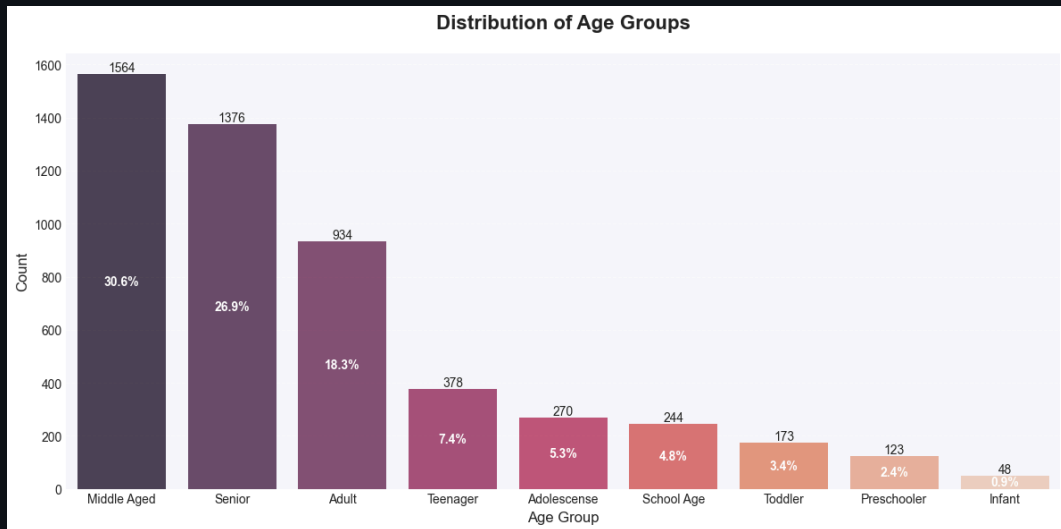
plt.xticks(rotation=0) # X eksen etiketlerini yatay hizalar

ax.set_facecolor('#f8f9fa') # Grafik arka plan rengini belirler
```

```
total = age_counts.sum() # toplam sayıyı hesaplar
for i, v in enumerate(age_counts.values):
    percentage = (v/total) * 100 # Yüzde hesaplar
    ax.text(i, v/2, f'{percentage:.1f}%', # Yüzdeyi barın ortasına ekl
            ha='center',
            va='center',
            color='white', # Yüzdeyi beyaz renkte yazdırır
            fontweight='bold')

plt.tight_layout() # Grafik öğelerini sıkıştırır

plt.show() # Grafiği ekranda gösterir
```



1 2 3 4 Yaş Grupları Oluşturulması

- Data içinde farklı yaş grupları olduğundan, daha detaylı analiz yapılabilmesi için yeni bir sütun/değişken oluşturulmuştur.
- Yeni oluşturduğumuz Age Group sütunu, grafik üzerinde gösterilmiştir. 🇮🇹

In [14]: `df.duplicated().sum()`

Out[14]: `np.int64(0)`

In []: `df = df[df['Gender'] != 'Other'] # 'Gender' sütununda 'Other' olan satırları kaldır`

In [16]: `df.Gender.value_counts()`

Out[16]:

| | |
|---------------------------|------|
| Gender | |
| Female | 2994 |
| Male | 2115 |
| Name: count, dtype: int64 | |

🚧 Veri Temizliği ve Düzenleme

- Gender sütununda tek bir değerin "Other" olduğunu tespit ettik.
- Bu sebeple, ilgili satırı verimizden sildik.
- Sonraki analizlerin daha anlamlı ve doğru olabilmesi için bu düzenlemeyi yaptık. 🛠️ 📊

Missing Value

```
In [17]: df.isnull().sum()
```

```
Out[17]: ID                0
Gender                0
Age                  0
Hypertension         0
Heart_Disease        0
Ever_Married         0
Work_Type            0
Residence_Type       0
Avg_Glucose_Level    0
BMI                  201
Smoking_Status       0
Stroke               0
Age_Group            0
dtype: int64
```

```
In [ ]: df[df['BMI'] >= 45] # 'BMI' değeri 45 ve daha büyük olan satırları fil
```

```
Out[ ]:   ID  Gender  Age  Hypertension  Heart_Disease  Ever_Married  Work_
```

| | | | | | | | |
|------|-------|--------|------|-----|-----|-----|------|
| 21 | 13861 | Female | 52.0 | 1 | 0 | Yes | empl |
| 66 | 17004 | Female | 70.0 | 0 | 0 | Yes | Pr |
| 79 | 42117 | Male | 43.0 | 0 | 0 | Yes | empl |
| 113 | 41069 | Female | 45.0 | 0 | 0 | Yes | Pr |
| 163 | 20426 | Female | 78.0 | 1 | 0 | No | Pr |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 4976 | 63656 | Female | 18.0 | 0 | 0 | No | Pr |
| 5009 | 40732 | Female | 50.0 | 0 | 0 | Yes | empl |
| 5015 | 50140 | Female | 44.0 | 0 | 0 | Yes | Gov |
| 5057 | 38349 | Female | 49.0 | 0 | 0 | Yes | Gov |
| 5103 | 22127 | Female | 18.0 | 0 | 0 | No | Pr |

160 rows × 13 columns

In []:

```

Q1 = df['BMI'].quantile(0.25) # BMI'nin 1. çeyrek değeri (Q1)
Q3 = df['BMI'].quantile(0.75) # BMI'nin 3. çeyrek değeri (Q3)
IQR = Q3 - Q1 # Çeyrekler arası aralık (IQR)

x = Q1 - 3 * IQR # Alt sınır (outlier'lar için)
y = Q3 + 3 * IQR # Üst sınır (outlier'lar için)

outliers = df[(df['BMI'] < x) | (df['BMI'] > y)] # Alt ve üst sınırlar
outliers # Outlier (aykırı) değerleri gösterir

```

Out[]:

| | ID | Gender | Age | Hypertension | Heart_Disease | Ever_Married | Work |
|------|-------|--------|------|--------------|---------------|--------------|------|
| 358 | 66333 | Male | 52.0 | 0 | 0 | Yes | empl |
| 544 | 545 | Male | 42.0 | 0 | 0 | Yes | Pr |
| 928 | 41097 | Female | 23.0 | 1 | 0 | No | Pr |
| 1559 | 37759 | Female | 53.0 | 0 | 0 | Yes | Pr |
| 2128 | 56420 | Male | 17.0 | 1 | 0 | No | Pr |
| 2764 | 20292 | Female | 24.0 | 0 | 0 | Yes | Pr |
| 4188 | 70670 | Female | 27.0 | 0 | 0 | Yes | Pr |
| 4209 | 51856 | Male | 38.0 | 1 | 0 | Yes | Pr |

In []:

```

plt.style.use('seaborn-v0_8-darkgrid') # Seaborn koyu ızgara stilini k
plt.figure(figsize=(12, 8)) # Grafik boyutunu ayarlar

# Boxplot grafiği ile yaş grubu başına BMI dağılımını gösterir
ax = sns.boxplot(data=df,
                  x='Age_Group',
                  y='BMI',
                  palette='rocket', # Renk paleti
                  width=0.7, # Boxplot genişliği
                  fliersize=3, # Aykırı değerlerin boyutu
                  linewidth=1.5) # Boxplot çizgi kalınlığı

plt.title('BMI Distribution by Age Group', pad=20, fontsize=16, fontwei
plt.xlabel('Age Group', fontsize=12) # X eksen etiketi
plt.ylabel('BMI', fontsize=12) # Y eksen etiketi

ax.set_facecolor('#f8f9fa') # Grafik arka plan rengini belirler

ax.grid(axis='y', linestyle='--', alpha=0.7) # Y ekseninde kesik çizgi

# Her yaş grubu için medyan değerini grafiğe ekler
for i in range(len(df['Age_Group'].unique())):
    age_group = df['Age_Group'].unique()[i]
    median = df[df['Age_Group'] == age_group]['BMI'].median()
    ax.text(i, df['BMI'].max(), f'Median: {median:.1f}',
            ha='center', va='bottom', fontsize=9)

```

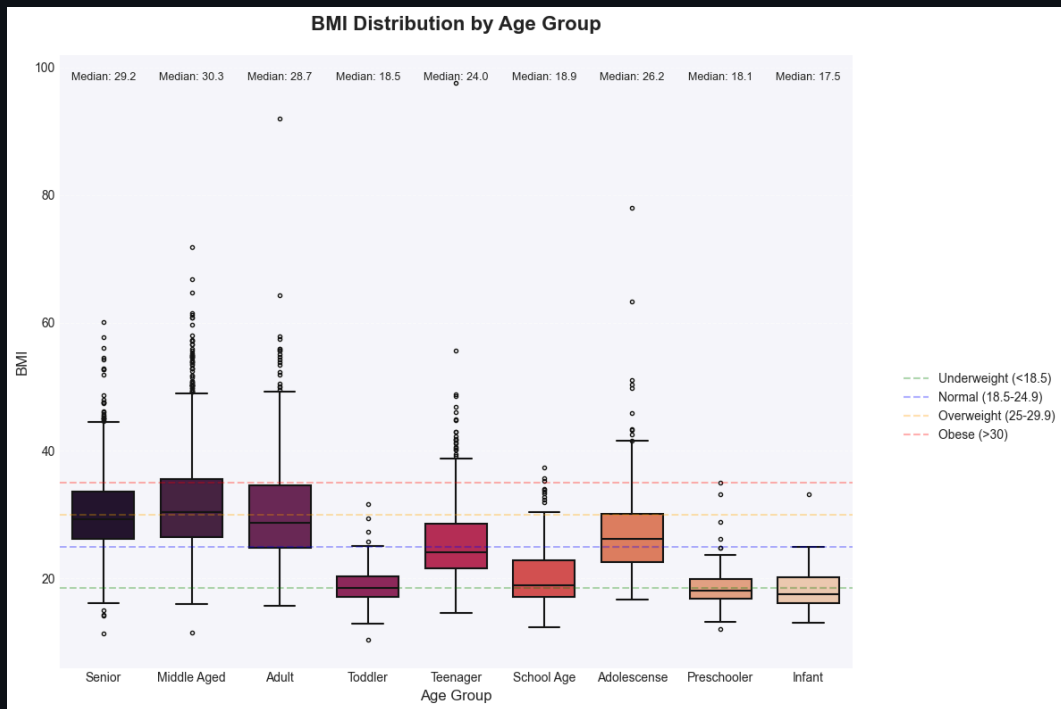


```
# BMI kategorilerine göre referans çizgileri ekler
plt.axhline(y=18.5, color='green', linestyle='--', alpha=0.3, label='Un
plt.axhline(y=24.9, color='blue', linestyle='--', alpha=0.3, label='Nor
plt.axhline(y=29.9, color='orange', linestyle='--', alpha=0.3, label='O
plt.axhline(y=34.9, color='red', linestyle='--', alpha=0.3, label='Obes

plt.legend(bbox_to_anchor=(1.05, 0.5), loc='upper left') # Efsane etik

plt.tight_layout() # Grafik öğelerini sıkıştırır

plt.show() # Grafiği ekranda gösterir
```



Missing Value Doldurma Aşaması

- Age Group'lara ait BMI dağılımlarını gözlemledik ve değişkenlik olduğunu fark ettik.
- Bu sebeple, missing value doldurma işlemini her bir age group özelinde yapmaya karar verdik. 🔄

```
In [ ]: bmi_avg_value = df.groupby(['Age_Group', 'Gender'])['BMI'].median().res
# 'Age_Group' ve 'Gender' sütunlarına göre 'BMI' değerinin medyanını he
```

```
In [22]: bmi_avg_value
```

```
Out[22]:
```

| | Age_Group | Gender | BMI |
|---|-------------|--------|-------|
| 0 | Adolescence | Female | 26.50 |
| 1 | Adolescence | Male | 25.75 |
| 2 | Adult | Female | 28.10 |
| 3 | Adult | Male | 29.90 |

| | | | |
|----|-------------|--------|-------|
| 4 | Infant | Female | 17.35 |
| 5 | Infant | Male | 18.30 |
| 6 | Middle Aged | Female | 29.80 |
| 7 | Middle Aged | Male | 30.80 |
| 8 | Preschooler | Female | 17.95 |
| 9 | Preschooler | Male | 18.10 |
| 10 | School Age | Female | 18.90 |
| 11 | School Age | Male | 18.80 |
| 12 | Senior | Female | 29.20 |
| 13 | Senior | Male | 29.40 |
| 14 | Teenager | Female | 24.15 |
| 15 | Teenager | Male | 23.95 |
| 16 | Toddler | Female | 18.10 |
| 17 | Toddler | Male | 18.90 |

```
In [ ]: df['BMI'] = df['BMI'].fillna(df.groupby(['Gender', 'Age_Group'])['BMI']
# Eksik BMI değerlerini, aynı 'Gender' ve 'Age_Group' değerlerine sahip
```

```
In [24]: df['BMI'].isnull().sum()
```

```
Out[24]: np.int64(0)
```

```
In [ ]: def categorize_BMI(bmi):
# BMI'yi kategorilere ayıran fonksiyon
if bmi < 18.5:
return 'Underweight' # Zayıf (<18.5)
elif 18.5 <= bmi <= 24.9:
return 'Normal Weight' # Normal kilo (18.5-24.9)
elif 24.9 < bmi <= 29.9:
return 'Overweight' # Fazla kilolu (25-29.9)
elif 29.9 < bmi <= 34.9:
return 'Moderately Obese' # Orta derecede obez (30-34.9)
elif 34.9 < bmi <= 40:
return 'Severely Obese' # Ciddi derecede obez (35-40)
else:
return 'Extreme Obese' # Aşırı obez (>40)
```

```
In [ ]: df['BMI_Group'] = df['BMI'].apply(categorize_BMI)
# 'BMI' sütunundaki her değeri 'categorize_BMI' fonksiyonu ile kategori
```

BMI Değerlerine Dayalı Gruplandırma Çalışması 🏋️ 📊

- Bu çalışmada, **BMI** değerlerine dayalı olarak bireyleri **DSÖ'nün (Dünya Sağlık Örgütü)** belirlediği sınıflandırmalara göre gruplandırmak amacıyla bir **"BMI_Group"** sütunu oluşturulmuştur. 📝
- Bu sütun, veriyi daha detaylı analiz edebilmek, **BMI ile ilişkili sağlık göstergeleri** üzerinde derinlemesine inceleme yapabilmek için kullanılmıştır. 💪

In []:

```
plt.style.use('seaborn-v0_8-darkgrid') # Grafik stilini seaborn'un darkgrid
plt.figure(figsize=(12, 6)) # Grafik boyutlarını belirliyor

bmi_counts = df['BMI_Group'].value_counts() # 'BMI_Group' sütunundaki
ax = sns.barplot(x=bmi_counts.index,
                 y=bmi_counts.values,
                 palette='rocket', # Renk paletini belirliyor
                 alpha=0.8) # Saydamlık oranını ayarlıyor

plt.xlabel('BMI Groups', fontsize=12) # X eksenini etiketini belirliyor
plt.ylabel('Count', fontsize=12) # Y eksenini etiketini belirliyor

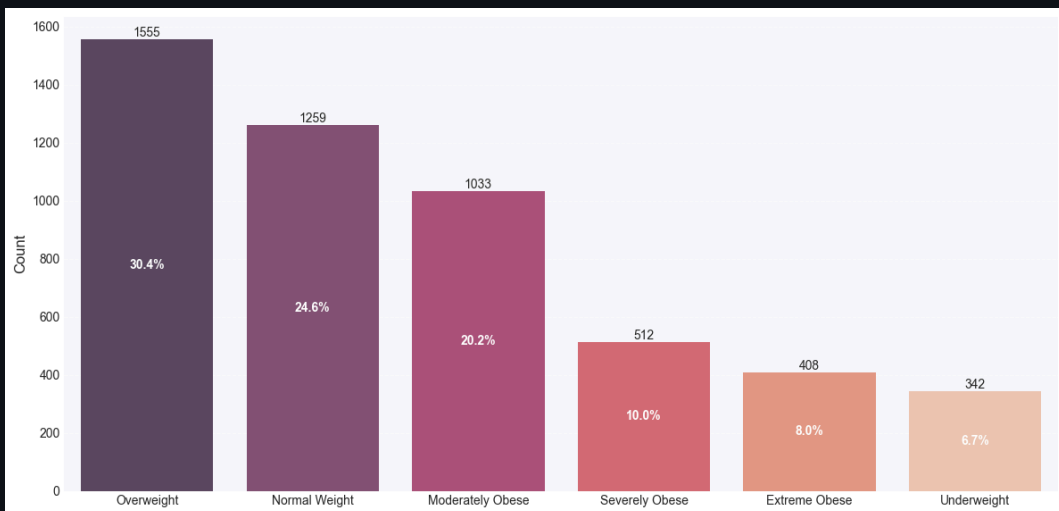
# Barların üzerine değerlerini yazıyor
for i, v in enumerate(bmi_counts.values):
    ax.text(i, v, str(v), ha='center', va='bottom', fontsize=10)

ax.grid(axis='y', linestyle='--', alpha=0.7) # Y ekseninde grid çizgisini
plt.xticks(rotation=0) # X eksenini etiketlerini döndürmeden yerleştiriyor
ax.set_facecolor('#f8f9fa') # Grafik arka plan rengini değiştiriyor

total = bmi_counts.sum() # Toplam sayıyı hesaplıyor
for i, v in enumerate(bmi_counts.values):
    percentage = (v/total) * 100 # Yüzdeyi hesaplıyor
    ax.text(i, v/2, f'{percentage:.1f}%',
            ha='center',
            va='center',
            color='white',
            fontweight='bold') # Yüzdeyi bar üzerinde beyaz renkte yazıyor

plt.tight_layout() # Grafik düzenini sıkıştırıyor

plt.show() # Grafiği gösteriyor
```



BMI Gruplarının Son Durumu

Bu grafik, **missing value** doldurma işleminden sonra **BMI gruplarının** son durumunu göstermektedir.




Açıklamalar:

- Grafik, veri setindeki eksik değerlerin doldurulmasının ardından, her bir BMI grubundaki değişiklikleri ve dağılımları görselleştirmektedir.
- Bu işlem, veri setinin daha doğru ve güvenilir hale gelmesini sağlamaktadır.

In []:

