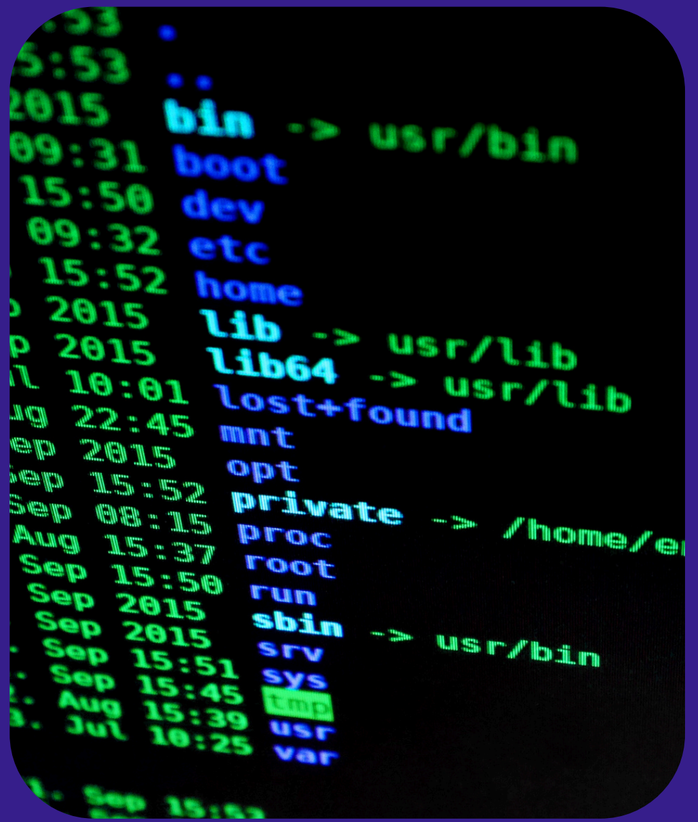
WEB ÜZERİNDEN BİLGİYE ERİŞİM ÖDEV SUNUMU

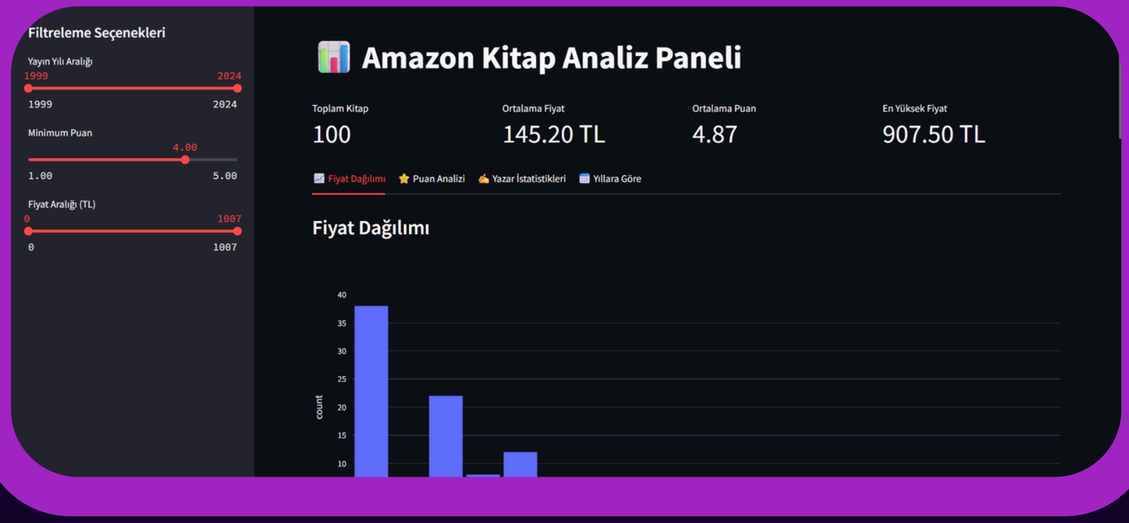
HIDIR SAMET YALÇINKAYA
220509015

Veri Mühendisi



Amazon Kitap Verileriyle Web Dashboard Uygulaması

Bu proje, Amazon’dan elde edilen verileri sadece toplamakla kalmayıp, onları işleyerek analiz eden ve grafiksel olarak sunan etkileşimli bir dashboard ile veriye anlam kazandırmayı amaçlamaktadır.



Kullanılan Teknolojiler

1. Python
2. Streamlit
3. Selenium
4. Plotly

Sistem Mimarisi

1. Veri Toplama: Amazon'daki kitap bilgilerini otomatik olarak tarayıp kaydeden bir sistem kurduk.
2. Veri Düzenleme: İlk topladığımız verileri anlaşılır ve kullanışlı hale getirmek için JSON formatına çevirdik.
3. Veri Analizi: Düzenlenmiş veriler üzerinde anlamlı sonuçlar çıkarmak için Python ile çeşitli hesaplamalar yaptık.
4. Sunum: Elde ettiğimiz verileri kullanıcı dostu grafikler ve tablolarla görselleştirdik.

Dashboard

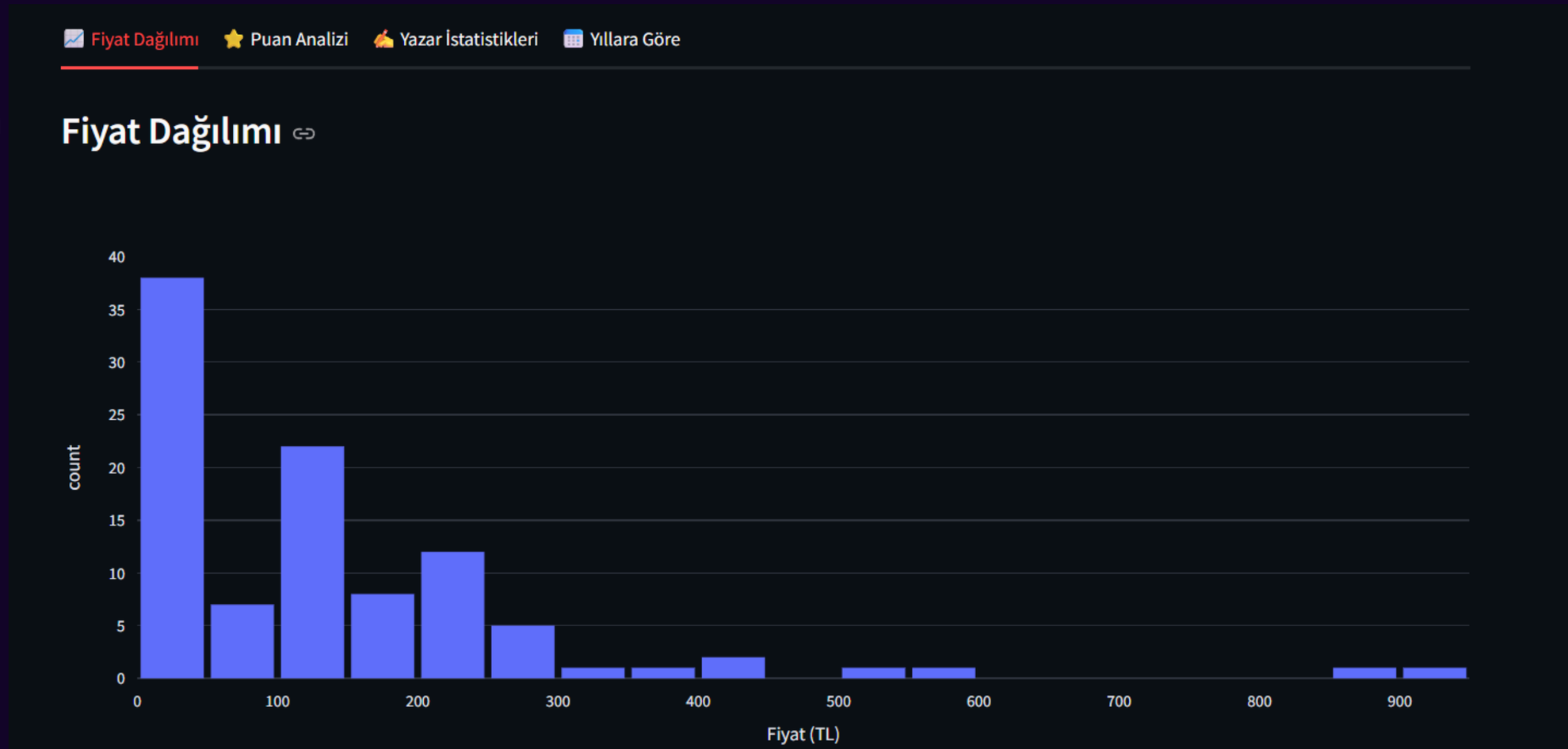
Web arayüzünün en üst bölümünde, veri üzerinde yapılan işlemler sonucunda toplam kitap sayısı, ortalama fiyat, ortalama puan ve en yüksek fiyat gibi genel bilgiler kullanıcıya sunulmaktadır. Bu veriler, data.json dosyasından çekilerek dinamik olarak hesaplanmakta ve görselleştirilmektedir.

