WEB KAZIMADA KULLANILAN PYTHON KÜTÜPHANELERİNİN İNCELENMESİ

YÜKSEK LİSANS TEZİ

Murat Ali ÖZ

Danışman
Dr. Ögr. Üyesi Naim KARASEKRETER

İNTERNET VE BİLİŞİM TEKNOLOJİLERİ YÖNETİMİ ANABİLİM DALI

Şubat 2025

AFYON KOCATEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

WEB KAZIMADA KULLANILAN PYTHON KÜTÜPHANELERİNİN İNCELENMESİ

Murat Ali ÖZ

Danışman
Dr. Ögr. Üyesi Naim KARASEKRETER

İNTERNET VE BİLİŞİM TEKNOLOJİLERİ YÖNETİMİ
ANABİLİM DALI

Şubat 2025

TEZ ONAY SAYFASI

Murat Ali ÖZ tarafından hazırlanan "Web Kazımada Kullanılan Python Kütüphanelerinin İncelenmesi" adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 03/02/2025 tarihinde aşağıdaki jüri tarafından **oy birliği** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü **İnternet ve Bilişim Teknolojileri Yönetimi Anabilim Dalı'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman **:** Dr. Öğr. Üyesi Naim KARASEKRETER

Başkan **:** Doç.Dr. Mehmet Akif ŞAHMAN
Selçuk Üniversitesi Teknoloji Fakültesi

…… İmza ……

Üye **:** Dr. Öğr. Üyesi Naim KARASEKRETER
Afyon Kocatepe Üniversitesi Mühendislik Fakültesi

…… İmza ……

Üye **:** Dr. Öğr. Üyesi Caner BALIM
Afyon Kocatepe Üniversitesi Mühendislik Fakültesi

…… İmza ……

Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../..... tarih ve
……. sayılı kararıyla onaylanmıştır.

…….
Prof. Dr. Bekir YALÇIN
Enstitü Müdürü

BİLİMSEL ETİK BİLDİRİM SAYFASI

Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

03/02/2025

İmza

Murat Ali Öz

ÖZET

Yüksek Lisans Tezi

WEB KAZIMADA KULLANILAN
PYTHON KÜTÜPHANELERİNİN İNCELENMESİ

Murat Ali ÖZ
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
İnternet ve Bilişim Teknolojileri Yönetimi Anabilim Dalı
Danışman: Dr. Ögr. Üyesi Naim KARASEKRETER

Veri, günümüzde kuruluşlar için önemli bir varlık haline gelmiş ve internet, başlıca veri kaynağı olmuştur. Web kazıma, HTML (Hyper Text Markup Language - Hiper Metin İşaretleme Dili) dokümanlarından yapılandırılmış verileri otomatik olarak çekmeyi sağlar. Bu veriler; ürün ve hizmet bilgilerini toplama, fiyat karşılaştırması yapma, iletişim bilgileri edinme, haber ve blog takibi yapma, müşteri geri bildirimlerini analiz etme gibi çeşitli amaçlar için kullanılabilir. Web kazıma süreci, bir web sayfası ile web robotu arasındaki otomatik veri aktarımını ifade eder.

Bu tez çalışmasında, BeautifulSoup, Selenium ve Scrapy gibi web kazımada kullanılan Python kütüphaneleri detaylı olarak incelenmiş, hangi tür web sitelerinde ve hangi amaçlar için daha uygun oldukları belirlenmeye çalışılmıştır. Web kazıma, web sitelerinden otomatik olarak veri toplama tekniği olup, günümüzde veri analizi, makine öğrenimi ve veri madenciliği gibi çeşitli alanlarda kullanılmaktadır.

Web sayfalarının yapısını tanımlamak için HTML işaretleme dili kullanılırken, stil özelliklerini tanımlamak için CSS (Cascading Style Sheets - Basamaklı Stil Sayfaları) stil dili kullanılır. Python, web geliştirme, veri bilimi ve yapay zeka gibi birçok alanda kullanılan yüksek seviyeli bir programlama dilidir. BeautifulSoup, Selenium ve Scrapy kütüphaneleri, web kazımada kullanılan başlıca Python kütüphanelerdir.

Belirtilen kütüphaneler (BeautifulSoup, Selenium, Scrapy) kullanılarak çeşitli web sitelerinden (emlak ilan sitesi, haber sitesi, online market ve e-ticaret sitesi) veri kazıma kodları yazılmış ve çalışma süreleri ölçülmüştür. Ölçümler, Windows, Linux ve bulut sunucu sistemlerinde ayrı ayrı gerçekleştirilmiştir.

Scrapy ‘de yazılan kodlar, diğer kütüphanelere göre daha fazla satır içerebilir, ancak daha yapılandırılmış bir yaklaşım sunar. BeautifulSoup, basit ve statik siteler için hızlı sonuçlar verirken, dinamik içerikler için Selenium ve Scrapy daha etkilidir. İşletim sistemleri kodların çalışma sürelerini fazla etkilemezken, bulut sunucudaki kodlar biraz daha yavaş çalışmıştır.

BeautifulSoup, veri kazımaya yeni başlayanlar için basit ve kullanıcı dostu bir seçenektir. Statik veya az dinamik siteler için uygundur. Selenium, dinamik içeriğe sahip JavaScript içeriği yoğun olan sitelerden veri çekmek ve kullanıcı etkileşimlerini simüle etmek için güçlü bir araçtır. Scrapy ise, büyük ölçekli projeler ve karmaşık siteler için yüksek performanslı bir araçtır. Bilimsel çalışmalar ve projeler için doğru kütüphanenin seçilmesi, web kazıma tekniklerinin verimliliği ve başarısı için önemlidir. Bu tez, web kazıma kullanılarak yapılan bilimsel çalışmalara rehberlik edecek nitelikte olup, gelecekteki araştırmalar için de referans niteliği taşımaktadır.

2025, xiii + 103 sayfa

Anahtar Kelimeler: Web kazıma, Veri toplama, Python, Selenium, BeautifulSoup, Scrapy.