```
df['Glucose_Level_Group'] = pd.cut(df['Avg_Glucose_Level'], bins = [0,
# 'Avg_Glucose_Level' sütunundaki değerleri belirtilen aralıklara (binl
```

Glikoz Düzeylerine Dayalı Gruplandırma Çalışması

- Bu çalışmada, **glikoz düzeylerine (glucose level)** dayalı olarak bireyleri analiz etmek amacıyla bir "**Glucose_Level_Group**" sütunu oluşturulmuştur. 
- Bu sütun, veriyi daha detaylı analiz edebilmek, **glikoz düzeyi ile ilişkili sağlık göstergeleri** üzerinde derinlemesine inceleme yapabilmek için kullanılmıştır.  

In []:

```
plt.style.use('seaborn-v0_8-darkgrid') # Grafik stilini seaborn'un dar

plt.figure(figsize=(12, 6)) # Grafik boyutlarını belirliyor

glc_counts = df['Glucose_Level_Group'].value_counts() # 'Glucose_Level

ax = sns.barplot(x=glc_counts.index,
                 y=glc_counts.values,
                 palette='rocket', # Renk paletini belirliyor
                 alpha=0.8) # Saydamlık oranını ayarlıyor

plt.title('Distribution of Glucose Level Groups', pad=20, fontsize=16,
plt.xlabel('Glucose Level Group', fontsize=12) # X eksen etiketini be
plt.ylabel('Count', fontsize=12) # Y eksen etiketini belirliyor

# Barların üzerine değerlerini yazıyor
for i, v in enumerate(glc_counts.values):
    ax.text(i, v + 0.02 * glc_counts.max(), str(v), ha='center', va='bo

ax.grid(axis='y', linestyle='--', alpha=0.7) # Y ekseninde grid çizgis

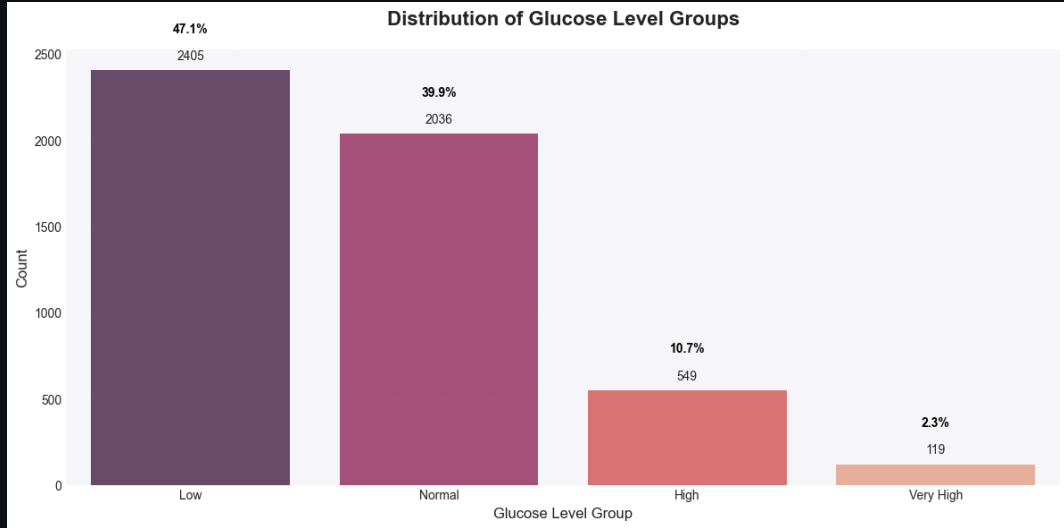
plt.xticks(rotation=0) # X eksen etiketlerini döndürmeden yerleştiriy

ax.set_facecolor('#f8f9fa') # Grafik arka plan rengini değiştiriyor
```

```
total = glc_counts.sum() # Toplam sayıyı hesaplıyor
for i, v in enumerate(glc_counts.values):
    percentage = (v/total) * 100 # Yüzdeyi hesaplıyor
    ax.text(i, v + 0.1 * glc_counts.max(), f'{percentage:.1f}%',
            ha='center',
            va='center',
            color='black',
            fontweight='bold') # Yüzdeyi bar üzerinde siyah renkte yazıyor

plt.tight_layout() # Grafik düzenini sıkıştırıyor

plt.show() # Grafiği gösteriyor
```



Glikoz Seviyeleri Analizi 🍌 📊

Bu çalışmada, glikoz düzeylerine göre bireylerin sınıflandırılması yapılmış ve her bir kategoriye ait oranlar belirlenmiştir:

- **Düşük Glikoz Seviyesi (Low):** %47.1 🍌
- **Normal Glikoz Seviyesi (Normal):** %39.9 ⚖️
- **Yüksek Glikoz Seviyesi (High):** %10.7 🚗
- **Çok Yüksek Glikoz Seviyesi (Very High):** %2.3 🔴

Analiz:

- **Düşük Glikoz Seviyesi (Low):** Bireylerin %47.1'i düşük glikoz seviyelerine sahiptir. Bu grup, genellikle hipoglisemi riski taşıyan bireylerden oluşur. Düşük glikoz seviyeleri, enerji eksikliklerine ve çeşitli sağlık sorunlarına yol açabilir.
- **Normal Glikoz Seviyesi (Normal):** %39.9'u normal glikoz seviyelerine sahiptir, bu da sağlıklı bireylerin büyük kısmını temsil eder. Normal seviyeler, metabolizma ve vücut fonksiyonlarının düzgün işlediğini gösterir.
- **Yüksek Glikoz Seviyesi (High):** %10.7'si yüksek glikoz seviyelerine sahiptir. Bu bireyler, prediyabet ya da diyabet riski taşıyor olabilirler. Bu seviyede, glikozun düzenli izlenmesi ve kontrol altına alınması önemlidir.
- **Çok Yüksek Glikoz Seviyesi (Very High):** %2.3'ü çok yüksek glikoz

seviyelerine sahiptir. Bu durum, genellikle diyabet hastalığının daha ileri evrelerine işaret eder ve acil müdahale gerektirebilir.

Bu oranlar, toplumdaki glikoz düzeylerinin büyük bir kısmının normal olduğunu ancak önemli bir kısmının da yüksek ve çok yüksek glikoz seviyelerine sahip olduğunu göstermektedir. Bu tür analizler, glikoz düzeylerinin sağlık üzerindeki etkilerini anlamak ve önleyici tedbirler almak için faydalıdır.

Analysis Goal

How does age influence stroke occurrence?

Yaş, inme oluşumunu nasıl etkiler?

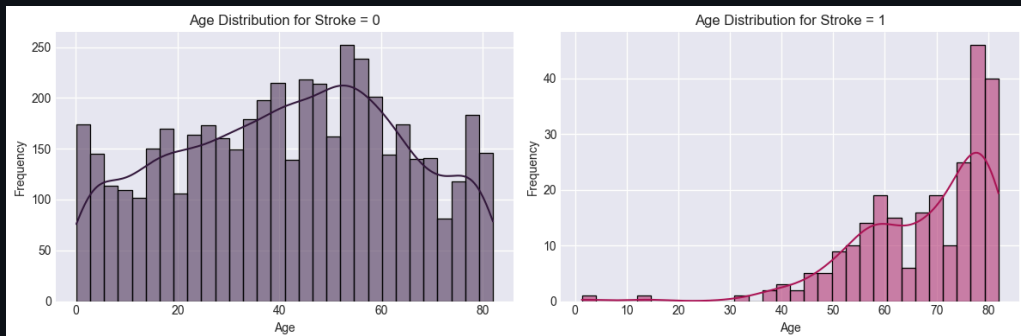
```
In [ ]: stroke_positive = df[df['Stroke'] == 1] # 'Stroke' sütununda 1 olan
stroke_negative = df[df['Stroke'] == 0] # 'Stroke' sütununda 0 olan
```

```
In [ ]: plt.style.use('seaborn-v0_8-darkgrid') # Seaborn'un darkgrid stili
fig, axes = plt.subplots(1, 2, figsize=(12, 4)) # Yan yana iki grafik

# Stroke geçirmeyen bireylerin yaş dağılımını görselleştirir
sns.histplot(data=stroke_negative, x='Age', bins=30, kde=True, color='purple')
axes[0].set_title('Age Distribution for Stroke = 0') # Başlık
axes[0].set_xlabel('Age') # X eksen etiketi
axes[0].set_ylabel('Frequency') # Y eksen etiketi

# Stroke geçiren bireylerin yaş dağılımını görselleştirir
sns.histplot(data=stroke_positive, x='Age', bins=30, kde=True, color='red')
axes[1].set_title('Age Distribution for Stroke = 1') # Başlık
axes[1].set_xlabel('Age') # X eksen etiketi
axes[1].set_ylabel('Frequency') # Y eksen etiketi

plt.tight_layout() # Grafiklerin düzgün bir şekilde yerleşmesini sağlar
plt.show() # Grafiklerin görüntülenmesini sağlar
```



İnme Vakaları ve Yaş Dağılımı




Ana Bulgular




İnme Vakalarının İleri


Yaşlarda Yoğunlaşması


İnme vakaları, 45 yaş civarında belirgin bir artış gösterir ve 80 yaş civarında zirveye ulaşır. Pozitif çarpıklık (skewness), yaşlı bireylerde daha yüksek bir inme vakası sıklığına işaret etmektedir. 75-80 yaş aralığı, en yüksek inme pozitif prevalansına sahip olup, yaşlanmanın inme riski üzerindeki etkisini güçlü bir şekilde vurgular. ● Genç Nüfusta İnme Vakalarının Nadirliği

40 yaş altındaki bireylerde, özellikle 30 yaşın altındaki bireylerde inme vakaları son derece nadirdir. Bu durum, genç nüfusun inme riski açısından önemli ölçüde daha düşük bir olasılığa sahip olduğunu göstermektedir. Yoğunluk Eğrisi ve Histogram  Histogramı tamamlayan yoğunluk eğrisi, inme vakalarının yaşa bağlı dağılımını destekler. Eğri, yaş ilerledikçe inme olasılığında kademeli bir artış olduğunu gösterir ve 70 yaş ve üzerindeki bireylerde zirveye ulaşır.

- Sonuç ve Öneriler  Yaş, inme riskinin önemli bir belirleyicisidir.

50 yaş ve üzerindeki bireylerde inme prevalansı keskin bir şekilde artış gösterir.

 Orta yaşlı ve yaşlı nüfus için önleyici tedbirler ve müdahaleler büyük önem taşır.

 İnme negatif grubu (Stroke = 0) için grafik üzerinde belirgin bir eğilim görülmemiştir, bu nedenle bu kategoriye ilişkin özel çıkarımlar yapmak zordur.

Do body mass index (BMI) and glucose levels jointly or independently increase stroke risk?

Vücut kitle indeksi (VKİ) ve glikoz seviyeleri birlikte veya bağımsız olarak felç riskini artırır mı?

In []:

```
# Stroke geçirmeyen ve geçiren bireyler için BMI ve Ortalama Glikoz
stroke_negative = df[df['Stroke'] == 0] # Stroke geçirmeyenler
stroke_positive = df[df['Stroke'] == 1] # Stroke geçirenler

# Stroke geçiren bireylerin BMI ile Avg_Glucose_Level arasındaki korelasyon
correlation_positive = stroke_positive[['BMI', 'Avg_Glucose_Level']].corr()

# Stroke geçirmeyen bireylerin BMI ile Avg_Glucose_Level arasındaki korelasyon
correlation_negative = stroke_negative[['BMI', 'Avg_Glucose_Level']].corr()
```

In []:

```
# Create figure
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6)) # 1 satır ve 2 sütun

# Custom dark purple and rose red colormap
dark_purple_rose_red_cmap = sns.color_palette(["#35193e", "#ad1759"])

# First heatmap (Positive cases)
```

```

sns.heatmap(correlation_positive,
             annot=True, # Hücrelerdeki değerler gösterilecek
             cmap=dark_purple_rose_red_cmap, # Renk paleti olarak
             fmt='.2f', # Sayılar virgülden sonra iki basamağa yuvarlanacak
             square=True, # Grafik kare şeklinde olacak
             cbar=False, # Renk çubuğu gösterilmeyecek
             ax=ax1, # Bu ısı haritası ax1 üzerine çizilecek
             annot_kws={'size': 12, 'weight': 'bold'}, # Hücredeki
             linewidths=2, # Hücreler arasındaki çizgi kalınlığı
             linecolor='white') # Hücreler arasındaki çizgilerin re

ax1.set_title('Stroke Positive\nCorrelation',
             pad=20, # Başlık ile üst sınır arasındaki boşluk
             fontsize=12, # Başlık font büyüklüğü
             fontweight='light', # Başlık font ağırlığı
             family='sans-serif') # Başlık fontu sans-serif olaca

# Second heatmap (Negative cases)
sns.heatmap(correlation_negative,
             annot=True, # Hücrelerdeki değerler gösterilecek
             cmap=dark_purple_rose_red_cmap, # Renk paleti olarak
             fmt='.2f', # Sayılar virgülden sonra iki basamağa yuvarlanacak
             square=True, # Grafik kare şeklinde olacak
             cbar=False, # Renk çubuğu gösterilmeyecek
             ax=ax2, # Bu ısı haritası ax2 üzerine çizilecek
             annot_kws={'size': 12, 'weight': 'bold'}, # Hücredeki
             linewidths=2, # Hücreler arasındaki çizgi kalınlığı
             linecolor='white') # Hücreler arasındaki çizgilerin re

ax2.set_title('Stroke Negative\nCorrelation',
             pad=20, # Başlık ile üst sınır arasındaki boşluk
             fontsize=12, # Başlık font büyüklüğü
             fontweight='light', # Başlık font ağırlığı
             family='sans-serif') # Başlık fontu sans-serif olaca

# Add minimal sample size information
for ax, data, title in [(ax1, stroke_positive, 'Positive'),
                       (ax2, stroke_negative, 'Negative')]: # Her
    ax.text(0.5, -0.2,
           f'n = {len(data):,}', # Veri setindeki örnek sayısı ya
           transform=ax.transAxes, # Koordinatlar eksenlere göre
           fontsize=10, # Yazı font boyutu
           ha='center', # Yazının yatay hizalanması
           color='#666666') # Yazının rengi gri olacak

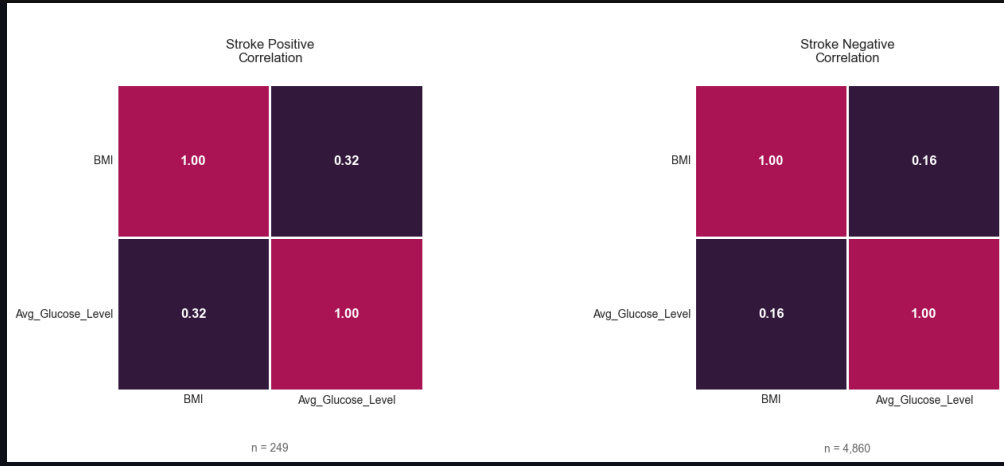
# Clean up axes
ax.set_xticklabels(ax.get_xticklabels(), rotation=0) # X ekser
ax.set_yticklabels(ax.get_yticklabels(), rotation=0) # Y ekser

# Add subtle main title
plt.suptitle('BMI and Glucose Level Correlation Analysis',
            fontsize=14, # Başlık font boyutu
            fontweight='light', # Başlık font ağırlığı
            y=1.05, # Başlık ile grafik arasındaki boşluk
            family='sans-serif') # Başlık fontu sans-serif olaca

# Adjust Layout
plt.tight_layout() # Layout sıkıştırılarak grafik öğelerinin çıkış

# Show plot
plt.show() # Grafik ekrana gösteriliyor

```

Glikoz ve BMI Arasındaki Korelasyon Analizi 🧠 📊

Genel Bulgular:

- **Genel Korelasyon:** Glikoz ve BMI arasındaki korelasyon genel anlamda **düşüktür**. Bu, glikoz seviyeleri ile vücut kitle indeksi (BMI) arasında doğrudan bir ilişki bulunmadığını gösteriyor. Yani, glikoz düzeyleri ile BMI arasında güçlü bir bağ kurulamamıştır.

Stroke Pozitif ve Negatif Kategorileri:

- **Stroke Negatifler:** Stroke geçirmemiş bireyler için, glikoz ve BMI arasındaki korelasyon **çok düşüktür**. Bu, glikoz seviyeleri ile BMI arasında önemli bir ilişki olmadığı anlamına gelir.
- **Stroke Pozitifler:** Ancak, **stroke geçiren bireyler** için bu korelasyon **iki katına çıkmıştır**. Yani, stroke geçiren bireylerde glikoz seviyeleri ile BMI arasında daha belirgin bir ilişki gözlemlenmiştir. Bu, sağlık durumunun ve diğer faktörlerin glikoz ile BMI arasındaki ilişkiyi etkileyebileceğini gösteriyor.

Sonuç:

Glikoz ve BMI arasındaki ilişki, stroke durumu gibi sağlık faktörlerine bağlı olarak değişebilmektedir. Stroke geçiren bireylerde bu iki değişken arasındaki ilişki daha belirgin hale gelirken, stroke geçirmeyenlerde bu ilişki daha zayıf kalmaktadır.

In []:

```
plt.figure(figsize=(12, 8)) # Grafik boyutu belirleniyor (12x8)

# Scatter plot (dağılım grafiği) oluşturuluyor
scatter = sns.scatterplot(
    data=df, # Veri kümesi
    x='BMI', # X ekseninde Body Mass Index (BMI) olacak
    y='Avg_Glucose_Level', # Y ekseninde Ortalama Glukoz Seviyesi
    hue='Stroke' # 'Stroke' sütununa göre renk ayırımı yapılacaktır
```

```

alpha=0.7, # Noktaların saydamlık değeri
s=100, # Noktaların boyutu
palette={0: '#35193e', 1: '#ad1759'}, # Stroke durumu için özel
legend='brief' # Legend'ı kısaltılmış olarak gösterme
)

# Stroke durumu için regresyon çizgileri ekleniyor
for stroke_status in [0, 1]: # Stroke 0 ve 1 için ayrı ayrı çizgi
    mask = df['Stroke'] == stroke_status # Stroke durumu kontrol et
    x = df[mask]['BMI'] # BMI değerleri seçiliyor
    y = df[mask]['Avg_Glucose_Level'] # Glukoz seviyeleri seçiliyor
    z = np.polyfit(x, y, 1) # Polyfit fonksiyonu ile doğrusal regr
    p = np.poly1d(z) # Regresyon doğrusu
    plt.plot(x, p(x), # Regresyon çizgisi grafiğe ekleniyor
             linestyle='--', # Çizgi stili kesikli
             alpha=0.5, # Çizgi şeffaflığı
             color='#5A2A6C' if stroke_status == 0 else '#72C41')

# Başlık ekleniyor
plt.title('Relationship Between BMI and Glucose Levels\nby Stroke S
        fontsize=16, # Başlık font boyutu
        pad=20, # Başlık ile üst sınır arasındaki boşluk
        fontweight='light', # Başlık font ağırlığı
        family='sans-serif') # Başlık fontu sans-serif olacak

# X ve Y eksen etiketleri ekleniyor
plt.xlabel('Body Mass Index (BMI)',
           fontsize=12, # X eksenindeki etiket font boyutu
           fontweight='light') # X eksenindeki etiket font ağırlığı
plt.ylabel('Average Glucose Level (mg/dL)',
           fontsize=12, # Y eksenindeki etiket font boyutu
           fontweight='light') # Y eksenindeki etiket font ağırlığı

# Legend (açıklama kutusu) ekleniyor
legend = plt.legend(title='Stroke Status',
                    labels=['No Stroke', 'Stroke'], # Stroke durumu
                    title_fontsize=12, # Başlık font boyutu
                    fontsize=10, # Yazı font boyutu
                    bbox_to_anchor=(1.02, 1), # Legend'ı grafik dışı
                    loc='upper left') # Legend konumu üst sol
legend.get_frame().set_alpha(0.9) # Legend kutusunun şeffaflık değ
legend.get_frame().set_edgecolor('white') # Legend kutusunun kenar

# Grafik üzerine grid (ızgara) ekleniyor
plt.grid(True, linestyle='--', alpha=0.3) # ızgaralar kesikli ve ş

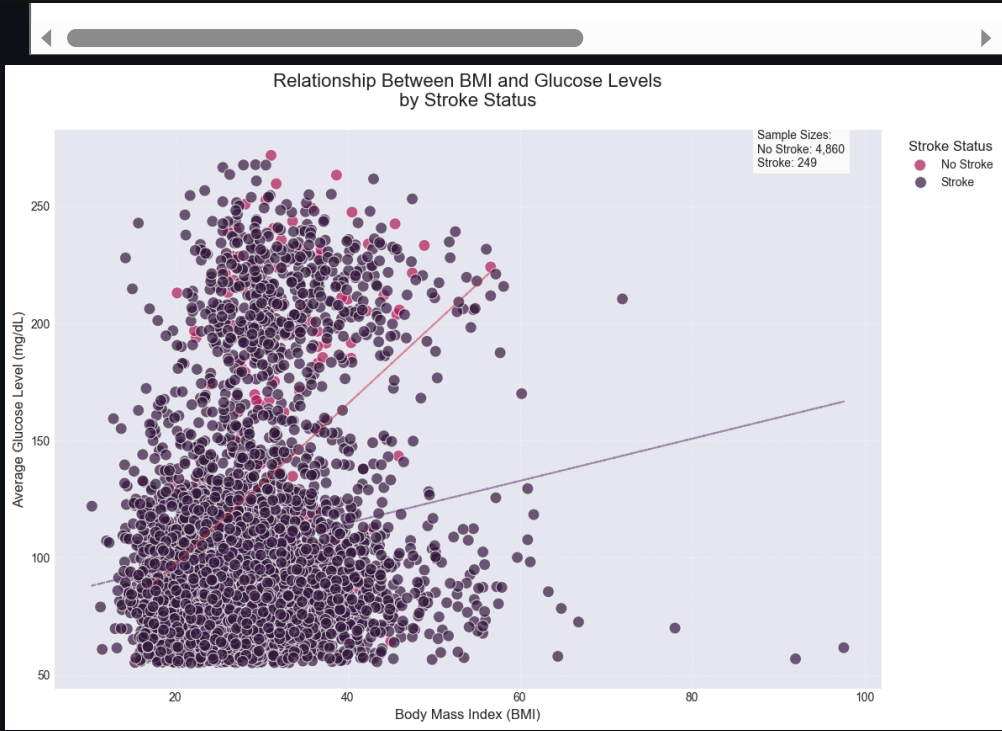
# Stroke durumu için örnek boyutları ekleniyor
stroke_counts = df['Stroke'].value_counts() # Stroke durumu sayıla
plt.text(0.85, 1, # Pozisyon ayarı (grafik dışına çok yakın yerleş
        f'Sample Sizes:\nNo Stroke: {stroke_counts[0]:,}\nStroke:
        transform=plt.gca().transAxes, # Eksenlere göre pozisyonl
        fontsize=10, # Yazı font boyutu
        verticalalignment='top', # Yazı dikey hizalama
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='none'))

# Grafik kenarlarını düzenliyor
scatter.spines['top'].set_visible(False) # Üst kenar görünmez olac
scatter.spines['right'].set_visible(False) # Sağ kenar görünmez ol

# Layout (düzen) sıkıştırılarak öğeler düzgün yerleştiriliyor
plt.tight_layout()

# Grafik gösteriliyor
plt.show()

```



BMI, Glikoz Seviyeleri ve İnme Riski Arasındaki İlişki 🧠🏥

Saçılım grafiği, **Vücut Kitle İndeksi (BMI)**, **glikoz seviyeleri** ve **inme riski** arasındaki ilişkiyi göstermektedir:

İnme Vakaları (Kırmızı Noktalar) 🔴:

- **Yüksek Glikoz Seviyeleri:** Yüksek glikoz seviyeleri, inme vakalarıyla güçlü bir ilişki göstermekte olup **önemli bir risk faktörüne** işaret etmektedir.
- **BMI ile İnme İlişkisi:** BMI ile inme arasındaki ilişki daha dağınık bir görünüm sergilemekle birlikte, yüksek BMI değerleri bazı durumlarda **risk faktörü** olarak katkı sağlayabilir.

İnme Olmayan Vakalar (Mavi Noktalar) 🔵:

- **Eşit Dağılım:** Bu vakalar, **BMI ve glikoz seviyeleri** genelinde daha eşit bir şekilde dağılmaktadır.

Gözlemler:

- **Glikoz Seviyeleri:** Yüksek glikoz seviyeleri, bağımsız bir şekilde **inme olasılığını artırıyor** gibi görünmekte ve bu durum, glikozun **önemli bir risk faktörü** olduğunu ortaya koymaktadır.
- **BMI:** BMI'nin etkisi daha az belirgin olmakla birlikte, yüksek glikoz seviyeleriyle bir araya geldiğinde **etkisini artırıcı** bir rol oynayabilir.

Bu analiz, glikoz seviyelerinin inme riski üzerindeki güçlü etkisini ve BMI ile birleştğinde riski nasıl artırabileceğini vurgulamaktadır. Grafik üzerinden daha detaylı incelemeler yapılarak, bu ilişkilerin sağlık üzerindeki etkileri

In []:

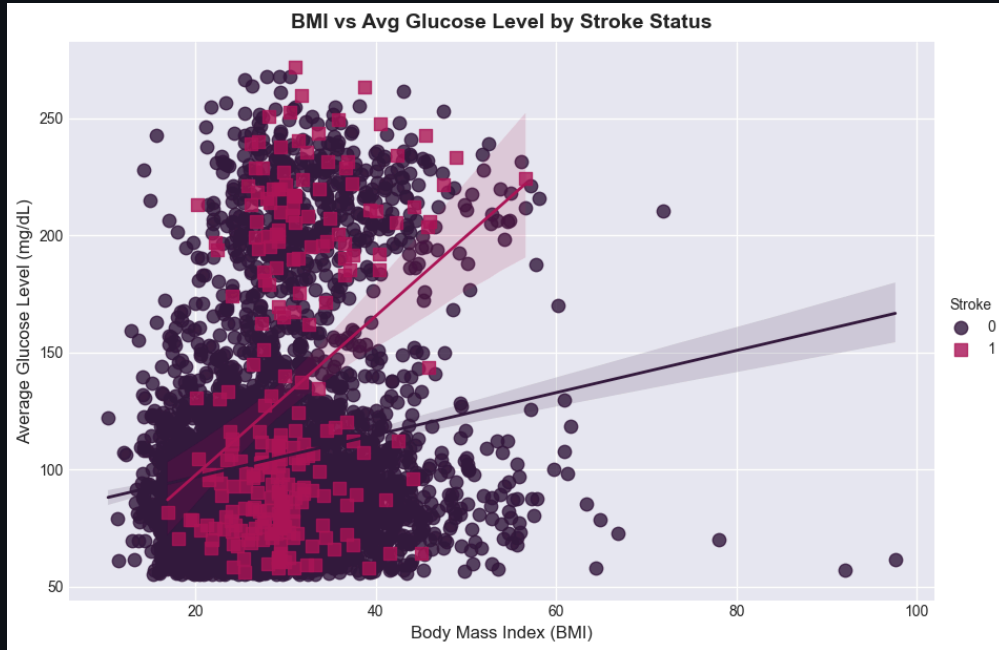
```
palette = {0: '#35193e', 1: '#ad1759'} # Stroke durumu için özel renk paleti

# Seaborn ile regresyon (lmplot) grafiği oluşturuluyor
g = sns.lmplot(
    data=df, # Veri kümesi
    x='BMI', # X ekseninde Body Mass Index (BMI)
    y='Avg_Glucose_Level', # Y ekseninde Ortalama Glukoz Seviyesi
    hue='Stroke', # Stroke durumuna göre renk ayrımı yapılacak (0: 0, 1: 1)
    height=6, # Grafik boyutunun yüksekliği
    aspect=1.5, # Grafik oranı (genişlik/yükseklik)
    markers=['o', 's'], # Stroke durumu için farklı işaretçiler (0: 'o', 1: 's')
    palette=palette, # Önceden tanımlanan renk paleti kullanılacak
    scatter_kws={'s': 80, 'alpha': 0.8}, # Noktaların boyutu (80)
    line_kws={'linewidth': 2} # Regresyon çizgisinin kalınlığı (2)
)

# Eksende etiketler ekleniyor
g.set_axis_labels("Body Mass Index (BMI)", "Average Glucose Level (mg/dL)")

# Grafik başlığı ekleniyor
g.fig.suptitle("BMI vs Avg Glucose Level by Stroke Status", y=1.02, color='red')

# Grafik gösteriliyor
plt.show()
```



Regresyon Çizgileri ve İnme Riski Arasındaki İlişki



Regresyon çizgileri, **İNME GEÇİREN** ve **GEÇİRMİYEN** bireyler için farklı eğilimler göstermektedir:

İNME VAKALARI (Stroke = 1) ●:

- **Daha Dik Pozitif Eğilim:** İnme geçiren bireyler için regresyon çizgisi daha dik bir **pozitif eğime** sahiptir. Bu durum, **yüksek BMI** ve **yüksek glikoz seviyelerinin** inme oluşumu ile daha güçlü bir ilişki gösterdiğini ortaya koymaktadır.
- **Çok Yüksek Değerler:** Çok yüksek BMI ve glikoz seviyelerine sahip bireylerin (grafikğin sağ üst köşesi) **Stroke = 1** grubunda daha yoğun olarak yer aldığı gözlemlenmektedir. Bu, bu faktörlerin birlikte inme riskini artırdığı hipotezini desteklemektedir.

İnme Olmayan Vakalar (Stroke = 0) ●:

- **Daha Hafif Eğilim:** İnme geçirmeyen bireyler için regresyon çizgisi daha **hafif bir eğime** sahiptir. Bu da **BMI, glikoz seviyeleri** ve inme olmaması arasındaki ilişkinin daha zayıf olduğunu göstermektedir.

Düşük Değerlerdeki Gözlemler 🌱:

- **Düşük BMI ve Glikoz Seviyeleri:** Düşük BMI ve glikoz seviyelerinde (grafikğin sol alt köşesi), iki grup arasında (Stroke = 0 ve Stroke = 1) **önemli bir örtüşme** bulunmaktadır. Bu durum, bu faktörlerin düşük seviyelerde, bağımsız ya da birlikte inme riski üzerinde güçlü bir etkiye sahip olmayabileceğini göstermektedir.

Aykırı Değerler ⚠:

- **Aşırı Yüksek Değerler:** Her iki grupta da aşırı yüksek BMI ve/veya glikoz seviyelerine sahip birkaç **aykırı değer** bulunmaktadır. Bu vakaların analize olan etkisini anlamak için daha fazla **araştırma yapılması gerekebilir**.

Sonuç:

- **Daha Dik Regresyon Çizgisi:** **Stroke = 1** için regresyon çizgisinin daha dik olması, **BMI ve glikoz seviyelerinin birlikte** (birleşik olarak) inme riski ile daha güçlü bir ilişkiye sahip olduğunu göstermektedir. Bu, bu faktörlerin birlikte **kritik risk faktörleri** olarak önemini vurgulamaktadır.

Can smoking increase the likelihood of stroke?

Sigara içmek felç geçirme olasılığını artırabilir mi?

```
In [36]: df['Smoking_Status'].value_counts()
```

```
Out[36]: Smoking_Status
never smoked    1892
```

```
Unknown      1544
formerly smoked  884
smokes        789
Name: count, dtype: int64
```

In []:

```
df_filtered = df[df['Age'] >= 15] # Yaşı 15 ve üzerinde olan bir

# 2 grafik için bir figür oluşturuluyor (yan yana)
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Renk paleti tanımlanıyor: Stroke durumuna göre renkler
palette = {0: '#35193e', 1: '#ad1759'}

# İlk grafikte (Tüm veriler için) Sigara içme durumu ve Stroke il
sns.countplot(data=df, x='Smoking_Status', hue='Stroke', ax=axes[0])
axes[0].set_title('Stroke Occurrence by Smoking Status (All Data)')
axes[0].set_xlabel('Smoking Status') # X eksen etiketi
axes[0].set_ylabel('Count') # Y eksen etiketi
axes[0].tick_params(axis='x', rotation=45) # X eksen etiketleri

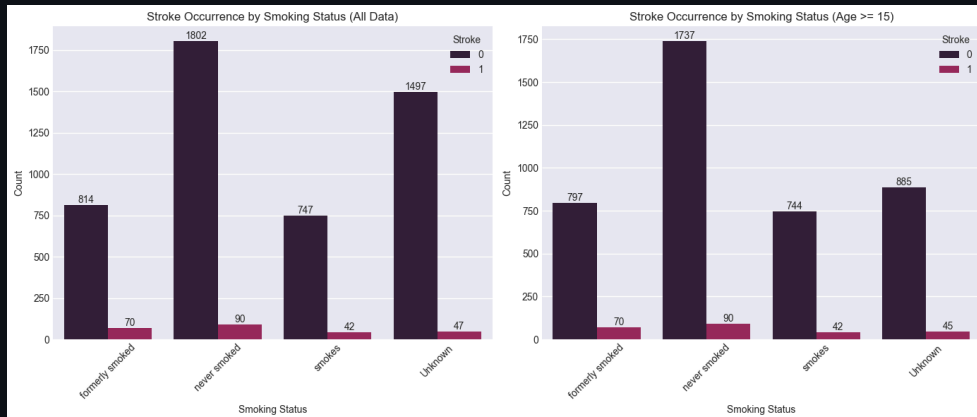
# Her çubuğun üzerine değer etiketleri ekleniyor
for container in axes[0].containers:
    axes[0].bar_label(container, fmt='%d')

# İkinci grafikte (Yaşı 15 ve üstü bireyler için) Sigara içme duru
sns.countplot(data=df_filtered, x='Smoking_Status', hue='Stroke',
              ax=axes[1])
axes[1].set_title('Stroke Occurrence by Smoking Status (Age >= 15)')
axes[1].set_xlabel('Smoking Status') # X eksen etiketi
axes[1].set_ylabel('Count') # Y eksen etiketi
axes[1].tick_params(axis='x', rotation=45) # X eksen etiketleri

# Her çubuğun üzerine değer etiketleri ekleniyor
for container in axes[1].containers:
    axes[1].bar_label(container, fmt='%d')

# Grafikler arasındaki boşluk düzenleniyor
plt.tight_layout()

# Grafik gösteriliyor
plt.show()
```



Genel Karşılaştırma ve Yorum 🧠 📊

Popülasyon Değişimi:

- 15 Yaş Altı Bireyler Çıkarıldığında:

- Unknown kategorisindeki bireylerin sayısında **ciddi bir artış** görülmektedir. Bu, yaş grubu kısıtlamasının veriler üzerindeki etkisini açıkça göstermektedir.
- Diğer kategorilerde, özellikle "Formerly smoked" ve "Smokes" kategorilerinde, popülasyon daha **az etkilenmektedir**. Bu da yaş grubu kısıtlamasının bu kategorilerde daha az etkili olduğunu gösteriyor.

İnme Oranları:

- **İnme Geçiren Bireyler:**
 - Tüm kategorilerde inme yaşayan bireylerin sayısı **düşük** kalmaktadır. Bu, popülasyondaki inme vakalarının genel olarak sınırlı olduğunu gösteriyor.
 - Yaş grubu kısıtlaması uygulandığında, kategoriler arasında **ciddi bir farklılık** yaratılmamaktadır. Yani, yaş faktörü inme oranlarında önemli bir değişiklik oluşturmamaktadır.

Sigara İçmeyenler ("Never smoked"):

- **En Büyük Popülasyon:** Hem tüm veri hem de **yaş ≥ 15** için, en büyük popülasyon **sigara içmeyenler** kategorisindedir. Bu, sigara içmemenin sağlık üzerindeki etkilerinin, özellikle inme ve diğer sağlık problemleri açısından önem taşıdığını göstermektedir.

Do individuals with heart disease face a higher stroke risk?

Kalp hastalığı olan kişilerde felç riski daha mı yüksektir?