Dashboard

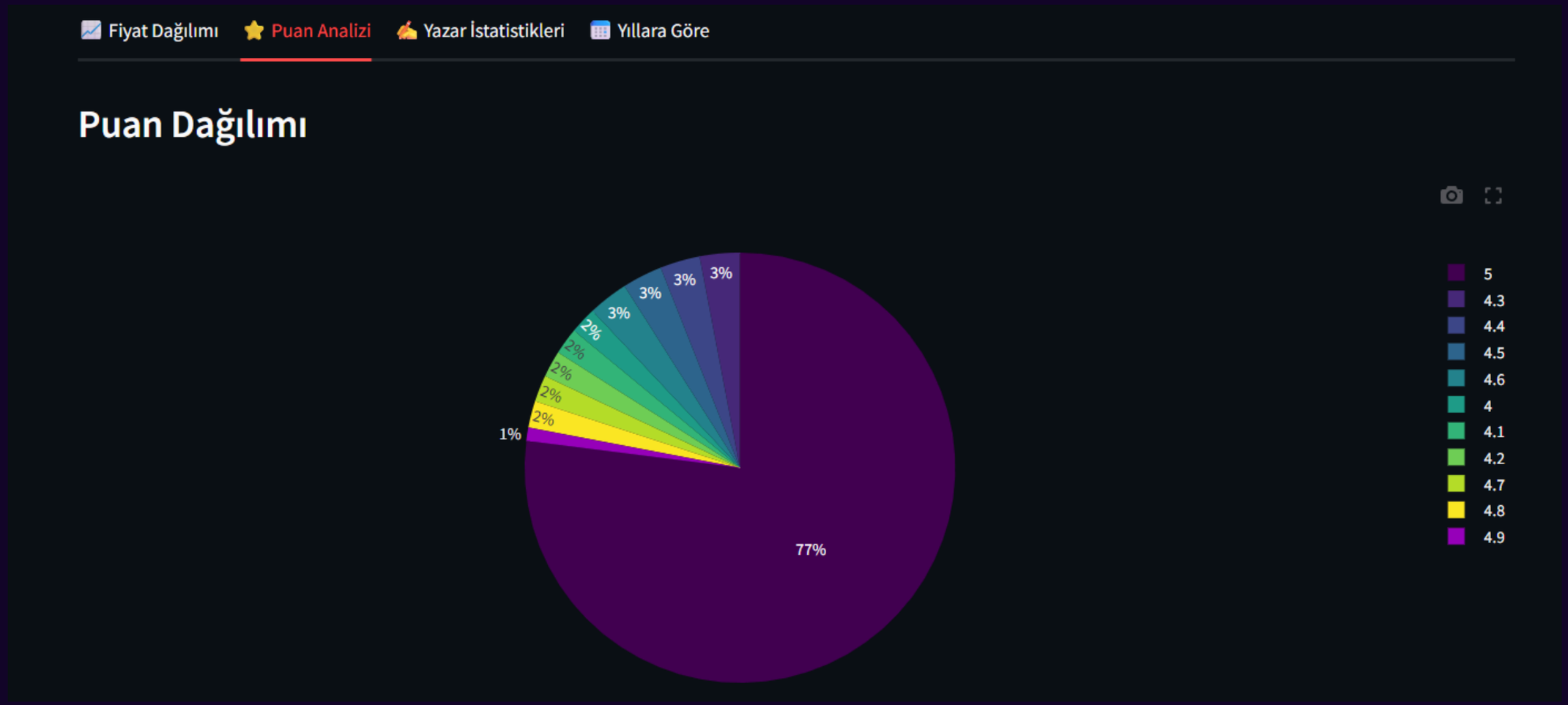
Sayfanın sol tarafında filtreleme seçenekleri yer almaktadır. Kullanıcılar, kitapların yayın yıllarını, puanlarını ve fiyat aralıklarını dinamik olarak seçerek, istedikleri kriterlere uygun verileri kolayca filtreleyebilir ve sonuçları anında görebilirler.

Sayfanın orta kısmında dinamik bir panel yer almaktadır. Bu panel üzerinden kullanıcılar; fiyat dağılımı, puan analizi, yazar istatistikleri ve yıllara göre kitap dağılımı gibi farklı analiz türlerini seçerek ilgili grafikleri görüntüleyebilirler. Bu özellik sayesinde veriler daha anlaşılır hale gelir ve kullanıcıların yorum yapması kolaylaşır. Bu yapıyı oluşturarak, veriyi etkili şekilde işleyebilen vizyoner bir proje ortaya koymuş oldum.

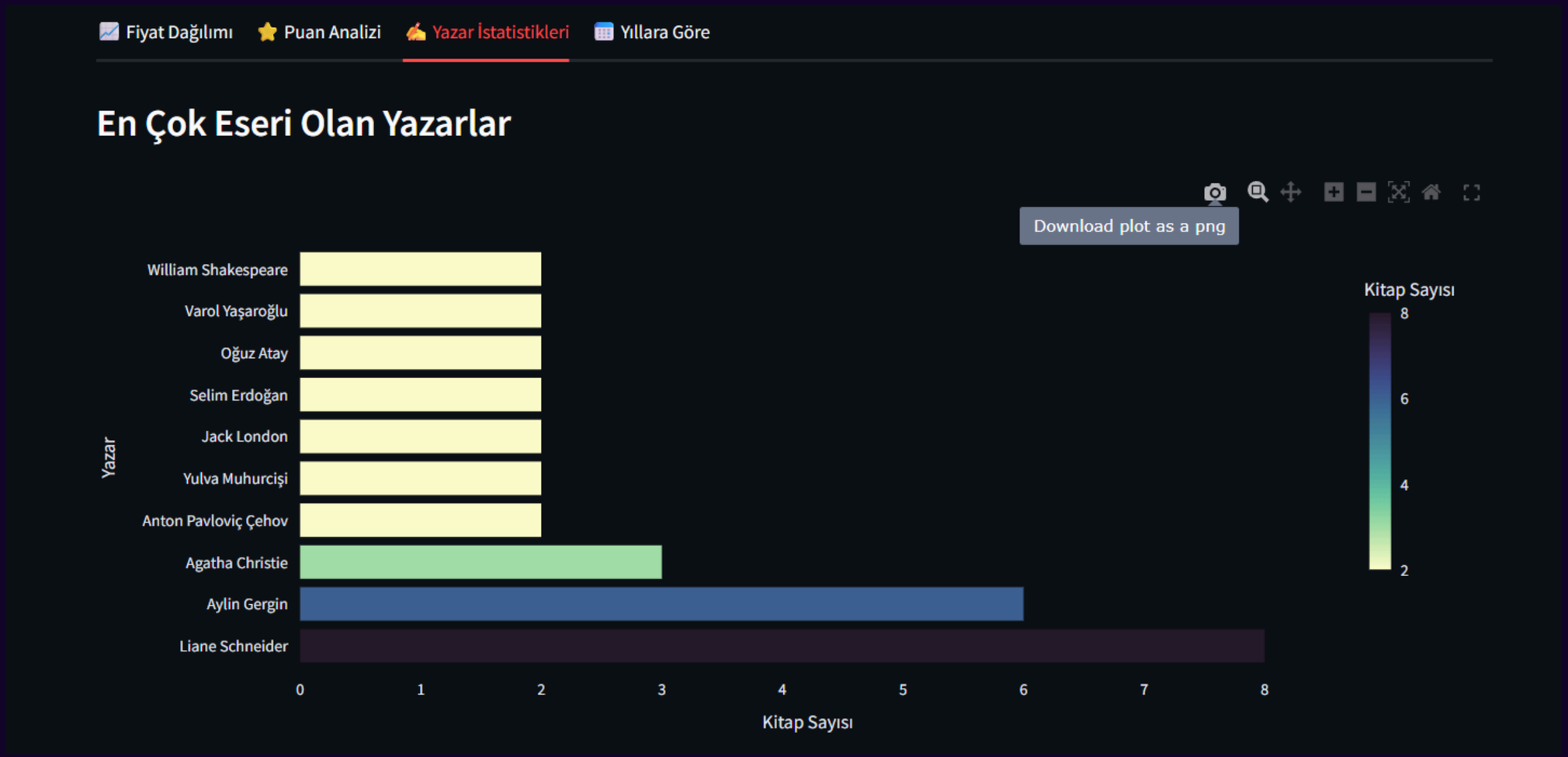
Sayfanın bu bölümünde, kitapların fiyat dağılımını gösteren bir sütun grafiği yer almaktadır. Bu grafik sayesinde fiyatların genel dağılımı görsel olarak analiz edilebilir.