ABSTRACT

M.Sc.Thesis

EXAMINATION OF PYTHON LIBRARIES
USED IN WEB SCRAPING

Murat Ali ÖZ

Afyon Kocatepe University

Graduate School of Natural andAppliedSciences

Department of Internet and Information Technologies Management

Supervisor: Asst. Prof. Dr. Naim KARASEKRETER

Data has become a crucial asset for organizations today, with the internet being the primary source of data. Web scraping allows for the automatic extraction of structured data from HTML documents. This data can be used for various purposes such as collecting product and service information, price comparison, obtaining contact details, tracking news and blogs, and analyzing customer feedback. The web scraping process refers to the automatic data transfer between a web page and a web robot.

In this thesis, Python libraries used in web scraping, such as BeautifulSoup, Selenium, and Scrapy, have been examined in detail, and their suitability for different types of websites and purposes has been determined. Web scraping is a technique for automatically collecting data from websites and is currently used in various fields such as data analysis, machine learning, and data mining.

HTML is used to define the structure of web pages, while CSS is used to define style properties. Python is a high-level programming language used in many fields, including web development, data science, and artificial intelligence. BeautifulSoup, Selenium, and Scrapy are the main Python libraries used in web scraping.

Using the specified libraries (BeautifulSoup, Selenium, Scrapy), data scraping codes were written for various websites (real estate listing site, news site, online market, and

e-commerce site), and their execution times were measured. Measurements were conducted separately on Windows, Linux, and cloud server systems.

Codes written in Scrapy may contain more lines compared to other libraries but offer a more structured approach. BeautifulSoup provides quick results for simple and static sites, while Selenium and Scrapy are more effective for dynamic content. Operating systems do not significantly affect the execution times of the codes, but the codes on the cloud server ran slightly slower.

BeautifulSoup is a simple and user-friendly option for beginners in data scraping, suitable for static or minimally dynamic sites. Selenium is a powerful tool for extracting data from sites with dynamic content and heavy JavaScript, capable of simulating user interactions. Scrapy, on the other hand, is a high-performance tool for large-scale projects and complex sites. Selecting the right library for scientific studies and projects is crucial for the efficiency and success of web scraping techniques. This thesis serves as a guide for scientific studies conducted using web scraping and acts as a reference for future research..

2025, xiii + 103 pages

Keywords: Web scraping, Data collection, Python, Selenium, BeautifulSoup, Scrapy.

TEŞEKKÜR

Bu araştırmanın konusu, deneysel çalışmaların yönlendirilmesi, sonuçların değerlendirilmesi ve yazımı aşamasında yapmış olduğu büyük katkılarından dolayı tez danışmanım Sayın Dr. Ögr. Üyesi Naim KARASEKRETER'e, araştırma ve yazım süresince yardımlarını esirgemeyen her konuda öneri ve eleştirileriyle yardımlarını gördüğüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Bu araştırma boyunca maddi ve manevi desteklerinden dolayı aileme teşekkür ederim.

Murat Ali ÖZ

Afyonkarahisar 2025

İÇİNDEKİLER DİZİNİ

Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar	
AI	Artificial Intelligence (Yapay Zeka)
API	Application Programming Interface (Uygulama Programlama Arayüzü)
CCNA	Cisco Certified Network Associate (Cisco Sertifikalı Ağ Uzmanı)
CSS	Cascading Style Sheets (Basamaklı Stil Sayfaları)
CSV	Comma-Separated Values (Virgülle Ayrılmış Değerler)
FTP	File Transfer Protocol (Dosya Transfer Protokolü)
HTML	Hyper Text Markup Language (Hiper Metin İşaretleme Dili)
IDE	Integrated Development Environment (Entegre Geliştirme Ortamı)
JSON	JavaScript Object Notation (JavaScript Nesne Notasyonu)
NLP	Natural Language Processing (Doğal Dil İşleme)
OSI	Open Systems Interconnection (Açık Sistemler Ara Bağlantısı)
PCA	Principal Component Analysis (Temel Bileşenler Analizi)
PIP	Python Package Installer (Python Paket Yükleyicisi)
RAM	Random Access Memory (Rastgele Erişimli Bellek)
S3	Amazon Simple Storage Service (Amazon Basit Depolama Servisi)
URL	Uniform Resource Locator (Tekdüzen Kaynak Konumlayıcısı)
VM	Virtual Machine (Sanal Makine)
WEB	World Wide Web (Dünya Çapında Ağ)
XML	Extensible Markup Language (Genişletilebilir İşaretleme Dili)
XPATH	XML Path Language (XML Yol Dili)

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

RESİMLER DİZİNİ

Sayfa

1. GİRİŞ

Günümüzde veri, kuruluşlar için önemli bir varlık haline gelmiş ve internet, bu verilerin başlıca kaynağı olmuştur. Bu durum, web sitelerindeki bilgilerin otomatik olarak toplanmasını sağlayan web kazıma (web scraping) tekniklerinin önemini artırmıştır. Web kazıma, web sayfalarından yapılandırılmış verilerin otomatik olarak çekilmesi süreci olup, elde edilen bu veriler, analiz, makine öğrenimi, veri madenciliği gibi birçok farklı alanda kullanılmaktadır.

Web kazıma, yalnızca büyük veri analizi için değil, aynı zamanda rekabet analizi, pazar araştırması ve fiyat takibi gibi çeşitli iş süreçleri için de kritik bir araç haline gelmiştir. İşletmeler, web kazıma sayesinde büyük miktarda veriyi hızlı ve kolay bir şekilde toplayarak, veri odaklı karar alma yeteneklerini artırabilirler. Örneğin, e-ticaret sitelerindeki müşteri yorumları, emlak verileri, kargo firmaları hakkındaki şikayetler ve hatta sosyal medya mesajları gibi çeşitli veriler web kazıma teknikleri ile toplanabilmekte ve analiz edilebilmektedir.

Bu tez çalışmasında, öncelikle web kazıma kavramı, amaçları ve temel teknolojileri açıklanacaktır. Daha sonra, incelenen Python kütüphanelerinin (BeautifulSoup, Selenium, Scrapy) özellikleri, avantajları, dezavantajları ve kullanım alanları detaylı olarak ele alınacaktır. Bu kütüphaneler, web sitelerinden veri toplama süreçlerinde farklı avantaj ve dezavantajlara sahiptirler ve farklı türdeki web siteleri için daha uygun çözümler sunarlar.