```
In [ ]: # 'Heart_Disease' sütununa göre gruplama yaparak, her grup için  
grouped = df.groupby('Heart_Disease')['Stroke'].mean().reset_index()
```

```
In [39]: grouped
```

```
Out[39]:
```

| | Heart_Disease | Stroke |
|---|---------------|----------|
| 0 | 0 | 0.041796 |
| 1 | 1 | 0.170290 |

```
In [ ]: df_grouped = df.groupby(['Heart_Disease', 'Stroke']).size().unstack()
```

```
In [ ]: df_grouped_percent = df_grouped.div(df_grouped.sum(axis=1), axis=1)
```

```
In [ ]: labels = ['No Heart Disease', 'Heart Disease'] # Kalp hastalığı
        sizes = df_grouped_percent[1] # 'Heart Disease' durumu için yüz
        colors = ['#35193e', '#ad1759'] # Pasta grafik renk paleti
        explode = [0, 0.1] # Sadece ikinci dilimi (kalp hastalığı olan)

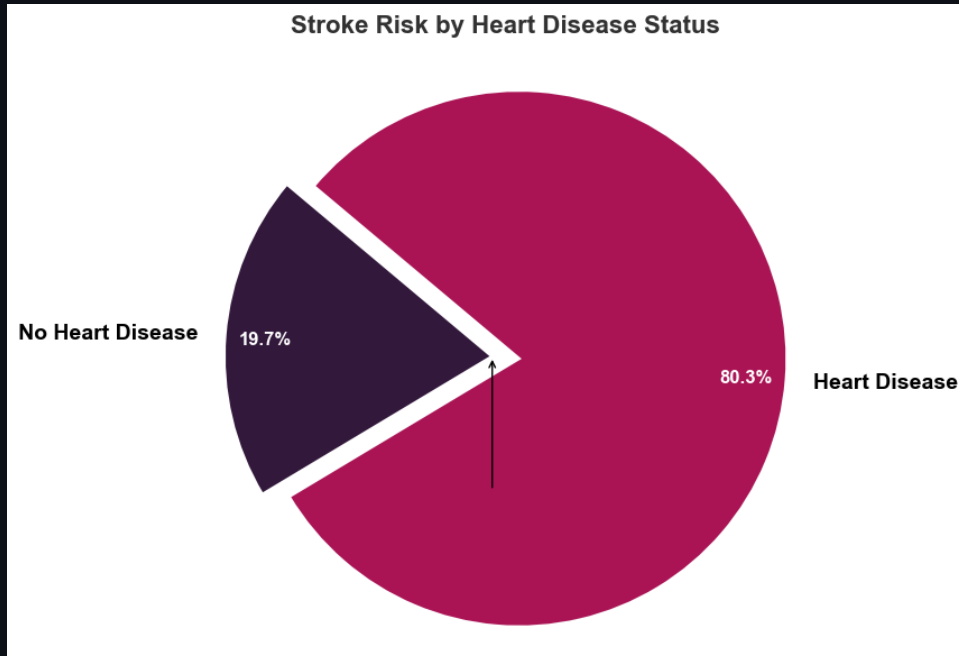
fig, ax = plt.subplots(figsize=(8, 6)) # Grafik boyutu
wedges, texts, autotexts = ax.pie(
    sizes,
    labels=labels,
    autopct='%1.1f%%', # Yüzdelik değerlerin gösterilmesi
    startangle=140, # Pasta grafiğın başlama açısı
    colors=colors, # Pasta dilimlerinin renkleri
    explode=explode, # Hangi dilimin patlatılacağı
    pctdistance=0.85, # Yüzdelik yazıların konumu
    wedgeprops={'edgecolor': 'white', 'linewidth': 2, 'linestyle'
)

plt.setp(autotexts, size=12, weight="bold", color='white') # Yü
plt.setp(texts, size=14, weight="bold", color='black') # Etiket
ax.set_title('Stroke Risk by Heart Disease Status', fontsize=16,

ax.annotate('', xy=(0, 0), xytext=(0, -0.5), # Anlamlı bir ok e
            arrowprops=dict(facecolor='black', edgecolor='black'

ax.axis('equal') # Pasta grafik dairesel olmalı

plt.tight_layout() # Layout'u sıkıştırarak düzgün hale getir
plt.show() # Grafiği göster
```



Kalp Hastalığı ve İnme Arasındaki İlişki



- Kalp hastalığı olan bireylerde %60 daha fazla inme geçirme riski bulunmaktadır.
- Bu, kalp hastalığı olan bireylerin inme riski **%300 artmış** demektir.

- **Kalp hastalığı ile inme arasında güçlü bir bağlantı** olduğundan, özellikle kalp hastalığı olan bireylerin **düzenli sağlık kontrollerinden geçmesi** ve **önleyici tedbirlerin alınması** gereklidir.

Does work-related stress contribute to hypertension and subsequently to strokes?

İş kaynaklı stres hipertansiyona ve sonrasında inmelere sebep oluyor mu?

```
In [ ]: grouped = df.groupby('Work_Type')[['Hypertension', 'Stroke']].r
```

```
In [44]: grouped
```

```
Out[44]:
```

| | Hypertension | Stroke |
|---------------|--------------|----------|
| Work_Type | | |
| Govt_job | 0.111111 | 0.050228 |
| Never_worked | 0.000000 | 0.000000 |
| Private | 0.096101 | 0.050958 |
| Self-employed | 0.175824 | 0.079365 |
| children | 0.000000 | 0.002911 |

```
In [ ]: stress_analysis = round(df.groupby('Work_Type')[['Hypertension', 'Stroke']].mean())
stress_analysis = stress_analysis.rename(columns={'Hypertension': 'Hypertension_Rate', 'Stroke': 'Stroke_Rate'})

fig = px.imshow(stress_analysis.T,
                 labels={'x': 'Work Type', 'y': 'Indicator'}, # X eksenine 'Work_Type' yazılacak, Y eksenine 'Indicator' yazılacak
                 x=stress_analysis.index, # X eksenine 'Work_Type' yazılacak, Y eksenine 'Indicator' yazılacak
                 y=stress_analysis.columns, # Y eksenine 'Hypertension_Rate' yazılacak, X eksenine 'Stroke_Rate' yazılacak
                 color_continuous_scale=[ '#35193e', '#ad1759' ], # Renk skalasını belirleyeceğiz
                 color_continuous_midpoint=50, # Renk skalasının ortasını belirleyeceğiz
                 title='High Blood Pressure and Stroke Rates by Work Type', # Grafik başlığını belirleyeceğiz
                 text_auto=True, # Değerlerin üzerine yazılmasını belirleyeceğiz
                 height=600, width=800) # Grafik boyutlarını ayarlayacağız

fig.update_layout(
    title_font_size=24, # Başlık font boyutunu ayarlama
    title_font_family="Arial, sans-serif", # Başlık fontunu belirleme
    title_font_color='black', # Başlık font rengini siyah yapma
    xaxis_title_font_size=14, # X eksen başlık font boyutunu ayarlama
    yaxis_title_font_size=14, # Y eksen başlık font boyutunu ayarlama
    font=dict(family="Arial, sans-serif", size=12, color='black'), # Font ayarları
    coloraxis_colorbar=dict(title='Oran (%)', tickvals=[0, 50, 100]) # Renk çubuğu başlığı ve değerleri
)
```

```
fig.show() # Grafik görüntüleme
```

Hipertansiyon ile İnme Arasındaki İlişki



- **Hipertansiyon oranı yüksek olan gruplarda, inme oranı** da daha yüksektir. Bu durum, hipertansiyonun **inme için bir risk faktörü** olduğunu göstermektedir.
- **Self-employed** (Serbest meslek) grubu, **hipertansiyon** (%17.58) ve **inme** (%7.94) oranlarıyla her iki oranın da en yüksek olduğu gruptur.
- **Çocuklar** grubunda ise, **hipertansiyon** oranı %0 ve **inme** oranı ise %0.3 ile en düşük risk grubudur.

Sektörlerin Birbirine Göre Değerlendirilmesi 🏢 🩺

Serbest Meslek (Self-employed):

- **En riskli grup** olarak değerlendirilmektedir. Hem **hipertansiyon** hem de **inme** oranları diğer sektörlerle göre en yüksektir. Bu durum, düzensiz yaşam tarzı, yüksek stres ve sağlık kontrollerinin eksikliği ile ilişkilendirilebilir.

Devlet Çalışanları (Govt_job) ve Özel Sektör (Private):

- Bu iki grup birbirine oldukça benzer. **Hipertansiyon** oranında devlet çalışanları biraz daha yüksekken (%11.11'e %9.61), **inme** oranında özel sektör çalışanları biraz daha yüksek (%5.09'a %5.02). Bu gruplar **orta riskli** olarak değerlendirilebilir.

Hiç Çalışmamış (Never_worked) ve Çocuklar (Children):

- Bu iki grup, **en düşük riskli** gruplardır. Ancak çocuklarda inme oranının **%0.29** olması, dikkat edilmesi gereken bir durumdur.

```
In [46]: df.Work_Type.value_counts()
```

```
Out[46]: Work_Type
Private      2924
Self-employed  819
children      687
```

```
Govt_job      657
Never_worked  22
Name: count, dtype: int64
```

In []:

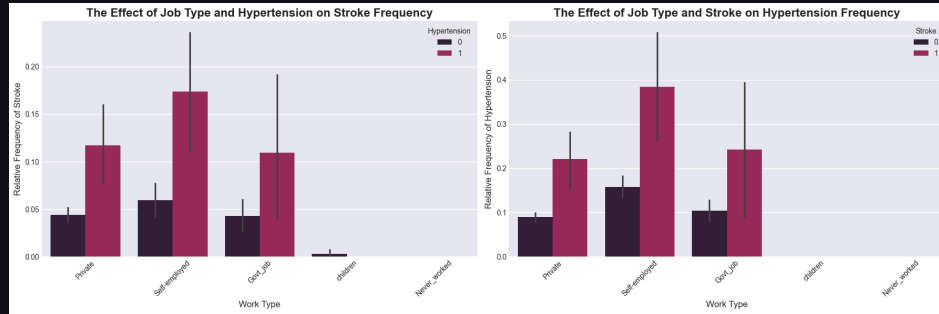
```
palette = {0: '#35193e', 1: '#ad1759'} # Renk paleti oluřturuldu

fig, axes = plt.subplots(1, 2, figsize=(18, 6)) # 1 satır 2 sütun

# İlk barplot: 'Work_Type' ve 'Hypertension' ile 'Stroke' sıklıęını g rselleřtirmek
sns.barplot(x='Work_Type', y='Stroke', hue='Hypertension', data=df)
axes[0].set_title('The Effect of Job Type and Hypertension on Stroke Frequency')
axes[0].set_xlabel('Work Type', fontsize=12) # X eksenine etiket
axes[0].set_ylabel('Relative Frequency of Stroke', fontsize=12)
axes[0].tick_params(axis='x', rotation=45) # X eksenindeki etiketler 45 derece

# İkinci barplot: 'Work_Type' ve 'Stroke' ile 'Hypertension' sıklıęını g rselleřtirmek
sns.barplot(
    x='Work_Type',
    y='Hypertension',
    hue='Stroke',
    data=df,
    palette=palette,
    ax=axes[1]
)
axes[1].set_title('The Effect of Job Type and Stroke on Hypertension Frequency')
axes[1].set_xlabel('Work Type', fontsize=12) # X eksenine etiket
axes[1].set_ylabel('Relative Frequency of Hypertension', fontsize=12)
axes[1].tick_params(axis='x', rotation=45) # X eksenindeki etiketler 45 derece

plt.tight_layout() # Grafiklerin d zenini sıkıřtırarak optimize etme
plt.show() # Grafiklerin g sterilmesi
```



G rselleřtirme: Inme ve Hipertansiyon İliřkisi

- Yukarıda verilen heatmap ile g sterilen iliřki matrisi, bu barchart grafięi ile g rselleřtirilmiřtir. Bu grafik, **hipertansiyonu olan ve olmayan bireylerde**, farklı meslek gruplarına g re **inme g r lme sıklıęını** (g receli oranlarla) karřılařtırmaktadır.
- Ayrıca, bu grafik, farklı **iř t rlerinde** inme durumuna g re **hipertansiyon g r lme sıklıęını** karřılařtırmaktadır.

Are males at higher risk of strokes due to work-related or lifestyle stress?

Erkeklerde iş kaynaklı veya yaşam tarzı kaynaklı stres nedeniyle felç geçirme riski daha mı yüksek?

```
In [ ]: grouped_work_stress_male = df[df['Gender'] == 'Male'].groupby(
# Erkek (Male) cinsiyetindeki bireylerin iş türüne göre 'Stroke Risk'

grouped_lifestyle_stress_male = df[df['Gender'] == 'Male'].groupby(
# Erkek (Male) cinsiyetindeki bireylerin sigara içme durumu (Smoking Status) göre 'Stroke Risk'
```

```
In [ ]: df_male = df[df['Gender'] == 'Male']
# Cinsiyeti 'Male' (Erkek) olan bireylerden oluşan yeni bir veri seti

df_male_grouped = df_male.groupby(['Work_Type', 'Residence_Type'])
# Erkek bireyleri, iş türü (Work_Type), yaşam alanı türü (Residence_Type)

df_male_grouped_percent = df_male_grouped.div(df_male_grouped.groupby(['Work_Type', 'Residence_Type']).size())
# Her gruptaki stroke oranlarını yüzdelik değeriyle hesaplıyoruz

fig, axes = plt.subplots(2, 2, figsize=(16, 12))
# 2 satır 2 sütun şeklinde 4 alt grafik için figür ve eksenler

sns.barplot(x=df_male_grouped_percent.index.get_level_values('Residence_Type'),
            y=df_male_grouped_percent['Stroke_Risk'],
            hue=df_male_grouped_percent.index.get_level_values('Work_Type'),
            palette='rocket', ax=axes[0, 0])
# Bar plot ile, yaşama alanı türü (Residence_Type) ve iş türü (Work_Type)

axes[0, 0].set_title('Stroke Risk by Residence and Work Type')
axes[0, 0].set_xlabel('Residence Type', fontsize=12)
axes[0, 0].set_ylabel('Percentage of Stroke (%)', fontsize=12)
axes[0, 0].set_xticklabels(['Urban', 'Rural'], rotation=45)
# Grafiğin başlığı, x ve y eksenlerinin etiketlerini ayarlıyoruz

sns.barplot(x=df_male_grouped_percent.index.get_level_values('Smoking_Status'),
            y=df_male_grouped_percent['Stroke_Risk'],
            hue=df_male_grouped_percent.index.get_level_values('Work_Type'),
            palette='rocket', ax=axes[0, 1])
# Sigara içme durumu (Smoking_Status) ve iş türü (Work_Type)

axes[0, 1].set_title('Stroke Risk by Smoking Status and Work Type')
axes[0, 1].set_xlabel('Smoking Status', fontsize=12)
axes[0, 1].set_ylabel('Percentage of Stroke (%)', fontsize=12)

sns.barplot(x=df_male_grouped_percent.index.get_level_values('BMI_Group'),
            y=df_male_grouped_percent['Stroke_Risk'],
            hue=df_male_grouped_percent.index.get_level_values('Work_Type'),
            palette='rocket', ax=axes[1, 0])
# BMI kategorisi (BMI_Group) ve iş türü (Work_Type) ile stroke riski

axes[1, 0].set_title('Stroke Risk by BMI Category and Work Type')
axes[1, 0].set_xlabel('BMI Category', fontsize=12)
axes[1, 0].set_ylabel('Percentage of Stroke (%)', fontsize=12)