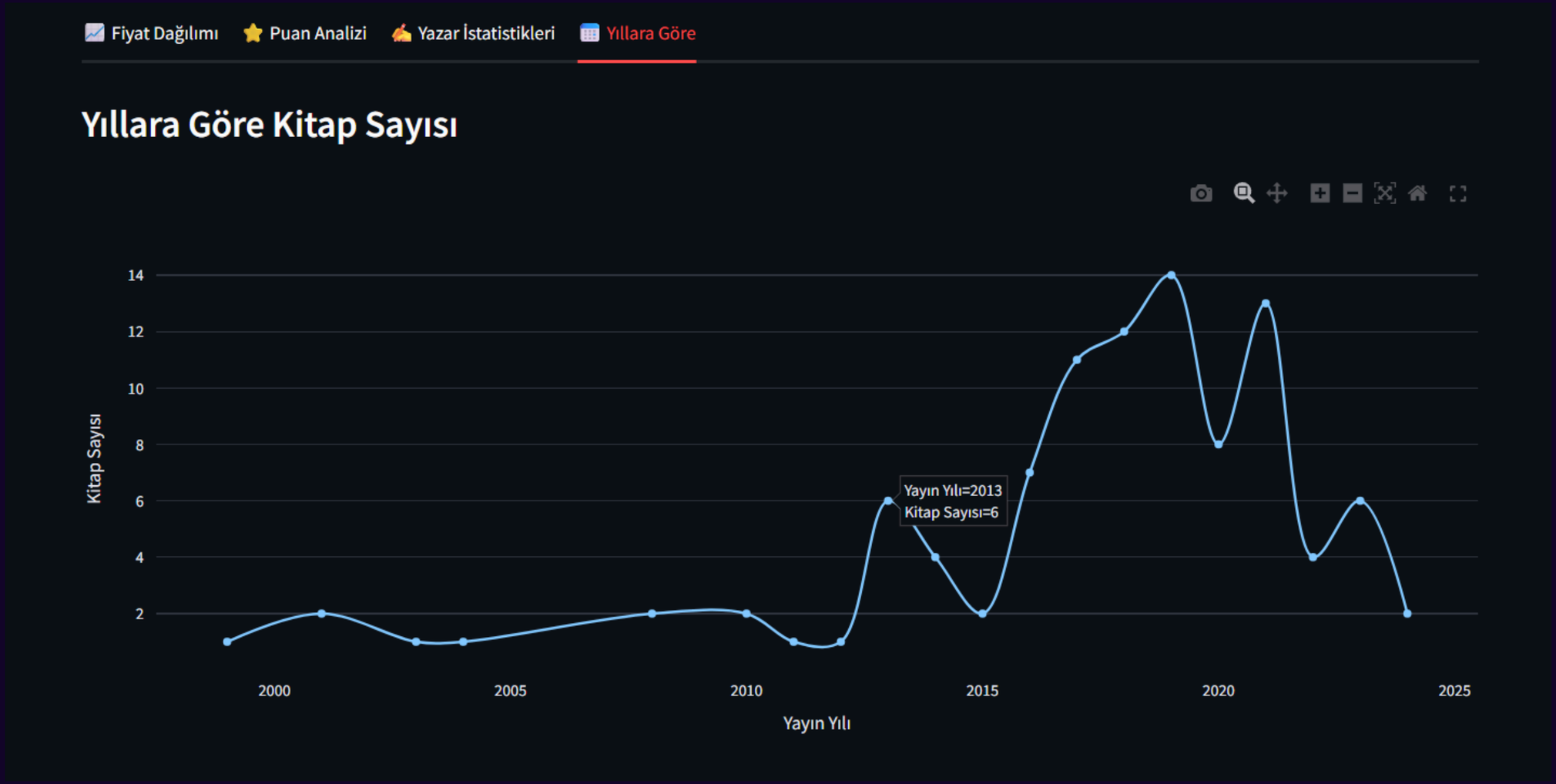
Sayfanın puan analizi kısmında, kitapların aldığı puanlara göre dağılımı gösteren bir pasta grafiği yer almaktadır. Bu grafik, veri setinde 4 ile 5 puan arasındaki kitaplar seçildiği için, yalnızca bu aralıktaki puanların dağılımını yansıtmaktadır.



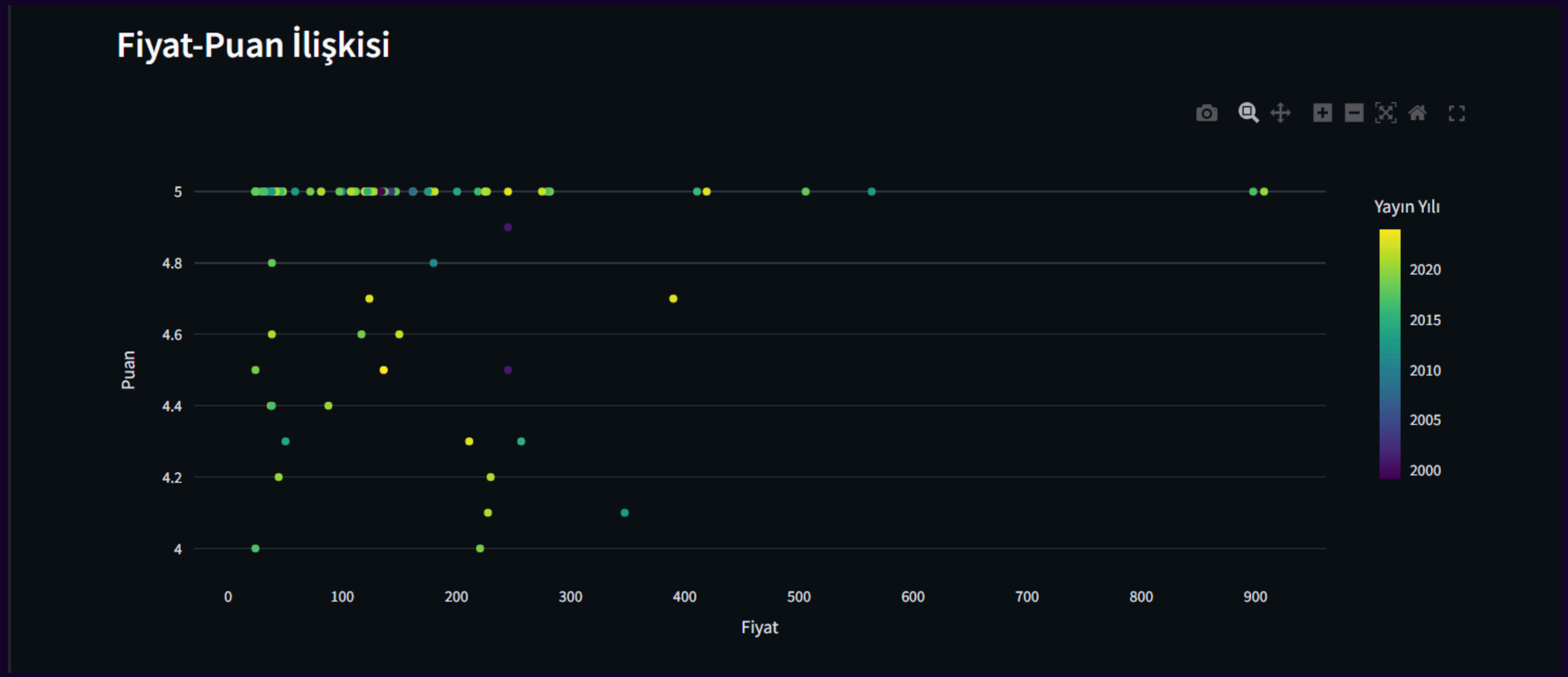
Sayfanın orta kısmında yer alan üçüncü dinamik bölümde, yazar istatistiklerini grafiksel olarak görebilmekteyiz. Bu bölümde, yazarların yazdığı kitap sayısına göre oluşturulmuş bir çubuk grafiği yer almakta ve böylece en üretken yazarları kolaylıkla tespit edebilmekteyiz.



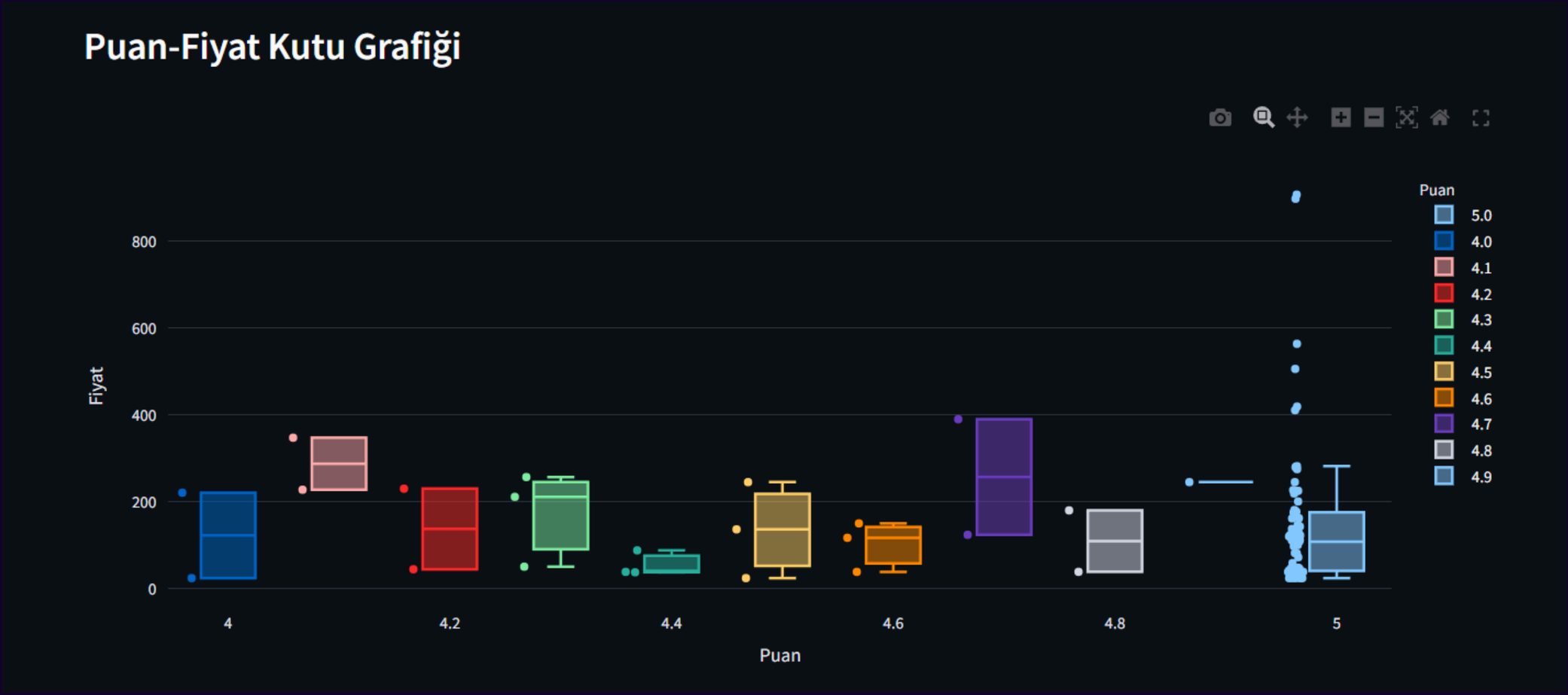
Bir sonraki bölümümüzde ise, yıllara göre kitap dağılımını gösteren bir çizgi grafiği yer almaktadır. Bu grafik sayesinde hangi yıllarda daha fazla kitap yayımlandığını, hangi yıllarda ise düşüş yaşandığını net bir şekilde görebiliriz. Böylece veriyi analiz ederek anlamlı ve yorumlanabilir bilgilere dönüştürmüş oluyoruz.