İncelenen Python kütüphaneleri ile, farklı web sitelerinden veri çekmek için yazılan kodlar ve bu kodların çalışma süreleri karşılaştırılacaktır. Bu karşılaştırmalar sonucunda, hangi kütüphanenin hangi tür web siteleri için daha uygun olduğu ve hangi durumlarda daha verimli çalıştığı ortaya konulacaktır.

2. LİTERATÜR BİLGİLERİ

2.1 Veri ve Web Kazımanın Tanımları

Veri, gerçekleri ve olayları temsil eden sayılar, metinler, görüntüler veya sesler gibi bilgilerdir. Genellikle bir dizi sayı veya metin olarak depolanır ve analiz edilebilir. Veri, günümüzde hemen hemen tüm organizasyonların varlıkları ve büyümeleri açısından oldukça önemlidir. İnternet, bireyler ve kuruluşlar için temel veri kaynağı haline gelmiştir (Almaqbali vd. 2019). Davenport ve Prusak (1998)'a göre, "Veri, sayılar, harfler veya semboller gibi herhangi bir biçimde saklanabilen ham bilgidir." Verilerin farklı türleri vardır. Örneğin, sayısal veriler sayılar veya sayısal değerler içerirken, metinsel veriler metin içeren verilerdir. Görüntülü veriler, görüntüler içeren verilerdir ve ses verileri, sesler içeren verilerdir. Bu farklı veri türleri, çeşitli analiz ve işlem süreçlerinde kullanılarak bilgiye dönüştürülebilir ve karar verme süreçlerinde önemli bir rol oynar.

Web kazıma, web sitelerinden otomatik olarak veri toplamak için kullanılan bir tekniktir. Bu veriler, analiz, makine öğrenimi veya veri madenciliği gibi işlemler için kullanılabilir. Web kazıma, genellikle bir web sitesinin HTML koduna erişmeyi, ilgili bilgileri çıkarmayı ve ardından bu bilgileri yapılandırılmış bir formata dönüştürmeyi içerir. Araştırmacıların, veri analistlerinin, işletmelerin, bilim insanlarının büyük miktarda veriyi hızlı ve kolay bir şekilde toplamasına olanak tanıyan bu teknik, veri odaklı karar verme yeteneğini artırabilir. Web kazıma, HTML dokümanlarından yapılandırılmış verileri çıkarmamıza olanak tanır ve verilerin JSON (JavaScript Object Notation - JavaScript Nesne Notasyonu) veya XML (Extensible Markup Language - Genişletilebilir İşaretleme Dili) gibi makine tarafından okunabilir formatta sağlanmadığı durumlarda son derece kullanışlıdır (Khder 2021).

İnternetin popülaritesi, her gün oluşturulan veri miktarının hızla artmasına neden olmuş ve bu da web sitelerinin kazınması ihtiyacını doğurmuştur. Arama motoru mühendisleri, Google gibi ilk tanınmış web kazıyıcıları yaratmışlardır. Bu kazıyıcılar, tüm İnternet üzerinde arama yapar, her web sayfasını tarar, bilgi çıkarır ve aranabilir bir dizin

oluşturur (Lawson 2015). Web kazıma, yapılandırılmamış web verilerini merkezi bir veri tabanında veya elektronik tabloda saklanabilen ve analiz edilebilen yapılandırılmış verilere dönüştürmek için kullanılan bir tekniktir (Sirisuriya 2015). Verileri yapılandırılmış bir formatta dosyalarda veya veri tabanında kullanarak ve saklayarak bir web sitesinden veri çıkarmak için kullanılır. Web kazıma, verileri aramak ve depolamak için belirli aralıklarla web sitesini ziyaret eden kişilerin manuel işlemini otomatikleştirir. Verilerin manuel olarak kopyalanıp dosyalara yapıştırılması zaman alıcı ve sıkıcı bir iştir. Otomatik web kazıma araçları aynı işi çok daha kısa sürede yapar (Pillai ve Amin 2020).

Web kazıma, web'den verileri otomatik bir şekilde indirmek, ayrıştırmak ve düzenlemek için bir aracının oluşturulması olarak tanımlanabilir. Bir insan son kullanıcının, bir web tarayıcısında tıklayıp ilginç bölümleri, örneğin bir elektronik tabloya kopyalayıp yapıştırması yerine, web kazıma bu görevi çok daha hızlı ve doğru bir şekilde yürütebilecek bir bilgisayar programına aktarır (VandenBroucke ve Baesens 2018). Büyük Veri senaryoları için, bilgilerin manuel olarak toplanması hem mümkün değildir hem de kopyalama/yapıştırma hatalarına açıktır. Web kazıma, genellikle bir web sayfası ile web robotu, kazıyıcı veya tarayıcı olarak adlandırılan bir araç seti kullanılarak uygulanan yapılandırılmış bir veri formatı arasındaki otomatik veri aktarımı sürecini ifade eder (Gheorghe vd. 2018). Herkes bir web kazıyıcı oluşturabilir. Google veya Bing'e yeni bir rakip olabilecek büyük bir uygulamayı hayata geçirmeye çalışacak kişi sayısı az olacaktır. Ancak kapsam bir veya iki web sayfasına daraltılabilir ve bilgiler yapılandırılmış bir şekilde çıkarılabilirse, sonuçlar bir veri tabanına veya yapılandırılmış dosyaya örneğin JSON, CSV (Comma-Separated Values - Virgülle Ayrılmış Değerler), XML ve Excel sayfalarına kolaylıkla aktarılabilir (Hajba 2018).

2.2 Web Kazıma İle İlgili Yapılan Çalışmalar

Web kazıma, internet sitelerinde bulunan ürün, hizmet, iletişim bilgilerini toplamak, fiyat karşılaştırması amacıyla veri elde etmek, haber veya blog gönderilerini derlemek, sitelerde bulunan kullanıcı geri bildirimlerini arşivlemek gibi amaçlar için kullanılabilir.

Web tarayıcınızın penceresinden web sayfasını sayfa sayfa görüntülemek yerine, otomatik olarak zengin bir veri seti toplayabilmek güzel olmaz mıydı? Bu tam olarak web kazımanın resme girdiği yerdir (VandenBroucke ve Baesens 2018).