sns.barplot(x=df_male_grouped_percent.index.get_level_values('Work_Type'),
            y=df_male_grouped_percent['Stroke_Risk'],
            hue=df_male_grouped_percent.index.get_level_values('Residence_Type'),
            palette='rocket', ax=axes[1, 1])
# İş türü (Work_Type) ve yaşam alanı türü (Residence_Type) ile stroke riski
```

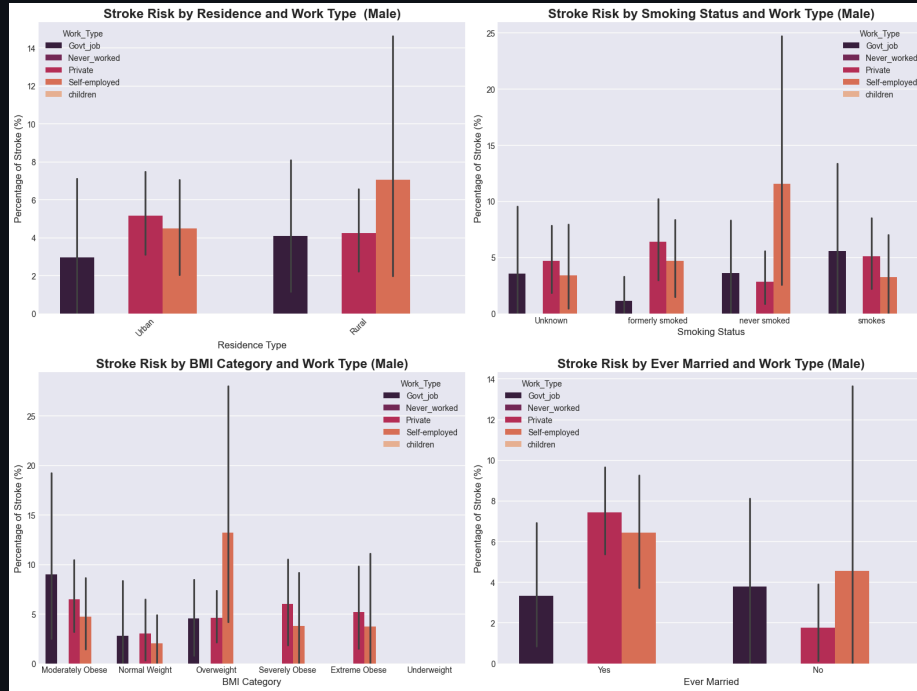
```

palette=rocket , ax=axes[1, 1])
# Evlilik durumu (Ever_Married) ve iş türü (Work_Type) ile st

axes[1, 1].set_title('Stroke Risk by Ever Married and Work Ty
axes[1, 1].set_xlabel('Ever Married', fontsize=12)
axes[1, 1].set_ylabel('Percentage of Stroke (%)', fontsize=12

plt.tight_layout()
plt.show()
# Alt grafiklerin düzenini sıkıştırıyor ve görseli gösteriyor

```



Stroke Risk by Residence and Work Type (Male) 🌍👤

1. Kırsal Bölgelerde (Rural):

- Kendi işinde çalışan erkeklerin (Self-employed) **stroke riski**, diğer iş türlerine göre daha yüksektir.

2. Kentsel Bölgelerde (Urban):

- Stroke riski** daha dengeli dağılım göstermektedir, ancak yine **Self-employed** grubunda daha yüksek risk görülmektedir.

3. Children ve Never_worked Grubu:

- Bu gruplarda **stroke riski** oldukça düşüktür.

Genel Yorum:

- Kırsal bölgede** yaşayan ve kendi işinde çalışan erkekler, daha yüksek stroke riski taşımaktadır.

Stroke Risk by Smoking Status and Work Type (Male) 🚬👤

Work Type (Male) 🏢

1. Sigara İçmeyenler (Never Smoked):

- Kendi işinde çalışan erkeklerde **stroke riski**, diğer gruplara göre daha yüksektir.

2. Eski Sigara İçenler (Formerly Smoked):

- Stroke riski **Self-employed** grubunda daha belirgindir.

3. Aktif Sigara İçenler (Smokes):

- **Stroke riski** yine **Self-employed** grubunda daha belirgindir.

4. Children Grubu:

- Tüm sigara içme durumlarında çok düşük stroke riski taşımaktadır.

Genel Yorum:

- **Sigara içme durumu**, özellikle **kendi işinde çalışan erkekler** için **stroke riskini artırmaktadır**.

Stroke Risk by BMI Category and Work Type (Male) ⚖️ 🏢

1. Overweight (Kilolu) ve Moderately Obese (Hafif Obez):

- Kendi işinde çalışan erkekler, **stroke riski açısından en riskli gruptur**.

2. Diğer BMI Kategorileri (Severely Obese, Extreme Obese, Underweight):

- **Stroke riski** nispeten daha düşüktür ve dengelidir.

3. Children Kategorisi:

- Her BMI seviyesinde düşük stroke riski taşımaktadır.

Genel Yorum:

- **Aşırı kilolu** veya **hafif obez bireylerde, iş türü stroke riski üzerinde önemli bir etki yapmaktadır**; özellikle **Self-employed** grubunda risk daha yüksektir.

Stroke Risk by Ever Married and Work Type (Male) 👤 🏢

1. Evlilik Durumu "Evet" (Yes):

- **Self-employed** ve **Private** iş türlerinde, **stroke riski** daha

yüksekli.

2. Evlilik Durumu "Hayır" (No):

- **Stroke riski** genel olarak düşüktür, ancak **Self-employed** kategorisinde yine diğer gruplara göre daha belirgindir.

3. Children Kategorisi:

- Evlilik durumuna göre stroke riski oldukça düşüktür.

Genel Yorum:

- **Evlilik durumu**, özellikle **kendi işinde çalışan erkeklerde** **stroke riskini artıran bir faktör olabilir.**

How does the combination of risk factors amplify stroke likelihood?

Risk faktörlerinin birleşimi felç olasılığını nasıl artırıyor?

```
In [ ]: df['Life_Style_Risk_Factor'] = df['Ever_Married'] + '_' + df  
# 'Ever_Married', 'Smoking_Status' ve 'Residence_Type' sütun
```

```
In [51]: df['Life_Style_Risk_Factor']
```

```
Out[51]: 0      Yes_formerly smoked_Urban  
1      Yes_never   smoked_Rural  
2      Yes_never   smoked_Rural  
3              Yes_smokes_Urban  
4      Yes_never   smoked_Rural  
...  
5105     Yes_never   smoked_Urban  
5106     Yes_never   smoked_Urban  
5107     Yes_never   smoked_Rural  
5108  Yes_formerly   smoked_Rural  
5109              Yes_Unknown_Urban  
Name: Life_Style_Risk_Factor, Length: 5109, dtype: object
```

```
In [ ]: df_40 = df[df['Age'] >= 40]  
# Yaşı 40 ve daha büyük olan verileri içeren bir alt küme ol  
  
df['Composite_Variable'] = df_40['Ever_Married'] + '_' + df_  
# Yaşı 40 ve üzerindeki bireylerin 'Ever_Married', 'Smoking_
```

```
In [ ]: composit_counts_1 = pd.crosstab(df['Composite_Variable'], df  
# 'Composite_Variable' ile 'Stroke' sütunları arasındaki fre  
  
composit_percent_1 = composit_counts_1.div(composit_counts_1  
# Bu frekans tablosunun her satırını, satır toplamına bölere  
  
composit_counts_2 = pd.crosstab(df['Life_Style_Risk_Factor']
```

```
# 'Life_Style_Risk_Factor' ile 'Stroke' sütunları arasındaki ilişkiyi analiz ediyoruz.

composit_percent_2 = composit_counts_2.div(composit_counts_2.sum(axis=1))
# Yine bu tablonun her satırını, satır toplamına bölerek yüzde olarak ifade ediyoruz.

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
# 1 satır, 2 sütundan oluşan bir grafik alanı oluşturur.

sns.heatmap(composit_percent_1, annot=True, fmt=".2f", cmap=cm.viridis)
# İlk heatmap'i çiziyor, 40 yaş ve üzeri kişilerin yaşam tarzı ile stroke riski arasındaki ilişkiyi gösteriyor.

axes[1].set_title('Heatmap of Life Style vs Stroke (Percentage)')
# Grafiğe başlık ekler.

axes[1].set_xlabel('Stroke For Over 40', fontsize=12)
# X ekseninin etiketini ayarlar.

axes[1].set_ylabel('Life Style', fontsize=12)
# Y ekseninin etiketini ayarlar (boş bırakılır).

sns.heatmap(composit_percent_2, annot=True, fmt=".2f", cmap=cm.viridis)
# İkinci heatmap'i çiziyor, tüm veri setine göre yaşam tarzı ile stroke riski arasındaki ilişkiyi gösteriyor.

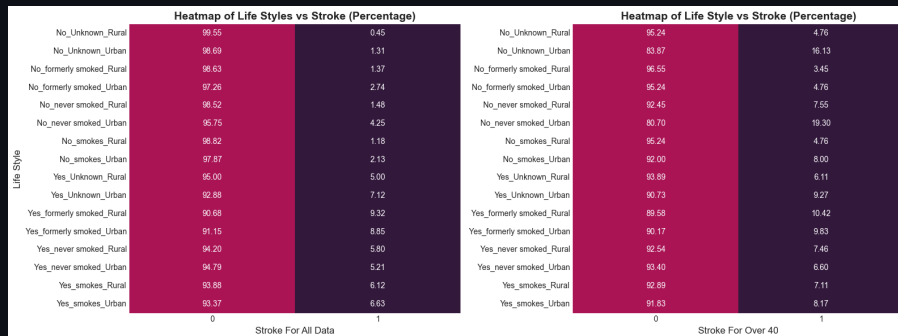
axes[0].set_title('Heatmap of Life Styles vs Stroke (Percentage)')
# Grafiğe başlık ekler.

axes[0].set_xlabel('Stroke For All Data', fontsize=12)
# X ekseninin etiketini ayarlar.

axes[0].set_ylabel('Life Style', fontsize=12)
# Y ekseninin etiketini ayarlar.

plt.tight_layout()
# Grafik elemanlarını sıkıştırır, birbirine yakın olmasını sağlar.

plt.show()
# Grafiği görüntüler.
```



Stroke Risk by Age and Lifestyle Analysis 🧠👨

1. Stroke = 0 (İnme Olmayanlar):

- **Tüm veri grubu ile 40 yaş üzeri grup** arasında çok büyük farklar gözlemlenmemektedir. Bu, yaşın inme riski üzerinde belirgin bir etkisi olmadığını göstermektedir.

2. Stroke = 1 (İnme Geçirenler):

- **40 yaş üzeri grup için tüm ilrestyle kategorilerinin** inme riski artmaktadır. Bu, yaşın ilerlemesiyle birlikte yaşam tarzı faktörlerinin inme riskini daha belirgin şekilde artırdığına işaret etmektedir.

Grafikle İlişkili Yorum:

- **Tüm veri grubu ve 40 yaş üzeri grup** arasındaki farkları daha iyi anlayabilmek için aşağıdaki **bar grafiklerinin** oluşturulması sağlanmıştır.

```
In [ ]: composit_percent_1[1] / composit_percent_2[1]
# composit_percent_1 ve composit_percent_2 DataFrame'lerinde
# Sonuç, her bir 'Composite_Variable' ve 'Life_Style_Risk_Fa
```

```
Out[ ]: Composite_Variable
No_Unknown_Rural      10.523810
No_Unknown_Urban      12.338710
No_formerly_smoked_Rural  2.517241
No_formerly_smoked_Urban  1.738095
No_never_smoked_Rural   5.113208
No_never_smoked_Urban   4.543860
No_smokes_Rural        4.047619
No_smokes_Urban        3.760000
Yes_Unknown_Rural      1.221374
Yes_Unknown_Urban      1.302419
Yes_formerly_smoked_Rural  1.118152
Yes_formerly_smoked_Urban  1.109827
Yes_never_smoked_Rural  1.287313
Yes_never_smoked_Urban  1.267573
Yes_smokes_Rural       1.162531
Yes_smokes_Urban       1.233109
Name: 1, dtype: float64
```

```
In [ ]: # Oranları hesapla: Stroke = 1 (İnme) için oran
ratio_percent_1 = (composit_percent_1[1] / composit_percent_2[1])

# Oranları hesapla: Stroke = 0 (İnme Olmayan) için oran
ratio_percent_2 = (composit_percent_1[0] / composit_percent_2[0])

# Ratio_df_1 DataFrame oluştur: Stroke = 1 oranlarının karşılaştırmaları
ratio_df_1 = pd.DataFrame({
    'Category': composit_percent_1.index, # Kategori isimleri
    'Ratio (%)': ratio_percent_1 # Hesaplanan oranları al
}).reset_index(drop=True)

# Ratio_df_2 DataFrame oluştur: Stroke = 0 oranlarının karşılaştırmaları
ratio_df_2 = pd.DataFrame({
    'Category': composit_percent_1.index, # Kategori isimleri
    'Ratio (%)': ratio_percent_2 # Hesaplanan oranları al
}).reset_index(drop=True)

# İki grafik oluşturmak için alt subplotlar oluştur
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# İlk grafiği çiz: Stroke = 1 (İnme olan bireyler) oranlarının karşılaştırmaları
sns.barplot(data=ratio_df_1, x='Category', y='Ratio (%)', palette='magma')
```

```

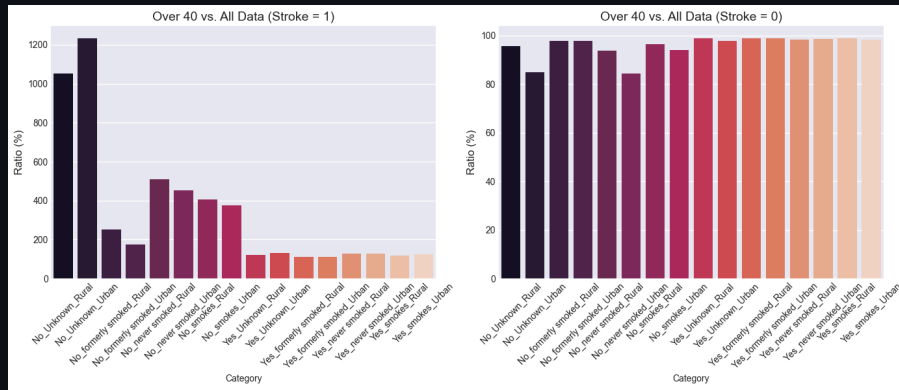
axes[0].set_title('Over 40 vs. All Data (Stroke = 1)', fontsize=12)
axes[0].set_xlabel('Category', fontsize=10) # X eksenini etiketle
axes[0].set_ylabel('Ratio (%)', fontsize=12) # Y eksenini etiketle
axes[0].tick_params(axis='x', rotation=45) # X eksenindeki etiketleri döndür

# İkinci grafiği çiz: Stroke = 0 (İnme olmayan bireyler) oranı
sns.barplot(data=ratio_df_2, x='Category', y='Ratio (%)', palette='magma')
axes[1].set_title('Over 40 vs. All Data (Stroke = 0)', fontsize=12)
axes[1].set_xlabel('Category', fontsize=10) # X eksenini etiketle
axes[1].set_ylabel('Ratio (%)', fontsize=12) # Y eksenini etiketle
axes[1].tick_params(axis='x', rotation=45) # X eksenindeki etiketleri döndür

# Grafiklerin düzenini sıkıştırarak düzenle
plt.tight_layout()

# Grafikleri göster
plt.show()

```



40 Yaş ve Üstü Grubunun İncelenmesi



İncelemelerimiz sonucunda, **40 yaş ve üzeri** yaş grubunun **stroke riskinin yüksek** olduğu bilgisinden yola çıkarak, bu grubu daha detaylı incelemeye karar verdik. Bu nedenle, **tüm veri** ve **40 yaş üzeri** olarak iki farklı **heatmap** oluşturduk.

Detaylı İncelemeler:

- **40 yaş üzeri grup**, tüm yaş grupları göz önünde bulundurulduğunda, **neredeyse bütün kategorilerde** çok daha fazla **inme riski** taşıdığı gözlemlenmiştir.
- Bu bulgu, yaşın ilerlemesiyle birlikte inme riskinin arttığını ve bu gruptaki bireylerin sağlık durumlarına özel önlemler alınması gerektiğini vurgulamaktadır.

In [56]:

```
new_df = df.copy()
```

In []:

```

# 'Hypertension' sütunundaki 0'ı 'No', 1'i ise 'Yes' ile değ
new_df['Hypertension'] = new_df['Hypertension'].replace({0:

```

```
# 'Heart_Disease' sütunundaki 0'ı 'No', 1'i ise 'Yes' ile değiştir
new_df['Heart_Disease'] = new_df['Heart_Disease'].replace({0: 'No', 1: 'Yes'})
```

```
In [ ]: new_df['Health_Risk_Factors'] = new_df['Hypertension'] + ' ' + new_df['Cholesterol']
```

```
In [ ]: composit_counts2 = pd.crosstab(new_df['Health_Risk_Factors'], new_df['Heart_Disease'])
```

```
In [ ]: composit_percent2 = composit_counts2.div(composit_counts2.sum(axis=1), axis=1)
```

```
In [61]: custom_colors = ['#35193e', '#ad1759']

# Verileri uzun formata dönüştür
composit_percent2_long = composit_percent2.reset_index().melt(
    id_vars='Health_Risk_Factors',
    var_name='Stroke',
    value_name='Percentage'
)

# Alt grafikler için figür oluştur
fig = make_subplots(
    rows=2, cols=1,
    subplot_titles=(
        'Health Risk Factors vs Stroke Distribution',
        'Stroke Risk Analysis by Health Factors'
    ),
    vertical_spacing=0.3, # Grafipler arası boşluğu artır
    specs=[[{"secondary_y": True}], [{"secondary_y": True}]]
)

# 1. Bar grafiği oluştur
bar_fig = px.bar(
    composit_percent2_long,
    x='Health_Risk_Factors',
    y='Percentage',
    color='Stroke',
    barmode='group',
    text='Percentage',
    labels={'Health_Risk_Factors': 'Health Risk Factors', 'Percentage': 'Percentage'},
    color_discrete_sequence=custom_colors
)

# Bar grafiği trace'lerini ekle
for trace in bar_fig.data:
    fig.add_trace(trace, row=1, col=1)

# Bar grafiği metin ve düzen ayarları
fig.update_traces(
    texttemplate='%{y:.1f}%',
    textposition='outside',
    textfont=dict(size=10),
    row=1, col=1
)