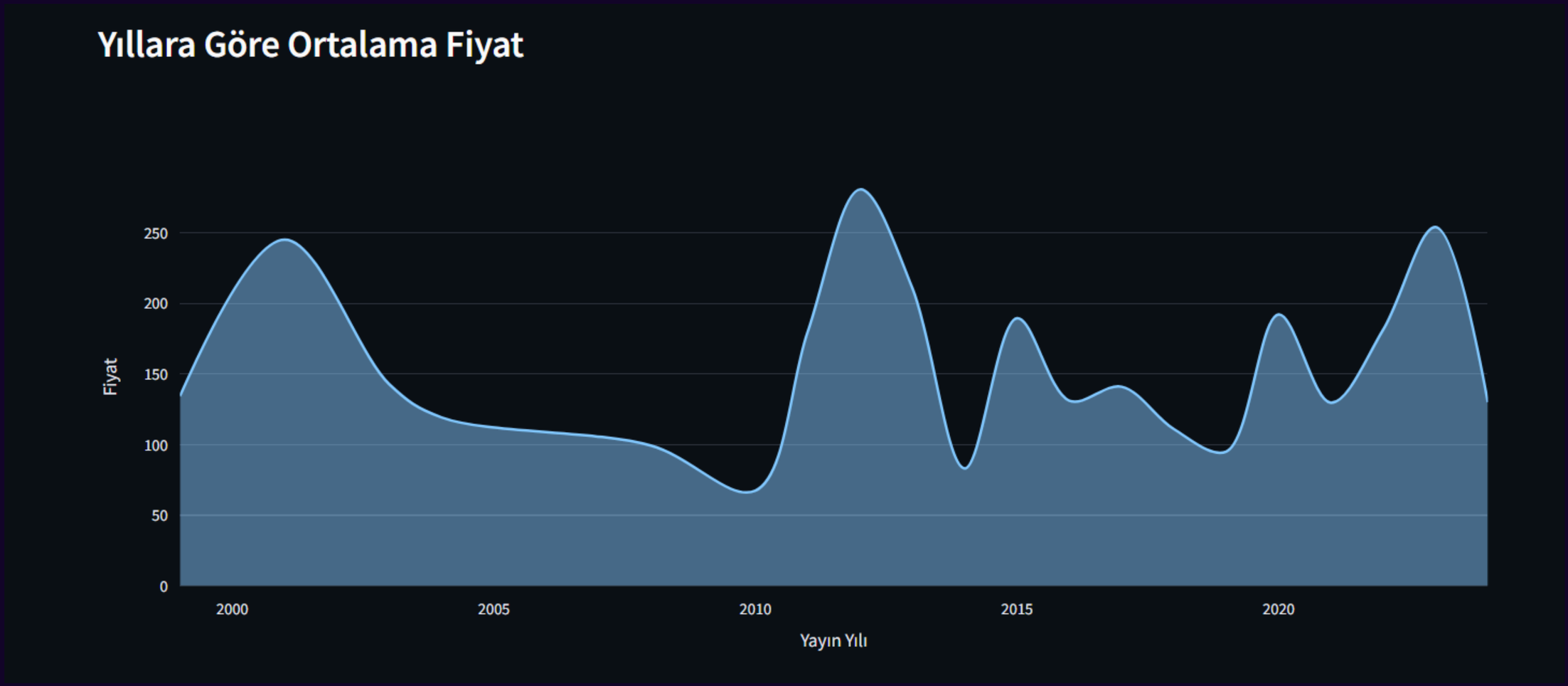
Sayfamızda ilk kısım olan “Fiyat Dağılımı” bölümünü açtığımızda, ikinci bir grafik olarak “Fiyat-Puan İlişkisi”ni gösteren bir dağılım grafiği karşımıza çıkmaktadır. Bu grafik sayesinde kitapların fiyatlarına göre aldıkları puanları detaylı bir şekilde inceleyebiliriz. Ayrıca, veri noktaları yıllara göre renklendirilmiş olup, bu sayede yıllara göre fiyat ve puan eğilimlerini de görsel olarak analiz edebilmemizi sağlar.



Sayfanın “Puan Analizi” kısmındaki ikinci bölümde ise bir kutu grafiği yer almaktadır. Bu grafik, kitapların puanlarının fiyatlara göre nasıl dağıldığını göstermektedir. Bu sayede belirli puan aralıklarındaki kitapların fiyat seviyelerini görerek, fiyat-puan ilişkisini daha detaylı analiz edebiliriz.



Yıllara göre dağılım bölümümüzde yer alan bir diğer grafik ise, yıllara göre ortalama kitap fiyatlarını gösteren çizgi-alan grafiğidir. Bu grafik sayesinde her yıl için kitapların ortalama fiyat seviyesini görsel olarak inceleyebilir, zaman içerisindeki fiyat değişimlerini kolayca analiz edebiliriz.



Web sayfamızın alt kısmında, filtrelenmiş verileri içeren bir veri tablosu yer almaktadır. Bu tabloda, sol taraftaki üç filtreleme seçeneğine göre veriler dinamik olarak listelenir. Tablo, dikeyde kaydırılabilir olup, kullanıcı filtrelediği verileri kolayca inceleyebilir.

Dashboard Web Sayfamızın En Alt Kısmı

Sayfanın en alt kısmında, istediğimiz gibi filtrelediğimiz veriyi CSV formatında indirebiliyoruz. Bu sayede veriyi kayıt altına alarak ileride kullanmak üzere saklayabiliriz.

Filtreleme Seçenekleri

Yayın Yılı Aralığı

1999 — 2024

Minimum Puan

4.00

1.00 — 5.00

Fiyat Aralığı (TL)

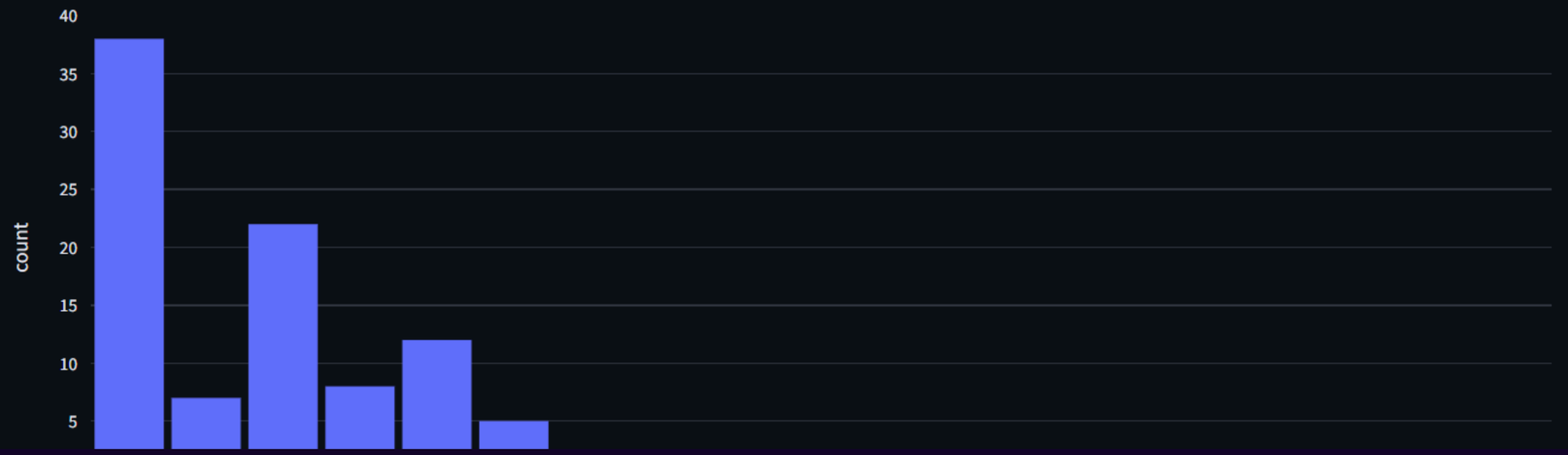
0 — 1007

📊 Amazon Kitap Analiz Paneli

Toplam Kitap	Ortalama Fiyat	Ortalama Puan	En Yüksek Fiyat
100	145.20 TL	4.87	907.50 TL