Özen vd. (2022) 'nin yaptıkları araştırmada, Kapadokya'da faaliyet gösteren ve yöresel lezzetler sunan turistler tarafından büyük ilgi gören, kadın girişimci restoranları hakkında yapılan yorumlar incelenmiştir. Google Maps'te yer alan 4 restorana ait 139 İngilizce yorum analiz edilmiş ve turistlerin tercih nedenleri ortaya konulmuştur. Araştırmada 4 restorana ait 139 yorum analiz edilmiştir. Veri seti web kazıma tekniği kullanılarak toplanmıştır. Metin madenciliği ve birliktelik teknikleri kullanılarak analiz yapılmıştır.

Yüksel (2023) tarafından yapılan araştırmada e-ticaret platformlarında bulunan müşteri yorumları incelenmiştir. Bu yorumlar, ürünün ve iletişim sürecinin iyileştirilmesi, pazarlama stratejilerinin geliştirilmesi açısından önemlidir. 6714 müşteri şikâyeti analiz edilerek dört temel kategori belirlenmiştir: ürün materyali, kargo, paketleme ve beden. Veri web kazıma metodu ile elde edilmiştir. Elde edilen veri içerik analizine tabi tutulmuştur.

Kuyucuk ve Çallı (2022) çalışmalarında kargo şirketlerinin Kovid-19 salgını sırasında son kilometre faaliyetlerini nasıl yönettiklerini araştırıyor ve olumsuz sonuçlara çözüm önerileri sunuyor. Çalışmada kullanılan veriler, Şubat 2020 ile Eylül 2021 tarihleri arasında sikayetvar.com üzerinden kargo firmaları hakkında yapılan şikayetleri içermekte olup Python dili ve Scrapy modülü web scraping yöntemleri kullanılarak toplanmıştır. Konulara göre elde edilen eğitim verilerinin değerlendirilmesine dayalı olarak şikayetlerin kategorize edilmesi için çok etiketli sınıflandırma algoritmaları kullanıldı. Sonuçlar, en çok şikâyet edilen konuların paket teslimatı ile ilgili konular olduğunu ve önemli bir kısmının da gecikme sorunları olduğunu gösterdi.

Üzümcü ve Eligüzel (2023) çalışmalarında, Gaziantep şehrinde potansiyel bir alıcının mali durumu ve belirli mülk özellikleri göz önüne alındığında, birinin hangi mahallede yaşayabileceğini tahmin etmeyi amaçlamışlardır. Web sitesinden emlak verilerini

toplamak için, web kazıma kullanılmıştır. Veriler toplandıktan sonra bir evin mahallesini tahmin etmek; karar ağaçları, rastgele orman ve ekstra ağaçları içeren makine öğrenimi algoritmaları kullanılarak yapılmıştır. Sonuçlar, tüm algoritmaların %80'in üzerinde performans doğruluğuyla iyi sonuçlar ürettiğini göstermektedir. Ancak bu algoritmalar arasında karar ağacı sınıflandırması en iyi performansı sunmaktadır.

Yılmaz ve Selvi (2023) çalışmalarında web scraping teknikleri kullanılarak toplanan otomobil satış verilerinin makine öğrenmesi algoritmaları kullanılarak analiz edilmesini ve fiyat tahmin modelinin oluşturulmasını amaçlamıştır. Analiz için gerekli veriler web kazıma yoluyla elde edilmiş, Selenium ve BeautifulSoup kullanılarak toplanmıştır. Çeşitli veri ön işleme adımları uygulanarak analize hazırlanmıştır. Özellik seçimi ve boyut küçültme için Lasso regresyon ve PCA (Principal Component Analysis - Temel Bileşenler Analizi) analizi, hiper parametre ayarlama için GridSearchCV yöntemi kullanılmıştır. Sonuçlar makine öğrenmesi algoritmaları ile değerlendirilmiştir.

Agüero-Torales vd. (2019) çalışmalarında turizm sektörü ürün ve hizmetlerini, insanların TripAdvisor.com, Booking.com gibi seyahat sitelerinde ve buna benzer diğer platformlarda sıklıkla yazdıkları yorumları incelemişlerdir. Bu yorumlar, hangi restoranların rezerve edileceği karar verme süreci üzerinde derin bir etkiye sahiptir. Bu sosyal medya verilerinin (TripAd-visor.com) kapsamlı analizi için web kazıma yapılmış ve bulut tabanlı bir yazılım aracı kullanılmıştır. Yazılım dili olarak Python ve web tarama kütüphanesi olarak Scrapy kullanılmıştır.

Seliverstov vd. (2020) çalışmalarında, Web'de yayınlanan incelemelere göre Kuzeybatı Federal Bölgesi'ndeki trafik güvenliğini analiz etmişlerdir. Analizlerinde, duygu sınıflandırıcısına dayanan bir otomatik inceleme sınıflandırma sistemi geliştirmişlerdir. Kullanıcıların yol durumuna ilişkin bilgilerini içeren internet kaynaklarını analiz edilmiştir. Verilerin toplanması için web kazıma kullanılmıştır. Web kazıma için Scrapy ve Python 3'te yazılmış bir web tarayıcısı tasarlanmıştır.

Pillai ve Amin (2019) çalışmalarında, iş portallarından bilgi taramasıyla iş tanımlarından bilgi çıkararak Hindistan BT sektörünün gereksinimlerini analiz etmek

için basit bir metodoloji geliştirdiler. Analizi gerçekleştirmek için piyasa değerine göre en iyi 10 BT şirketi seçildi. Web sitelerinden Java, CCNA (Cisco Certified Network Associate - Cisco Sertifikalı Ağ Uzmanı), SAP, Agile gibi her anahtar kelimeyi içeren iş ilanlarının sayısı çıkarıldı. Her becerinin popülerliği hesaplandı. Her şirketin iş arama sitesinden ayrı ayrı iş bağlantıları çıkarıldı ve ardından her iş sayfası ayrı ayrı tarandı. İşin adı, işin son tarihi, iş yeri, iş tanımı, nitelikler, deneyim vb. ayrıntılar çıkarılıp kaydedildi. Web kazıma işlemi için Selenium ve Python'un BeautifulSoup modülleri kullanılmıştır.

