

T.C.

DOĞUŞ ÜNİVERSİTESİ

İKTİSADİ VE İDARİ BİLİMLER FAKÜLTESİ

YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

PYTHON İLE VERİ ANALİZİ

Berivan YALÇIN

202002009027

ÖĞR. GÖR. Oktay CESUR

İstanbul,2024

**ÖNSÖZ**

Bu çalışma, dünya genelinde en yaygın ve ölümcül kanser türlerinden biri olan akciğer kanseri ölüm oranları üzerine kapsamlı bir analiz sunmaktadır. Proje kapsamında kullanılan veri seti, Kaggle platformundan alınmış ve Python programlama dili kullanılarak detaylı bir şekilde incelenmiştir. Bu analiz sürecinde, verilerin ön işleme aşamasından başlayarak, istatistiksel analizler ve makine öğrenimi modellerinin oluşturulmasına kadar çeşitli aşamalar gerçekleştirilmiştir.

Projenin temel amacı, akciğer kanseri ölüm oranlarına etki eden demografik, çevresel ve genetik faktörleri daha iyi anlamak ve bu faktörlerin etkilerini nicel verilerle ortaya koymaktır. Bu kapsamda, kanser hastalarının demografik, tıbbi ve tedavi süreçlerine dair veriler analiz edilmiş ve anlamlı bulgular elde edilmiştir. Çalışmamız, bu ciddi sağlık sorunu ile mücadelede stratejik planlamaya ve toplum sağlığının iyileştirilmesine katkı sağlamayı amaçlamaktadır.

Bu projeyi gerçekleştirme sürecinde, danışman hocam öğretim görevlisi Oktay Cesur’un değerli rehberliği, ailemin ve arkadaşlarımın desteği benim için büyük bir motivasyon kaynağı olmuştur. Ayrıca, veri setini sağlayan Kaggle topluluğuna teşekkürlerimi sunarım.

Saygılarımla,

Berivan Yalçın

**ÖZET**

Bu çalışmada, akciğer kanseri hastalarının demografik, tıbbi ve tedavi süreçlerine dair bilgilerin yer aldığı bir veri seti kullanılarak çeşitli analizler gerçekleştirilmiştir. Veri seti, hastaların yaş, cinsiyet, ülke, sigara içme durumu, vücut kitle indeksi, kolesterol seviyesi, hipertansiyon, astım ve tedavi türleri gibi bilgileri içermektedir. Her satır bir hastayı temsil etmektedir ve çeşitli özellikler hakkında bilgi sunmaktadır.

Analiz sürecinde, Python programlama dili ve çeşitli veri bilimi kütüphaneleri kullanılmıştır. Pandas, NumPy, Matplotlib, Seaborn ve SciPy gibi kütüphaneler, verilerin işlenmesi ve görselleştirilmesi için etkin bir şekilde kullanılmıştır. Ayrıca, eksik verilerin yönetimi için Missingno, coğrafi verilerin görselleştirilmesi için Folium ve coğrafi işlemler için Geopy kütüphaneleri kullanılmıştır.

Veri seti üzerinde uygulanan istatistiksel testler arasında Kolmogorov-Smirnov Testi, T-Testi, Levene Testi, Chi-kare Testi ve Korelasyon Testi yer almaktadır. Bu testler, veri setindeki örüntüleri ve ilişkileri anlamamıza yardımcı olmuştur.

Sonuç olarak, bu çalışma, akciğer kanseri ile ilgili değerli bilgiler sunmakta ve bu bilgilerin kanserle mücadele ve tedavi süreçlerinde kullanılması için temel oluşturmaktadır. Elde edilen bulgular, kanser araştırmaları, tedavi planlaması ve hastaların sağlık durumlarının analizi için önemli bir kaynak teşkil etmektedir.

**Anahtar Kelimeler:** Akciğer Kanseri, Veri Analizi, Python, İstatistiksel Testler, Demografik Faktörler, Makine Öğrenimi, Veri Görselleştirme, Kaggle

**ABSTRACT**

In this study, various analyses were conducted using a dataset containing information on the demographic, medical, and treatment processes of lung cancer patients. The dataset includes information such as patients' age, gender, country, smoking status, body mass index, cholesterol level, hypertension, asthma, and treatment types. Each row represents a patient and provides information on various attributes.

In the analysis process, the Python programming language and various data science libraries were utilized. Libraries such as Pandas, NumPy, Matplotlib, Seaborn, and SciPy were effectively used for data processing and visualization. Additionally, Missingno was used for managing missing data, Folium for visualizing geographical data, and Geopy for geographical operations.

The statistical tests applied to the dataset include the Kolmogorov-Smirnov Test, T-Test, Levene's Test, Chi-Square Test, and Correlation Test. These tests helped us understand patterns and relationships within the dataset.

In conclusion, this study provides valuable insights into lung cancer and forms a basis for using this information in cancer combat and treatment processes. The findings obtained are an important resource for cancer research, treatment planning, and the analysis of patients' health conditions.

**Keywords:** Lung Cancer, Data Analysis, Python, Statistical Tests, Demographic Factors, Machine Learning, Data Visualization, Kaggle

**BÖLÜM 1: TEORİK KISIM**

**1. PYTHON NEDİR ?**

Python, web uygulamaları, yazılım geliştirme, veri bilimi ve makine öğrenimi (ML) alanlarında yaygın olarak kullanılan bir programlama dilidir. Geliştiriciler, etkili, öğrenmesi kolay ve çeşitli platformlarda çalışabilen bir dil olduğu için Python'ı tercih ederler. Python yazılımı ücretsiz indirilebilir, her türlü sistemle sorunsuz entegrasyon sağlar ve geliştirme sürecini hızlandırır.

**1.1 Python'ın avantajlarından bazıları şunlardır:**

1. **Kolay Okunabilirlik ve Anlaşılabilirlik**: İngilizceye benzer basit söz dizimi sayesinde Python kodu kolayca okunabilir ve anlaşılabilir.
2. **Yüksek Üretkenlik**: Diğer dillere kıyasla daha az kodla aynı işlevselliği sağlayabilme özelliği sayesinde geliştiriciler daha verimli çalışabilir.
3. **Geniş Kütüphane Desteği**: Python, geniş bir standart kütüphane ve çeşitli üçüncü parti kütüphanelerle birçok görev için hazır çözümler sunar, böylece tekrar kullanılabilir kod yazma ihtiyacını azaltır.
4. **Çoklu Platform Desteği**: Windows, macOS, Linux gibi farklı işletim sistemlerinde çalışabilir olmasıyla çeşitli ortamlarda kullanım kolaylığı sağlar.
5. **Geniş Topluluk Desteği**: Küresel çapta büyük ve aktif bir geliştirici topluluğuna sahip olması, sorunlarınıza hızlı çözümler bulmanızı sağlar.

**1.2 Python'u kullanmanın farklı alanlarda örnekleri şunlardır:**

* **Web Geliştirme**: Sunucu tarafı işlevleri için çeşitli Python çerçevelerinden yararlanarak web uygulamaları geliştirebilirsiniz.
* **Otomasyon ve Betikler**: Günlük görevlerin otomatikleştirilmesi için Python betikleri kullanılır, örneğin dosya yönetimi, veri işleme gibi.
* **Veri Bilimi ve Makine Öğrenimi**: Veri temizleme, özellik seçimi, tahmin modelleri oluşturma gibi veri bilimi işlemleri için yaygın olarak kullanılır.
* **Yazılım Geliştirme**: Hata izleme, otomatik derleme, prototip geliştirme gibi yazılım geliştirme süreçlerinde de tercih edilir.
* **Yazılım Test Otomasyonu**: Python birim testi çerçeveleri ile yazılım testlerini otomatize edebilir ve sürekli entegrasyon/distribution (CI/CD) süreçlerinde kullanılabilir.

**2. VERİ ANALİZİ**

Veri analizi, verileri inceleme, temizleme, dönüştürme ve modelleme işlemlerini içererek, faydalı bilgiler bulma, sonuçları bilgilendirme ve karar vermeyi destekleme amacı taşır. İşletme, bilim ve sosyal bilimler gibi çeşitli alanlarda kullanılan veri analizi, çeşitli teknikleri bünyesinde barındırır ve günümüz iş dünyasında karar verme süreçlerini daha bilimsel hale getirerek işletmelerin daha etkin çalışmasına yardımcı olur.

Veri madenciliği, özellikle tahmin edici bilgi keşfine odaklanan özel bir veri analiz tekniğidir, iş zekası ise genellikle iş bilgilerine odaklanarak büyük ölçüde veri birleştirmeye dayanır.[2] İstatistiksel uygulamalarda, veri analizi tanımlayıcı istatistikler, keşifsel veri analizi (EDA) ve doğrulayıcı veri analizi (CDA) gibi bölümlere ayrılabilir. EDA, verilerdeki yeni özellikleri keşfetmeye yönelikken, CDA mevcut hipotezleri doğrulamaya veya yalanlamaya odaklanır. Tahmine dayalı analitik, tahmin veya sınıflandırma için istatistiksel modellerin uygulanmasına odaklanırken, metin analizi ise yapılandırılmamış metin kaynaklarından bilgi çıkarmak ve sınıflandırmak için istatistiksel, dilsel ve yapısal teknikler kullanır. Bu farklı alanlar veri analizinin çeşitli türlerini oluşturur.

**3. VERİ TEMİZLEME**

Veri temizleme veya veri sürtme, bir veri tabanından yanlış verileri veya kayıtları tespit etme, düzeltme veya kaldırma işlemidir. Bu süreç, düzgün şekilde formatlanmamış verileri düzenleme, çift veri veya kayıtları birleştirme veya kaldırma gibi adımları içerir. Temizlenen veriler genellikle "kirli veri" olarak adlandırılır ve veri kalitesinin korunması için önemlidir. Büyük işletmeler genellikle bu tür hataları otomatik araçlar ve algoritmalar kullanarak tespit eder ve düzeltirler; örneğin, eksik posta kodları gibi yaygın hataları düzeltmek için.

Güçlü büyük veri çevreleri, veri kalitesini korumak için titiz veri temizleme araçları ve süreçleri kullanarak, veri kümelerine duyulan güveni tüm kullanıcılar için yüksek tutarlar.

**3.1 Veri Temizleme Adımları**

Veri temizleme adımları, veri tabanından veya veri kümesinden yanlış, eksik veya gereksiz verileri tespit etmek ve düzeltmek veya kaldırmak için uygulanan bir dizi işlemdir. İşte genel olarak veri temizleme adımları:

**1.Veri Keşfi ve Anlama**: İlk adım veri kümesini anlamaktır. Veri yapıları, alanlar ve potansiyel hatalar hakkında bilgi edinmek için veriye genel bir bakış yapılır.

**2. Eksik veya Boş Verilerin İncelenmesi**: Veri kümesinde eksik veya boş olan veriler belirlenir. Bu adım, eksik verilerin nedenlerini anlamak ve bunları doldurma veya çıkarma kararı vermek için kullanılır.

**3. Anormal Veri Noktalarının Belirlenmesi**: Veri setindeki anormal veya aykırı değerler tespit edilir. Bu değerler genellikle diğer veri noktalarından önemli ölçüde farklıdır ve analiz sonuçlarını yanıltabilir.

**4. Çift Kayıtların Tespiti**: Veri kümesinde aynı kayıtların tekrarlanıp tekrarlanmadığı kontrol edilir. Bu adım, veri bütünlüğünü sağlamak için önemlidir.

**5. Veri Birleştirme ve Standartlaştırma**: Birden fazla kaynaktan gelen veriler birleştirilir ve standart formata getirilir. Örneğin, farklı yazım biçimlerindeki isimlerin standart bir formata dönüştürülmesi.

**6. Veri Validasyonu**: Veri kümesinin belirlenen kurallara ve kriterlere uyup uymadığı kontrol edilir. Örneğin, belirli bir aralık dışındaki sayısal değerlerin geçersiz kabul edilmesi.

**7. Veri Düzeltme ve Dönüşüm**: Tespit edilen hatalar düzeltilir ve veri kümesi belirli bir formata veya yapıya dönüştürülür. Örneğin, tarih formatlarının standartlaştırılması veya sayısal değerlerin birimlerinin dönüştürülmesi.

**8. Dokümantasyon ve Raporlama**: Yapılan temizleme işlemleri ve elde edilen sonuçlar belgelenir. Bu, ilerideki analizler veya tekrarlanabilirlik için önemlidir.

**9. Sonuçların Doğrulanması**: Temizleme işlemlerinin ardından, veri kümesinin doğruluğunu ve uygunluğunu doğrulamak için son kontroller yapılır.

**4.VERİ MANİPÜLASYONU**

Veri manipülasyonu, verileri daha anlaşılır hale getirmek için düzenleme sürecidir. Örneğin, işletmelerde çalışan bilgilerinin alfabetik olarak sıralanması, belirli bir çalışanın bilgilerine hızlı erişim sağlar ve bu da yönetim süreçlerini kolaylaştırır. Web sitesi sahipleri için, trafiği ve popüler içerikleri izlemek için kullanılan web sunucusu günlükleri bu tür düzenlemelerden faydalanabilir.Muhasebeciler, üretim maliyetleri, vergi yükümlülükleri ve fiyatlandırma eğilimlerini belirlemek için verileri düzenlemektedir. Borsa tahmincileri ise hisse senedi performanslarını öngörmek için bu teknikleri kullanır. Ayrıca, bilgisayar programları veya web sayfaları gibi dijital ortamlarda bilgilerin daha anlaşılır bir şekilde gösterilmesi için de veri biçimlendirme önemlidir.

**5. VERİ GÖRSELLEŞTİRME**

"Bir resim bin sözcüğe bedeldir" denilen söz, bugünün büyük veri çağında işletmeler için daha da anlamlı hale gelmiştir. Şirketler, çeşitli veri türlerinden, hem şirket içinde hem de bulut tabanlı kaynaklardan gelen bilgi akışıyla karşı karşıyadır ve hangi bilgilerin önemli olduğunu belirlemek giderek zorlaşmaktadır.

Görseller, bilgiyi analiz etmeyi ve önemli konuları hızla görmeyi sağlar. İnsanlar genellikle görsellere metinden daha hızlı ve etkili bir şekilde tepki verirler; beyne gelen bilgilerin yüzde 90'ı görseldir ve görseller metne kıyasla 60.000 kat daha hızlı işlenir. Bu gerçekler, bilgi analizinde ve iletmekte veri görselleştirmesinin önemini vurgular.

İyi bir veri görselleştirmesi, verileri analiz etmek ve bu analizlere dayanarak kararlar almak için son derece önemlidir. İnsanlar, ilişkileri hızlı ve kolay bir şekilde görsel olarak görebilir ve anlayabilirler. Bu, sadece ham sayıların yer aldığı tablolarda gözden kaçabilecek trendleri tespit etmelerini sağlar.

Grafikler, özel bir eğitim gerektirmeden geniş bir kullanıcı kitlesi tarafından anlaşılabilir ve yorumlanabilir. Doğru bir şekilde tasarlandığında, grafikler bilgi iletmekle kalmaz, aynı zamanda bu bilgilerin etkisini artırır, dikkat çeker ve insanların ilgisini çeker. Bu, sadece elektronik tabloların yapamayacağı bir etki yaratır.

**6.VERİ SETİ TANITIMI**

Bu veri seti, kanser hastalarının demografik bilgilerini, hastalık durumlarını, tedavi süreçlerini ve sonuçlarını içermektedir. Aşağıda her sütunun anlamı verilmiştir:

1. **id**: Hasta kimlik numarası
2. **age**: Hastanın yaşı
3. **gender**: Hastanın cinsiyeti (Female: Kadın, Male: Erkek)
4. **country**: Hastanın yaşadığı ülke
5. **diagnosis\_date**: Kanser teşhis tarihi
6. **cancer\_stage**: Kanserin evresi (Stage I, II, III, IV)
7. **beginning\_of\_treatment\_date**: Tedaviye başlama tarihi
8. **family\_history**: Ailede kanser geçmişi olup olmadığı (Yes: Var, No: Yok)
9. **smoking\_status**: Sigara içme durumu (Never Smoked: Hiç sigara içmemiş, Passive Smoker: Pasif içici, Former Smoker: Eski sigara içicisi)
10. **bmi**: Vücut kitle indeksi
11. **cholesterol\_level**: Kolesterol seviyesi (mg/dL)
12. **hypertension**: Hipertansiyon durumu (1: Var, 0: Yok)
13. **asthma**: Astım durumu (1: Var, 0: Yok)
14. **cirrhosis**: Siroz durumu (1: Var, 0: Yok)
15. **other\_cancer**: Başka bir kanser durumu (1: Var, 0: Yok)
16. **treatment\_type**: Tedavi türü (Combined: Kombine, Radiation: Radyasyon, Surgery: Cerrahi, Chemotherapy: Kemoterapi)
17. **end\_treatment\_date**: Tedavinin bitiş tarihi
18. **survived**: Sağkalım durumu (1: Hayatta, 0: Vefat)

Örnekler:

* **id 1**: 64 yaşındaki Alman bir kadın, 2016 yılında Stage III kanser teşhisi almış ve tedavisine başlamış. Kombine tedavi uygulanmış ve 2017'de tedavi bitmiş. Vefat etmiş.
* **id 2**: 50 yaşındaki Çek bir erkek, 2023 yılında Stage III kanser teşhisi almış. Radyasyon tedavisi görmüş ve 2024'te tedavi bitmiş. Vefat etmiş.

Bu veri seti, kanser hastalarının demografik ve klinik özellikleri ile tedavi süreçlerini incelemek için kullanılabilir.