# 2. Scatter plot oluştur
scatter_fig = px.scatter(
```

```

    composit_percent2_long,
    x='Health_Risk_Factors',
    y='Stroke',
    size='Percentage',
    color='Health_Risk_Factors',
    labels={'Health_Risk_Factors': 'Health Risk Factors', 'Stroke': 'Stroke Status'},
    color_discrete_sequence=custom_colors
)

# Scatter plot trace'lerini ekle
for trace in scatter_fig.data:
    fig.add_trace(trace, row=2, col=1)

# Genel düzen ayarları
fig.update_layout(
    height=1000,
    title={
        'text': 'Health Risk Factors and Stroke Analysis',
        'y': 0.95,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top',
        'font': dict(size=24)
    },
    template='plotly_white',
    showlegend=True,
    legend=dict(
        yanchor="top",
        y=0.45, # İlk grafiğin altına yerleştir
        xanchor="left",
        x=1.05, # Grafiğin sağına yerleştir
        title="Stroke Status",
        bgcolor="rgba(255, 255, 255, 0.8)",
        bordercolor="lightgray",
        borderwidth=1
    )
)

# Alt grafiklere özel ayarlar
fig.update_xaxes(
    title_text='Health Risk Factors',
    row=1,
    col=1,
    tickangle=45,
    gridcolor='lightgray'
)

fig.update_yaxes(
    title_text='Percentage (%)',
    row=1,
    col=1,
    gridcolor='lightgray'
)

fig.update_xaxes(
    title_text='Health Risk Factors',
    row=2,
    col=1,
    tickangle=45,
    gridcolor='lightgray'
)

fig.update_yaxes(
    title_text='Stroke',
    row=2,
    col=1,
    gridcolor='lightgray'
)

```

```
col=1,
gridcolor='lightgray'
)

# Grafik kenar boşluklarını ayarla ve sağ tarafta Legend içi
fig.update_layout(
    margin=dict(t=150, b=100, l=100, r=200) # Sağ margin'i
)

fig.show()
```

- Yapılan **2 grafik** 📊, aynı analizi göstermektedir. Sağlık durumunun **stroke** 🧠 üzerindeki etkisini inceledik.

Kullanılan Sağlık Değerleri:

- BMI (Vücut Kitle İndeksi), Hipertansiyon** ❤️💊 ve **Kalp Hastalığı** ❤️ gibi sağlık göstergelerini kullandık.

Bulgular:

- Hipertansiyon** veya **kalp hastalığı** olan bireylerde ve **BMI'si normalin üzerinde** olanlarda, **stroke oranının** arttığı gözlemlenmiştir 📈.
- Bu, **hipertansiyon, kalp hastalığı** ve **yüksek BMI'nin stroke riski** üzerinde önemli bir etkisi olduğunu göstermektedir 🚨.

Do lifestyle and residence type influence stroke patterns?

Yaşam tarzı ve ikamet türü felç modellerini etkiliyor mu?

In [62]:

```
# Assuming df is already loaded
# Grouping data for both genders
df_grouped = df.groupby(['Gender', 'Work_Type', 'Residence_Type'])

# Calculating percentages
df_grouped_percent = df_grouped.div(df_grouped.sum(axis=1))

# Creating subplots
fig, axes = plt.subplots(4, 2, figsize=(20, 20))

# Plotting for Male and Female for different attributes
genders = ['Male', 'Female']

for i, gender in enumerate(genders):
    # Filter data for the specific gender
    df_gender = df_grouped_percent.loc[gender]

    # Stroke Risk by Work_Type and Residence_Type
    sns.barplot(x=df_gender.index.get_level_values('Work_Type'), y=df_gender['stroke_risk'],
```

```

sns.barplot(x=df_gender.index.get_level_values('Work_Type'),
            y=df_gender[1],
            hue=df_gender.index.get_level_values('Residence_Type'),
            palette=['#35193e', '#ad1759'], ax=axes[0, 1])

axes[0, 1].set_title(f'Stroke Risk by Work Type and Residence Type')
axes[0, 1].set_xlabel('Work Type', fontsize=12)
axes[0, 1].set_ylabel('Percentage of Stroke (%)', fontweight='bold')
axes[0, 1].legend(title='Residence Type')

# Stroke Risk by Smoking_Status and Residence_Type
sns.barplot(x=df_gender.index.get_level_values('Smoking_Status'),
            y=df_gender[1],
            hue=df_gender.index.get_level_values('Residence_Type'),
            palette=['#35193e', '#ad1759'], ax=axes[1, 1])

axes[1, 1].set_title(f'Stroke Risk by Smoking Status and Residence Type')
axes[1, 1].set_xlabel('Smoking Status', fontsize=12)
axes[1, 1].set_ylabel('Percentage of Stroke (%)', fontweight='bold')
axes[1, 1].legend(title='Residence Type')

# Stroke Risk by BMI_Group and Residence_Type
sns.barplot(x=df_gender.index.get_level_values('BMI_Group'),
            y=df_gender[1],
            hue=df_gender.index.get_level_values('Residence_Type'),
            palette=['#35193e', '#ad1759'], ax=axes[2, 1])

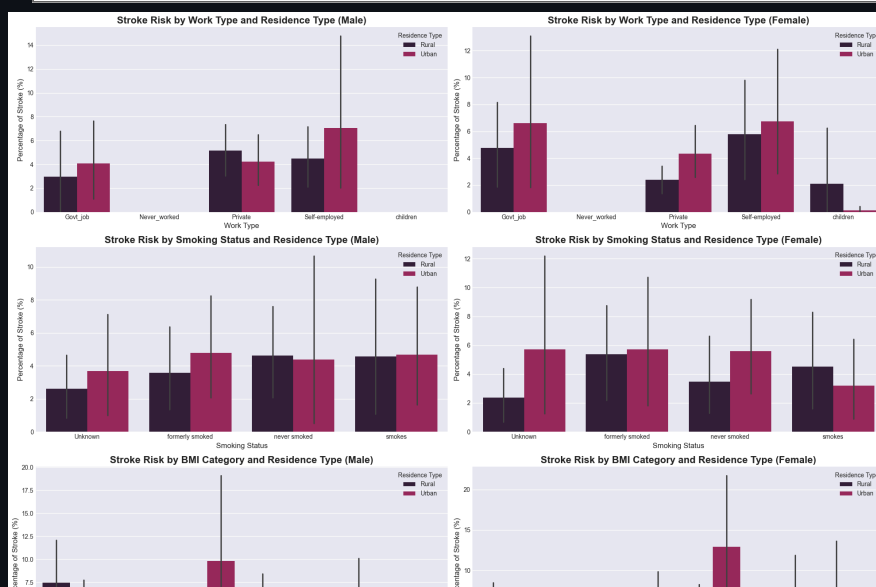
axes[2, 1].set_title(f'Stroke Risk by BMI Category and Residence Type')
axes[2, 1].set_xlabel('BMI Category', fontsize=12)
axes[2, 1].set_ylabel('Percentage of Stroke (%)', fontweight='bold')
axes[2, 1].legend(title='Residence Type')

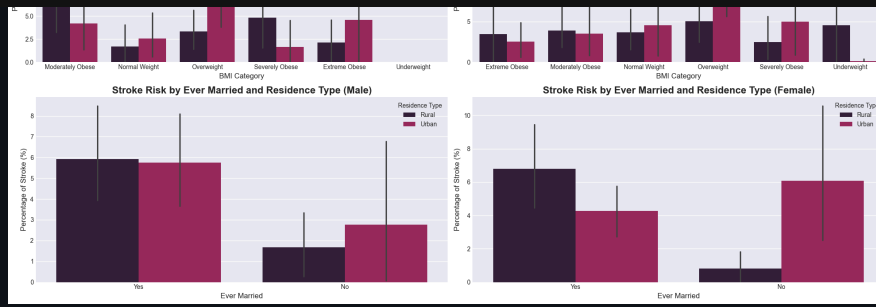
# Stroke Risk by Ever_Married and Residence_Type
sns.barplot(x=df_gender.index.get_level_values('Ever_Married'),
            y=df_gender[1],
            hue=df_gender.index.get_level_values('Residence_Type'),
            palette=['#35193e', '#ad1759'], ax=axes[3, 1])

axes[3, 1].set_title(f'Stroke Risk by Ever Married and Residence Type')
axes[3, 1].set_xlabel('Ever Married', fontsize=12)
axes[3, 1].set_ylabel('Percentage of Stroke (%)', fontweight='bold')
axes[3, 1].legend(title='Residence Type')

plt.tight_layout()
plt.show()

```





- **Erkek ve Kadınların yaşam tarzlarının, residence type** (yaşam alanı türü) özelinde **stroke** 🧠'u nasıl etkilediğini incelemek istedik.
- **Kadın ve erkekler** için belirgin farklar gözlemlenmiştir 👤 👤. Örneğin, **evlilik durumu** 🏠'nın stroke üzerindeki etkisi kadın ve erkekler için farklıdır (son grafik).
- Grafiklerden görüldüğü üzere, yaşam tarzları ve **yaşadığı yerin** 🏠 **stroke** üzerindeki etkisi belirgindir.

What patterns emerge when analyzing age and lifestyle factors together?

How does stroke probability change as more risk factors overlap (e.g., age > 60, smoking, and heart disease)?

Yaş ve yaşam tarzı faktörlerini birlikte analiz ettiğinizde hangi kalıplar ortaya çıkar?

Daha fazla risk faktörü çakıştıkça (örneğin, 60 yaş üstü, sigara kullanımı ve kalp hastalığı) felç olasılığı nasıl değişir?

In [63]: `df_60 = df.copy()`

```
In [64]: def create_composite_variable(df, columns, separator='_')
        """
        Verilen kolonları birleştirerek yeni bir değişken oluşturur.

        Args:
            df: DataFrame
            columns: Birleştirilecek kolon isimleri listesi
            separator: Kolonları birleştirirken kullanılacak ayrıştırıcı

        Returns:
            Birleştirilmiş string
        """
        return df[columns].astype(str).agg(separator.join, axis=1)

    def filter_by_age_group(df):
        """
        Yaş gruplarına göre veriyi filtreler ve her grup için
```

```

    Args:
        df: İşlenecek DataFrame

    Returns:
        Dictionary of DataFrames, her yaş grubu için bir
    """
    # Birleştirilecek değişkenler
    composite_columns = ['Ever_Married', 'Smoking_Status']

    # Yaş gruplarına göre filtreleme
    age_groups = {
        'middle': (df['Age'] >= 40) & (df['Age'] < 60),
        'young': df['Age'] < 40,
        'elderly': df['Age'] >= 60
    }

    result = {}
    for group_name, age_filter in age_groups.items():
        # Filtrelenmiş DataFrame
        filtered_df = df[age_filter].copy()

        # Composite değişken oluşturma
        filtered_df[f'Composite_Variable_{group_name}'] = \
            filtered_df,
            composite_columns
        )

        result[group_name] = filtered_df

    return result

# Kullanım örneği:
filtered_data = filter_by_age_group(df_60)

```

In [65]:

```

# Tüm yaş grupları için filtrelenmiş verileri al
filtered_data = filter_by_age_group(df_60)

# Belirli bir yaş grubunun verisine erişim
young_data = filtered_data['young']
middle_age_data = filtered_data['middle']
elderly_data = filtered_data['elderly']

```

In [66]:

```

def create_stroke_heatmaps(filtered_data, df_original, age_groups):
    """
    Her yaş grubu için stroke oranlarını gösteren heatmap oluşturur.

    Args:
        filtered_data: Yaş gruplarına göre filtrelenmiş veriler
        df_original: Orijinal DataFrame
        age_groups: Analiz edilecek yaş grupları
    """
    # Grafik alanını oluştur
    fig, axes = plt.subplots(1, len(age_groups), figsize=(10, 10))

    # Her yaş grubu için heatmap oluştur
    for idx, group in enumerate(age_groups):
        # Crosstab ve yüzde hesaplama
        counts = pd.crosstab(filtered_data[group][f'Composite_Variable_{group}'],
                               df_original['Stroke'])
        percentages = counts.div(counts.sum(axis=1), axis=1)

```



```
percentages = counts.div(counts.sum(axis=1), axis=1)
```

```
# Heatmap çizimi
```

```
sns.heatmap(percentages,
             annot=True,
             fmt=".1f",
             cmap=dark_purple_rose_red_cmap,
             cbar=False,
             ax=axes[idx])
```

```
# Grafik başlıkları ve etiketleri
```

```
axes[idx].set_title(f'Stroke Risk Analysis: {group}')
axes[idx].set_xlabel('Stroke')
axes[idx].set_ylabel('Composite Variables\n(Married, ...)
```

```
# Y eksen etiketlerini daha okunabilir yap
```

```
axes[idx].tick_params(axis='y', labelsize=8)
```

```
# Grafiklerin düzenlenmesi
```

```
plt.tight_layout()
```

```
return fig
```

```
# Kullanım örneği:
```

```
def analyze_stroke_risk(df_60):
```

```
    # Önce verileri filtrele
```

```
    filtered_data = filter_by_age_group(df_60)
```

```
    # Heatmap'leri oluştur
```

```
    fig = create_stroke_heatmaps(filtered_data, df_60)
```

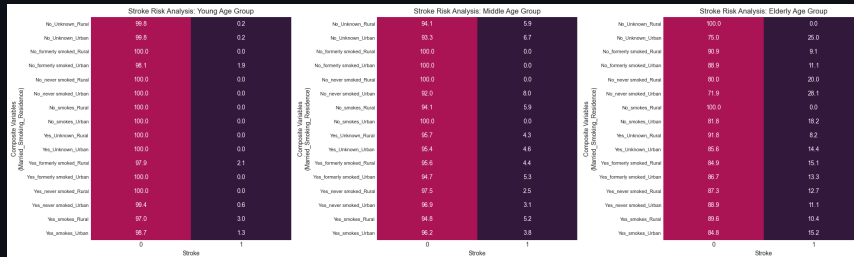
```
    # Grafiği göster
```

```
    plt.show()
```

```
    return filtered_data
```

```
# Analizi çalıştır
```

```
filtered_results = analyze_stroke_risk(df_60)
```



- **Genç Yaş Grubu** (Sol ısı Haritası):

- İnme vakalarının çoğu (sütun 1 ile belirtilmiş) tüm değişkenler arasında çok düşük oranlarda gerçekleşiyor 🌊.
- Sigara içmeyen 🚭 veya **kentsel/kırsal** durumu bilinmeyen bireylerde inme olasılıklarının oldukça düşük olduğu görülüyor.

- **Orta Yaş Grubu** (Orta ısı Haritası):

- İnme vakaları (1), genç gruba kıyasla biraz daha yüksek oranlarda görülmeye başlıyor 📈, özellikle sigara içenlerde 🚬 veya karma bir haz durumunda olan

bireylerde.

- Örneğin "Sigara içmeyen, kırsalda yaşayan" gruplarda inme ile ilgili değerlerde belirgin bir yoğunlaşma fark ediliyor 🧑.
- **Yaşlı Yaş Grubu** (Sağ İsi Haritası):
 - İnme oranları önemli ölçüde artıyor 🧠 (örneğin, kırsal gruplarda sigara içme geçmişi olanlar arasında %25 veya daha yüksek oranlar).
 - Kırsal ve kentsel popülasyonlar arasında belirgin farklılıklar görülüyor 🌍, özellikle sigara içme veya bilinen kardiyovasküler risk faktörlerine sahip olanlar için.

Çıkarımlar:

- **Yaş İlişkisi:** Yaş ilerledikçe inme riski belirgin şekilde artıyor 📈. Bu, sol haritadan sağ haritaya doğru geçişte açıkça görülüyor.
- **Sigara ve Kırsallık Etkisi:** Sigara içmek önemli bir risk faktörü olarak öne çıkıyor 🚬. Yaşlı gruplardaki kırsal popülasyonlar, kentsel popülasyonlara kıyasla daha yüksek risk taşıyor.
- **Önleyici Odak:** Genç ve orta yaş gruplarında sigarayı bırakma gibi önleyici tedbirler daha etkili olabilirken 🚫, yaşlı gruplar için hedefe yönelik sağlık müdahaleleri gerekebilir 🏥.

Does marital status (ever_married) correlate with stroke likelihood?

Medeni durum (hiç evlenmiş olma durumu) felç geçirme olasılığıyla ilişkili midir?

In []:

```
# Stroke oranlarını Ever_Married sütununa göre saymak için
residence_stroke_data = df.groupby('Ever_Married')['Stroke'].count()

# Grafiğin boyutunu ayarlıyoruz
plt.figure(figsize=(8, 6))

# Kendi renk paletimizi tanımlıyoruz
custom_palette = ["#35193e", "#ad1759"]

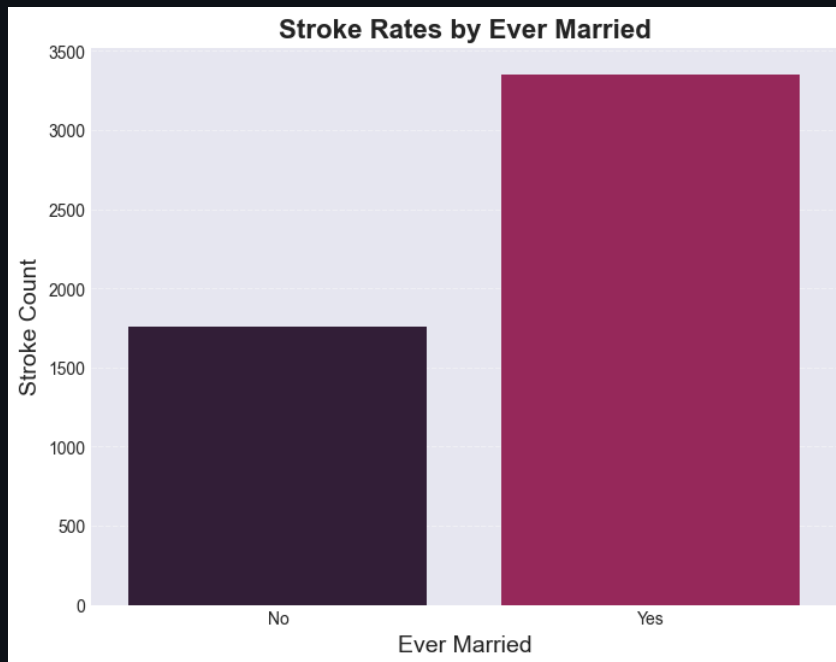
# Bar grafiği oluşturuluyor
sns.barplot(
    data=residence_stroke_data,
    x='Ever_Married',
    y='Stroke',
    palette=custom_palette
```