Plotnikov vd. (2020) çalışmalarında Post Bank ile ilgili bir dizi müşteri geri bildirim verisini incelemiştir. Veri kaynağı olarak banki.ru web sitesi seçilmiş ve web kazıma için BeautifulSoup kullanılmıştır. 16.659 geri bildirim verisi derlenmiştir. Veri seti, banka müşterilerinin bankacılık hizmetlerine olan güven düzeyini değerlendirmek için kullanılmıştır.

Altamimi ve Alayba (2023) çalışmalarında, 01/01/2021 ile 31/12/2021 arasındaki bir yıllık dönemde toplanan Arapça haber makalelerine dayanan modern standart Arapça veri seti hazırlamışlardır. Toplamda 12 Arapça haber sitesinden 500,000'in üzerinde makale toplandı; bunların seçimi spor, ekonomi, yerel haberler, siyaset, teknoloji, turizm gibi çeşitli konulara göre yapıldı. Bu veri kümesinin elde edilmesi için web kazıma yöntemi kullanıldı. Web kazımada Python requests ve BeautifulSoup kütüphaneleri kullanılmıştır.

Syed vd. (2023) çalışmalarında, web sitelerinden havayolları şirketlerine ait müşteri yorumlarını toplayarak değerlendirmişlerdir. Veriler, çeşitli havayolu inceleme web sayfalarının URL'si (Uniform Resource Locator - Tekdüzen Kaynak Konumlayıcısı) belirtilerek ve inceleme, başlık, derecelendirme, tavsiye değeri vb. gibi unsurlar bulunarak toplanmiştir. Web kazıma işlemi, Python requests ve BeautifulSoup kütüphaneleri kullanılarak yapılmıştır.

Yapılan bir çalışmada, Twitter'dan, bloglardan ve diğer sosyal medyadan web kazıma ile alınan mesajlar, depresyon ve intihar düşüncesi kalıplarını belirlemeye yönelik

öngörücü bir model oluşturmak için kullanılan bir veri seti oluşturmak üzere kullanıldı. (İnt. Kyn.1)

Cavallo ve Rigobon (2016), “Milyar Fiyat Projesi: Ölçüm ve Araştırma için Çevrimiçi Fiyatların Kullanılması" başlıklı çalışmalarında, birden fazla ülke için sağlam bir günlük fiyat endeksi oluşturmak amacıyla kullanılan çevrimiçi fiyat bilgilerinin bir veri kümesini toplamak için web kazıma kullanmıştır. Proje kapsamında 2010 yılında her gün 50 ülkeden 300 satıcıdan 5 milyon fiyat bilgisi bir fiyat endeksi oluşturmak amacıyla toplanıyordu.

Brown vd. (2024) yaptıkları çalışmada, web kazıma süreçlerinde dikkate alınması gereken yasal, etik, kurumsal ve bilimsel faktörleri incelemişlerdir. Özellikle, büyük metin veri kümelerine dayanan üretken yapay zekanın artan önemiyle birlikte, platformların resmi kanallar aracılığıyla verilere erişimi kısıtlaması sonucu araştırmacıların web kazımaya daha fazla yönelmesiyle ortaya çıkan yeni zorlukları ele almışlardır. Makale, ABD merkezli araştırmacılara yönelik kapsamlı bir çerçeve sunarak, web kazımanın nasıl bilimsel olarak meşru ve etik bir şekilde yapılabileceğine dair öneriler sunmaktadır. Web kazıma, veri toplama ve araştırma süreçlerinde önemli bir araç olarak görülmekte ve bu sürecin yasal ve etik boyutları üzerinde durulmaktadır.

Galvez-Hernandez vd. (2024) yaptıkları çalışmada, devlet web sitelerinden elde edilen kapsamlı metin verilerini, analiz edilebilir veri setlerine dönüştürmek için 8 adımlı bir yöntem sunmuştur. Yöntem, web kazıma, metin madenciliği ve mekansal örtüşme analizi tekniklerini birleştirir. Özellikle, bu yöntem, sağlık ve sosyal araştırmacılar tarafından yaşlıların sosyal bağlantılarını artırmayı hedefleyen sağlık varlıklarının (örneğin, aktiviteler ve programlar) belirlenmesinde kullanılır. Web kazıma, bu makalede, doğrudan devlet web sitelerinden veri çekme ve bunları analiz edilebilir formatlara dönüştürme aracı olarak kritik bir rol oynar.

Naing vd. (2024) yaptıkları çalışmada, internet üzerinden referans makaleleri otomatik olarak toplamak için web kazıma kullanan bir sistem geliştirmişlerdir. Google Scholar gibi mevcut araçların yetersiz kaldığı durumlarda, bu sistem, Selenium web kazıma

yazılımını kullanarak internetten veri toplar ve BERT modelini kullanarak arama hedefleriyle benzerlikleri inceler. Web kazıma, burada araştırmacılar için referans makaleleri toplama sürecini otomatikleştirme ve daha verimli hale getirme amacı taşır. Python Flask ve Angular gibi teknolojilerle entegre edilerek kullanıcı dostu bir sistem sunulmaktadır.

Pant vd. (2024) yaptıkları çalışmada, web kazıma teknolojilerini incelemişlerdir. Özellikle, BeautifulSoup kütüphanesi kullanarak web sayfalarından veri çekme yöntemi ele alınmaktadır.Web kazıma, bu çalışmada, web sayfalarından bilgi çıkarma ve analiz etme süreçlerinde kullanılan bir temel teknik olarak değerlendirilmektedir. Çalışmada, farklı web kazıma araçları ve yaklaşımları da karşılaştırılmaktadır. Farklı araçların birbirlerine göre avantaj ve dezavantajları belirtilmiştir.