**BÖLÜM 2: UYGULAMA KISMI**

**1.Gerekli Kütüphanelerin Yüklenmesi**

import matplotlib.pyplot as plt # Grafik oluşturmak için matplotlib kütüphanesini içe aktarır ve 'plt' olarak kısaltır.

import pandas as pd # Veri analizi ve manipülasyonu için pandas kütüphanesini içe aktarır ve 'pd' olarak kısaltır.

import numpy as np # Bilimsel hesaplamalar ve veri işleme için numpy kütüphanesini içe aktarır ve 'np' olarak kısaltır.

import scipy.stats as stats # İstatistiksel analizler için scipy.stats kütüphanesini içe aktarır ve 'stats' olarak kısaltır.

from scipy.stats import ttest\_ind # Bağımsız iki örnek t-testi için gerekli fonksiyonu içe aktarır.

from scipy.stats import levene # Grupların varyanslarının homojenliğini test etmek için gerekli fonksiyonu içe aktarır.

import seaborn as sns # Veri görselleştirmesi için seaborn kütüphanesini içe aktarır ve 'sns' olarak kısaltır.

import missingno as msno # Eksik verilerin görselleştirilmesi için kullanılır

import folium # Haritalar ve coğrafi veriler üzerinde interaktif harita görselleştirmesi yapmak için kullanılır

import geopy # Coğrafi veri işleme için kullanılır

**2.Csv Dosyamızı Yüklüyoruz**

#Veri seti bir CSV dosyasından okunur. Dosya adı "lung-cancer-data copy.csv" olarak belirtilmiştir ve karakter kodlaması "latin1" olarak ayarlanmıştır.

data = pd.read\_csv("lung-cancer.csv", encoding='latin1')

**3. Missigno Kütüphanesi ile Eksik Veriler Görselleştirildi**

print(data.isnull().sum()) # Eksik verileri görselleştirme

import missingno as msno

msno.matrix(data)





**4.Veri Setimizin Temel İstatistiksel Analizlerini Yapıyoruz**

# İlk 5 gözlemi görüntüle

print("Veri Seti Örneği (5 Gözlem)")

display(data.head())

# Veri setinin boyutlarını kontrol et

print("Veri Seti Boyutu:")

print(data.shape)

# Eksik değerlerin kontrolü ve temizleme

print("Eksik Değerlerin Kontrolü:")

missing\_values = data.isnull().sum()

print(missing\_values.to\_frame('Eksik Değerler'))

# Veri setini ekrana yazdırma

print("Veri Seti:")

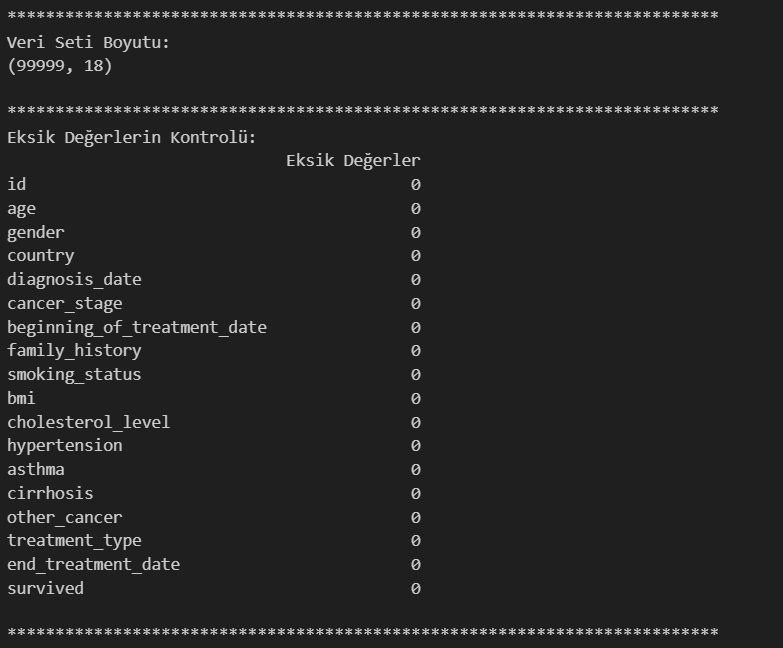
display(data)

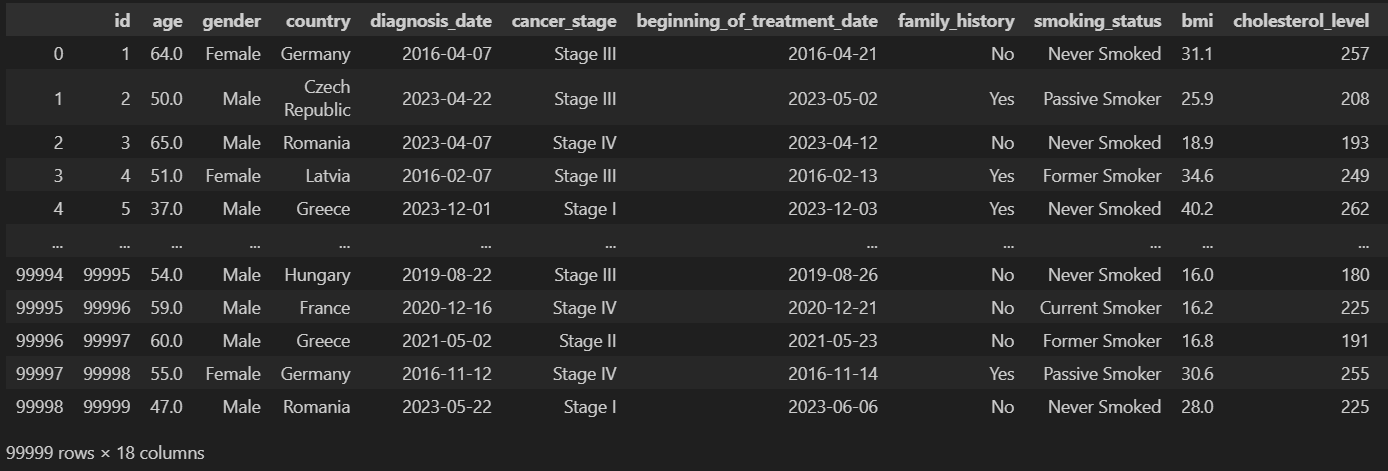
# Veri seti istatistiksel analizi

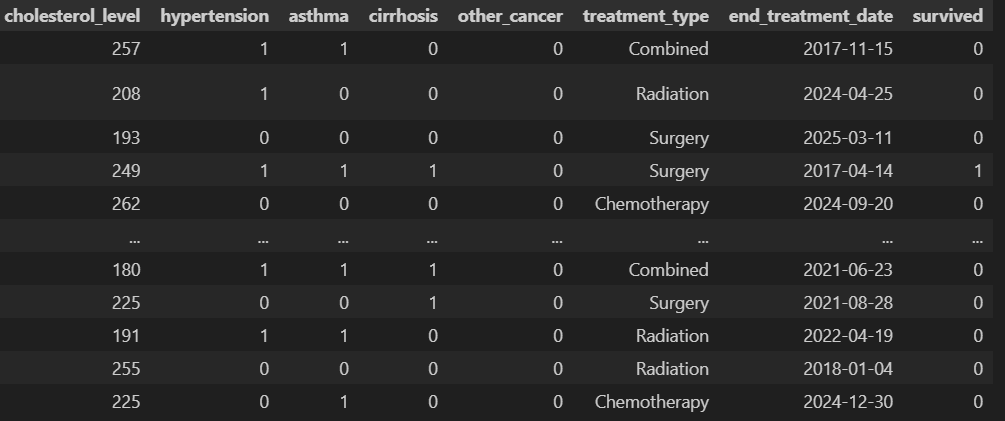
print("Veri Seti İstatistiksel Analizi:")

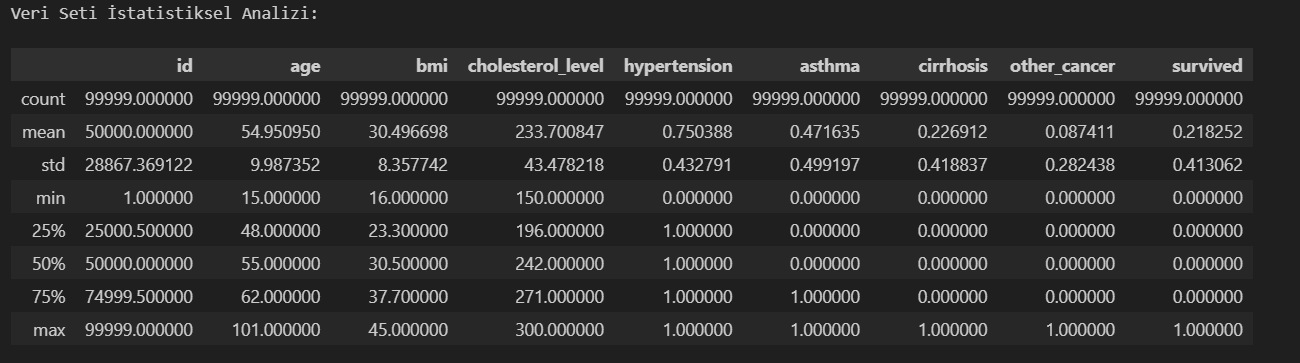
display(data.describe())











**5.Ölenlerin Cinsiyet Dağılımını Pasta Grafiği ile Gösteriyoruz**

# Verilen değerler

olenler = {'Female': 39159, 'Male': 39015}

# Toplam sayıları hesapla

toplam\_olenler = sum(olenler.values())

# Yüzde hesapla

yuzde\_olenler = {k: (v / toplam\_olenler \* 100) for k, v in olenler.items()}

# Sonuçları ekrana yazdır

print("Ölenlerin Cinsiyet Dağılımı Yüzdesi ve Sayıları:")

for k, v in olenler.items():

print(f"{k}: {v} kişi ({yuzde\_olenler[k]:.2f}%)")

# Grafik oluşturur

fig, ax = plt.subplots(figsize=(6, 6))

# Renkleri tanımlar (Cinsiyete göre: Kadın: Pembe, Erkek: Mavi)

colors = ['#ff69b4', '#1e90ff']

# Ölenlerin cinsiyet dağılımı

died\_counts = [olenler['Female'], olenler['Male']]

labels = [f"Female: {olenler['Female']} ({yuzde\_olenler['Female']:.2f}%)", f"Male: {olenler['Male']} ({yuzde\_olenler['Male']:.2f}%)"]

ax.pie(died\_counts, labels=labels, autopct='%1.1f%%', colors=colors)

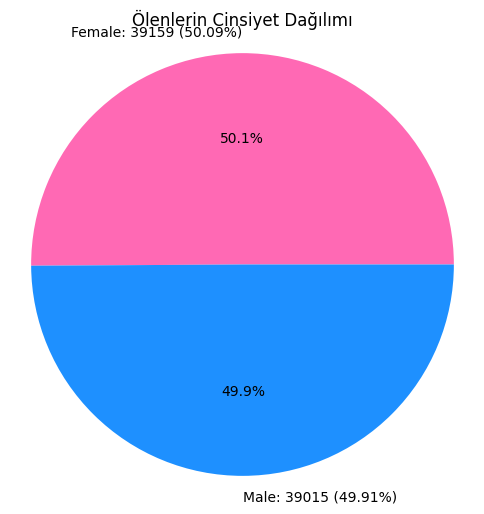
ax.set\_title('Ölenlerin Cinsiyet Dağılımı')

# Dairesel Grafik için oranları eşitler

plt.axis('equal')

# Grafiği gösterir

plt.show()



**6.Ailedeki Kanser Öyküsünü Pasta Grafiği ile Gösteriyoruz**

# 'family\_history' sütunundaki değerlerin sayısını hesapla

family\_history\_counts = data['family\_history'].value\_counts()

# Sonucu ekrana yazdır

print(family\_history\_counts)

# Renkleri tanımla (Yes: Pembe, No: Mavi)

colors = ['#ff9999', '#1e90ff']

# Pasta grafiği oluştur

plt.figure(figsize=(8, 6))

plt.pie(family\_history\_counts, labels=family\_history\_counts.index, autopct='%1.1f%%', colors=colors, startangle=140)

plt.title('Ailedeki Kanser Hastalarının Varlığı')

plt.axis('equal')

plt.show()

data = pd.read\_csv("lung-cancer.csv")

df = pd.DataFrame(data)

# 'gender' ve 'family\_history' sütunlarını gruplayarak sayılarını hesapla

grouped = df.groupby('gender')['family\_history'].value\_counts().unstack()

# Renkleri tanımla (Var: Yeşil, Yok: Mavi)

colors = ['#1f77b4', '#2ca02c']

# Pasta grafiği oluştur

plt.figure(figsize=(8, 6))

patches, texts, autotexts = plt.pie(grouped.values.flatten(), labels=['Female - Yok', 'Female - Var', 'Male - Yok', 'Male - Var'],

autopct='%1.1f%%', colors=colors, startangle=140)

# Pasta dilimlerinin yüzdelik oranlarını ekrana yazdırma

for i, autotext in enumerate(autotexts):

autotext.set\_color('white')

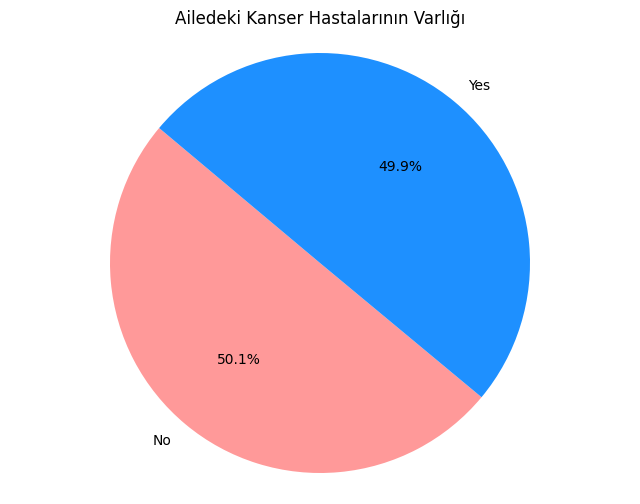
autotext.set\_fontsize(12)

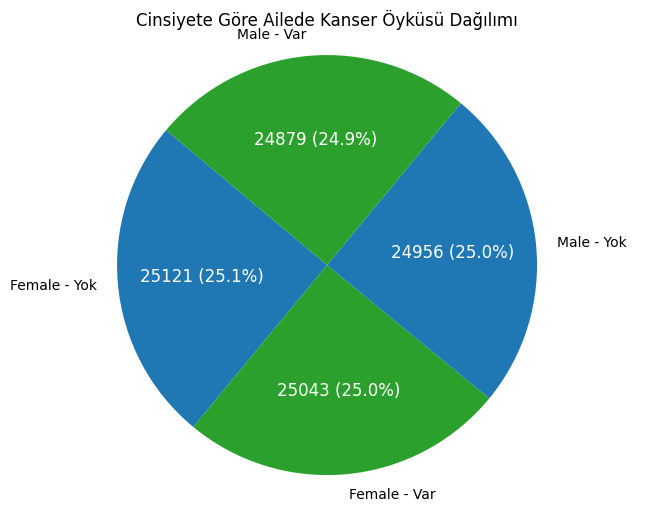
autotext.set\_text(f"{grouped.values.flatten()[i]} ({autotext.get\_text()})")

plt.title('Cinsiyete Göre Ailede Kanser Öyküsü Dağılımı')

plt.axis('equal')

plt.show()





**7.Veri Setindeki Belirli Değişkenlerin Dağılımını Pasta Grafiği İle Gösterir**

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv("lung-cancer.csv")

# İlgili değişkenlerin listesi

degiskenler = ["gender", "family\_history", "hypertension", "asthma", "cirrhosis",

"other\_cancer", "survived", "cancer\_stage", "smoking\_status",

"treatment\_type"]

# Subplotları oluşturmak için figür oluşturma

plt.figure(figsize=(15, 15))

# Renk paleti tanımlama

colors = ['#66c2a5', '#fc8d62', '#8da0cb', '#e78ac3']

# Her bir değişken için subplot oluşturma

for i, degisken in enumerate(degiskenler):

# Her bir değişkenin değerlerinin sayısını hesaplama

df = data[degisken].value\_counts()

# Subplot eklemek

plt.subplot(5, 2, i+1)

# Pasta grafiği oluşturma

plt.pie(df, labels=df.index, autopct='%1.1f%%', colors=colors, startangle=140)

plt.title(degisken) # Subplot başlığı ekleme

# Dairesel grafik için oranları eşitleme

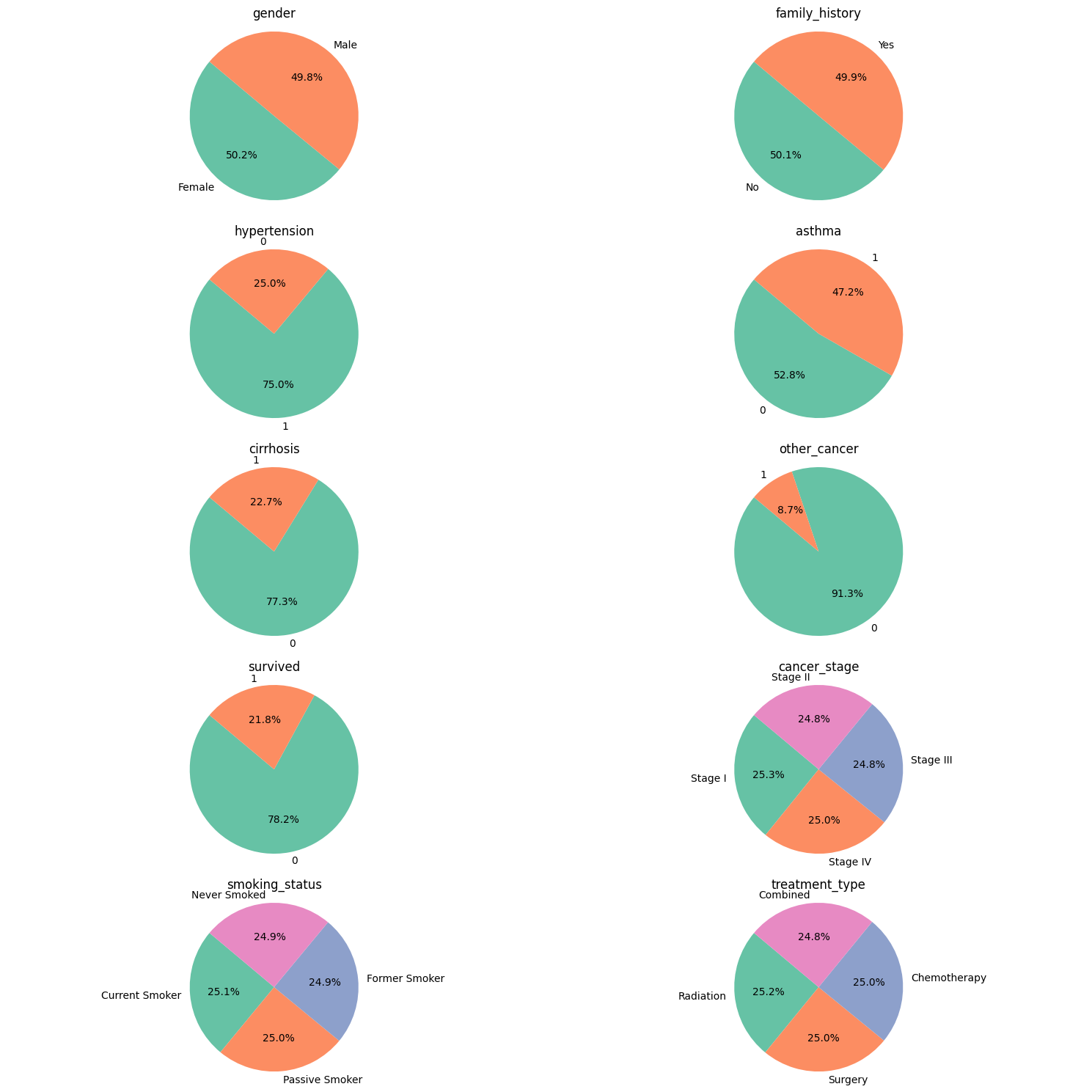
plt.axis('equal')