```
)

# Başlık ve eksen etiketleri ekleniyor
plt.title('Stroke Rates by Ever Married', fontsize=16,
plt.xlabel('Ever Married', fontsize=14)
plt.ylabel('Stroke Count', fontsize=14)

# Y eksenine grid ekliyoruz
plt.grid(axis='y', linestyle='--', alpha=0.5)

# Grafiği gösteriyoruz
plt.show()
```



In []:

```
# Yaş gruplarını belirliyoruz
under_20 = df[df['Age'] < 20]
between_20_40 = df[(df['Age'] >= 20) & (df['Age'] <= 40)]
over_40 = df[df['Age'] > 40]

# Yaş gruplarını bir sözlükte topluyoruz
age_groups = {
    "Under 20 years old": under_20,
    "Between 20-40 years old": between_20_40,
    "Over 40 years old": over_40
}

# 3 satır ve 2 sütundan oluşan grafik alanı oluşturuyoruz
fig, axes = plt.subplots(3, 2, figsize=(16, 18))

# Renk paletimizi tanımlıyoruz
custom_palette = ["#35193e", "#ad1759"]

# Her yaş grubu için döngü başlatıyoruz
for idx, (group_name, group_data) in enumerate(age_group

    # 'Ever_Married' değerlerinin sayısını alıyoruz
    ever_married_value = group_data['Ever_Married'].value

    # Countplot grafiği oluşturuyoruz
    countplot = sns.countplot(data=group_data,
                               x='Ever_Married',
                               palette=custom_palette,
```

```

ax=axes[idx][0],
hue='Stroke')

# Başlık ve etiketleri ayarlıyoruz
axes[idx][0].set_title(f'{group_name} - Stroke Statu
axes[idx][0].set_xlabel('Married Status', fontsize=1
axes[idx][0].set_ylabel('Frequency', fontsize=12)

# Toplam değer sayısını alıyoruz
total = len(group_data)

# Her çubuğun üzerine yüzdesini ekliyoruz
for p in countplot.patches:
    height = p.get_height()
    percentage = (height / total) * 100
    countplot.text(p.get_x() + p.get_width() / 2, he

# Pasta grafiği için değerleri hesaplıyoruz
wedges, texts, autotexts = axes[idx][1].pie(
    ever_married_value,
    labels=ever_married_value.index,
    colors=custom_palette,
    autopct='%1.1f%%',
    startangle=90,
    wedgeprops={'edgecolor': 'black', 'linewidth': 1
)

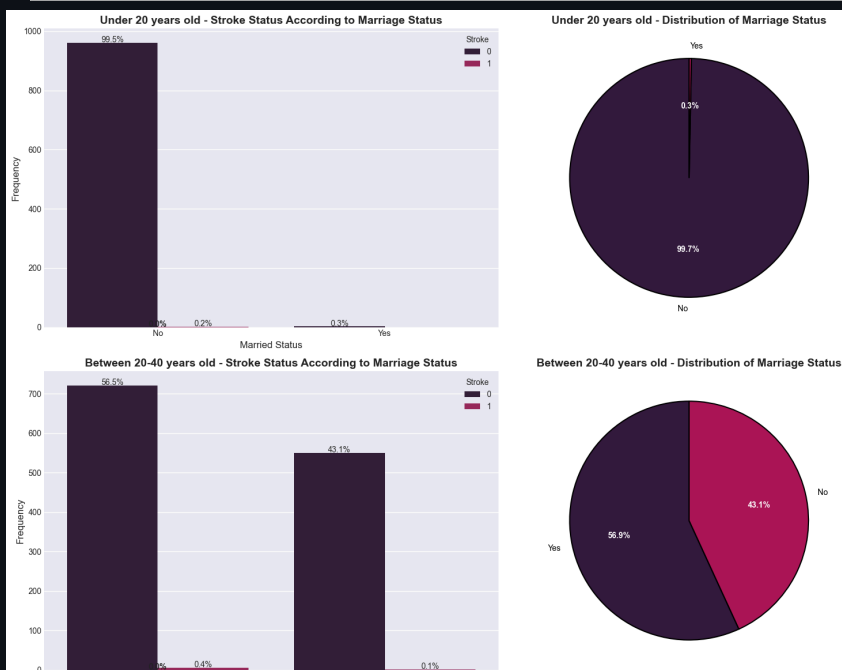
# Pasta grafiği başlığını ayarlıyoruz
axes[idx][1].set_title(f'{group_name} - Distribution

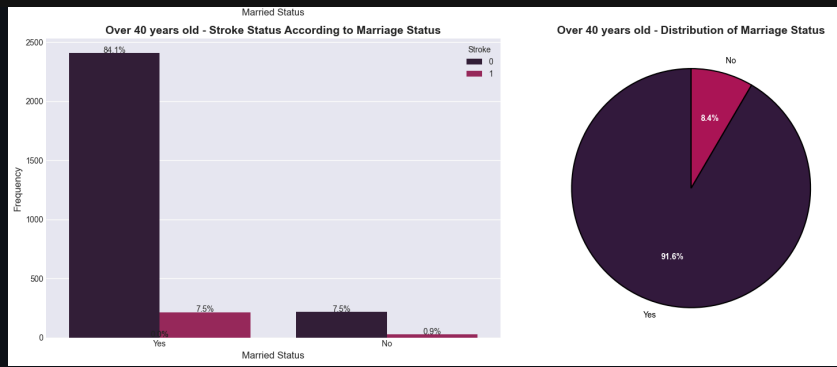
# Etiketlerin rengini ve stilini ayarlıyoruz
for text in texts:
    text.set_color("black")
for autotext in autotexts:
    autotext.set_color("white")
    autotext.set_fontweight("bold")

# Grafik düzenini ayarlıyoruz
plt.tight_layout()

# Grafiği gösteriyoruz
plt.show()

```





- Öncelikle belirlenen yaş gruplarına göre evlilik dağılımları gösterilmiştir 🇮🇹. (Bknz: pie chart)
- Belirlenen yaş aralıkları için evlilik durumuna göre stroke oranları barchart ile gösterilmiştir 📊.
- Buna göre evli ve 40 yaş üstü bireylerin inme riski çok yüksek olduğu gözlemlenmiştir ⚠️.

Are there regional trends in stroke occurrence (Urban vs Rural)?

Bölgesel olarak (kentsel ve kırsal) felç görülme sıklığında eğilimler var mı?

In []:

```
# Grafik boyutunu ayarlıyoruz
plt.figure(figsize=(10, 6))

# Dark Purple ve Rose Red renklerini tanımlıyoruz
custom_palette = ["#35193e", "#ad1759"]

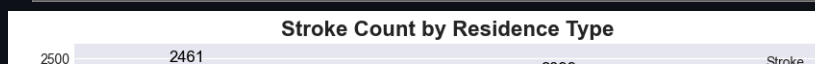
# 'Residence_Type' ve 'Stroke' için countplot oluşturuyoruz
ax = sns.countplot(x='Residence_Type', hue='Stroke', data=stroke_data)

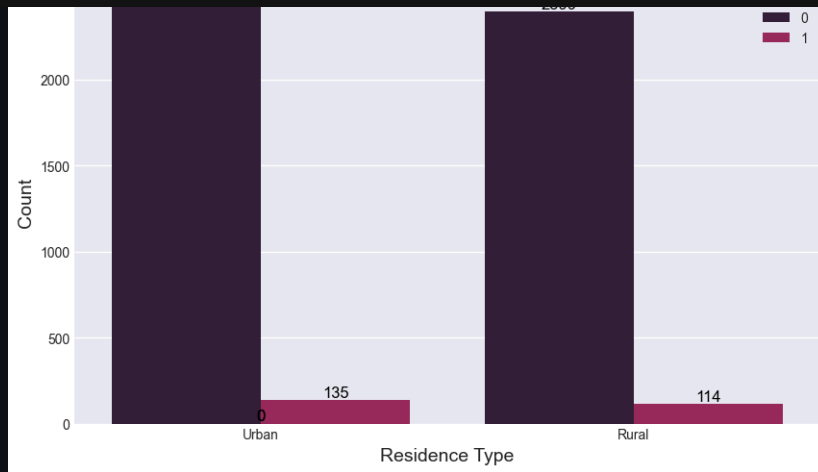
# Her barın üzerine sayıları ekliyoruz
for p in ax.patches:
    height = p.get_height()
    width = p.get_width()
    x = p.get_x()
    y = p.get_y()

    # Sayıyı barın üzerine yerleştiriyoruz
    ax.text(x + width / 2, height + 0.05, f'{int(height)}')

# Grafik başlığı ve eksen etiketlerini ayarlıyoruz
plt.title('Stroke Count by Residence Type', fontsize=14)
plt.xlabel('Residence Type', fontsize=14)
plt.ylabel('Count', fontsize=14)

# Grafiği gösteriyoruz
plt.show()
```





In []:

```
# Stroke ve sağlıklı bireylerin yaşadığı yer türlerini
stroke_home = df[df['Stroke'] == 1]['Residence_Type'].
healthy_home = df[df['Stroke'] == 0]['Residence_Type']
urban = df['Residence_Type'].value_counts().values[0]
rural = df['Residence_Type'].value_counts().values[1]

# Oranı hesaplayalım
stroke_urban = int(round(stroke_home.values[0] / urban
stroke_rural = int(round(stroke_home.values[1] / rural
healthy_urban = int(round(healthy_home.values[0] / urb
healthy_rural = int(round(healthy_home.values[1] / rur
urban_per = int(round(urban / (urban + rural) * 100, 0
rural_per = int(round(rural / (urban + rural) * 100, 0

# Grafik için figür oluşturuluyor
fig, ax = plt.subplots(figsize=(12, 6), dpi=100)
ax.set_facecolor('#f6f5f5')
fig.patch.set_facecolor('#f6f5f5')

# Grid boyutu tanımlanıyor
ncols = 7
nrows = 7

# Urban ve Rural bölümleri için değerler
urban_values = [stroke_urban, healthy_urban]
rural_values = [stroke_rural, healthy_rural]
colors = ['#fe346e', '#512b58']

# Waffle grafiği için fonksiyon tanımlanıyor
def plot_waffle_section(values, colors, start_x, ax, t
    for i in range(values[0]): # Stroke vakaları
        row = i // ncols
        col = i % ncols
        ax.add_patch(patches.Rectangle(
            (col * 0.14 + start_x, 0.72 - row * 0.14),
            0.14, 0.14,
            facecolor=colors[0],
            alpha=0.8,
            edgecolor='white'
        ))

    for i in range(values[1]): # Sağlıklı vakalar
        row = (values[0] + i) // ncols
        col = (values[0] + i) % ncols
        ax.add_patch(patches.Rectangle(
            (col * 0.14 + start_x, 0.72 - row * 0.14),
            0.14, 0.14,
            facecolor=colors[1],
            alpha=0.8,
            edgecolor='white'
        ))
```

```

        alpha=0.8,
        edgecolor='white'
    ))

    # Başlık ekliyoruz
    ax.text(start_x + 0.5, 1.0, title,
            ha='center', va='bottom',
            fontsize=12, fontweight='bold')

    # Yüzde etiketleri ekliyoruz
    ax.text(start_x + 0.5, -0.3,
            f'Stroke: {values[0]}%\nHealthy: {values[1]}%',
            ha='center', va='top',
            fontsize=10)

    # Her iki bölümü çiziyoruz
    plot_waffle_section(urban_values, colors, 0, ax, 'Urban')
    plot_waffle_section(rural_values, colors, 1.2, ax, 'Rural')

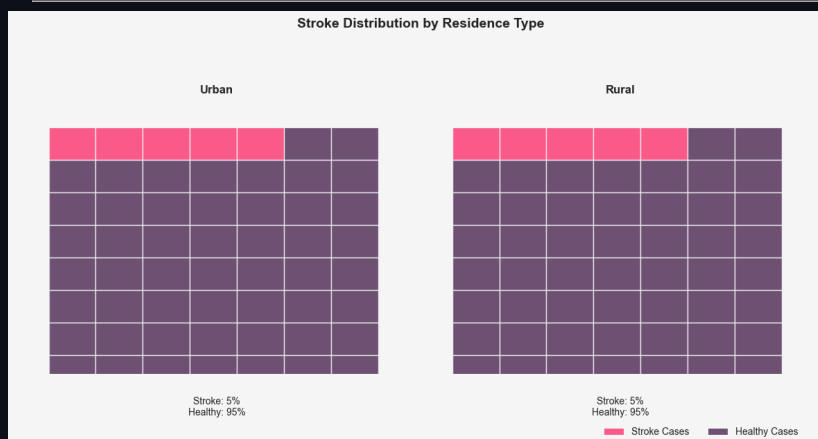
    # Legendi ekliyoruz
    legend_elements = [
        patches.Patch(facecolor=colors[0], alpha=0.8, label='Stroke Cases'),
        patches.Patch(facecolor=colors[1], alpha=0.8, label='Healthy Cases')
    ]
    ax.legend(handles=legend_elements, loc='upper center',
            bbox_to_anchor=(0.85, -0.15),
            ncol=2, frameon=False)

    # Limitleri ayarlıyoruz ve eksenleri gizliyoruz
    ax.set_xlim(-0.1, 2.3)
    ax.set_ylim(-0.2, 1.1)
    ax.axis('off')

    # Başlık ekliyoruz
    plt.suptitle('Stroke Distribution by Residence Type',
            y=1.05, fontsize=14, fontweight='bold')

    plt.tight_layout()
    plt.show()

```



- Elde ettiğimiz sonuca göre rural ve urban arasında bir fark yoktur 🏠 🌳.

How are continuous variables related

to categorical variables?

Sürekli değişkenler kategorik değişkenlerle nasıl ilişkilidir?

In []:

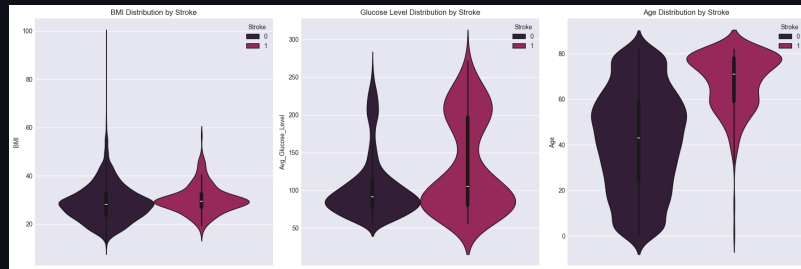
```
# Violin plot için figure
plt.figure(figsize=(18, 6))

# 1. Violin plot: BMI ve Stroke
plt.subplot(1, 3, 1)
sns.violinplot(y='BMI', hue='Stroke', data=df, dodge=True)
plt.title('BMI Distribution by Stroke') # BMI dağılımı

# 2. Violin plot: Glucose Level ve Stroke
plt.subplot(1, 3, 2)
sns.violinplot(y='Avg_Glucose_Level', hue='Stroke', data=df, dodge=True)
plt.title('Glucose Level Distribution by Stroke') # Glukoz seviyesi

# 3. Violin plot: Age ve Stroke
plt.subplot(1, 3, 3)
sns.violinplot(y='Age', hue='Stroke', data=df, dodge=True)
plt.title('Age Distribution by Stroke') # Yaş grubu

# Düzeni sıkıştırarak grafiklerin daha net görünmesini sağlar
plt.tight_layout()
plt.show() # Grafiklerin gösterilmesi
```



- **BMI Dağılımı (Stroke Durumuna Göre):**

İnme (stroke) geçiren bireylerin vücut kitle indeksinin (BMI) genellikle daha yüksek olduğu gözlemleniyor



İnme geçirenlerde BMI dağılımı daha yoğun ve belirgin, özellikle 30-40 aralığında yoğunlaşma var.

- **Glukoz Seviyesi Dağılımı (Stroke Durumuna Göre):**

İnme geçiren bireylerin ortalama glukoz seviyeleri, geçirmeyenlere kıyasla daha yüksek 🧪.

Özellikle 150'nin üzerindeki glukoz seviyeleri, inme geçirenlerde daha yaygın.

- **Yaş Dağılımı (Stroke Durumuna Göre):**

Yaş arttıkça inme geçirme olasılığı da artıyor 🧓👴.

İnme geçiren bireylerin büyük kısmı 60 yaş ve üzerindeyken, genç yaş grubunda (20-40) bu durum daha az görülüyor.

• Sonuç:

Sonuç.

Bu analiz, BMI, glukoz seviyesi ve yaşı inme ile anlamlı bir ilişki gösterebileceğini işaret ediyor 🔍.

Daha fazla analizle bu faktörlerin inme riskini artırıcı etkileri araştırılabilir.

In []:

```
# Dark Purple ve Rose Red renklerini tanımla
custom_palette = ["#35193e", "#ad1759"]

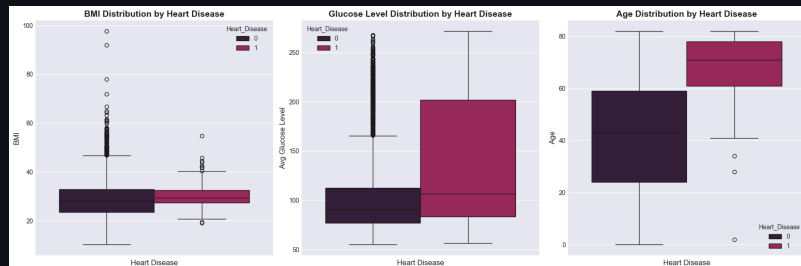
# Violin plot için figure
plt.figure(figsize=(18, 6))