Meyberg vd. (2024) yaptıkları çalışmada, Berlin'deki kiralık dairelerin fiyatlarını tahmin etmek için web kazıma yöntemini kullanmışlardır. İki farklı çevrimiçi emlak portalından daire listeleme verileri toplanarak, dairelerin özellikleri ve şehir içindeki konumlarına göre kira fiyatları tahmin edilmiştir. Eksik değerler çoklu imputasyon yöntemiyle ele alınır ve tahmin modeli yarı parametrik bir yaklaşım kullanır. Web kazıma, bu çalışmada emlak piyasası verilerini toplama ve analiz etme amacıyla kullanılmıştır.

Wibawa ve Dewa (2024) yaptıkları çalışmada, Endonezya'daki pansiyon piyasası trendlerini analiz etmek için web kazıma ve iş zekasını kullanmışlardır. E-ticaret platformlarından 658 veri noktası çekilerek, pansiyonların özellikleri ve fiyatları arasındaki ilişkiler incelenmiştir. Çoklu Lojistik Regresyon kullanılarak, pansiyon fiyatlarını etkileyen faktörler belirlenir. Web kazıma, bu araştırmada e-ticaret sitelerinden veri toplama ve pazar trendlerini analiz etme amacıyla kritik bir rol oynamıştır.

Farias vd. (2024) yaptıkları çalışmada, Python kullanarak web kazımanın bilimsel araştırmalarda teorik çerçeve oluşturma sürecinde nasıl etkili bir araç olduğunu incelemişlerdir. BeautifulSoup ve Requests kütüphaneleri kullanılarak, web sitelerinden

veri çekme süreçleri otomatikleştirilmiştir. Ayrıca, Tkinter kütüphanesi ile kullanıcı dostu bir grafik arayüzü oluşturulmuş ve Pandas kütüphanesi ile veriler Excel formatında kaydedilmiştir. Web kazıma, burada akademik makalelerden veri toplama, analiz etme ve teorik çerçeve oluşturma süreçlerini destekleyen bir yöntem olarak vurgulanmaktadır.

Goulas ve Karamitros (2024) yaptıkları çalışmada, cerrahi araştırmalarda web kazımanın yenilikleri nasıl tetiklediğini ve hasta sonuçlarını nasıl iyileştirdiğini analiz etmişlerdir. Özellikle, uluslararası cerrahi araştırma iş birliklerine dair web kazıma ile elde edilen bilgiler paylaşılır. Web kazıma, bu alanda büyük veri analizi yapma, trendleri belirleme ve iş birliklerini ortaya çıkarma gibi amaçlarla kullanılır.

Zoszak vd. (2024) yaptıkları çalışmada, MS hastaları için çevrimiçi beslenme bilgilerini toplamak amacıyla web kazıma tekniklerini kullanmışlardır. Web sayfalarındaki içerikler arşivlenmiş, beslenme ile ilgili bilgiler manuel olarak çıkarılmış ve web siteleri türlerine göre sınıflandırılmıştır. Web kazıma, bu çalışmada, belirli bir hasta grubu için internetteki sağlık bilgilerini toplama ve analiz etme amacıyla kullanılmıştır.

Prasetyani vd. (2024) yaptıkları çalışmada, Google Haritalar'dan elde edilen popüler zaman bilgileri ve web kazıma tekniklerini kullanarak otoyol dinlenme alanlarının kullanım kalıplarını analiz etmişlerdir. Web kazıma, bu araştırmada, coğrafi konum verilerini ve kullanıcı davranışlarını elde etmek için kullanılmıştır.

Zhekova ve Yumer (2024) yaptıkları çalışmada, tarım web sitelerinden bilgi çıkarmak için bir JavaScript web kazıma aracı geliştirmişlerdir. Bu araç, dinamik web sayfalarından veri çekmek ve bunları analiz etmek için kullanılmıştır. Web kazıma, bu çalışmada, tarım sektöründeki web sitelerinden veri toplama ve bu verileri analiz ederek, sektörel kararlar için bir temel oluşturma amacını taşımaktadır.

Pandey vd. (2024) yaptıkları çalışmada, Springer ve Nature gibi bilimsel veri tabanlarından web kazıma yoluyla bilgi toplamayı incelemişlerdir. Basra Üniversitesi için akademik veri elde etmek amacıyla, web kazıma yöntemleri uygulanmış ve elde

edilen veriler analiz edilmiştir. Çalışmada veri toplama, ön işleme ve analiz süreçlerine odaklanılmıştır. Web kazıma, akademik araştırmalar için bilimsel makalelerden veri toplama ve analiz etme aracı olarak kullanılmıştır.

Chandrawat vd. (2024) yaptıkları çalışmada, e-ticaret alanında karar verme süreçlerini optimize etmek için web kazıma tekniklerinin potansiyelini incelemişlerdir. Gelişmiş web kazıma yöntemlerinden yararlanarak, çeşitli e-ticaret alanlarından veri çıkaran ve analiz eden bir çerçeve geliştirilmiştir. Bu veriler, ürün bilgileri ve fiyatlandırma stratejileri gibi önemli unsurları içermektedir. Araştırma, etik ve sorumlu bir şekilde kullanıldığında web kazımanın, e-ticaret işletmelerinin rekabet avantajı elde etmesi ve genel karar verme süreçlerini geliştirmesi için güçlü bir araç haline gelebileceğini göstermeyi amaçlamaktadır.

Al-Madhhachi ve Mahmood (2024) yaptıkları çalışmada, çevrimiçi kazıma tekniklerini kullanarak bilimsel veri çıkarma alanını, özellikle Basra Üniversitesi'ndeki Springer ve Nature arşivlerine odaklanarak incelemişlerdir. Web kazımanın teorik temellerini açıklamayı ve çevrimiçi kaynaklardan yapılandırılmış veri elde etmenin önemini vurgulamışlardır. Çalışma, dinamik içerik, captcha'lar ve IP engelleme gibi birçok sorunu ele almakta ve bu engeller için yenilikçi çözümler önermektedir. Üniversitenin araştırma hedefleri, titiz bir veri toplama ve hazırlama süreciyle dikkatlice oluşturulmuş zengin bir veri seti ile desteklenmiştir. Sonuçlar, web kazımanın etkinliğini ve ön işleme süreçlerinin önemli etkisini vurgulamaktadır. Bu çalışma, mevcut akademik araştırma metodolojisini geliştirmekle kalmayıp, aynı zamanda Basra Üniversitesi'nin veri odaklı ve etkili akademik çalışmalar yürütme çabalarını da ileriye taşımaktadır.