# Alt başlık ayarları

plt.tight\_layout()

# Grafiği gösterme

plt.show()



**8. Sayısal Değişkenler Arasındaki İlişkiyi Görmek İçin (age,bmi ve cholestrol\_level) Korelasyon Analizi Yaptım**

# Veri setini yükleme

df = pd.read\_csv('lung-cancer.csv')

# Korelasyon matrisi için seçilen sayısal sütunlar

numeric\_columns = ['age', 'bmi', 'cholesterol\_level']

numeric\_data = df[numeric\_columns]

# Korelasyon matrisini hesaplama

correlation\_matrix = numeric\_data.corr()

# Korelasyon matrisini ekrana yazdırma

print("Korelasyon Matrisi:")

print(correlation\_matrix)

# Korelasyon matrisi için heatmap oluşturma

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True)

plt.title('Korelasyon Matrisi Heatmap')

plt.show()

# Scatter plotlar için kullanılacak değişkenler

scatter\_variables = [('age', 'bmi'), ('age', 'cholesterol\_level'), ('bmi', 'cholesterol\_level')]

# Scatter plotları oluşturma

plt.figure(figsize=(18, 5))

for i, (x\_var, y\_var) in enumerate(scatter\_variables, start=1):

plt.subplot(1, 3, i)

sns.scatterplot(x=x\_var, y=y\_var, data=df, hue='survived', palette='viridis', alpha=0.8)

plt.title(f'{x\_var} vs {y\_var} Scatter Plot')

plt.xlabel(x\_var)

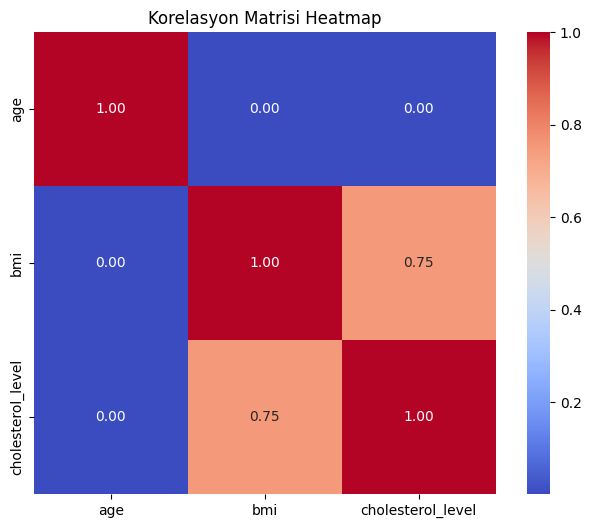
plt.ylabel(y\_var)

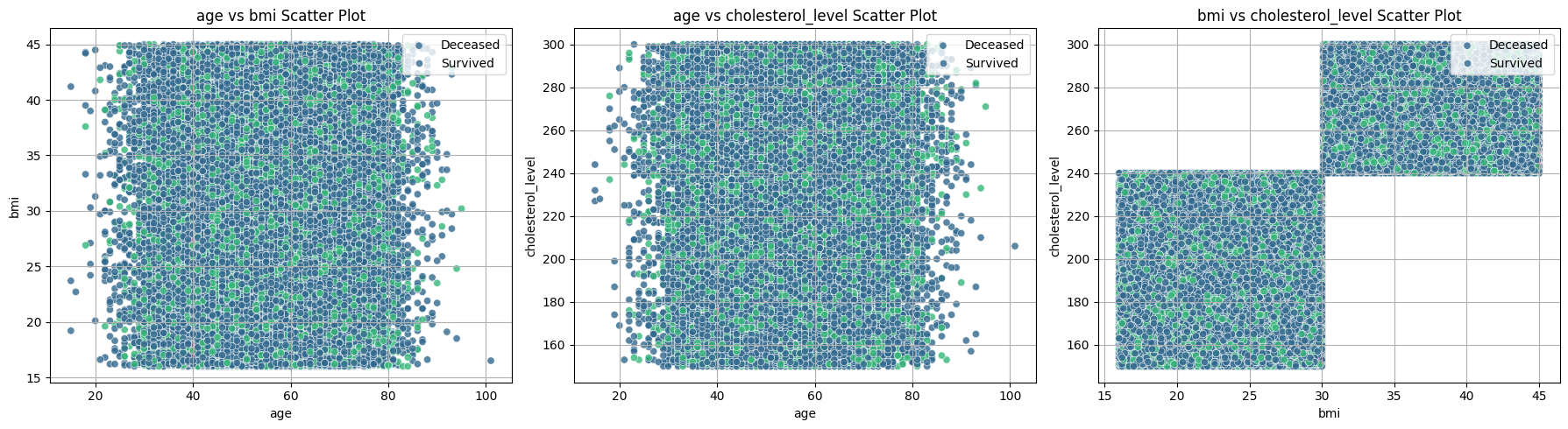
plt.legend(loc='upper right', labels=['Deceased', 'Survived'])

plt.grid(True)

plt.tight\_layout()

plt.show()





**8.1 Bu kod bloğu, bir akciğer kanseri veri setini yükler, veri setindeki sayısal sütunları seçer, bu sütunlar arasındaki korelasyon matrisini hesaplar ve bir heatmap (ısı haritası) kullanarak bu korelasyonları görselleştirir.**



**9. 9.1 Ülke (country) ve Sigara İçme Durumu (smoking\_status) için kros-tablasyon ve chi-kare testi**

from scipy.stats import chi2\_contingency

# Veri setini yükleme

df = pd.read\_csv('lung-cancer.csv')

# Kros-tablasyon oluşturma

country\_smoking\_cross = pd.crosstab(df['country'], df['smoking\_status'])

# Kros-tablo görselleştirme

plt.figure(figsize=(10, 6))

colors = sns.color\_palette('husl', n\_colors=len(df['smoking\_status'].unique()))

country\_smoking\_cross.plot(kind='bar', stacked=True, ax=plt.gca(), color=colors)

plt.title('Country vs Smoking Status Distribution')

plt.xlabel('Country')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.legend(title='Smoking Status')

plt.tight\_layout()

plt.show()

# Chi-Kare Testi

chi2, p\_value, dof, expected = chi2\_contingency(country\_smoking\_cross)

print("Chi-Kare Testi Sonuçları:")

print(f"Chi-Kare İstatistiği: {chi2}")

print(f"p-value: {p\_value}")

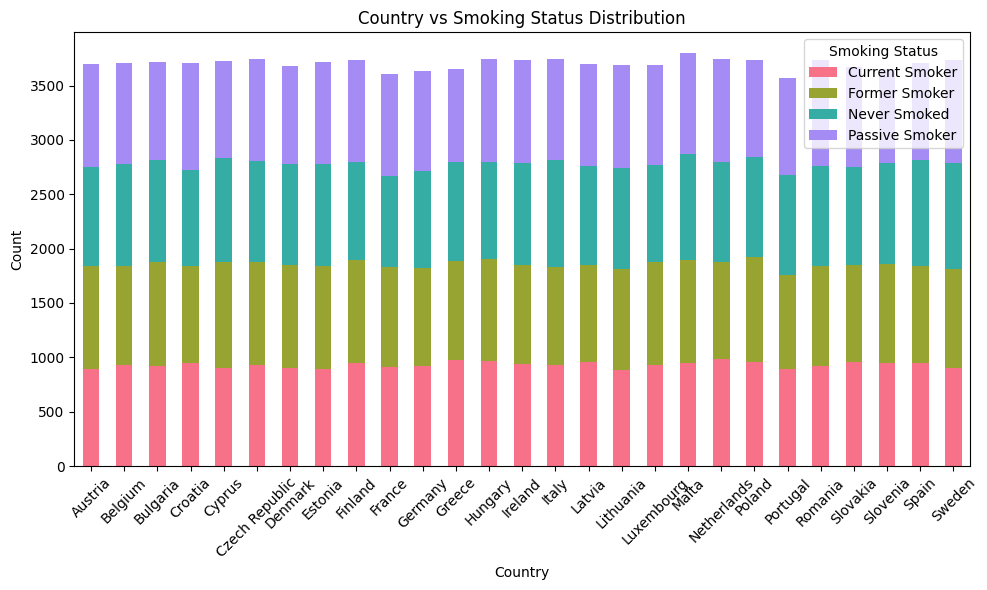
print("Ölçeklenmiş Beklenen Frekanslar Tablosu:")

print(expected)

# Verileri ekrana basma

print("\nCountry vs Smoking Status Distribution:")

print(country\_smoking\_cross)

**10. Kanser Evresi (cancer\_stage) ve Tedavi Türü (treatment\_type) için kros-tablasyon ve chi-kare testi**

from scipy.stats import chi2\_contingency

# Veri setini yükleme (örnek olarak kullanılmıştır, gerçek veri setinize göre ayarlayın)

df = pd.read\_csv('lung-cancer.csv')

# Kros-tablasyon oluşturma

cancer\_treatment\_cross = pd.crosstab(df['cancer\_stage'], df['treatment\_type'])

# Chi-kare testi yapma

chi2, p\_value, dof, expected = chi2\_contingency(cancer\_treatment\_cross)

# Sonuçları ekrana yazdırma

print("Chi-kare Testi Sonuçları:")

print(f"Chi-kare İstatistiği: {chi2}")

print(f"p-value: {p\_value}")

print("Ölçeklenmiş Beklenen Frekanslar Tablosu:")

print(expected)

# Kros-tablo görselleştirme

plt.figure(figsize=(8, 6))

sns.heatmap(cancer\_treatment\_cross, annot=True, cmap='YlGnBu', fmt='d', cbar=False)

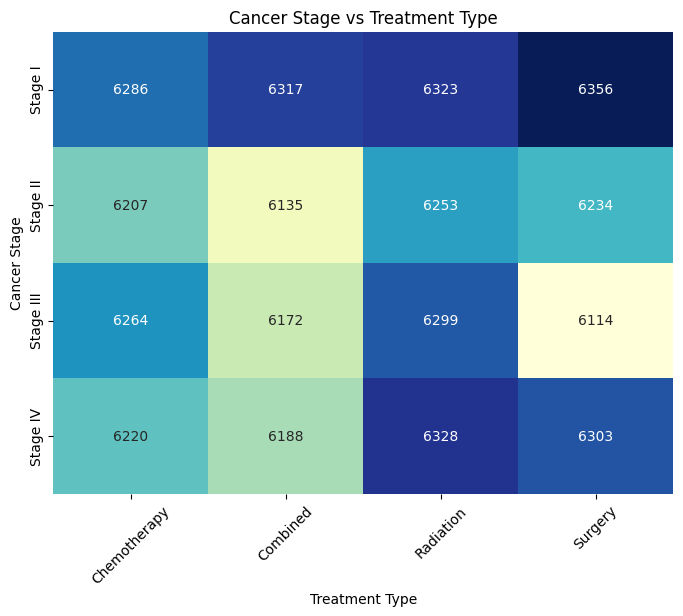
plt.title('Cancer Stage vs Treatment Type')

plt.xlabel('Treatment Type')

plt.ylabel('Cancer Stage')

plt.xticks(rotation=45)

plt.show()



10. Sigara İçme Durumu (smoking\_status) ve Kanser Evresi (cancer\_stage) için kros-tablasyon ve chi-kare testi

from scipy.stats import chi2\_contingency

# Veri setini yükleme

df = pd.read\_csv('lung-cancer.csv')

# Kros-tablasyon oluşturma

smoking\_cancer\_cross = pd.crosstab(df['smoking\_status'], df['cancer\_stage'])

# Renk paleti

colors = sns.color\_palette('Pastel1', n\_colors=len(df['cancer\_stage'].unique()))

# Kros-tablo görselleştirme

plt.figure(figsize=(10, 6))

smoking\_cancer\_cross.plot(kind='bar', ax=plt.gca(), color=colors)

plt.title('Smoking Status vs Cancer Stage')

plt.xlabel('Smoking Status')

plt.ylabel('Count')

plt.xticks(rotation=0)

plt.legend(title='Cancer Stage')

plt.tight\_layout()

plt.show()

# Chi-Kare Testi

chi2, p\_value, dof, expected = chi2\_contingency(smoking\_cancer\_cross)

print("Chi-Kare Testi Sonuçları:")

print(f"Chi-Kare İstatistiği: {chi2}")

print(f"p-value: {p\_value}")

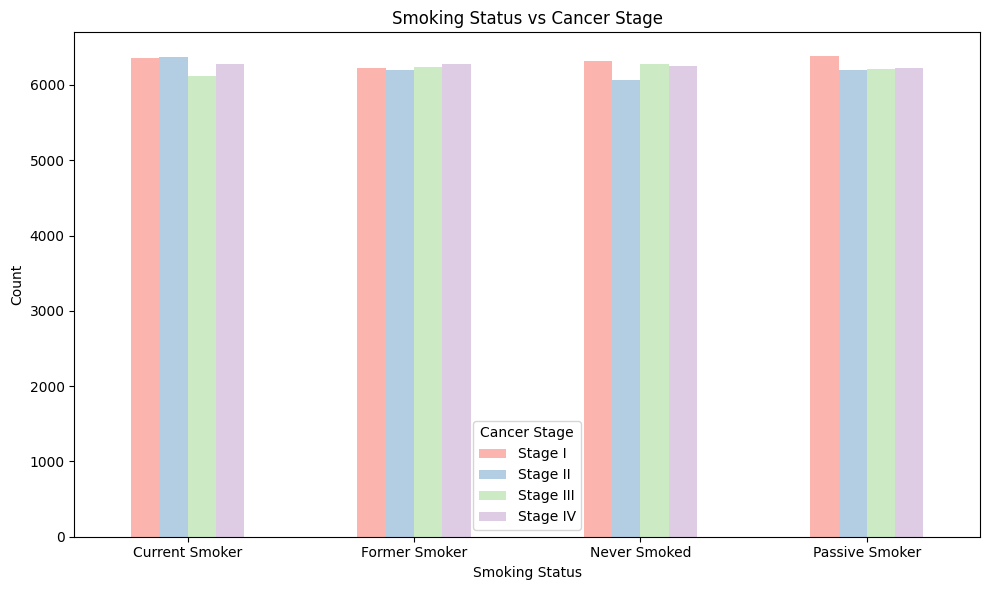
print("Ölçeklenmiş Beklenen Frekanslar Tablosu:")

print(expected)

# Verileri ekrana basma

print("\nSmoking Status vs Cancer Stage:")

print(smoking\_cancer\_cross)



**11. family\_history ve treatment\_type adlı iki kategorik değişken arasındaki ilişkiyi incelemek için Chi-Kare testi yapar ve sonuçları görselleştirir**

# 2. İlgilendiğimiz değişkenler

var1 = 'family\_history'

var2 = 'treatment\_type'

# 3. Çapraz tabloyu oluştur

contingency\_table = pd.crosstab(lung\_cancer\_data[var1], lung\_cancer\_data[var2])

# 4. Chi-Kare testi uygula

chi2, p, dof, expected = chi2\_contingency(contingency\_table)

# 5. P değerini yazdır

print(f"Chi-Kare İstatistiği: {chi2}")

print(f"P-value: {p}")

# 6. Çapraz tabloyu görselleştir (isteğe bağlı)

plt.figure(figsize=(8, 6))

sns.heatmap(contingency\_table, annot=True, cmap='viridis', cbar=True, fmt='d')

plt.title(f'Chi-Kare Testi: {var1} vs {var2}')

plt.xlabel(var2)

plt.ylabel(var1)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 7. P değerini yorumla

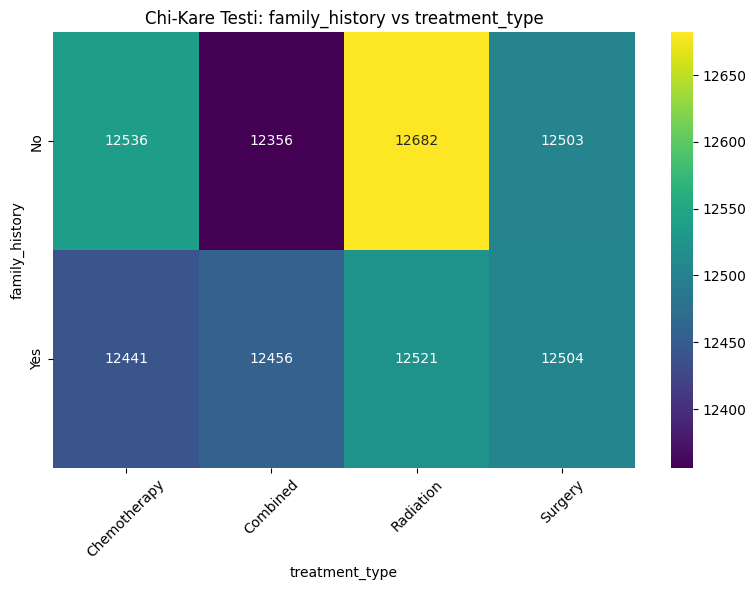
alpha = 0.05

if p < alpha:

print("İki değişken arasında anlamlı bir ilişki vardır (Null hipotez reddedilir).")

else:

print("İki değişken arasında anlamlı bir ilişki yoktur (Null hipotezi kabul edilir).")



**12. Tedavi türü ile hayatta kalma durumu arasındaki ilişkiyi incelemek için chi-kare testi uygulandı**

import pandas as pd

from scipy.stats import chi2\_contingency

# Veri setini yükle

lung\_cancer\_data = pd.read\_csv('lung-cancer.csv')

# İlgilendiğimiz değişkenler

var1 = 'survived'

var2 = 'treatment\_type'

# Çapraz tabloyu oluştur

contingency\_table = pd.crosstab(lung\_cancer\_data[var1], lung\_cancer\_data[var2])

# Chi-Kare testi uygula

chi2, p, dof, expected = chi2\_contingency(contingency\_table)

# P değerini yazdır

print(f"Chi-Kare İstatistiği: {chi2}")

print(f"P-value: {p}")

# P değerini yorumla

alpha = 0.05

if p < alpha:

print("Hayatta kalma durumu ile tedavi türü arasında anlamlı bir ilişki vardır (Null hipotez reddedilir).")

else:

print("Hayatta kalma durumu ile tedavi türü arasında anlamlı bir ilişki yoktur (Null hipotezi kabul edilir).")

Chi-Kare İstatistiği: 4.840750200685195 P-value: 0.18383656352123692 Hayatta kalma durumu ile tedavi türü arasında anlamlı bir ilişki yoktur (Null hipotezi kabul edilir).