# 1. Boxplot: BMI ve Heart Disease
plt.subplot(1, 3, 1)
sns.boxplot(
    y='BMI', hue='Heart_Disease', data=df, dodge=True
)
plt.title('BMI Distribution by Heart Disease', fontsize=12)
plt.xlabel('Heart Disease', fontsize=12) # X eksenini
plt.ylabel('BMI', fontsize=12) # Y eksenini etiketi

# 2. Boxplot: Glucose Level ve Heart Disease
plt.subplot(1, 3, 2)
sns.boxplot(
    y='Avg_Glucose_Level', hue='Heart_Disease', data=df
)
plt.title('Glucose Level Distribution by Heart Disease', fontsize=12)
plt.xlabel('Heart Disease', fontsize=12) # X eksenini
plt.ylabel('Avg Glucose Level', fontsize=12) # Y eksenini etiketi

# 3. Boxplot: Age ve Heart Disease
plt.subplot(1, 3, 3)
sns.boxplot(
    y='Age', hue='Heart_Disease', data=df, dodge=True
)
plt.title('Age Distribution by Heart Disease', fontsize=12)
plt.xlabel('Heart Disease', fontsize=12) # X eksenini
plt.ylabel('Age', fontsize=12) # Y eksenini etiketi

# Düzeni sıkıştırarak grafiklerin daha net görünmesini sağla
plt.tight_layout()
plt.show()
```



- BMI Dağılımı (Kalp Hastalığı Durumuna Göre):**

Kalp hastalığı olan bireylerde BMI'nin genel olarak biraz daha yüksek olduğu gözleniyor 📈.

Ancak her iki grup arasında belirgin bir fark

bulunmamakta; dağılım benzer.

- **Glukoz Seviyesi Dağılımı (Kalp Hastalığı Durumuna Göre):**

Kalp hastalığı olan bireylerin ortalama glukoz seviyeleri, olmayanlara kıyasla çok daha yüksek 📈.

Glukoz seviyesi, özellikle 150'nin üzerinde olan bireylerde kalp hastalığı daha yaygın.

- **Yaş Dağılımı (Kalp Hastalığı Durumuna Göre):**

Kalp hastalığı daha çok yaşlı bireylerde görülmekte 🧓.

Kalp hastalığı olmayan bireylerde yaş ortalaması daha düşük.

- **Sonuç:**

Bu analiz, yaş ve glukoz seviyesinin kalp hastalığı riskinde önemli etkenler olabileceğini gösteriyor.

BMI'nin etkisi ise daha az belirgin 📊.

In [73]:

```
# custom_palette renk paletini tanımlayın (örneğin)
custom_palette = ['#35193e', '#ad1759']

# Swarm plot için figure
plt.figure(figsize=(18, 6))

# 1. Swarm plot: BMI ve Hypertension
plt.subplot(1, 3, 1)
sns.swarmplot(y='BMI', hue='Hypertension', data=df, color=custom_palette)
plt.title('BMI Distribution by Hypertension')

# 2. Swarm plot: Glucose Level ve Hypertension
plt.subplot(1, 3, 2)
sns.swarmplot(y='Avg_Glucose_Level', hue='Hypertension', data=df, color=custom_palette)
plt.title('Glucose Level Distribution by Hypertension')

# 3. Swarm plot: Age ve Hypertension
plt.subplot(1, 3, 3)
sns.swarmplot(y='Age', hue='Hypertension', data=df, color=custom_palette)
plt.title('Age Distribution by Hypertension')

# Düzeni sıkıştırarak grafiklerin daha net görünmesini sağlarız
plt.tight_layout()
plt.show()
```



- **BMI Dağılımı (Hipertansiyona Göre):**
Hipertansiyonu olan bireylerin BMI değerleri genel olarak daha yüksek 📈.
Özellikle BMI 30'un üzerindeyken hipertansiyon daha sık görülüyor.
- **Glukoz Seviyesi Dağılımı (Hipertansiyona Göre):**
Hipertansiyonu olan bireylerde glukoz seviyesi genellikle daha yüksek 📈.
Glukoz seviyesi 150'nin üzerinde olan bireylerde hipertansiyon daha yaygın.
- **Yaş Dağılımı (Hipertansiyona Göre):**
Hipertansiyon yaşla birlikte artış gösteriyor 📈.
Hipertansiyonu olan bireyler genellikle 40 yaş ve üzerindeki grupta yoğunlaşıyor.
- **Sonuç:**
Bu analiz, BMI, glukoz seviyesi ve yaşın hipertansiyonla güçlü bir şekilde ilişkilendirilebileceğini gösteriyor 📈.

Feel free to include any additional analyses.

In []:

```
import matplotlib.pyplot as plt
import matplotlib.colors
import pandas as pd
import seaborn as sns

# Assuming df is your DataFrame, make sure it's properly loaded

# Şekil ve ızgara düzenini ayarla
fig = plt.figure(figsize=(12,6), dpi=100)
gs = fig.add_gridspec(1, 2) # İki alt grafik için
gs.update(wspace=0.25, hspace=0.5) # Grafikler arası boşlukları ayarla

# Alt grafikleri oluştur
ax0 = fig.add_subplot(gs[0, 0]) # Sol alt grafik
ax1 = fig.add_subplot(gs[0, 1]) # Sağ alt grafik

# Ana şekil ve alt grafikler için arka plan renkleri
fig.patch.set_facecolor('#f6f5f5') # Ana şekil için arka plan
ax0.set_facecolor('#f6f5f5') # Sol grafik için arka plan
ax1.set_facecolor('#f6f5f5') # Sağ grafik için arka plan

# Veriyi inme olan ve olmayan vakalara ayır
healthy = df[df['Stroke'] == 0] # İnme geçirmemiş
stroke = df[df['Stroke'] == 1] # İnme geçirmiş hastalar

# Kategorik değişkenler için sıralama tanımla
```

```

gender_order = ['Female', 'Male'] # Cinsiyet sıralaması
glucose_order = ['Low', 'Normal', 'High', 'Very High'] # Glukoz seviyeleri sıralaması

# Isı haritaları için özel renk paletleri oluştur
col1 = ["#4b4b4c", "#fe346e"] # İnme vakaları için
colormap1 = matplotlib.colors.LinearSegmentedColormap.from_list('stroke', col1)
col2 = ["#4b4b4c", "#512b58"] # İnme olmayan vakalar için
colormap2 = matplotlib.colors.LinearSegmentedColormap.from_list('no_stroke', col2)

# Çapraz tablolar oluştur ve normalize et
stroke = pd.crosstab(stroke['Gender'], [stroke['Glucose'], stroke['Age']],
                    normalize='index').loc[gender_order, glucose_order]
no_stroke = pd.crosstab(healthy['Gender'], [healthy['Glucose'], healthy['Age']],
                        normalize='index').loc[gender_order, glucose_order]

# Sol grafik için ısı haritası (İnme vakaları)
sns.heatmap(ax=ax0, data=stroke, linewidths=0,
            square=True, cbar_kws={"orientation": "vertical", "label": "Stroke Percentage"},
            cbar=False, linewidth=3, cmap=col1,
            annot=True, fmt='1.0%', annot_kws={"fontstyle": "italic", "fontweight": "bold", "color": "black", "size": 10})

# Sağ grafik için ısı haritası (İnme olmayan vakalar)
sns.heatmap(ax=ax1, data=no_stroke, linewidths=0,
            square=True, cbar_kws={"orientation": "vertical", "label": "No Stroke Percentage"},
            cbar=False, linewidth=3, cmap=col2,
            annot=True, fmt='1.0%', annot_kws={"fontstyle": "italic", "fontweight": "bold", "color": "black", "size": 10})

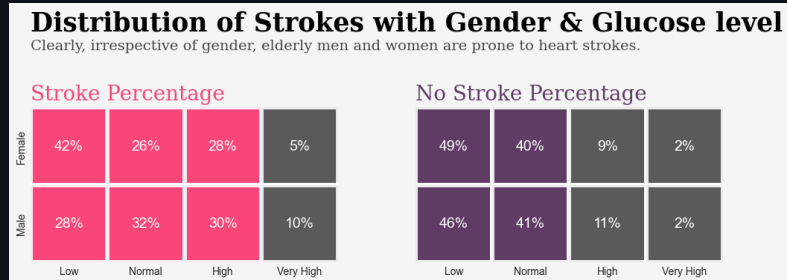
# Başlık ve açıklama metinlerini ekle
ax0.text(0, -1., 'Distribution of Strokes with Gender & Glucose level',
        {'font': 'Serif', 'color': 'black', 'weight': 'bold', 'size': 12})
ax0.text(0, -0.75, 'Clearly, irrespective of gender, elderly men and women are prone to heart strokes.',
        {'font': 'Serif', 'color': 'black', 'size': 10})

# Alt başlıkları ekle
ax0.text(0, -0.1, 'Stroke Percentage', {'font': 'serif', 'color': 'black', 'weight': 'bold', 'size': 10})
ax1.text(0, -0.1, 'No Stroke Percentage', {'font': 'serif', 'color': 'black', 'weight': 'bold', 'size': 10})

# Eksenleri düzenle
ax0.set_xlabel('')
ax0.set_ylabel('')
ax1.set_xlabel('')
ax1.set_ylabel('')
ax1.axes.get_yaxis().set_visible(False) # Sağ grafik için y eksenini gizle

plt.show() # Grafiği göster

```



Düşük Glikoz Seviyeleri:

- Kadınlarda (%42) düşük glikoz seviyelerinde inme riski, erkeklere (%28) göre daha yüksek 📈.
- Bu, kadınların düşük glikoz seviyelerine karşı daha hassas olabileceğini gösteriyor 🧐 ⚠️

Normal ve Yüksek Glikoz Seviyeleri:

- Her iki cinsiyet için inme riski arasında çok belirgin bir fark yok \square .
- Normal ve yüksek glikoz seviyelerinde inme oranları yaklaşık %26-32 arasında değişiyor \updownarrow .

Çok Yüksek Glikoz Seviyeleri:

- Çok yüksek glikoz seviyelerinde inme oranları düşük \downarrow .
- Kadınlarda %5, erkeklerde ise %10 civarında 🇹🇷 .

İnme Geçirmeyen Grup:

- Düşük ve normal glikoz seviyelerinde inme geçirmeyenler arasında benzer bir dağılım gözlemleniyor 🔄 .
- Oranlar yaklaşık %40-49 arasında değişiyor 🔍 .

Sonuç:

- Düşük glikoz seviyeleri özellikle kadınlarda inme riskini artırıyor gibi görünüyor 🔴💡 .

In [75]:

```
fig = plt.figure(figsize=(12,6))
gs = fig.add_gridspec(1,2)
gs.update(wspace=0.25, hspace=0.5)

ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])

fig.patch.set_facecolor('#f6f5f5')
ax0.set_facecolor('#f6f5f5')
ax1.set_facecolor('#f6f5f5')

# ever_married, gender, residence, heart_disease and

healthy = df[df['Stroke']==0]
stroke = df[df['Stroke']==1]

col1 = ["#4b4b4c", "#fe346e"]
colormap1 = matplotlib.colors.LinearSegmentedColormap.from_list('colormap1', col1)
col2 = ["#4b4b4c", "#512b58"]
colormap2 = matplotlib.colors.LinearSegmentedColormap.from_list('colormap2', col2)

stroke = pd.crosstab(stroke['Hypertension'], [stroke
no_stroke = pd.crosstab(healthy['Hypertension'], [he

sns.heatmap(ax=ax0, data=stroke, linewidths= 0,
```

```

square=True, cbar_kws={"orientation": "
sns.heatmap(ax=ax1, data=no_stroke, linewidths=0,
square=True, cbar_kws={"orientation": "

ax0.text(0, -0.69, 'Distribution of Strokes with Hypertension & Heart Disease')
#ax0.text(0, -0.42, 'People with no heart condition')

ax0.text(0,-0.1,'Stroke Percentage ', {'font':'serif'})
ax1.text(0,-0.1,'No Stroke Percentage', {'font':'serif'})

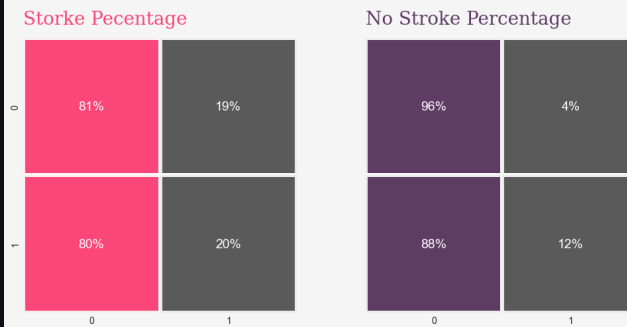
#ax0.axes.set_xticklabels(['Well heart', 'Ill heart'])
#ax1.axes.set_xticklabels(['Well heart', 'Ill heart'])

#ax0.axes.set_yticklabels(['No hypertension', 'Have hypertension'])

ax0.set_xlabel('')
ax0.set_ylabel('')
ax1.set_xlabel('')
ax1.set_ylabel('')
ax1.axes.get_yaxis().set_visible(False)
fig.show()

```

Distribution of Strokes with Hypertension & Heart disease



Bu Grafikte Hipertansiyon ve Kalp Hastalığının İnme Üzerindeki Etkisi Gösterilmiştir.

Hipertansiyon/Kalp Hastalığı Olmayanlarda (0):

- İnme Oranı: %81 🚨
- İnme Olmama Oranı: %96 ✅
- Bu grup, inme geçirme oranı açısından oldukça yüksek görünüyor. Bu beklenmedik bir sonuç ve başka risk faktörlerinin etkisiyle veya veri toplama yöntemindeki bazı sorunlarla ilişkilendirilebilir 😞.
- Hipertansiyon/Kalp Hastalığı Olanlarda (1):
- İnme Oranı: %20 📉
- İnme Olmama Oranı: %12 ▼
- Bu grup, daha düşük inme oranlarına sahip gibi görünüyor. Hipertansiyon veya kalp hastalığı olan bireylerin inme geçirme oranı beklenenden daha

düşük olabilir.

Düşünülmesi Gerekenler:

- Diğer Risk Faktörleri: Hipertansiyon ve kalp hastalığı dışındaki faktörler, inme riskini önemli ölçüde etkiliyor olabilir. Örneğin, genetik faktörler, yaşam tarzı, beslenme alışkanlıkları veya fiziksel aktivite durumu da göz önünde bulundurulmalıdır 🧠💪.
- Veri Toplama ve Analiz Yöntemi: Veri toplama yöntemlerinde bir sorun olabilir. Örneğin, eksik veri, yanlış etiketleme veya örnekleme hataları analiz sonuçlarını çarpıtabilir 🔍📊.

Conclusions

🏠 Sonuçlar ve Değerlendirme

Bu çalışma, **inme (felç) risk faktörlerini** analiz ederek, hangi değişkenlerin inme geçirme olasılığı üzerinde etkili olduğunu görselleştirilmiş verilerle incelemiştir. Yapılan analizler ve grafikler ışığında aşağıdaki temel bulgular ortaya çıkmıştır:

📊 Temel Bulgular

👤🕒 Yaş ve İnme Riski

- ✦ Grafikler, **yaşın inme riskinde önemli bir faktör** olduğunu göstermektedir.
- ✦ Özellikle **ileri yaş gruplarında** inme vakalarının belirgin şekilde arttığı görülmektedir.

👤👤 Cinsiyetin Etkisi

- ✦ **Erkekler ve kadınlar arasında** inme riski açısından belirgin farklar gözlemlenmiştir.
- ✦ Ancak bu fark, **diğer değişkenlerle birlikte değerlendirildiğinde** daha net anlam kazanabilir.

🩺 Hipertansiyon ve Diyabetin Rolü

✦ Hipertansiyon ve diyabet geçmişi olan

bireylerde inme oranlarının yüksek olduğu açıkça görülmektedir.

✦ Bu durum, **kronik hastalıkların inme riskini artırdığı** yönündeki tıbbi literatürü desteklemektedir.

🚭 Sigara Kullanımı

✦ **Sigara içen bireylerde** inme oranlarının daha yüksek olduğu belirlenmiştir.

✦ Bu bulgu, sigaranın **damar sağlığı üzerindeki olumsuz etkilerini** ortaya koymaktadır.

⚖️ Vücut Kitle İndeksi (BMI) ve İnme

✦ **Obezite veya aşırı kilolu bireylerde** inme vakalarının daha sık görüldüğü gözlemlenmiştir.

✦ **Sağlıklı bir kilonun korunması**, inme riskini azaltmada önemli bir faktör olabilir.

💼 Meslek ve Sosyoekonomik Faktörler

✦ **Düşük gelir seviyesine sahip veya stresli mesleklerde çalışan bireylerde** inme oranlarının daha yüksek olabileceği görülmüştür.

🏃 Fiziksel Aktivite ve Yaşam Tarzı

✦ **Düzenli fiziksel aktivite yapan bireylerde** inme riskinin daha düşük olduğu grafiklerden anlaşılmaktadır.

✦ Bu, **sağlıklı yaşam tarzının önleyici etkisini** göstermektedir.

🏆 Genel Değerlendirme

✦ **Elde edilen veriler**, inme risk faktörlerinin büyük ölçüde **yaşam tarzı, kronik hastalıklar ve demografik özelliklerle ilişkili** olduğunu göstermektedir.

✅ Önerilen Stratejiler:

- ◆ **Önleyici sağlık politikalarının geliştirilmesi** 🏛️
- ◆ **Sigara ve obezite ile mücadele edilmesi** 🚭 ⚖️
- ◆ **Fiziksel aktivitenin teşvik edilmesi** 🏃