📈 Fiyat Dağılımı | ⭐ Puan Analizi | ✍️ Yazar İstatistikleri | 🗓️ Yıllara Göre

Fiyat Dağılımı



Filtreleme Seçenekleri

Yayın Yılı Aralığı

1999 2024

1999 2024

Minimum Puan

4.00

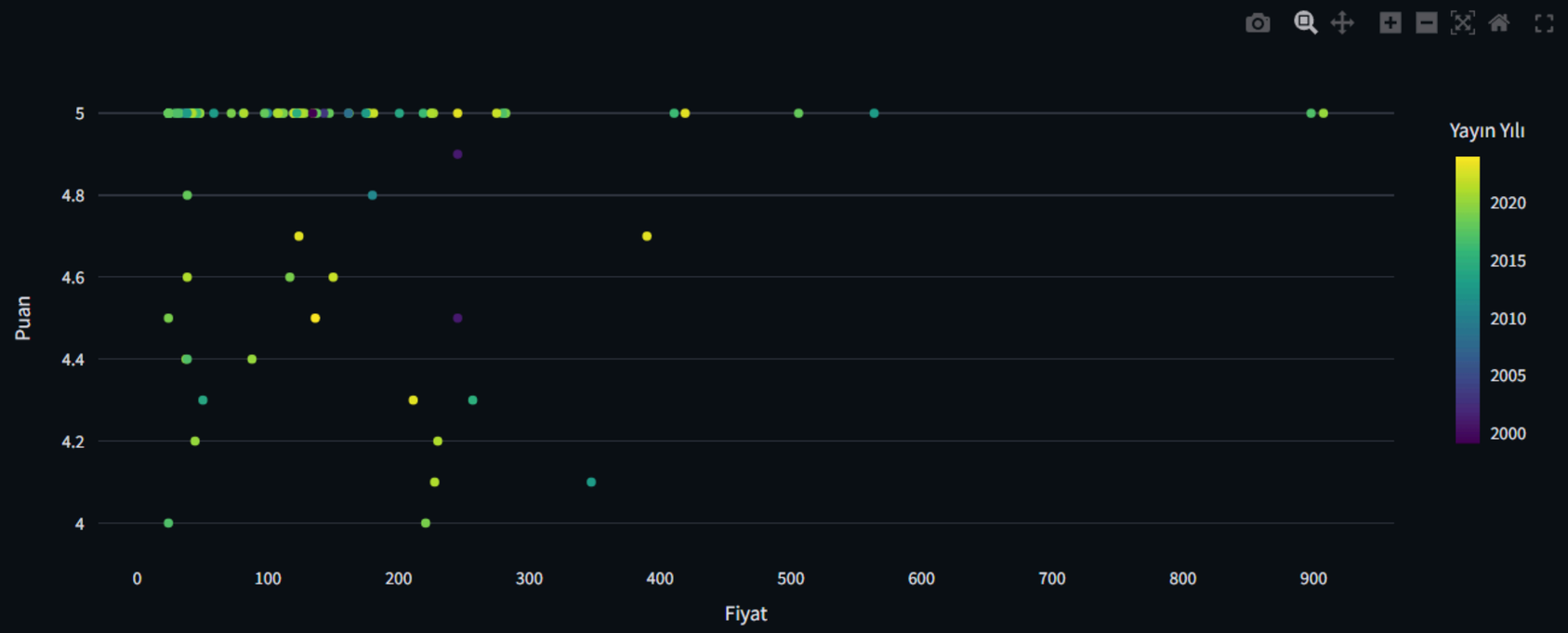
1.00 5.00

Fiyat Aralığı (TL)

0 1007

0 1007

Fiyat-Puan İlişkisi



Filtrelenmiş Veri Tablosu

Filtreleme Seçenekleri

Yayın Yılı Aralığı

1999 2024

1999 2024

Minimum Puan

4.00

1.00 5.00

Fiyat Aralığı (TL)

0 1007

0 1007

Filtrelenmiş Veri Tablosu

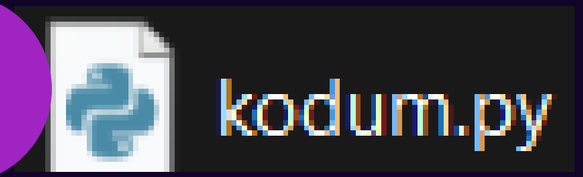
Yazar	Yayın Tarihi	Puan	Fiyat	Yayın Yılı
Byung-Chul Han	29 Eylül 2019	4.6 ⭐	116.84 TL	2019
Liane Schneider ve Aylin Gergin	2025-05-24	4.6 ⭐	38.34 TL	2021
[Yazar] Jason Schreier	2022-05-25	4.6 ⭐	150.00 TL	2022
Kentaro Miura ve Peren Ercan	31 Ekim 2024	4.5 ⭐	136.29 TL	2024
Felix Salten ve Yonca Aşçı Dalar	2019-03-18	4.5 ⭐	23.96 TL	2019
Kör Baykuş	2025-05-24	4.5 ⭐	245.20 TL	2001
Roger Hargreaves	2025-05-24	4.4 ⭐	87.88 TL	2020
Sabahattin Ali	2020-03-01	4.4 ⭐	37.40 TL	2020
Liane Schneider ve Aylin Gergin	2025-05-24	4.4 ⭐	38.34 TL	2017
İskender Pala	2025-05-24	4.3 ⭐	256.75 TL	2015
Rainer Maria Rilke ve Ahmet Cemal	2025-05-24	4.3 ⭐	50.32 TL	2014

Filtrelenmiş Veriyi İndir (CSV)

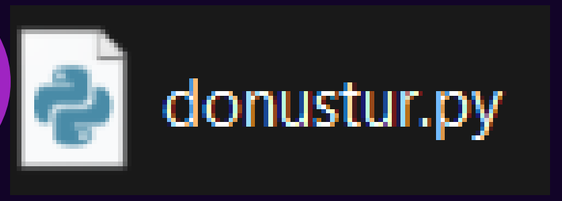
Kaynak Kodlarımızın İncelenmesi

- dashboard.py
- donustur.py
- kodum.py
- veri.json

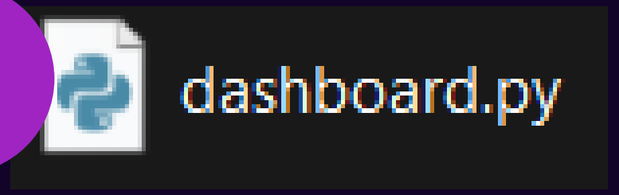
İşlem Sırası

1 

kodum.py dosyasını hazırlıyoruz. Bu dosya Selenium kullanarak Amazon’dan veri çekiyor ve verileri veri.txt dosyası olarak kaydediyor.

2 

donustur.py dosyasını hazırlıyoruz. Bu dosya veri.txt dosyasını okuyup, daha kullanışlı ve yapılandırılmış bir formata yani veri.json dosyasına dönüştürüyor.

3 

dashboard.py dosyasını oluşturuyoruz. Streamlit ile lokal bir web uygulaması olarak çalışan bu dosya, veri.json dosyasını kullanarak verileri işliyor, görselleştiriyor ve grafiklerle kullanıcıya sunuyor.

kodum.py dosyasının hazırlanması

Öncelikle, web tarayıcısını otomatik olarak kontrol etmek için Selenium kütüphanesini yüklüyorum.

Web sayfasındaki öğeleri seçmek, beklemek ve hataları yakalamak için gerekli modülleri içe aktarıyorum.

Ayrıca sayfa yüklenmesini beklemek ve rastgele sayfa seçmek için time ve random modüllerini kullanıyorum.

kodum.py > wait_for_products

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
from time import sleep
import random
```

kodum.py dosyasının hazırlanması

Burada Amazon kitap sayfasının adresini ve veri limitini belirledikten sonra, Chrome tarayıcısı için gerekli ayarları yapıyorum.

Önemli nokta, ChromeDriver dosyasını Google'ın resmi Chrome Testing sayfası olan https://googlechromelabs.github.io/chrome-for-testing/ sitesinden indiriyorum.

İndirirken Windows platformu için uygun sürümü seçip, indirdiğim chromedriver.exe dosyasını projenin içine koyuyorum.

Sonra bu dosyayı kullanarak Chrome'u otomatik olarak başlatıyorum.

```
11  BASE_URL = "https://www.amazon.com.tr/s?i=stripbooks&rh=n%3A12466496011&s=review-rank&page={page}&qid=1748017039&xpid=8wJiG5JLpILI9&ref=sr_pg_{page}"
12
13  PRODUCT_LIMIT = 100
14  OUTPUT_FILE = "veri.txt"
15
16  chrome_options = Options()
17  chrome_options.add_argument("--disable-gpu")
18  chrome_options.add_argument("--no-sandbox")
19
20  service = Service(r"chromedriver.exe")
21  driver = webdriver.Chrome(service=service, options=chrome_options)
```

Binary	Platform	URL	HTTP status
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/win32/chromedriver-win32.zip	200
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/win64/chromedriver-win64.zip	200
chrome-headless-shell	linux64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/linux64/chrome-headless-shell-linux64.zip	200
chrome-headless-shell	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/136.0.7103.113/mac-arm64/chrome-headless-shell-mac-arm64.zip	200

Chrome tarayıcı otomasyonu için gereken ChromeDriver dosyasını indirirken, Chrome for Testing sitesini kullanabilirsiniz. Bu site, Google tarafından sağlanmakta olup, işletim sisteminize (Windows, macOS, Linux) uygun en güncel ChromeDriver sürümünü indirmenize olanak tanır. Böylece programınız, Chrome tarayıcısını sorunsuz ve uyumlu şekilde kontrol edebilir. Siteye gidip sisteminize uygun dosyayı seçerek kolayca indirebilirsiniz.

URL: https://googlechromelabs.github.io/chrome-for-testing/

kodum.py dosyasının hazırlanması

Bu fonksiyon sayfadaki ürünlerin tamamen yüklenmesini sağlamak için 15 saniye kadar bekliyor.
Eğer ürünler gelmezse ya da CAPTCHA çıkarsa hata mesajı yazıyor ve işlemi durduruyor.
Bu sayede veri çekme işlemi sağlam şekilde ilerliyor.