**13. Bu kod, veri setindeki belirli değişkenler için Kolmogorov-Smirnov testini (KS testi) uygulayarak normal dağılıma uygunluğunu kontrol eder.**

# Kolmogorov-Smirnov testini uygulayacağımız verileri seçtiğimiz liste

degiskenler = ["age", "cholesterol\_level" ,"bmi"]

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım

sag = data[data['survived'] == 1]

olu = data[data['survived'] == 0]

# sag için Kolmogorov-Smirnov testi

print("Sağ:")

print("-----------------------------------------------------------------------------------------")

for degisken in degiskenler:

ks\_test = stats.kstest(sag[degisken], 'norm', args=(np.mean(sag[degisken]), np.std(sag[degisken])))

normallik\_testleri[f'Sag\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

# Ölü için Kolmogorov-Smirnov testi

print("\nÖlü:")

print("-----------------------------------------------------------------------------------------")

for degisken in degiskenler:

ks\_test = stats.kstest(olu[degisken], 'norm', args=(np.mean(olu[degisken]), np.std(olu[degisken])))

normallik\_testleri[f'Ölü\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

Sağ: ----------------------------------------------------------------------------------------- age: Normal dağılıma uymaz (p-value = 3.4387854479963536e-11) cholesterol\_level: Normal dağılıma uymaz (p-value = 2.791803214890963e-120) bmi: Normal dağılıma uymaz (p-value = 1.091065545045385e-71) Ölü: ----------------------------------------------------------------------------------------- age: Normal dağılıma uymaz (p-value = 4.6199293395107087e-32) cholesterol\_level: Normal dağılıma uymaz (p-value = 0.0) bmi: Normal dağılıma uymaz (p-value = 1.8996260561585353e-236)

**14. Bu kodun amacı, belirli bir özellik (hipertansiyon, astım, siroz, diğer kanser türleri, yaşama durumu gibi) altında belirlenen değişkenlerin (yaş, kolesterol seviyesi, BMI gibi) dağılımlarını incelemektir.**

columns = ["hypertension","asthma", "cirrhosis", "other\_cancer", "survived"]

for column in columns:

# Değişkenler

degiskenler = ["age", "cholesterol\_level", "bmi"]

# Verileri cinsiyete göre ayırma

columnA = data[data[column] == 1]

columnB = data[data[column] == 0]

fig, axes = plt.subplots(len(degiskenler), 2, figsize=(20, 5 \* len(degiskenler)))

# Her bir değişken için histogramların çizimi

for i, degisken in enumerate(degiskenler):

sns.histplot(columnA[degisken], bins=20, color='#1e90ff', kde=True, ax=axes[i, 0])

axes[i, 0].set\_title(f"{column} var : {degisken} Değişkeni Histogramı")

axes[i, 0].set\_xlabel("Değerler")

axes[i, 0].set\_ylabel("Frekans")

sns.histplot(columnB[degisken], bins=20, color='#ff69b4', kde=True, ax=axes[i, 1])

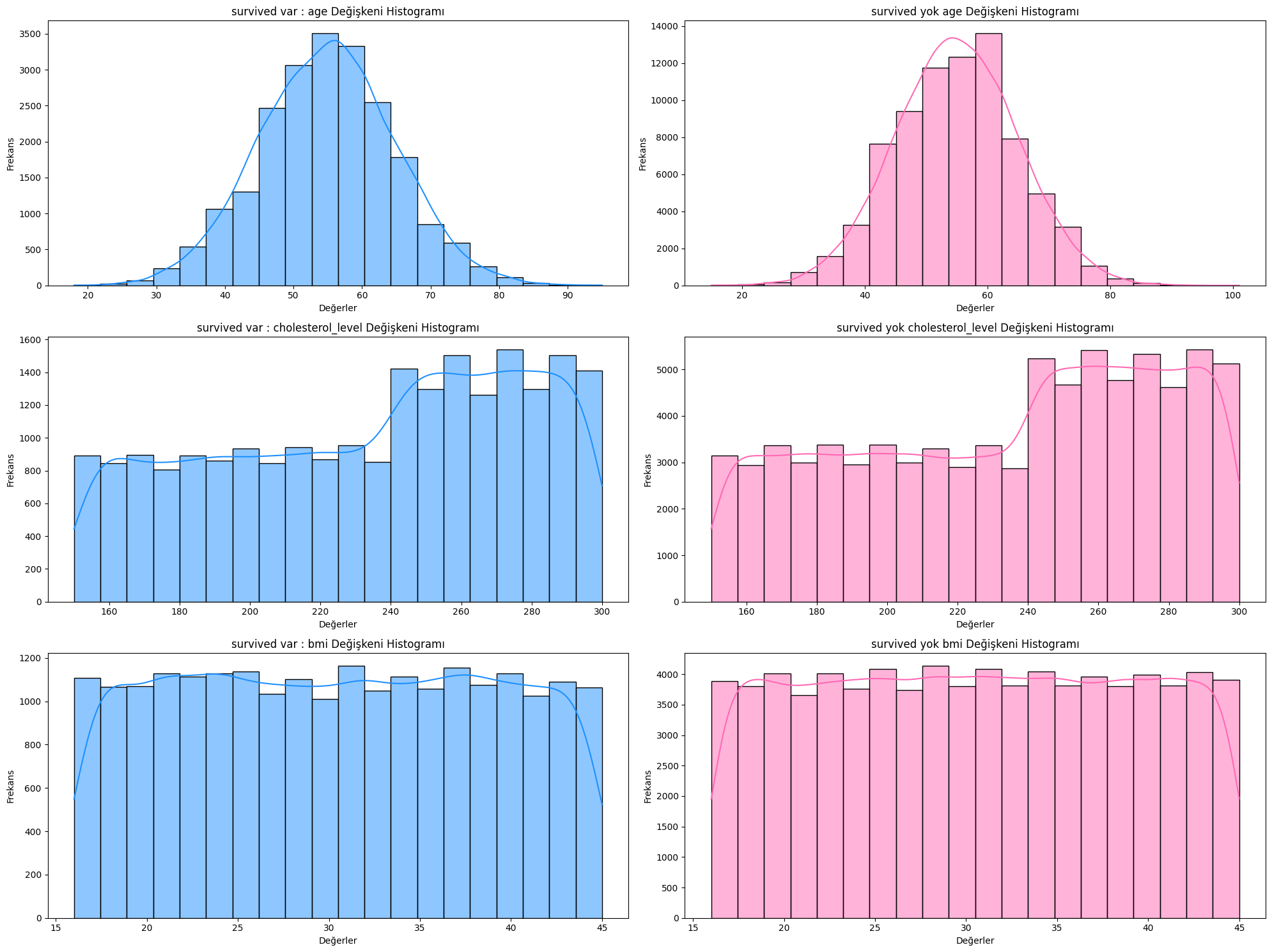
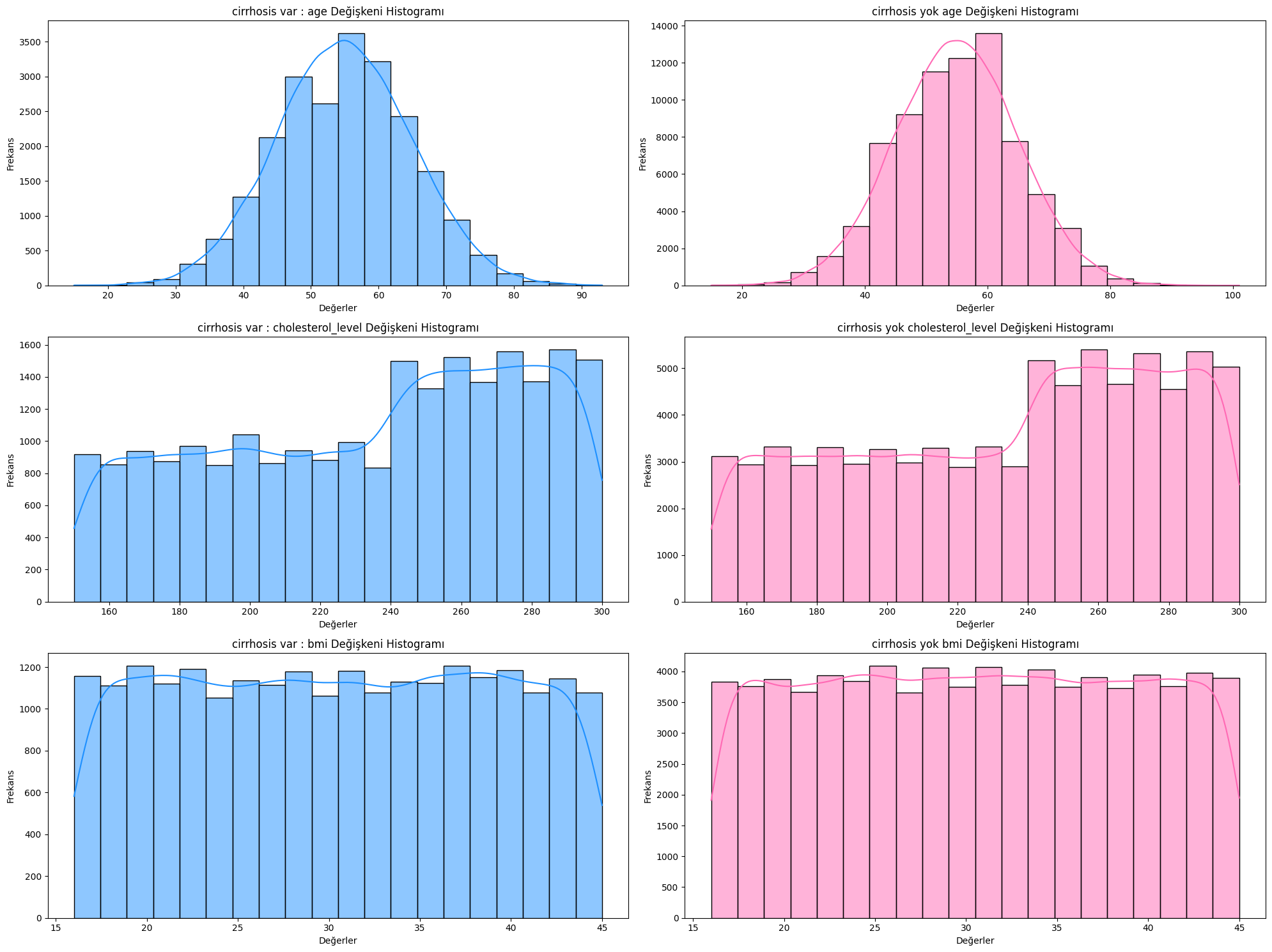
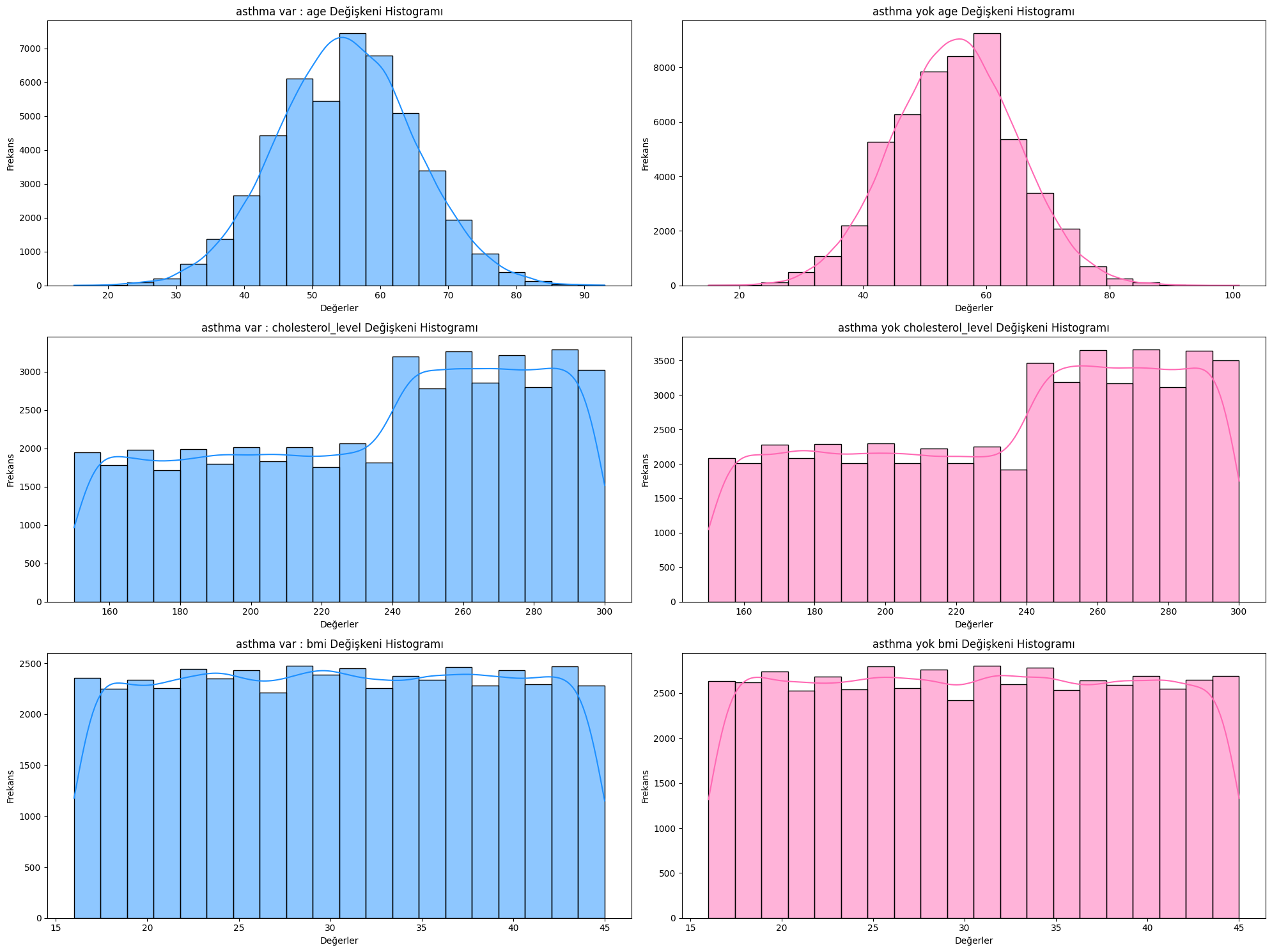
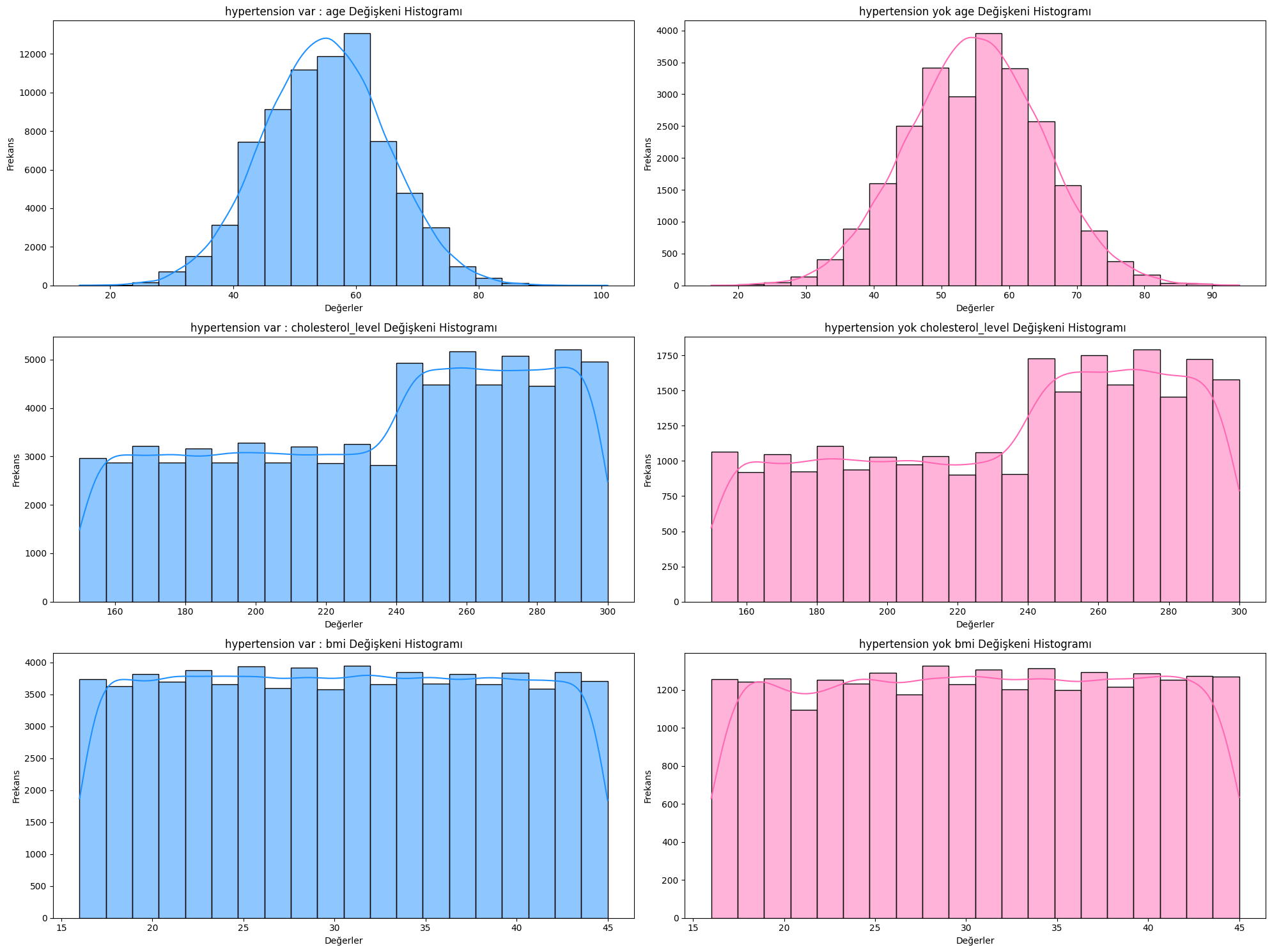
axes[i, 1].set\_title(f"{column} yok {degisken} Değişkeni Histogramı")

axes[i, 1].set\_xlabel("Değerler")

axes[i, 1].set\_ylabel("Frekans")

plt.tight\_layout()

plt.show()



**15. Bu kodun amacı, veri setindeki belirli değişkenlerin (yaş, kolesterol seviyesi, BMI) cinsiyete göre normal dağılıma uygunluğunu kontrol etmek için Kolmogorov-Smirnov testini kullanır.**

# Kolmogorov-Smirnov testini uygulayacağımız verileri seçtiğimiz liste

degiskenler = ["age", "cholesterol\_level" ,"bmi"]

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım

erkekler = data[data['gender'] == 'Male']

kadinlar = data[data['gender'] == 'Female']