Kumar vd. (2024) yaptıkları çalışmada, mobil telefon fiyatlarının tüketiciler arasındaki popülerliğini ve piyasadaki rekabetçi konumlarını belirlemedeki önemli rolünü incelemişlerdir. Müşteriler, bir mobil telefonun özelliklerini ve tasarımını değerlendirirken bütçelerini göz önünde bulundururlar, bu nedenle fiyatı önceden tahmin etmek, ihtiyaç ve beklentilerini karşılamak için önemlidir. Bu araştırmanın temel amacı, mobil telefonların gerçek piyasa değerini belirlemektir. Araştırmada, Flipkart

web sitesinden web kazıma yöntemi kullanılarak veri seti çıkarılmıştırve analizi yapılmıştır.

Chataut vd. (2024) yaptıkları çalışmada, haber web sitelerinden multimedya haber içeriği oluşturmak için tamamen otomatik bir sistem ortaya koymuşlardır. Sistem, web kazıma, görsel-işitsel içerik üretimi ve yapay zekayı birleştirerek yayınlanmaya hazır haber içeriği sunar. Sistem farklı kaynaklardan haberleri web kazıma yoluyla elde eder.

Abdillah vd. (2024) yaptıkları çalışmada, bilimsel makalelerin otomatik olarak indirilmesi için web kazıma yeteneklerine sahip bir Telegram botu tasarlamışlardır. Bu bot, web kazıma teknikleri ile belirli bilimsel yayın kaynaklarından makaleleri otomatik olarak indirir. Web kazıma, bu çalışmada, bilimsel makale toplama süreçlerini otomatikleştirmek amacıyla kullanılmaktadır.

Sharma ve Borkar (2024) yaptıkları çalışmada, dinamik web sayfalarından veri toplamak için farklı web kazıma stratejilerini karşılaştırır. BeautifulSoup, LXML ve Selenium gibi kütüphanelerin performansları değerlendirilir ve Selenium'un daha verimli olduğu sonucuna varılır. Web kazıma, bu çalışmada dinamik web sayfalarından veri toplama süreçlerinde farklı yaklaşımların verimliliğini belirlemek amacıyla kullanılır.

Rodrigo (2024) çalışmasında, web kazıma ve sinir ağlarını kullanarak uçuş fiyatlarını tahmin etmeyi amaçlayan bir web sitesi geliştirme projesini gerçekleştirmiştir. Web kazıma, bu proje kapsamında uçuş fiyatı verilerini toplama amacı ile kullanılmaktadır.

Anbu vd. (2024) yaptıkları çalışmada, IMDb'nin tüm zamanların en iyi 50 filmi listesinden veri çıkarmayı amaçlamışlardır. Bu amaçla, Python paket kütüphanesinde bulunan BeautifulSoup paketi kullanılarak web kazıma işlemi gerçekleştirilmiştir. Bu ana hedef doğrultusunda, IMDb'nin resmi web sitesi temel alınmış ve Python dili, Numpy, Pandas, Requests ve BeautifulSoup gibi Python paket kütüphanesindeki yerleşik fonksiyonlar kullanılarak veri kazıma işlemi yapılmıştır.

Akbar ve Ramadhani (2024) yaptıkları çalışmada, kullanıcıların internet üzerinde ürün verilerini arayıp toplamasına yardımcı olabilecek bir uygulama geliştirmişlerdir. Bu çalışmanın amacı, BeautifulSoup4 (Bs4) yöntemi kullanılarak Tokopedia pazar yeri web sayfasından web kazıma sonuçlarına dayalı bir ürün veri analizi oluşturmaktır. Web sitesi tasarımı, Python'un streamlit modülü kullanılarak yapılmıştır. Bu çalışmanın sonuçları, BeautifulSoup yöntemi kullanılarak Tokopedia web sayfasından kazınan verilerin analizini içermektedir.

2.3 Literatür Değerlendirmesi

Taranan literatür, web kazıma tekniklerinin ne kadar geniş bir uygulama alanına sahip olduğunu göstermektedir. Bu teknikler, çeşitli sektörlerdeki araştırmalar için veri toplamada kritik bir rol oynamaktadır. E-ticaret alanında, müşteri yorumlarının analizi ve fiyat karşılaştırması için web kazıma yaygın olarak kullanılmaktadır. Örneğin, bir çalışmada e-ticaret platformlarındaki müşteri şikayetleri incelenerek ürün ve hizmetlerin iyileştirilmesi hedeflenmiştir. Turizm sektöründe, web kazıma, turistlerin yorumlarını analiz ederek restoran tercihleri gibi konularda bilgi sağlamaktadır. Ayrıca, konaklama ve seyahat sitelerindeki veriler, turist davranışlarını anlamak için kullanılmaktadır. Kargo ve lojistik alanında, web kazıma, müşteri şikayetlerini analiz ederek hizmet kalitesini artırmaya yönelik çalışmalarda kullanılmaktadır. Kovid-19 salgını sırasında kargo şirketlerinin performansını değerlendiren çalışmalar bu duruma örnektir. Emlak sektöründe, web kazıma, emlak verilerini toplayarak fiyat tahminleri yapmak ve potansiyel alıcıların hangi mahallelerde yaşayabileceğini tahmin etmek için kullanılmaktadır. Bu çalışmalar, makine öğrenimi algoritmaları ile birleştirilerek daha doğru analizler yapılabilmektedir. Akademik araştırmalarda, web kazıma, bilimsel makaleleri toplamak, referansları belirlemek ve hatta bilimsel veri tabanlarından veri elde etmek için kullanılmaktadır. Bu, araştırmacıların literatür tarama süreçlerini hızlandırmasına yardımcı olmaktadır. Finans alanında, web kazıma ile hisse senedi fiyatları gibi veriler elde edilerek yatırım kararlarına destek olunmaktadır . Haber ve medya sektöründe web kazıma, haber sitelerinden içerik toplama, analiz etme ve hatta otomatik haber içerikleri üretmek için kullanılmaktadır. Sosyal medya platformlarından toplanan veriler de, depresyon ve intihar gibi konularda kalıpları belirlemek için

kullanılabilmektedir. Web kazıma sadece veri toplama aracı olarak değil, aynı zamanda yasal, etik, kurumsal ve bilimsel açılardan da incelenmektedir. Özellikle, büyük veri kümeleriyle çalışan araştırmacılar için bu konuların önemi vurgulanmaktadır.