```
26  def wait_for_products():
27      try:
28          WebDriverWait(driver, 15).until(
29              EC.presence_of_all_elements_located(
30                  (By.CSS_SELECTOR, "div.s-main-slot div.s-result-item[data-component-type='s-search-result']")
31              )
32          )
33          return True
34      except TimeoutException:
35          print("Ürünler yüklenemedi veya CAPTCHA engeli.")
36          return False
```

kodum.py dosyasının hazırlanması

Bu fonksiyon, her ürünün fiyat bilgisini parçalar halinde çekiyor.

Tamsayı ve kuruş kısımlarını ayrı ayrı alıp, düzgün formatta fiyatı oluşturuyoruz.

Eğer fiyat bilgisi yoksa ‘Fiyat yok’ olarak dönüyor. Böylece eksik veri sorununu önlüyoruz.

```
38  def extract_price(product):
39      try:
40          price_whole = product.find_element(By.CSS_SELECTOR, "span.a-price-whole").text.replace(".", "").replace(",", ".").strip()
41          price_fraction = product.find_element(By.CSS_SELECTOR, "span.a-price-fraction").text.strip()
42          price = f"{price_whole},{price_fraction} TL"
43          return price
44      except:
45          return "Fiyat yok"
```

kodum.py dosyasının hazırlanması

Burada toplamda 100 kitap bulana kadar rastgele sayfalara gidiyoruz.

Ziyaret ettiğimiz sayfaları takip edip aynı sayfaya tekrar gitmiyoruz.

Her sayfa açıldığında 3 saniye bekliyor, ürünlerin yüklenmesini kontrol ediyoruz.

Eğer sayfa yüklenmezse döngüyü sonlandırıyoruz.

```
47  while len(product_list) < PRODUCT_LIMIT:
48      page = random.randint(1, 70)
49      if page in pages_visited:
50          continue
51      pages_visited.add(page)
52      print(f"Sayfa {page} açılıyor...")
53
54      url = BASE_URL.format(page=page)
55      driver.get(url)
56      sleep(3)
57
58      if not wait_for_products():
59          break
60
61      products = driver.find_elements(By.CSS_SELECTOR, "div.s-main-slot div.s-result-item[data-component-type='s-search-result']")
62
63      for product in products:
64          if len(product_list) >= PRODUCT_LIMIT:
65              break
```

kodum.py dosyasının hazırlanması

Her ürün için:

- Öncelikle ürünün puanını alıyoruz, 4’ten düşükse atlıyoruz.
- Yazar bilgisi çekiyoruz, standart dışı ve gereksiz yazar isimlerini filtreliyoruz.
- Yayın tarihini alıyoruz, yoksa ‘Tarih yok’ yazıyoruz.

```
61      products = driver.find_elements(By.CSS_SELECTOR, "div.s-main-slot div.s-result-item[data-component-type='s-search-result']")
62
63      for product in products:
64          if len(product_list) >= PRODUCT_LIMIT:
65              break
66
67          try:
68              rating_str = product.find_element(By.CSS_SELECTOR, "span.a-icon-alt").get_attribute("innerHTML")
69              rating = float(rating_str.split()[0].replace(",", "."))
70              if rating < 4.0:
71                  continue
72          except:
73              continue
74
75          try:
76              author = product.find_element(By.CSS_SELECTOR, ".a-row.a-size-base.a-color-secondary .a-link-normal").text.strip()
77          except:
78              try:
79                  author = product.find_element(By.CSS_SELECTOR, ".a-row.a-size-base.a-color-secondary").text.strip()
80              except:
81                  author = "Yazar yok"
82
83          # Standart olmayan yazar bilgisi varsa atla
84          if any(x in author for x in ["yeni ürün", "İngilizce Bask", "Kolektif"]):
85              continue
86
87          try:
88              date = product.find_element(By.CSS_SELECTOR, ".a-row.a-size-base.a-color-secondary .a-text-normal").text.strip()
89          except:
90              date = "Tarih yok"
91
92          price = extract_price(product)
93
94          entry = f"Yazar: {author} | Yayın Tarihi: {date} | Puan: {rating} | Fiyat: {price}"
95
96          if entry not in product_list:
97              product_list.append(entry)
```

- Fiyat bilgisini az önceki fonksiyon ile çekiyoruz.
- Sonra tüm bu bilgileri tek satırda birleştirip listeye ekliyoruz.
- Aynı ürün birden fazla eklenmesin diye kontrol yapıyoruz.

kodum.py dosyasının hazırlanması

İstenen sayıdaki kitap verisini topladıktan sonra, bilgileri veri.txt dosyasına yazıyoruz.
Dosyaya yazarken her kitabı numaralandırıyoruz.
Son olarak başarı mesajı verip Chrome tarayıcısını kapatıyoruz.
Böylece veri çekme işlemi tamamlanmış oluyor.

```
99  print(f"Toplam {len(product_list)} kitap bulundu. Dosyaya yazılıyor...")
100
101 with open(OUTPUT_FILE, "w", encoding="utf-8") as f:
102     for i, entry in enumerate(product_list, 1):
103         f.write(f"{i}. {entry}\n\n")
104
105 print(f"✅ Veriler '{OUTPUT_FILE}' dosyasına kaydedildi.")
106
107 driver.quit()
108
```

donustur.py dosyasının hazırlanması

Öncelikle Python'da JSON işlemleri için json modülünü, metin içinde desen aramak için ise re yani düzenli ifadeler modülünü kullanıyoruz. Ayrıca, elimizdeki veri dosyasının ismini INPUT_FILE olarak veri.txt, dönüştüreceğimiz dosyanın ismini ise OUTPUT_FILE olarak veri.json olarak belirliyoruz.

donustur.py > ...

```python
import json
import re

INPUT_FILE = "veri.txt"
OUTPUT_FILE = "veri.json"
```

donustur.py dosyasının hazırlanması

Bu kısımda, elimizdeki veri.txt dosyasındaki her satırı tek tek parçalayan bir fonksiyon yazıyoruz.
Burada pattern dediğimiz düzenli ifadeyle, satırdaki Yazar, Yayın Tarihi, Puan ve Fiyat bilgilerini yakalıyoruz.
Eğer satır doğru formatta ise bu bilgileri alıyoruz, puan kısmındaki virgülü nokta ile değiştirip sayıya çevirmeye çalışıyoruz.
Başarılı olursa sayısal değeri alıyoruz, hata varsa None olarak bırakıyoruz.
Sonra bu değerleri bir sözlük içinde döndürüyoruz.
Eğer satır formata uymuyorsa None döndürüyoruz.

```python
def parse_line(line):
    # Satırdaki bilgileri düzenli ifadeyle yakalayıp parçala
    pattern = r"Yazar:\s*(.*?)\s*\|\s*Yayın Tarihi:\s*(.*?)\s*\|\s*Puan:\s*([\d.,]+)\s*\|\s*Fiyat:\s*(.*)"
    match = re.search(pattern, line)
    if match:
        yazar = match.group(1).strip()
        yayin_tarihi = match.group(2).strip()
        puan_str = match.group(3).replace(",", ".").strip()  # Virgülü noktaya çevir (float için)
        try:
            puan = float(puan_str)
        except ValueError:
            puan = None
        fiyat = match.group(4).strip()
        return {
            "Yazar": yazar,
            "Yayın Tarihi": yayin_tarihi,
            "Puan": puan,
            "Fiyat": fiyat
        }
    return None
```

donustur.py dosyasının hazırlanması

```python
def main():
    data = []
    with open(INPUT_FILE, "r", encoding="utf-8") as f:
        lines = f.readlines()
        for line in lines:
            line = line.strip()
            if line == "" or line[0].isdigit() == False:  # Satır boşsa veya numaralı değilse atla
                continue
            parsed = parse_line(line)
            if parsed:
                data.append(parsed)

    with open(OUTPUT_FILE, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

    print(f"✅ {len(data)} kayıt '{OUTPUT_FILE}' dosyasına dönüştürüldü.")