# Erkekler için Kolmogorov-Smirnov testi

print("Erkekler:")

print("-----------------------------------------------------------------------------------------")

for degisken in degiskenler:

ks\_test = stats.kstest(erkekler[degisken], 'norm', args=(np.mean(erkekler[degisken]), np.std(erkekler[degisken])))

normallik\_testleri[f'Erkekler\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

# Kadınlar için Kolmogorov-Smirnov testi

print("\nKadınlar:")

print("-----------------------------------------------------------------------------------------")

for degisken in degiskenler:

ks\_test = stats.kstest(kadinlar[degisken], 'norm', args=(np.mean(kadinlar[degisken]), np.std(kadinlar[degisken])))

normallik\_testleri[f'Kadinlar\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

Erkekler: ----------------------------------------------------------------------------------------- age: Normal dağılıma uymaz (p-value = 4.140512290933633e-24) cholesterol\_level: Normal dağılıma uymaz (p-value = 4.659133066004392e-286) bmi: Normal dağılıma uymaz (p-value = 5.907110249386319e-150) Kadınlar: ----------------------------------------------------------------------------------------- age: Normal dağılıma uymaz (p-value = 2.9156476792243383e-21) cholesterol\_level: Normal dağılıma uymaz (p-value = 1.478787108272669e-286) bmi: Normal dağılıma uymaz (p-value = 4.5217421018995287e-156)

**16.Bu kod, veri setindeki belirli değişkenlerin (yaş, kolesterol seviyesi, BMI) cinsiyete göre normal dağılıma uygunluğunu kontrol etmek için Kolmogorov-Smirnov testini kullanır.**

# Değişkenler

degiskenler = ["age", "cholesterol\_level", "bmi"]

# Verileri cinsiyete göre ayırma

erkekler = data[data['gender'] == 'Male']

kadinlar = data[data['gender'] == 'Female']

fig, axes = plt.subplots(len(degiskenler), 2, figsize=(20, 5 \* len(degiskenler)))

# Her bir değişken için histogramların çizimi

for i, degisken in enumerate(degiskenler):

# Erkekler

sns.histplot(erkekler[degisken], bins=20, color='#1e90ff', kde=True, ax=axes[i, 0])

axes[i, 0].set\_title(f"Erkekler: {degisken} Değişkeni Histogramı")

axes[i, 0].set\_xlabel("Değerler")

axes[i, 0].set\_ylabel("Frekans")

# Kadınlar

sns.histplot(kadinlar[degisken], bins=20, color='#ff69b4', kde=True, ax=axes[i, 1])

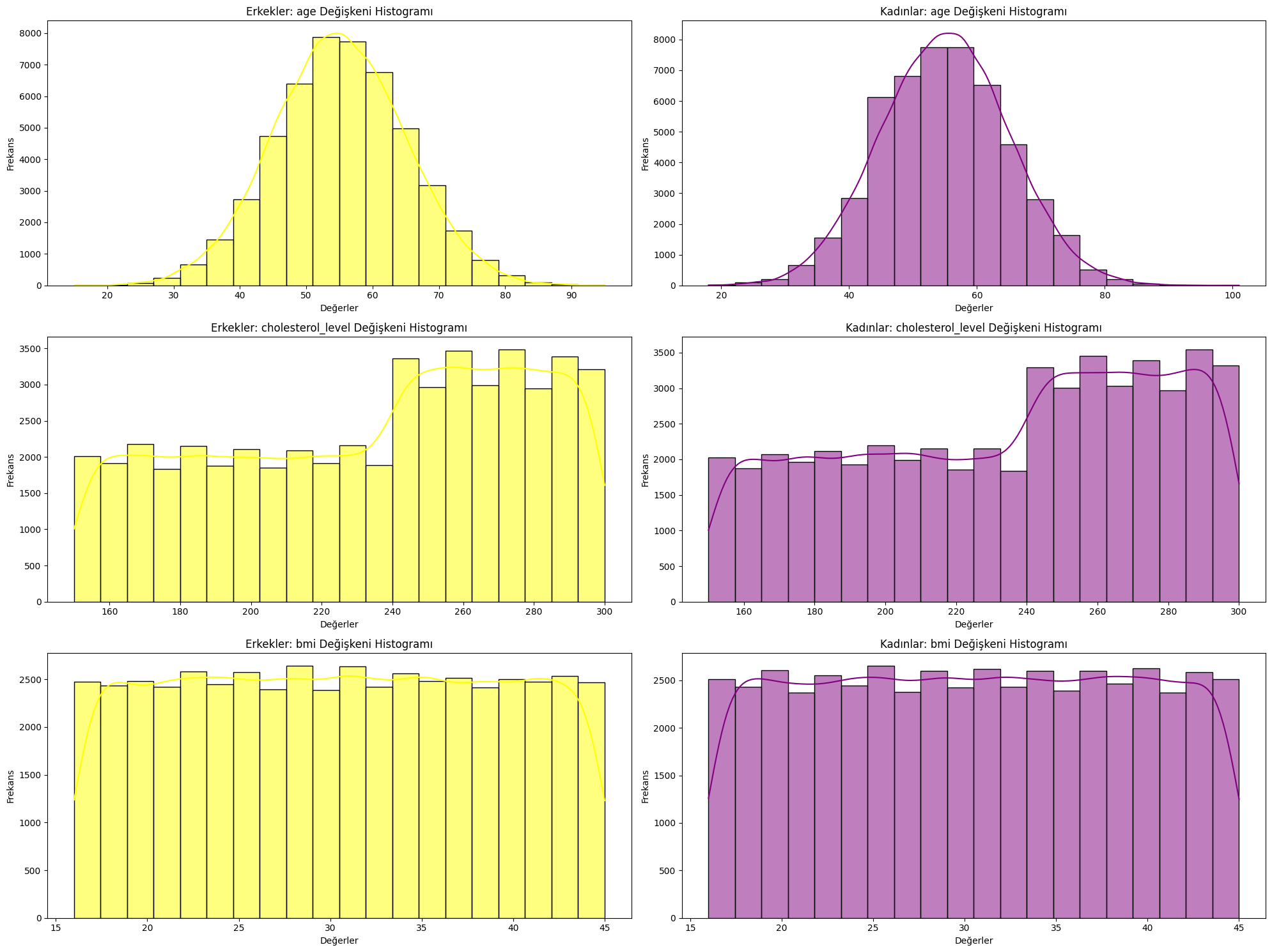
axes[i, 1].set\_title(f"Kadınlar: {degisken} Değişkeni Histogramı")

axes[i, 1].set\_xlabel("Değerler")

axes[i, 1].set\_ylabel("Frekans")

plt.tight\_layout()

plt.show()



**17. Erkekler için kolesterol seviyelerine göre normal, sınırda yüksek ve yüksek kolestrol gruplarını ayrı ayrı filtreleyerek Kolmogorov-Smirnov testini yapar.**

#Erkekler için normal, sınırda yüksek ve yüksek kolestrol seviyelerini ayrı ayrı filtreleyereke testimizi yapalım.

degiskenler = ["age", "cholesterol\_level","bmi"]

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım

erkekler = data[data['gender'] == 'Male']

cholesterol\_level = {

'Normal': (0,200 ),

'Sınırda\_yüksek': (200, 240),

'Yüksek': (240, np.inf)

}

print("\nErkeklerin kolestrol seviyelerine göre yapılan testler:")

for group\_name, (total\_min, total\_max) in cholesterol\_level.items():

cholesterol\_level = erkekler[(erkekler['cholesterol\_level'] >= total\_min) & (erkekler['cholesterol\_level'] < total\_max)]

print(f"\nKolestrol Seviyesi: {group\_name}")

for degisken in degiskenler:

ks\_test = stats.kstest(cholesterol\_level[degisken], 'norm', args=(np.mean(cholesterol\_level[degisken]), np.std(cholesterol\_level[degisken])))

normallik\_testleri[f'Erkekler\_{group\_name}\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

Erkeklerin kolestrol seviyelerine göre yapılan testler: Kolestrol Seviyesi: Normal age: Normal dağılıma uymaz (p-value = 5.683741487666462e-07) cholesterol\_level: Normal dağılıma uymaz (p-value = 3.026021640947003e-54) bmi: Normal dağılıma uymaz (p-value = 1.9516074158877902e-46) Kolestrol Seviyesi: Sınırda\_yüksek age: Normal dağılıma uymaz (p-value = 2.503194162530114e-07) cholesterol\_level: Normal dağılıma uymaz (p-value = 7.34914727295841e-48) bmi: Normal dağılıma uymaz (p-value = 2.0320113289808594e-34) Kolestrol Seviyesi: Yüksek age: Normal dağılıma uymaz (p-value = 8.493622496885339e-12) cholesterol\_level: Normal dağılıma uymaz (p-value = 4.231791055716357e-95) bmi: Normal dağılıma uymaz (p-value = 2.9145817117565947e-81)

**18.Erkekler için kolesterol seviyelerine göre normal, sınırda yüksek ve yüksek kolestrol gruplarını ayrı ayrı filtreleyerek Kolmogorov-Smirnov testinin görselleştirilmesini sağlar**

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from scipy import stats

# Veri setini yükleme

data = pd.read\_csv('lung-cancer.csv')

# Değişkenler

degiskenler = ["age", "cholesterol\_level", "bmi"]

# Normallik testlerinin sonuçlarını saklamak için boş bir sözlük

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım (örneğin, erkekler için)

erkekler = data[data['gender'] == 'Male']

# Kolesterol seviyeleri

cholesterol\_levels = {

'Normal': (0, 200),

'Sınırda\_yüksek': (200, 240),

'Yüksek': (240, np.inf)

}

print("\nErkeklerin kolestrol seviyelerine göre yapılan testler:")

for group\_name, (total\_min, total\_max) in cholesterol\_levels.items():

group\_data = erkekler[(erkekler['cholesterol\_level'] >= total\_min) & (erkekler['cholesterol\_level'] < total\_max)]

print(f"\nKolestrol Seviyesi: {group\_name}")

for i, degisken in enumerate(degiskenler):

ks\_test = stats.kstest(group\_data[degisken], 'norm', args=(np.mean(group\_data[degisken]), np.std(group\_data[degisken])))

normallik\_testleri[f'Erkekler\_{group\_name}\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

# Histogramları çizmek için tek bir subplot oluşturalım

fig, ax = plt.subplots(1, 1, figsize=(8, 6))

fig.suptitle(f"Erkekler - {group\_name} Kolestrol Seviyesi", fontsize=14)

# Her bir değişken için histogramları çizelim

color\_palette = ['skyblue', 'salmon', 'lightgreen'] # Renk paleti

for i, degisken in enumerate(degiskenler):

sns.histplot(group\_data[degisken], bins=20, kde=True, color=color\_palette[i], label=degisken, ax=ax)

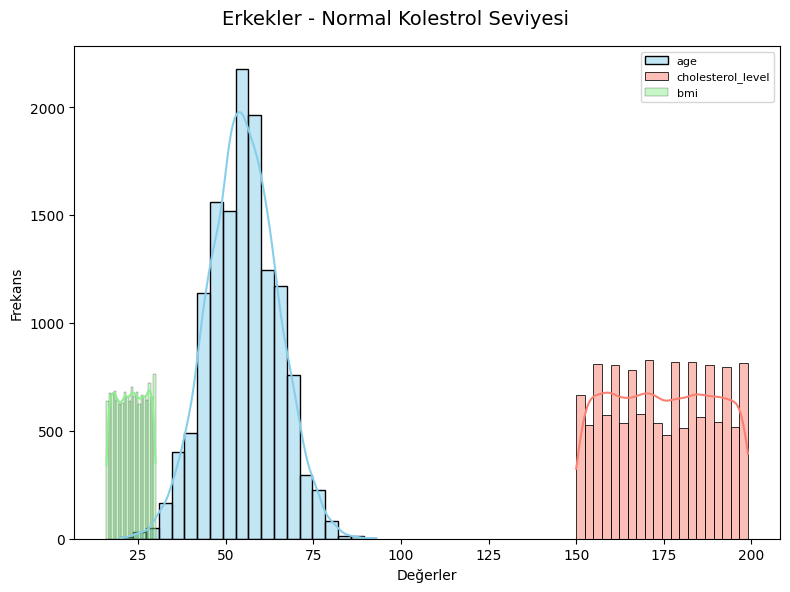
ax.set\_xlabel("Değerler", fontsize=10)

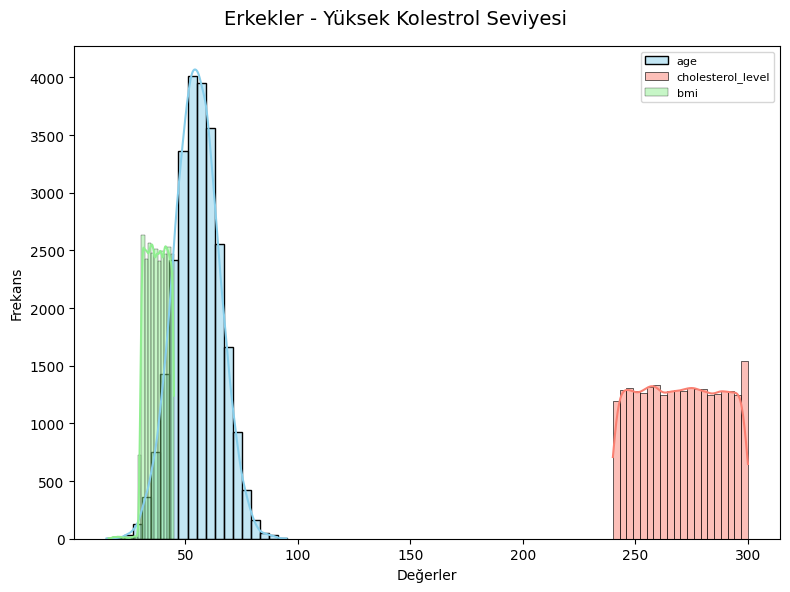
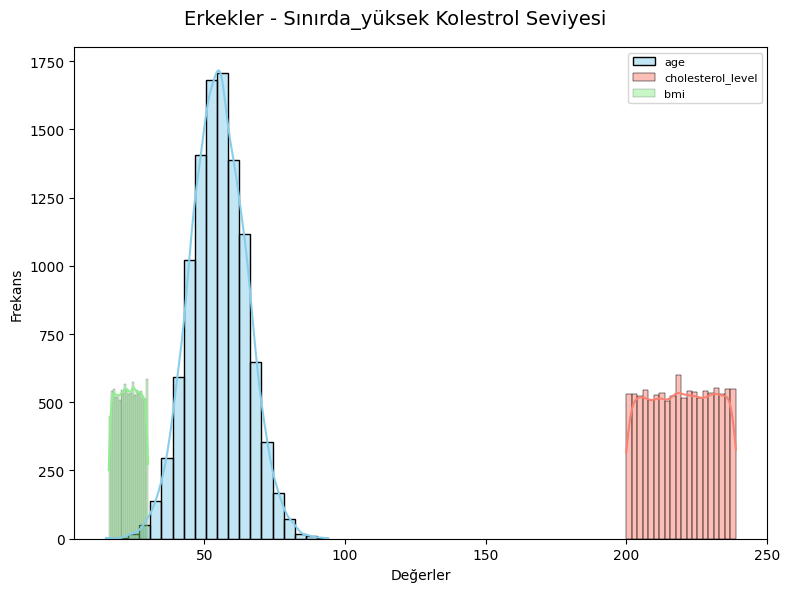
ax.set\_ylabel("Frekans", fontsize=10)

ax.legend(fontsize=8) # Değişken etiketleri için küçük bir legende sahip olunur

plt.tight\_layout()

plt.show()





**19. Kadınlar için kolesterol seviyelerine göre normal, sınırda yüksek ve yüksek kolestrol gruplarını ayrı ayrı filtreleyerek Kolmogorov-Smirnov testini yapar.**

#Kadınlar için normal, sınırda yüksek ve yüksek kolestrol seviyelerini ayrı ayrı filtreleyereke testimizi yapalım.

degiskenler = ["age", "cholesterol\_level","bmi"]

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım

kadinlar = data[data['gender'] == 'Female']

cholesterol\_level = {

'Normal': (0,200 ),

'Sınırda\_yüksek': (200, 240),

'Yüksek': (240, np.inf)

}

print("\nKadınların kolestrol seviyelerine göre yapılan testler:")

for group\_name, (total\_min, total\_max) in cholesterol\_level.items():

group\_data = kadinlar[(kadinlar['cholesterol\_level'] >= total\_min) & (kadinlar['cholesterol\_level'] < total\_max)]

print(f"\nKolestrol Seviyesi: {group\_name}")

for degisken in degiskenler:

ks\_test = stats.kstest(group\_data[degisken], 'norm', args=(np.mean(group\_data[degisken]), np.std(group\_data[degisken])))

normallik\_testleri[f'Kadınlar\_{group\_name}\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

Kadınların kolestrol seviyelerine göre yapılan testler: Kolestrol Seviyesi: Normal age: Normal dağılıma uymaz (p-value = 2.2369773087119188e-07) cholesterol\_level: Normal dağılıma uymaz (p-value = 8.358884028527105e-56) bmi: Normal dağılıma uymaz (p-value = 8.364599798197917e-48) Kolestrol Seviyesi: Sınırda\_yüksek age: Normal dağılıma uymaz (p-value = 9.205314205930924e-06) cholesterol\_level: Normal dağılıma uymaz (p-value = 1.82401003491283e-55) bmi: Normal dağılıma uymaz (p-value = 3.0551587448620506e-38) Kolestrol Seviyesi: Yüksek age: Normal dağılıma uymaz (p-value = 7.285166964973698e-12) cholesterol\_level: Normal dağılıma uymaz (p-value = 1.9231022294728605e-107) bmi: Normal dağılıma uymaz (p-value = 4.6515692530017956e-74)

**20. Kadınlar için kolesterol seviyelerine göre normal, sınırda yüksek ve yüksek kolestrol gruplarını ayrı ayrı filtreleyerek Kolmogorov-Smirnov testinin görselleştirilmesini sağlar**

# Değişkenler

degiskenler = ["age", "cholesterol\_level", "bmi"]

# Normallik testlerinin sonuçlarını saklamak için boş bir sözlük

normallik\_testleri = {}

# Verileri cinsiyete göre ayıralım (örneğin, kadınlar için)

kadinlar = data[data['gender'] == 'Female']

# Kolesterol seviyeleri

cholesterol\_levels = {

'Normal': (0, 200),

'Sınırda\_yüksek': (200, 240),

'Yüksek': (240, np.inf)