if __name__ == "__main__":
    main()
```

Burada programın ana çalıştığı fonksiyon var.
İlk önce boş bir liste data oluşturuyoruz.
veri.txt dosyasını açıp satır satır okuyoruz.
Her satırdaki boşlukları temizliyoruz, boş satırları veya numara ile başlamayan satırları atlıyoruz.
Geçerli satırları parse_line fonksiyonuna gönderiyoruz ve sözlük olarak dönen veriyi data listesine ekliyoruz.
Tüm satırlar okunduktan sonra, data listesini veri.json dosyasına JSON formatında yazıyoruz.
Son olarak kaç kayıt dönüştürdüğümüzü kullanıcıya bildiriyoruz.

dashboard.py dosyasının hazırlanması

1. Veri Yükleme ve Temizleme:
Kod başında streamlit, pandas, plotly gibi kütüphaneler içe aktarılır. load_data() fonksiyonuyla 'veri.json' dosyasındaki kitap verisi okunur, DataFrame'e çevrilir. Yayın tarihinden yıl çıkarılır, fiyat sütunu sayıya dönüştürülür ve boş yazarlar "Bilinmeyen Yazar" olarak doldurulur. Bu işlem cache'lenir, böylece performans artar.

2. Filtreler ve Sidebar:
Sayfanın sol tarafında kullanıcıya kitapları filtrelemek için üç seçenek sunulur: yayın yılı aralığı, minimum puan ve fiyat aralığı. Kullanıcı bu filtreleri seçince, DataFrame içindeki veriler buna göre süzülür

3. Özet Bilgiler ve Sayfa Başlığı:
Sayfanın üst kısmında filtrelenmiş veriye göre dört özet gösterge (metric) bulunur: toplam kitap sayısı, ortalama fiyat, ortalama puan ve en yüksek fiyat. Ayrıca sayfa başlığı ve özel CSS ile şık bir görünüm sağlanır.

4. Grafik Sekmeleri:
Dört farklı sekme var. İlkinde fiyat histogramı ve fiyat-puan ilişkisi grafiği; ikincisinde puan dağılımı ve puan-fiyat kutu grafiği; üçüncüsünde en çok kitap yazan yazarların çubuk grafiği; dördüncüsünde yıllara göre kitap sayısı ve ortalama fiyatın çizgi ve alan grafikleri gösteriliyor. Böylece veriyi farklı açılardan görsel olarak analiz etmek kolaylaşıyor.

5. Veri Tablosu ve İndirme:
Son olarak, filtrelenmiş veriler tablo halinde gösteriliyor ve kullanıcı istediğinde bu filtreli veriyi CSV olarak bilgisayarına indirebiliyor. Tablo, puan sırasına göre sıralanmış ve sütunlar kullanıcı dostu biçimde yapılandırılmış.

dashboard.py dosyasının hazırlanması

```python
# dashboard.py
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime
import json
import numpy as np
from collections import Counter

# Sayfa ayarları
st.set_page_config(
    page_title="Amazon Kitap Analiz Paneli",
    page_icon="📚",
    layout="wide",
    initial_sidebar_state="expanded"
)
```

```python
with tab2:
    st.subheader("Puan Dağılımı")
    fig = px.pie(
        filtered_df,
```

```python
with tab4:
    st.subheader("Yıllara Göre Kitap Sayısı")
    yearly_counts = filtered_df.groupby('Yayın Yılı').size().reset_index(name='Kitap Sayısı')
    fig = px.line(
        yearly_counts,
        x='Yayın Yılı',
        y='Kitap Sayısı',
        markers=True,
        line_shape='spline'
    )
    st.plotly_chart(fig, use_container_width=True)

    st.subheader("Yıllara Göre Ortalama Fiyat")
    yearly_price = filtered_df.groupby('Yayın Yılı')['Fiyat'].mean().reset_index()
    fig = px.area(
        yearly_price,
        x='Yayın Yılı',
        y='Fiyat',
        line_shape='spline'
    )
    st.plotly_chart(fig, use_container_width=True)

# Ham veri gösterimi
st.subheader("Filtrelenmiş Veri Tablosu")
st.dataframe(
    filtered_df.sort_values('Puan', ascending=False),
    column_config={
        "Yazar": st.column_config.TextColumn("Yazar", width="medium"),
        "Yayın Tarihi": st.column_config.DateColumn("Yayın Tarihi"),
        "Puan": st.column_config.NumberColumn("Puan", format="%.1f ⭐"),
        "Fiyat": st.column_config.NumberColumn("Fiyat", format="%.2f TL")
    },
    hide_index=True,
    use_container_width=True
)

# Veri indirme butonu
st.download_button(
    label="Filtrelenmiş Veriyi İndir (CSV)",
    data=filtered_df.to_csv(index=False).encode('utf-8'),
    file_name='filtrelenmis_kitap_verisi.csv',
    mime='text/csv'
)
```

```python
col1, c
col1.me
col2.me
col3.me
col4.me

# Grafi
tab1, t

with ta
    st.
    fig

    )
    fig
    st.

    st.
    fig = px.scatter(
        filtered_df,
```

dashboard.py dosyasının hazırlanması

dashboard.py > ...

```python
# dashboard.py
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime
import json
import numpy as np
from collections import Counter

# Sayfa ayarları
st.set_page_config(
    page_title="Amazon Kitap Analiz Paneli",
    page_icon="📚",
    layout="wide",
    initial_sidebar_state="expanded"
)

# Veriyi yükle
@st.cache_data
def load_data():
    with open('veri.json', 'r', encoding='utf-8') as f:
        data = json.load(f)
    df = pd.DataFrame(data)

    # Veri temizleme
    df['Yayın Yılı'] = df['Yayın Tarihi'].str.extract(r'(\d{4})').astype(float)
    df['Fiyat'] = pd.to_numeric(df['Fiyat'], errors='coerce')

    # Yazar işlemleri
    df['Yazar'] = df['Yazar'].fillna('Bilinmeyen Yazar')
    return df

df = load_data()
```

dashboard.py dosyasının hazırlanması

```python
# Sidebar filtreleri
st.sidebar.header("Filtreleme Seçenekleri")
min_year, max_year = int(df['Yay□n Y□l□'].min()), int(df['Yay□n Y□l□'].max())
year_range = st.sidebar.slider(
    "Yay□n Y□l□ Aral□□□",
    min_year, max_year, (min_year, max_year)
)

rating_filter = st.sidebar.slider(
    "Minimum Puan",
    1.0, 5.0, 4.0, 0.1
)

price_filter = st.sidebar.slider(
    "Fiyat Aral□□□ (TL)",
    0, int(df['Fiyat'].max()) + 100, (0, int(df['Fiyat'].max()) + 100)
)

# Filtreleme
filtered_df = df[
    (df['Yay□n Y□l□'] >= year_range[0]) &
    (df['Yay□n Y□l□'] <= year_range[1]) &
    (df['Puan'] >= rating_filter) &
    (df['Fiyat'] >= price_filter[0]) &
    (df['Fiyat'] <= price_filter[1])
]

# Ana sayfa
st.title("📊 Amazon Kitap Analiz Paneli")
st.markdown("""
<style>
div[data-testid="metric-container"] {
    background-color: rgba(28, 131, 225, 0.1);
    border: 1px solid rgba(28, 131, 225, 0.1);
    padding: 5% 5% 5% 10%;
    border-radius: 5px;
    color: rgb(30, 103, 119);
}
div[data-testid="metric-container"] > label {
    color: rgba(35, 86, 131);
}
</style>
""", unsafe_allow_html=True)
```

dashboard.py dosyasının hazırlanması

```python
# Metrikler
col1, col2, col3, col4 = st.columns(4)
col1.metric("Toplam Kitap", len(filtered_df))
col2.metric("Ortalama Fiyat", f"{filtered_df['Fiyat'].mean():.2f} TL")
col3.metric("Ortalama Puan", f"{filtered_df['Puan'].mean():.2f}")
col4.metric("En Yüksek Fiyat", f"{filtered_df['Fiyat'].max():.2f} TL")

# Grafikler
tab1, tab2, tab3, tab4 = st.tabs(["📈 Fiyat Dağılımı", "⭐ Puan Analizi", "✍️ Yazar İstatistikleri", "🗓️ Yıllara Göre"])

with tab1:
    st.subheader("Fiyat Dağılımı")
    fig = px.histogram(
        filtered_df,
        x='Fiyat',
        nbins=20,
        color_discrete_sequence=['#636EFA'],
        labels={'Fiyat': 'Fiyat (TL)'}
    )
    fig.update_layout(bargap=0.1)
    st.plotly_chart(fig, use_container_width=True)

    st.subheader("Fiyat-Puan İlişkisi")
    fig = px.scatter(
        filtered_df,
        x='Fiyat',
        y='Puan',
        color='Yayın Yılı',
        hover_data=['Yazar'],
        color_continuous_scale='viridis'
    )
    st.plotly_chart(fig, use_container_width=True)
```

dashboard.py dosyasının hazırlanması

```python
with tab2:
    st.subheader("Puan Dağılımı")
    fig = px.pie(
        filtered_df,
        names='Puan',
        color_discrete_sequence=px.colors.sequential.Viridis
    )
    st.plotly_chart(fig, use_container_width=True)

    st.subheader("Puan-Fiyat Kutu Grafiği")
    fig = px.box(
        filtered_df,
        x='Puan',
        y='Fiyat',
        color='Puan',
        points="all"
    )
    st.plotly_chart(fig, use_container_width=True)

with tab3:
    # Yazar analizi
    st.subheader("En Çok Eseri Olan Yazarlar")