}

print("\nKadınların kolestrol seviyelerine göre yapılan testler:")

for group\_name, (total\_min, total\_max) in cholesterol\_levels.items():

group\_data = kadinlar[(kadinlar['cholesterol\_level'] >= total\_min) & (kadinlar['cholesterol\_level'] < total\_max)]

print(f"\nKolestrol Seviyesi: {group\_name}")

for degisken in degiskenler:

ks\_test = stats.kstest(group\_data[degisken], 'norm', args=(np.mean(group\_data[degisken]), np.std(group\_data[degisken])))

normallik\_testleri[f'Kadınlar\_{group\_name}\_{degisken}'] = {'Kolmogorov-Smirnov': ks\_test}

if ks\_test.pvalue > 0.05:

print(f"{degisken}: Normal dağılıma uygundur (p-value = {ks\_test.pvalue})")

else:

print(f"{degisken}: Normal dağılıma uymaz (p-value = {ks\_test.pvalue})")

# Histogramları çizmek için tek bir subplot oluşturalım

fig, axes = plt.subplots(len(degiskenler), figsize=(10, 8))

fig.suptitle(f"Kadınlar - {group\_name} Kolestrol Seviyesi", fontsize=14)

# Her bir değişken için histogramları çizelim

color\_palette = ['lightblue', 'salmon', 'lightgreen'] # Renk paleti

for i, degisken in enumerate(degiskenler):

sns.histplot(group\_data[degisken], bins=20, kde=True, color=color\_palette[i], ax=axes[i])

axes[i].set\_title(f"{degisken} Değişkeni Histogramı", fontsize=12)

axes[i].set\_xlabel("Değerler", fontsize=10)

axes[i].set\_ylabel("Frekans", fontsize=10)

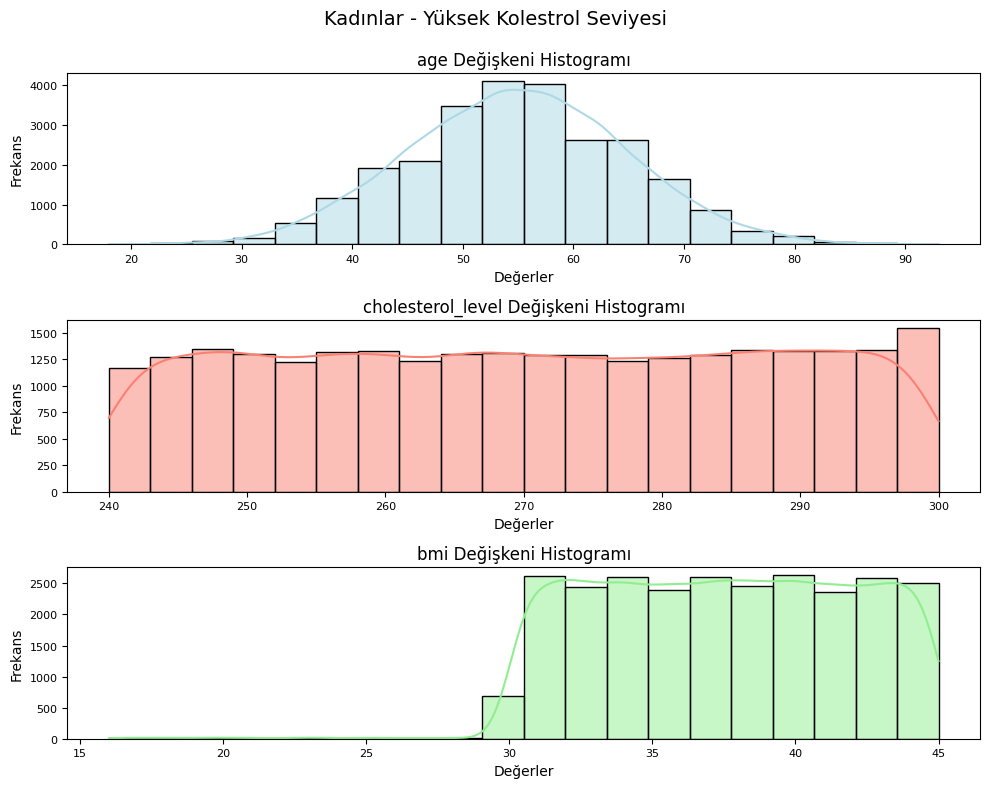
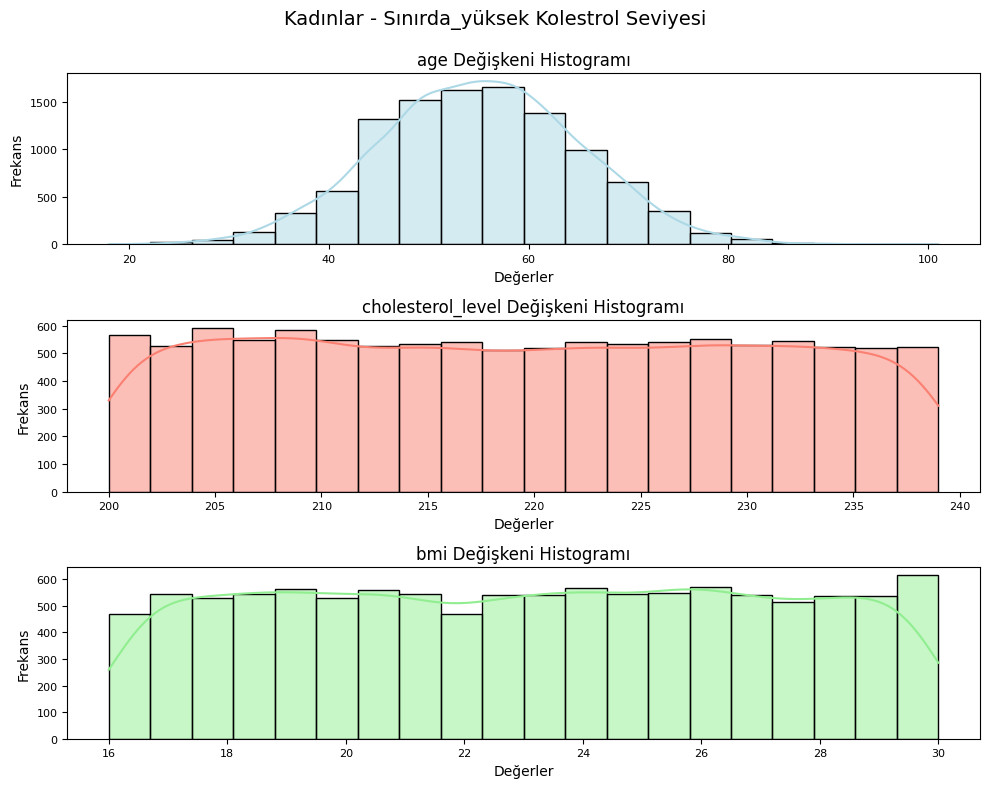
axes[i].tick\_params(axis='both', which='major', labelsize=8) # Eksen etiketlerinin boyutu

axes[i].tick\_params(axis='both', which='minor', labelsize=6) # Yardımcı eksen etiketlerinin boyutu

plt.tight\_layout()

plt.subplots\_adjust(top=0.9) # Başlık ile subplot arasındaki mesafe

plt.show()



**21.Veri setindeki yaş ve BMI değerlerini belirli aralıklara bölerek kategorik gruplar oluşturur ve her bir cinsiyet için ayrı ayrı Chi-Kare testi uygular**

from scipy.stats import chi2\_contingency

# Veri setini yükle

data = pd.read\_csv("lung-cancer.csv")

# Yaş ve BMI gruplarını tanımla

age\_bins = [0, 30, 40, 50, 60, 70, 100]

age\_labels = ['0-29', '30-39', '40-49', '50-59', '60-69', '70+']

bmi\_bins = [10, 18.5, 25, 30, 40, 60]

bmi\_labels = ['Zayıf', 'Sağlıklı', 'Kilolu', 'Şişman', 'Aşırı şişman']

# Yaş ve BMI gruplarını oluştur

data['age\_group'] = pd.cut(data['age'], bins=age\_bins, labels=age\_labels, right=False)

data['bmi\_group'] = pd.cut(data['bmi'], bins=bmi\_bins, labels=bmi\_labels, right=False)

# Kadınlar ve Erkekler için ayrı ayrı Chi-Kare testi yapalım

genders = ['Female', 'Male']

results = {}

for gender in genders:

gender\_data = data[data['gender'] == gender]

contingency\_table = pd.crosstab(gender\_data['age\_group'], gender\_data['bmi\_group'])

chi2, p, dof, expected = chi2\_contingency(contingency\_table)

results[gender] = {'chi2': chi2, 'p': p, 'contingency\_table': contingency\_table}

# Chi-Kare testi sonuçları

print(f"{gender} Chi-Kare İstatistiği: {chi2}")

print(f"{gender} P-değeri: {p}")

# Çapraz tabloyu görselleştir

plt.figure(figsize=(12, 8))

sns.heatmap(contingency\_table, annot=True, cmap='viridis', cbar=True, fmt='d')

plt.title(f'{gender} - Yaş Grupları ve BMI Grupları Arasındaki İlişki')

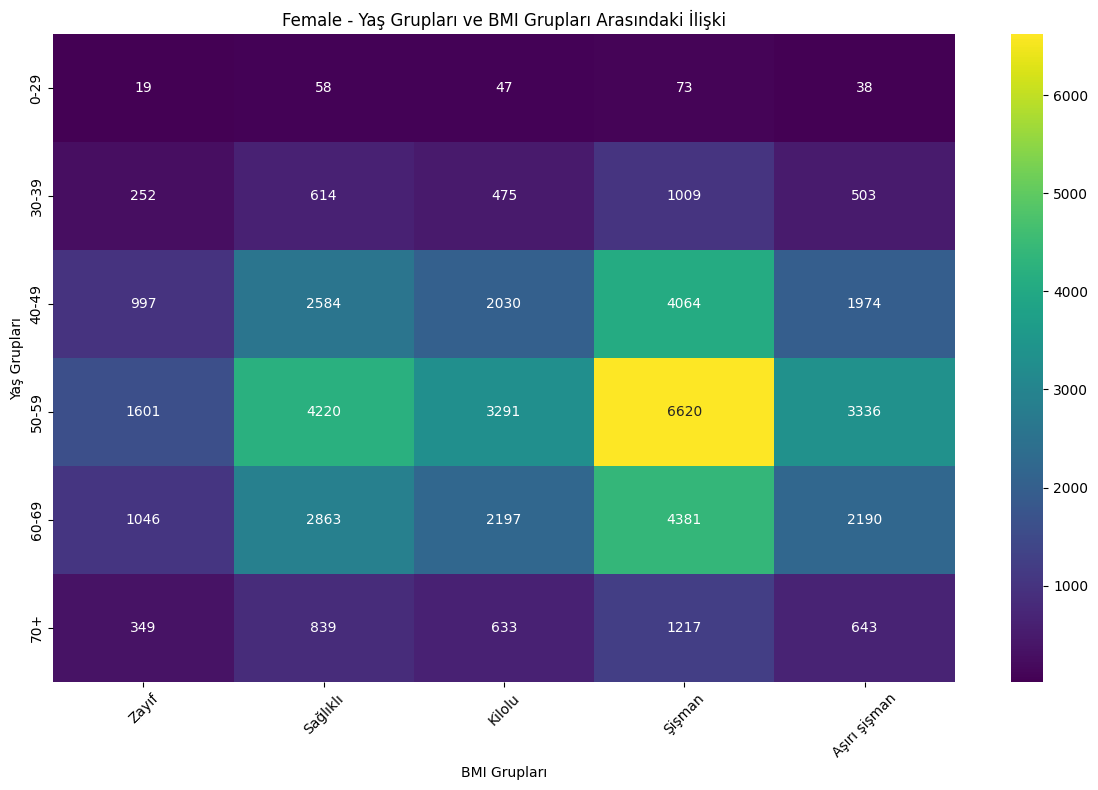
plt.xlabel('BMI Grupları')

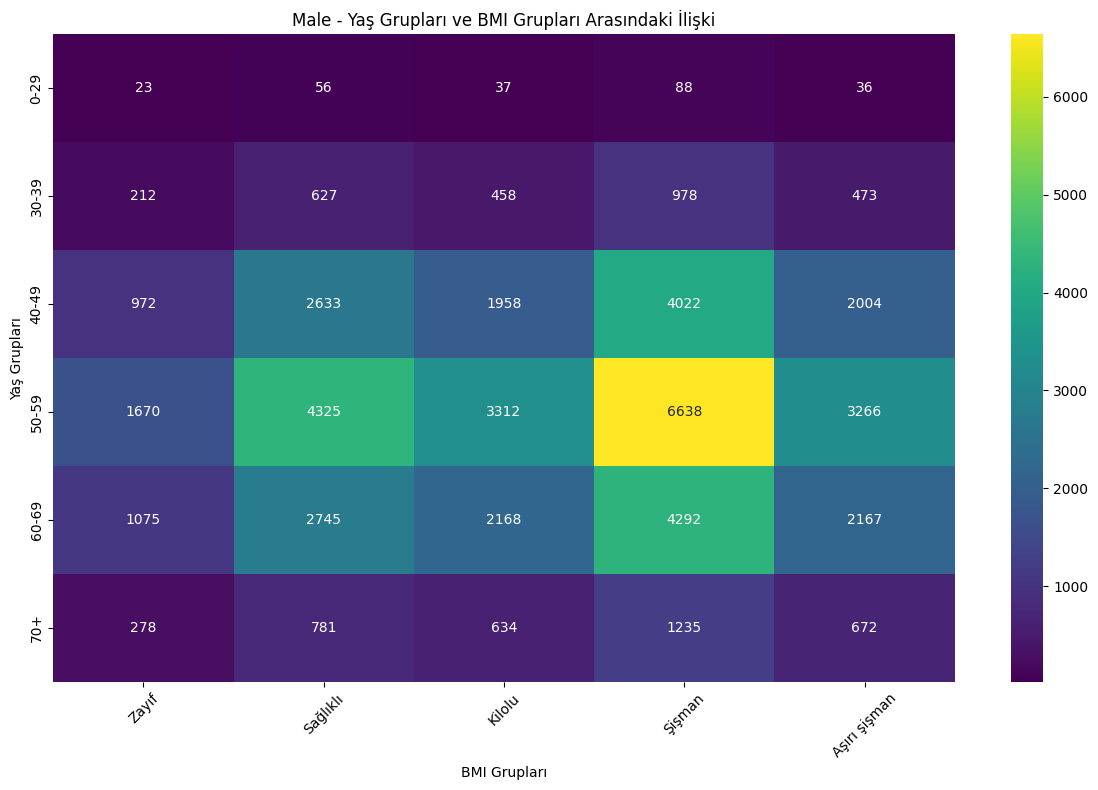
plt.ylabel('Yaş Grupları')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()





**21. Bu kod, hayatta kalanlar ve hayatta kalmayanlar arasında kolesterol seviyeleri açısından istatistiksel olarak anlamlı bir fark olup olmadığını test eder ve sonucu ekrana basar**

# t-testi

stat, p\_value = ttest\_ind(data[data["survived"]==1]["cholesterol\_level"], data[data["survived"]==0]["cholesterol\_level"])

if p\_value < 0.05:

print(f"İki grup arasında istatistiksel olarak anlamlı bir fark bulunmaktadır (p-value = {p\_value})")

else:

print(f"İki grup arasında istatistiksel olarak anlamlı bir fark bulunmamaktadır (p-value = {p\_value})")

İki grup arasında istatistiksel olarak anlamlı bir fark bulunmamaktadır (p-value = 0.8989779685697541)

**21.1. Kadınlar ve erkekler arasında "yüksek" kolestrol seviyesi grubundaki bireylerin kolesterol seviyeleri üzerinde varyans homojenliğini test eder.**

cholesterol\_level = {

'Normal': (0,200 ),

'Sınırda\_yüksek': (200, 240),

'Yüksek': (240, np.inf)

}

sag = kadinlar[kadinlar['cholesterol\_level'] >= 240]

olu = erkekler[erkekler['cholesterol\_level'] >= 240]

# Levene testini uygulama

levene\_test = stats.levene(sag['cholesterol\_level'], olu['cholesterol\_level'])

# Test sonucunu yazdırma

print("Levene Testi Sonucu:")

print(f"Statistic: {levene\_test.statistic}")

print(f"P-value: {levene\_test.pvalue}")

if levene\_test.pvalue > 0.05:

print("Yüksek - sag ve ölüler arasında cholesterol üzerinde varyans homojendir.")

else:

print("Yüksek - sag ve ölüler arasında cholesterol üzerinde varyans homojen değildir.")

Levene Testi Sonucu: Statistic: 0.9540287406868577 P-value: 0.32870048974924915 Yüksek - sag ve ölüler arasında cholesterol üzerinde varyans homojendir.

**22. Yaşlılar (50 yaş ve üzeri) ve gençler (25 yaş ve altı) grupları arasında belirli değişkenlerin (kolestrol seviyesi ve BMI) varyans homojenliğini test eder.**

# Yaşlılar ve gençler için veri filtreleme

yasli\_grup = data[data['age'] >= 50]

genc\_grup = data[data['age'] <= 25]

degiskenler = ["cholesterol\_level" ,"bmi"]

for degisken in degiskenler:

# Levene testi uygulama

levene\_test = levene(yasli\_grup[degisken], genc\_grup[degisken])

# Test sonucunu yazdırma

if levene\_test.pvalue > 0.05:

print(f"{degisken}: Yaşlılar ve gençler arasında varyans homojendir (p-value = {levene\_test.pvalue})")

else:

print(f"{degisken}: Yaşlılar ve gençler arasında varyans homojen değildir (p-value = {levene\_test.pvalue})")

cholesterol\_level: Yaşlılar ve gençler arasında varyans homojendir (p-value = 0.24392580875739472) bmi: Yaşlılar ve gençler arasında varyans homojendir (p-value = 0.2029370750847536)

**23.'cholesterol\_level' ve 'bmi' değişkenlerinin yaşa göre değişimini gösterir.**

# Kadın ve erkek verilerini ayırma

data\_female = data[data['gender'] == 'Female']

data\_male = data[data['gender'] == 'Male']

# Plot ayarları

lifts = ["cholesterol\_level", "bmi"]

titles = ['Cholesterol Seviyesi', 'BMI']

colors = ['orange', 'green']

genders = [('Kadın', data\_female, 'orange'), ('Erkek', data\_male, 'green')]

# Subplotlar oluşturma

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# For döngüsü ile plotları oluşturma

for i, (lift, title) in enumerate(zip(lifts, titles)):

for j, (gender, dataset, color) in enumerate(genders):

sns.lineplot(x='age', y=lift, data=dataset, color=color, ax=axes[i,j])

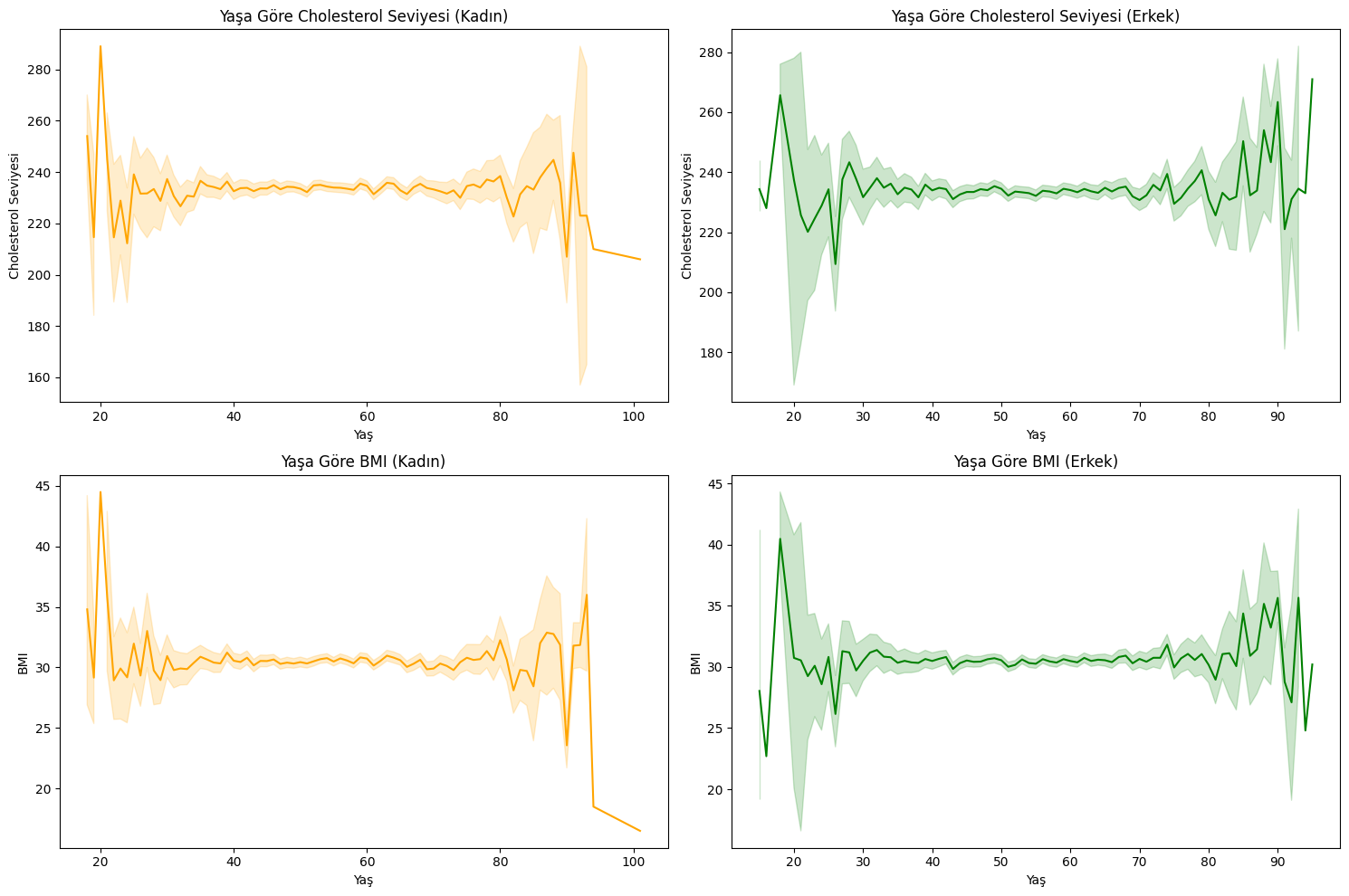
axes[i, j].set\_title(f"Yaşa Göre {title} ({gender})")

axes[i, j].set\_xlabel("Yaş")

axes[i, j].set\_ylabel(title)

plt.tight\_layout()

plt.show()



**24. Kadınlar ve erkekler arasında belirli değişkenlerin ("cholesterol\_level" ve "bmi") hayatta kalma durumuyla ilişkisini görselleştirir.**

# Veri setini yükle

data = pd.read\_csv("lung-cancer.csv")

# Cinsiyete göre ayrılmış veri kümeleri

female\_data = data[data['gender'] == 'Female']

male\_data = data[data['gender'] == 'Male']

# Renk paletini ayarla (mor ve lacivert tonları)

palette = sns.color\_palette(['indigo', 'orange'])

# Kutu grafiği (boxplot) ile görselleştirme

plt.figure(figsize=(14, 8))

# Cholesterol Level vs. Survival by Gender

plt.subplot(1, 2, 1)

sns.boxplot(x='survived', y='cholesterol\_level', hue='gender', data=data, orient='v', palette=palette)

plt.title('Cholesterol Level vs. Survival by Gender')

plt.xlabel('Survived')

plt.ylabel('Cholesterol Level')

plt.legend(title='Gender')

plt.xticks([0, 1], ['No', 'Yes'])

# BMI vs. Survival by Gender

plt.subplot(1, 2, 2)

sns.boxplot(x='survived', y='bmi', hue='gender', data=data, orient='v', palette=palette)

plt.title('BMI vs. Survival by Gender')

plt.xlabel('Survived')

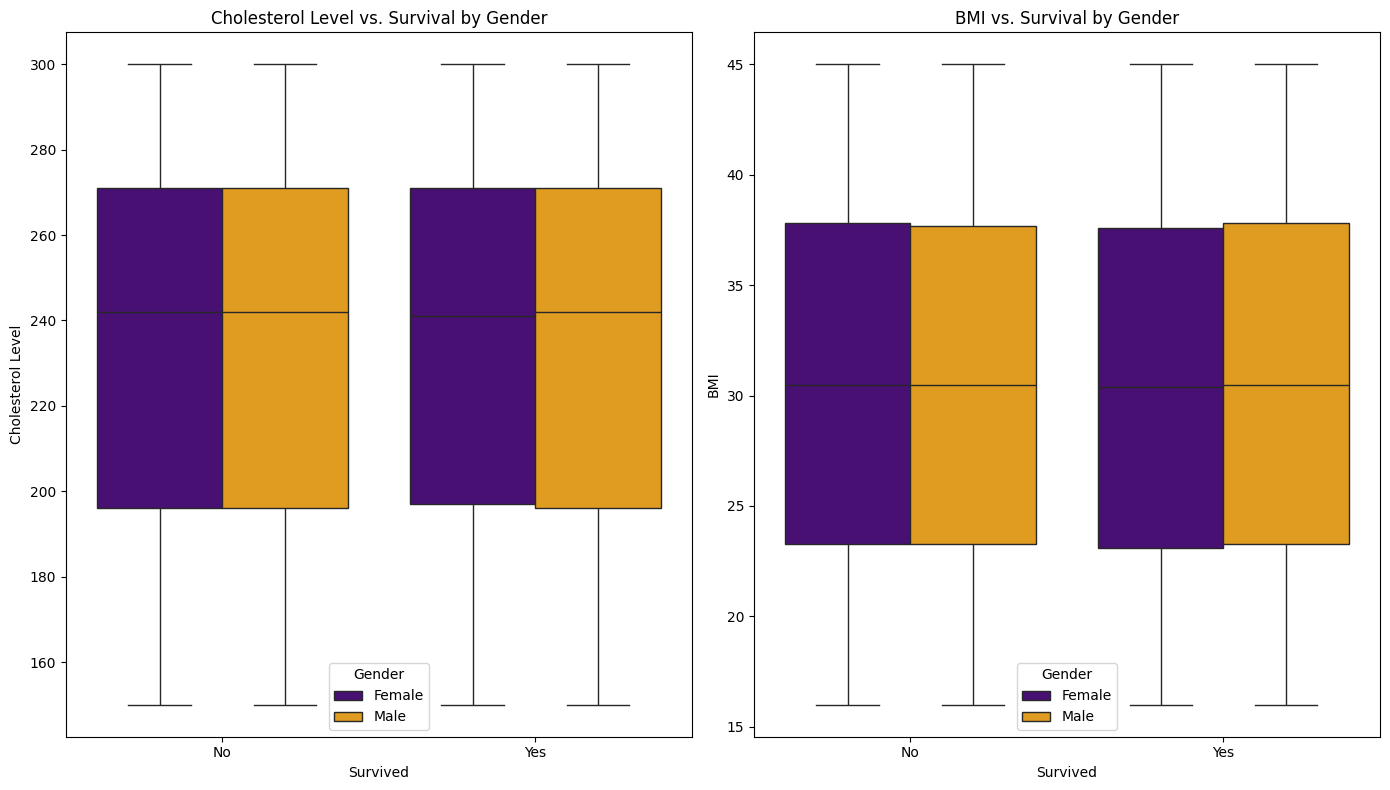
plt.ylabel('BMI')

plt.legend(title='Gender')

plt.xticks([0, 1], ['No', 'Yes'])

plt.tight\_layout()

plt.show()



**25. Veri kümesindeki ülkelerin kansere bağlı ölüm oranlarını hesaplar ve görselleştirir.**

# Verileri oku

file\_path = "lung-cancer.csv"

df = pd.read\_csv(file\_path)

# Ülkelere göre ölüm oranlarını hesapla

death\_rates = df.groupby('country')['survived'].mean().sort\_values()

# Ölüm oranlarını tabloda göster

print("Ülkelere Göre Kansere Bağlı Ölüm Oranları:")

print(death\_rates)

# Grafik oluştur

plt.figure(figsize=(10, 6))

death\_rates.plot(kind='bar', color='skyblue')

plt.title('Ülkelere Göre Kansere Bağlı Ölüm Oranları')

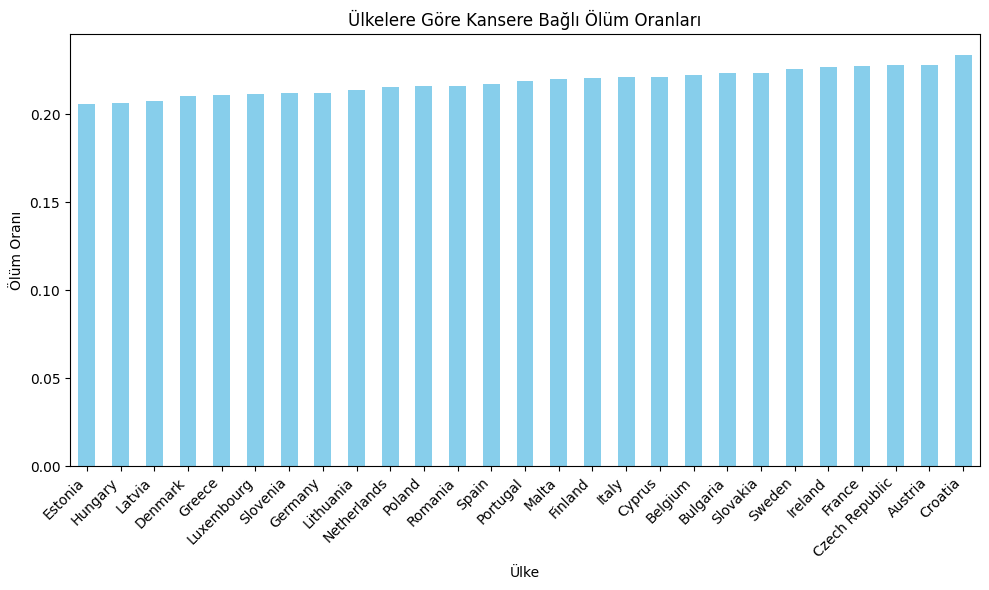
plt.xlabel('Ülke')

plt.ylabel('Ölüm Oranı')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

**26. Veri kümesindeki ülkelerin kanseri atlatma oranlarını hesaplar ve görselleştirir**

# Verileri oku

file\_path = "lung-cancer.csv"

df = pd.read\_csv(file\_path)

# Ülkelere göre kanseri atlatanların oranlarını hesapla

survival\_rates = df.groupby('country')['survived'].mean().sort\_values()

print("Ülkelere göre kanseri atlatanların Oranı:")

print(survival\_rates)

# Grafik oluştur

plt.figure(figsize=(12, 6))

survival\_rates.plot(kind='bar', color='lightgreen')

plt.title('Ülkelere Göre Kanseri Atlatma Oranları')

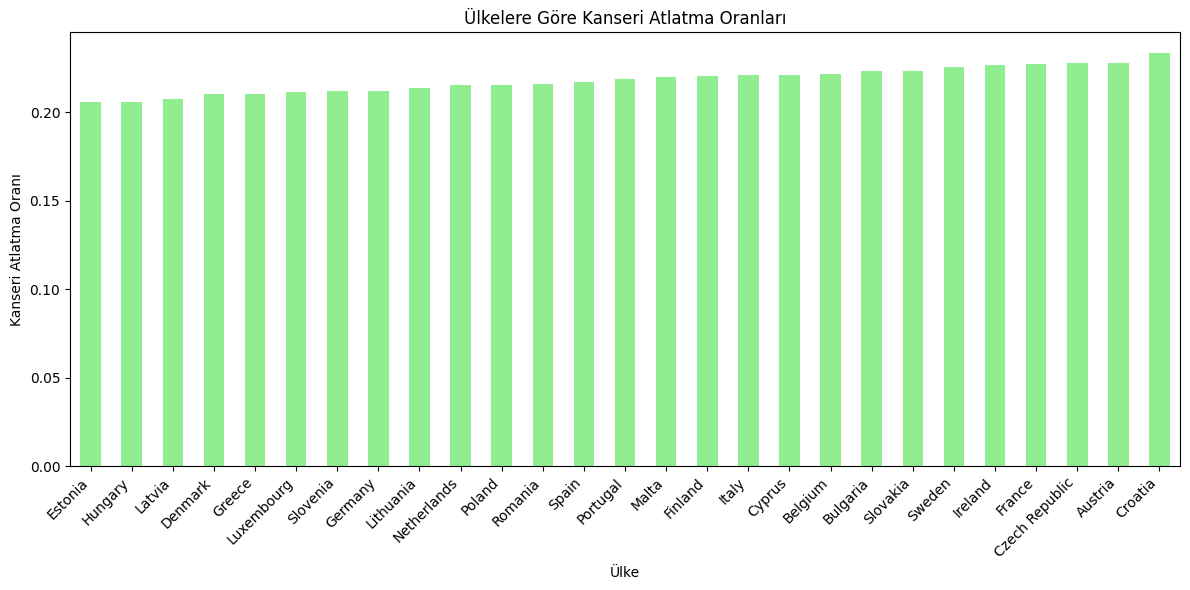
plt.xlabel('Ülke')

plt.ylabel('Kanseri Atlatma Oranı')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

**27. Ülkelere Göre Kansere Bağlı Ölenlerin Sayısı**

# Verileri oku

file\_path = "lung-cancer.csv"

df = pd.read\_csv(file\_path)

# Ülkelere göre ölüm sayılarını hesapla

death\_counts = df[df['survived'] == 0].groupby('country').size().sort\_values()

# Ölüm oranlarını tabloda göster

print("Ülkelere Göre Kansere Bağlı Ölenlerin Sayısı:")

print(death\_counts)

# Grafik oluştur

plt.figure(figsize=(12, 6))

death\_counts.plot(kind='bar', color='salmon')

plt.title('Ülkelere Göre Kansere Bağlı Ölüm Sayıları')

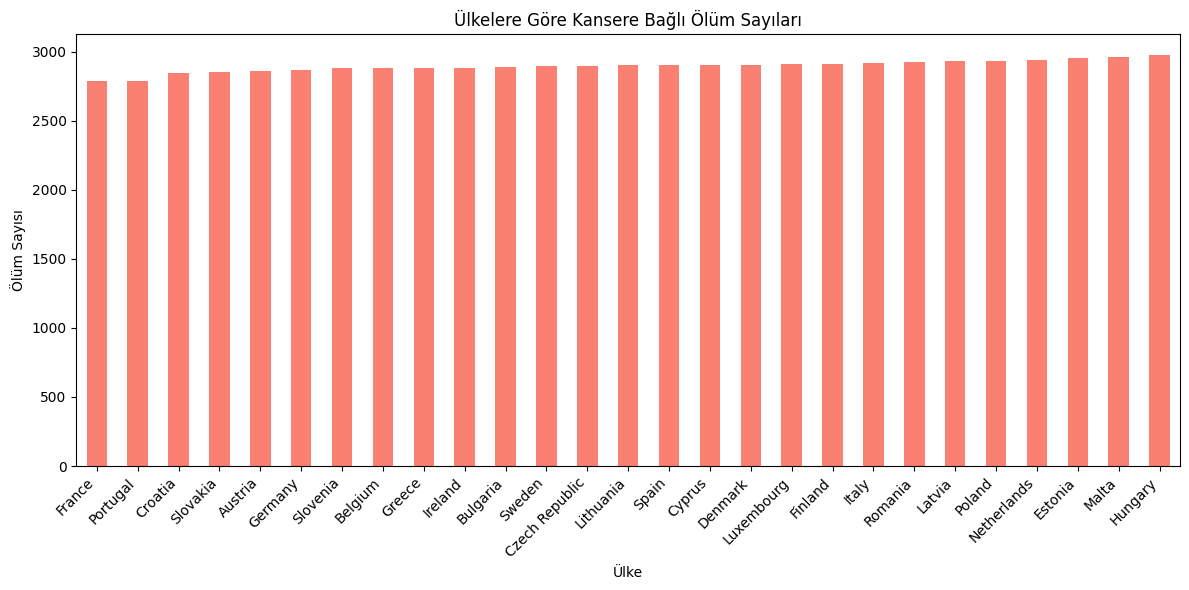
plt.xlabel('Ülke')

plt.ylabel('Ölüm Sayısı')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

**28. Ülkelere Göre Kanseri Atlatanların Sayısı**

import matplotlib.pyplot as plt

import pandas as pd

# Verileri oku

file\_path = "lung-cancer.csv"

df = pd.read\_csv(file\_path)

# Ülkelere göre kanseri atlatanların sayılarını hesapla

survived\_counts = df[df['survived'] == 1].groupby('country').size().sort\_values()

print("Ülkelere Göre Kanseri Atlatanların Sayısı:")

print(survived\_counts)

# Grafik oluştur

plt.figure(figsize=(12, 6))

survived\_counts.plot(kind='bar', color='lightblue')

plt.title('Ülkelere Göre Kanseri Atlatanların Sayısı')