    # Yazar isimlerini ayır
    all_authors = []
    for authors in filtered_df['Yazar']:
        if " ve " in authors:
            all_authors.extend([a.strip() for a in authors.split("ve")])
        elif "," in authors:
            all_authors.extend([a.strip() for a in authors.split(",")])
        else:
            all_authors.append(authors.strip())

    author_counts = Counter(all_authors)
    top_authors = pd.DataFrame(author_counts.most_common(10), columns=['Yazar', 'Kitap Sayısı'])

    fig = px.bar(
        top_authors,
        x='Kitap Sayısı',
        y='Yazar',
        orientation='h',
        color='Kitap Sayısı',
        color_continuous_scale='deep'
    )
    st.plotly_chart(fig, use_container_width=True)
```

dashboard.py dosyasının hazırlanması

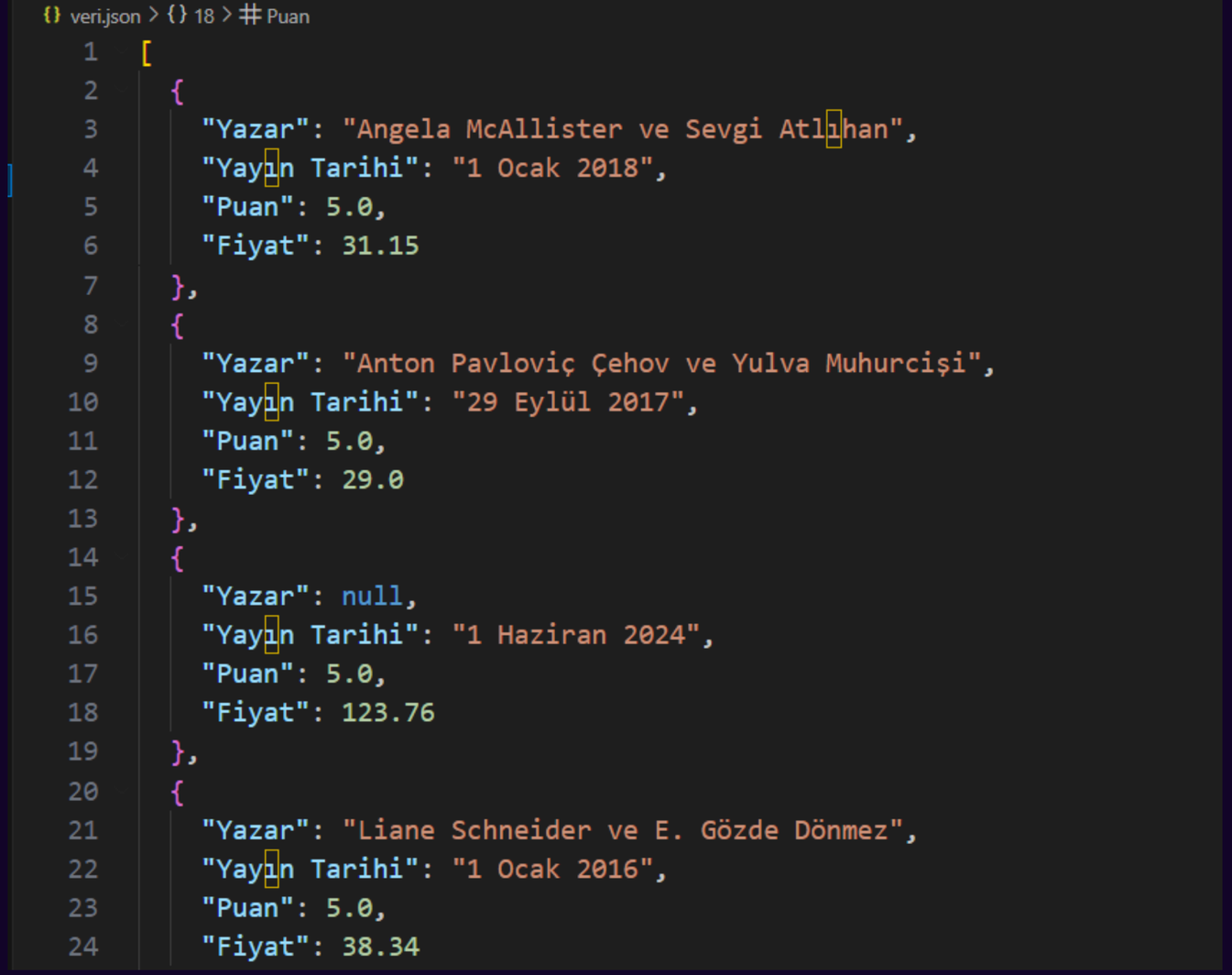
```python
with tab4:
    st.subheader("Yıllara Göre Kitap Sayısı")
    yearly_counts = filtered_df.groupby('Yayın Yılı').size().reset_index(name='Kitap Sayısı')
    fig = px.line(
        yearly_counts,
        x='Yayın Yılı',
        y='Kitap Sayısı',
        markers=True,
        line_shape='spline'
    )
    st.plotly_chart(fig, use_container_width=True)

    st.subheader("Yıllara Göre Ortalama Fiyat")
    yearly_price = filtered_df.groupby('Yayın Yılı')['Fiyat'].mean().reset_index()
    fig = px.area(
        yearly_price,
        x='Yayın Yılı',
        y='Fiyat',
        line_shape='spline'
    )
    st.plotly_chart(fig, use_container_width=True)

# Ham veri gösterimi
st.subheader("Filtrelenmiş Veri Tablosu")
st.dataframe(
    filtered_df.sort_values('Puan', ascending=False),
    column_config={
        "Yazar": st.column_config.TextColumn("Yazar", width="medium"),
        "Yayın Tarihi": st.column_config.DateColumn("Yayın Tarihi"),
        "Puan": st.column_config.NumberColumn("Puan", format="%.1f ⭐"),
        "Fiyat": st.column_config.NumberColumn("Fiyat", format="%.2f TL")
    },
    hide_index=True,
    use_container_width=True
)

# Veri indirme butonu
st.download_button(
    label="Filtrelenmiş Veriyi İndir (CSV)",
    data=filtered_df.to_csv(index=False).encode('utf-8'),
    file_name='filtrelenmis_kitap_verisi.csv',
    mime='text/csv'
)
```

veri.json Dosyasının Yapısının İncelenmesi



Veri.json dosyası, kitaplara ait bilgilerin JSON formatında tutulduğu bir dosyadır. Dosya, bir liste şeklindedir ve her öğesi bir kitabın özelliklerini içeren bir sözlüktür. Her kitap için yazar, yayın tarihi, puan ve fiyat bilgileri bulunmaktadır. Yazarlar bazen birden fazla isim içerir ve “ve” ile ayrılmıştır. Bazı kitaplarda yazar bilgisi boş (null) olabilir. Yayın tarihi, genellikle “gün ay yıl” biçimindedir. Puanlar 1.0 ile 5.0 arasında değişir ve ondalıklı sayı olarak verilmiştir.

Fiyat ise kitabın satış tutarını belirtir. Bu yapı, kitapların temel bilgilerini düzenli bir şekilde saklamak ve analizlerde kullanılmak üzere uygundur. Eksik yazar bilgileri daha sonra “Bilinmeyen Yazar” olarak işlenir ve yayın tarihinden sadece yıl bilgisi çıkarılarak analizlerde kullanılır.

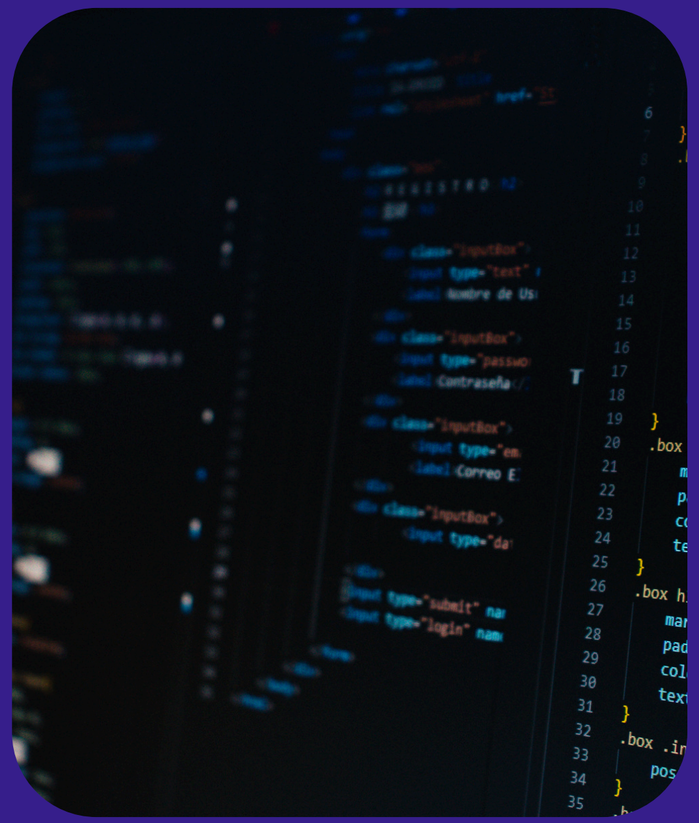
Programın Çalıştırılması

Bu projeyi çalıştırmak için terminal veya komut istemcisinde streamlit run .\dashboard.py komutunu kullanıyoruz. Bu komut, Streamlit kütüphanesini devreye alarak dashboard.py dosyasındaki kodu çalıştırır ve web tarayıcısında interaktif bir analiz paneli açar. Panel üzerinden kitap verilerini filtreleyebilir, grafiklerle görselleştirebilir ve detaylı analizlere kolayca ulaşabilirsiniz. Program çalışırken yaptığınız değişiklikler otomatik olarak güncellenir, böylece hızlıca sonuçları görebilirsiniz.

```
PS C:\Users\samet\OneDrive\Desktop\web_odev> streamlit run .\dashboard.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.112:8501
```



TEŞEKKÜR EDERİM