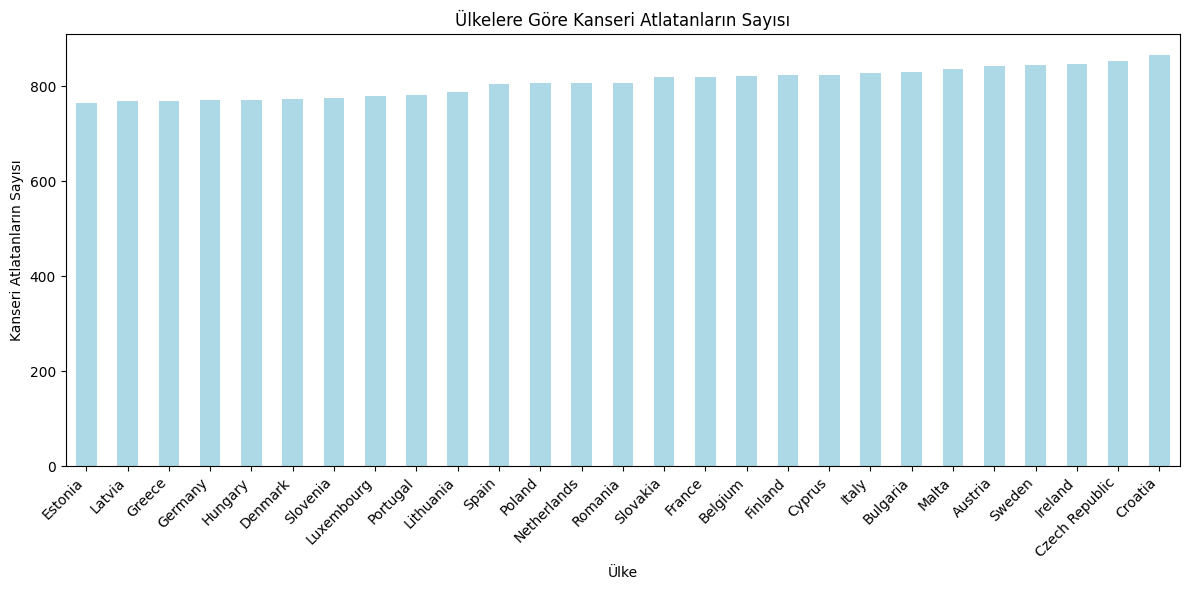
plt.xlabel('Ülke')

plt.ylabel('Kanseri Atlatanların Sayısı')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

**29. Bu kod iki farklı boxplot grafiği oluşturur:**

**1.Kanser Evrelerine Göre Yaş Dağılımı**

**2.Sigara İçme Durumuna Göre Kolesterol Seviyeleri**

**Her iki grafik de gruplar arasındaki dağılımları görselleştirerek karşılaştırmayı sağlar.**

# Kanser evrelerine göre yaş dağılımını karşılaştırma

plt.figure(figsize=(10, 6))

sns.boxplot(x='cancer\_stage', y='age', data=data, palette='viridis', hue='cancer\_stage', dodge=False)

plt.title('Kanser Evrelerine Göre Yaş Dağılımı')

plt.xlabel('Kanser Evresi')

plt.ylabel('Yaş')

plt.xticks(rotation=45)

plt.grid(axis='y')

plt.legend(title='Kanser Evresi')

plt.show()

# Sigara içme durumuna göre kolesterol seviyelerini karşılaştırma

plt.figure(figsize=(10, 6))

sns.boxplot(x='smoking\_status', y='cholesterol\_level', data=data, palette='pastel', hue='smoking\_status', dodge=False)

plt.title('Sigara İçme Durumuna Göre Kolesterol Seviyeleri')

plt.xlabel('Sigara İçme Durumu')

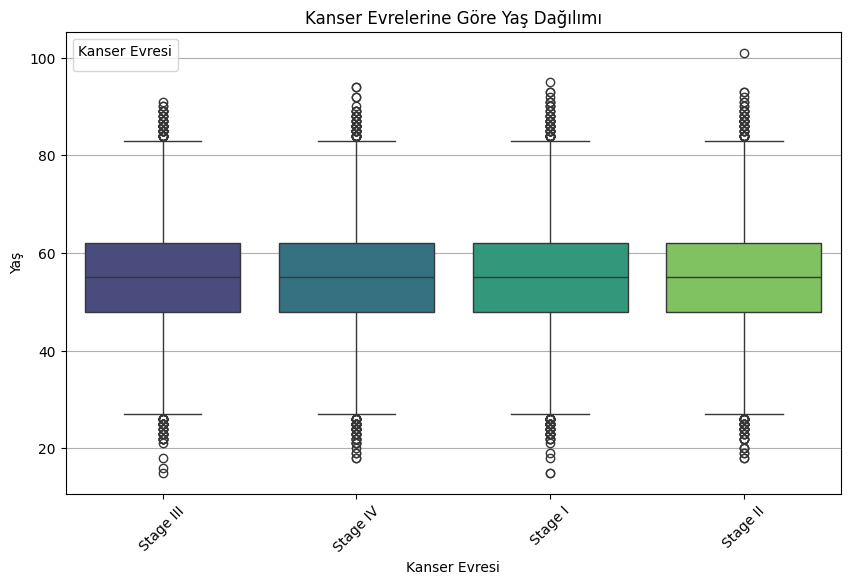
plt.ylabel('Kolesterol Seviyesi')

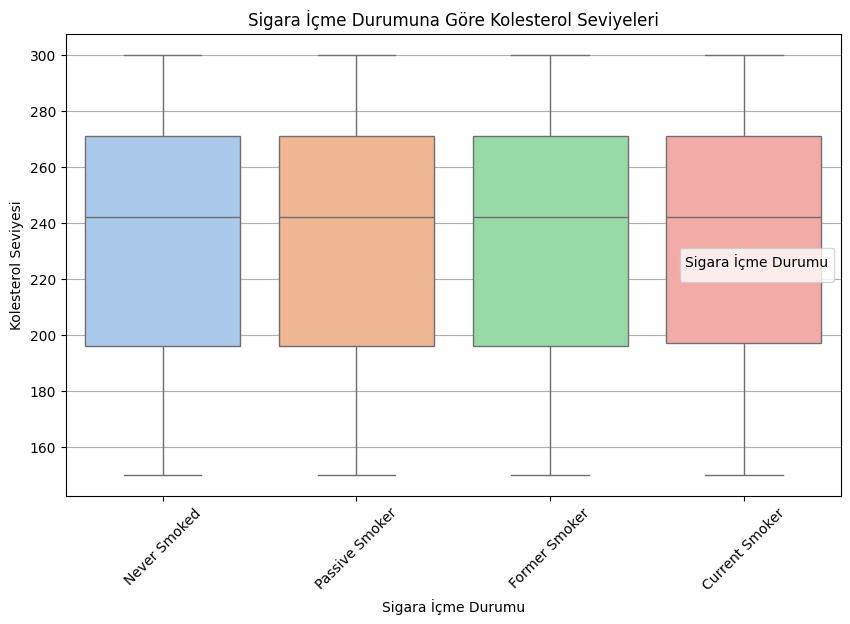
plt.xticks(rotation=45)

plt.grid(axis='y')

plt.legend(title='Sigara İçme Durumu')

plt.show()





**30. Kolesterol seviyelerinin normal dağılıma ne kadar uyduğunu görselleştirmek için bir Q-Q (Quantile-Quantile) plot oluşturur.**

import scipy.stats as stats

import matplotlib.pyplot as plt

# CSV dosyasını yükleyin

data = pd.read\_csv("lung-cancer.csv")

# Örnek bir sütunu seçin (örneğin, kolesterol seviyeleri)

sample\_data = data['cholesterol\_level'].dropna() # NaN değerleri kaldır

# QQ plot oluşturun

plt.figure(figsize=(8, 6))

stats.probplot(sample\_data, dist="norm", plot=plt)

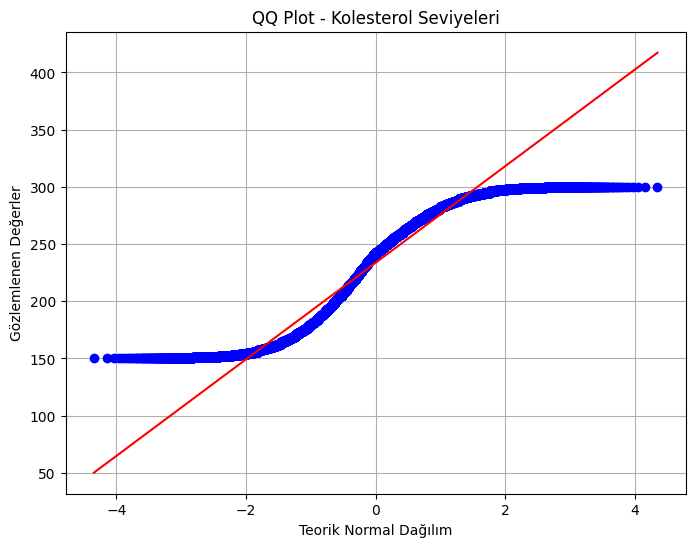
plt.title('QQ Plot - Kolesterol Seviyeleri')

plt.xlabel('Teorik Normal Dağılım')

plt.ylabel('Gözlemlenen Değerler')

plt.grid(True)

plt.show()



**31. Akciğer kanserinin ülke bazındaki vakalarının coğrafi dağılımını folium kütüphanesi ile görselleştirmektedir**

import pandas as pd

import folium

from geopy.geocoders import Nominatim

from collections import Counter

# CSV dosyasını okuyalım

df = pd.read\_csv('lung-cancer.csv')

# Ülke adlarını koordinatlara dönüştürmek için geopy kullanıyoruz

geolocator = Nominatim(user\_agent="lung-cancer-map")

# Her ülke için koordinatları depolamak için bir sözlük oluşturuyoruz

country\_coords = {}

# Ülkelerin koordinatlarını alıyoruz

for country in df['country'].unique():

location = geolocator.geocode(country)

if location:

country\_coords[country] = (location.latitude, location.longitude)

else:

print(f"Koordinatlar bulunamadı: {country}")

# Ülke bazında vaka sayılarını ve sigara içme durumlarını hesaplayalım

country\_counts = Counter(df['country'])

# Folium haritasını oluşturuyoruz

map = folium.Map(location=[20, 0], zoom\_start=2)

# En yüksek vaka sayısını bulalım

max\_cases = max(country\_counts.values())

# Her ülke için işaretçi ekliyoruz, vaka sayısına göre renklendirme yapacağız

for country, count in country\_counts.items():

if country in country\_coords:

lat, lon = country\_coords[country]

# Vaka sayısına göre daire işaretçileri ekliyoruz

radius = 10 + 10 \* (count / max\_cases)

color = 'red' if count == max\_cases else 'blue'

popup\_text = f"{country}: {count} cases"

folium.CircleMarker(location=[lat, lon], radius=radius, color=color, fill=True, fill\_color=color, popup=popup\_text).add\_to(map)

# Sigara içme durumuna göre ikon işaretçileri ekliyoruz

smoking\_status = df[df['country'] == country]['smoking\_status'].values[0]

if smoking\_status == 'Current Smoker':

icon = folium.Icon(color='orange', icon='cloud')

elif smoking\_status == 'Former Smoker':

icon = folium.Icon(color='green', icon='cloud')

elif smoking\_status == 'Never Smoked':

icon = folium.Icon(color='blue', icon='cloud')

else:

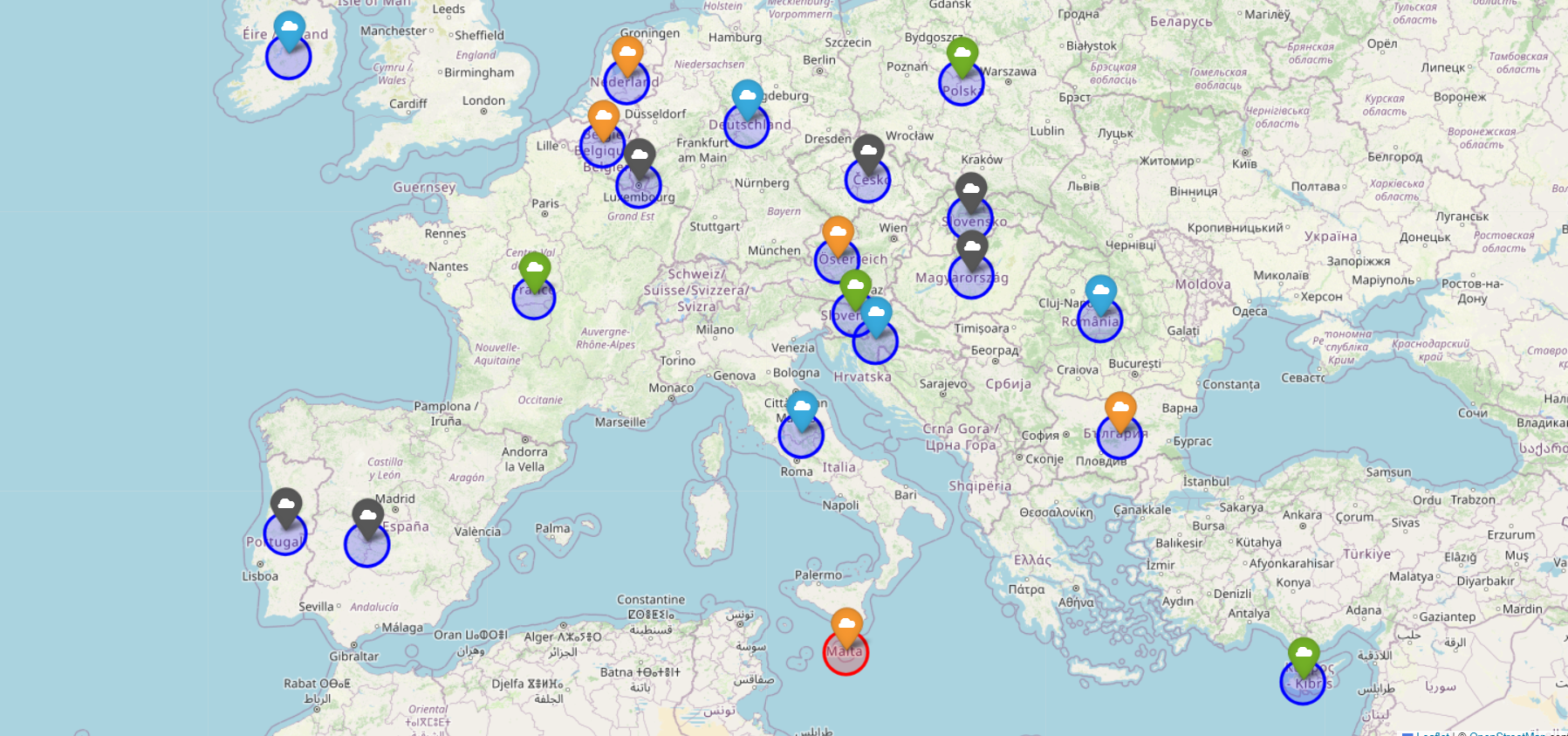
icon = folium.Icon(color='gray', icon='cloud')

folium.Marker(location=[lat+0.1, lon], icon=icon, popup=f"{country}: {smoking\_status} ({count} cases)").add\_to(map)

# Haritayı HTML dosyası olarak kaydediyoruz

map.save('lung\_cancer\_map.html')

print("Harita 'lung\_cancer\_map.html' dosyasına kaydedildi.")



**32. Bu test, iki farklı grup arasında sayısal bir değişkenin (burada BMI) ortalamaları arasında istatistiksel olarak anlamlı bir fark olup olmadığını belirlemek için kullanılır.**

from scipy.stats import ttest\_ind

# Veri setinizdeki sigara içenlerin ve içmeyenlerin BMI değerlerini gruplara ayırın

smokers\_bmi = data[data['smoking\_status'] == 'Current Smoker']['bmi']

non\_smokers\_bmi = data[data['smoking\_status'] == 'Never Smoked']['bmi']

# Bağımsız iki örnek t testi uygulayın

t\_statistic, p\_value = ttest\_ind(smokers\_bmi, non\_smokers\_bmi, equal\_var=False)

print("T statistic:", t\_statistic)

print("P value:", p\_value)

T statistic: -0.3450519684136582

P value: 0.7300567520917935

**33. Akciğer kanser veri setinde farklı kanser evrelerine (Stage I, Stage II, Stage III) sahip gruplar arasında BMI değerlerinin ortalamaları arasında istatistiksel bir fark olup olmadığını test ediyor.**

from scipy.stats import f\_oneway

# Örneğin, kanser evrelerine göre BMI değerlerini gruplara ayırın

stage1\_bmi = data[data['cancer\_stage'] == 'Stage I']['bmi']

stage2\_bmi = data[data['cancer\_stage'] == 'Stage II']['bmi']

stage3\_bmi = data[data['cancer\_stage'] == 'Stage III']['bmi']

# Anova testi uygulayın

f\_statistic, p\_value = f\_oneway(stage1\_bmi, stage2\_bmi, stage3\_bmi)

print("F statistic:", f\_statistic)

print("P value:", p\_value)

F statistic: 1.3291157657200303

P value: 0.2647174631004751

**34. Kanser evrelerine göre BMI değerlerinin farklı gruplara ayrılması ve bu gruplar arasında ANOVA (Analysis of Variance) testi yapılması için kullanılıyor.**

from scipy.stats import f\_oneway

# Örneğin, kanser evrelerine göre BMI değerlerini gruplara ayırın

stage1\_bmi = data[data['cancer\_stage'] == 'Stage I']['bmi']

stage2\_bmi = data[data['cancer\_stage'] == 'Stage II']['bmi']

stage3\_bmi = data[data['cancer\_stage'] == 'Stage III']['bmi']

# Anova testi uygulayın

f\_statistic, p\_value = f\_oneway(stage1\_bmi, stage2\_bmi, stage3\_bmi)

print("F statistic:", f\_statistic)

print("P value:", p\_value)

F statistic: 1.3291157657200303

P value: 0.2647174631004751

**34. Kaplan-Meier analizi kullanarak farklı tedavi tiplerine göre hayatta kalma sürelerini görselleştirmek için tasarlanmıştır**

from lifelines import KaplanMeierFitter

# Veri setini yükleyin (örneğin)

data = pd.read\_csv("lung-cancer.csv")

# Tarih sütunlarını datetime tipine dönüştürme

data['end\_treatment\_date'] = pd.to\_datetime(data['end\_treatment\_date'])

# Örneğin, farklı tedavi tipleri için Kaplan-Meier analizi yapalım

kmf = KaplanMeierFitter()

# Tedavi tiplerine göre hayatta kalma sürelerini hesaplayalım

treatment\_types = data['treatment\_type'].unique()

for treatment in treatment\_types:

treatment\_data = data[data['treatment\_type'] == treatment]

kmf.fit(treatment\_data['end\_treatment\_date'], event\_observed=treatment\_data['survived'], label=treatment)

kmf.plot\_survival\_function()

# Grafiği gösterelim

plt.title('Kaplan-Meier Analizi - Farklı Tedavi Tipleri')

plt.xlabel('Zaman (End Treatment Date)')

plt.ylabel('Hayatta Kalma Olasılığı')

plt.tight\_layout()

plt.show()