Çalışmalarda Python programlama dili ve bu dilin kütüphaneleri sıkça kullanılmıştır. Veri ön işleme ve analiz için de çeşitli makine öğrenimi algoritmaları kullanılmaktadır. Web kazıma sürecinde, HTML ve CSS bilgisi önemli bir rol oynamaktadır. Verilerin doğru bir şekilde tanımlanması ve çıkarılması için bu teknolojilerin bilinmesi gerekmektedir.

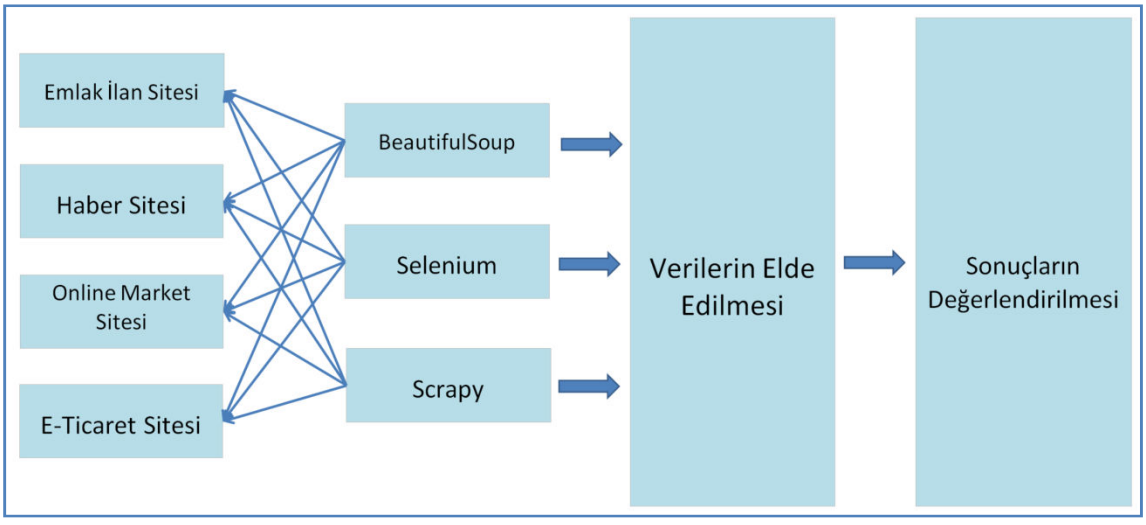
Bu tez çalışması, literatürdeki web kazıma çalışmalarına, Python kütüphanelerinin (BeautifulSoup, Selenium, Scrapy) karşılaştırmalı bir analizini yaparak katkıda bulunmayı amaçlamaktadır. Literatürde web kazıma tekniklerinin farklı alanlardaki uygulamaları çokça incelenmiş olsa da, bu kütüphanelerin performanslarını farklı işletim sistemlerinde (Windows, Linux ve bulut sunucular) karşılaştıran çalışmalar sınırlıdır. Bu tez çalışması, hangi kütüphanenin hangi tür web siteleri için daha uygun olduğunu ve hangi koşullarda daha verimli çalıştığını ortaya koyarak, araştırmacılara pratik bir rehber sunacaktır. Ayrıca, kod uzunluğu, çalışma süresi gibi metrikler açısından farklı kütüphanelerin karşılaştırılması, veri toplama projelerinin daha etkin planlanmasına katkı sağlayacaktır.

3. MATERYAL ve METOT

Bilgi kazıyacağımız web sayfaları HTML kodlarından oluşur ve CSS ile şekillendirilir. Bu sebeple, bu iki teknoloji web kazımada sürekli olarak kullanılır. Bilgi kazımak için Python dili kullanılmıştır ve Python ile kod yazarken üç farklı kütüphane kullanılmıştır: BeautifulSoup, Selenium ve Scrapy. Kodlar işletim sisteminin kütüphanelerin çalışma süresine etkisini gözlemlemek amacıyla, kişisel bilgisayarda, Windows ve sanal makinede çalışan bir Linux işletim sistemlerinde çalıştırılmıştır. Veri merkezi internet bağlantısının çalışma süresine etkisi olup olmadığını tespit etmek amacıyla, Linux işletim sistemi kullanan bir bulut sunucuda da performans ölçülmüştür.

Bilgi kazınacak örnek siteler seçilmiş ve HTML ile CSS yapıları incelenmiştir. Bu sitelerden kazınacak bilgiler belirlenmiş ve buna yönelik kodlar yazılmıştır. Aynı bilgiler, her üç kütüphane kullanılarak ayrı ayrı kazınmıştır.

Yazılan bu kodların karşılaştırılması için kullanılan metrik; kodun karakter sayısı, satır sayısı ve ayrı ayrı platformlardaki hızıdır. Bu metrikler kullanılarak kütüphaneler arasında karşılaştırma yapılmıştır. Kütüphanelerin karşılaştırılması için izlenen aşamalar Şekil 3.1'de ifade edilmiştir.



Şekil 3.1 İncelemede takip edilen aşamalar

3.1 HTML ve CSS

HTML, web sayfaları üretmek için özel olarak geliştirilen bir işaretleme dilidir. Türkçeye Hiper Metin İşaretleme Dili olarak çevrilebilir. İnternetin temelini oluşturan ve bir web standardı olarak kabul edilen HTML, 90'lı yıllardan bu yana sürekli olarak gelişim göstermektedir. HTML, web sitelerinin yapısını tanımlamak için kullanılır ve metin, tablo, görsel ve link gibi her türlü nesne HTML etiketleri kullanılarak belirtilir. HTML, bir programlama dili olarak sınıflandırılamaz çünkü kendi başına yürütülebilir programlar oluşturamaz. Bunun yerine, işaretleme dili olarak sınıflandırılır. HTML, web sayfalarının yapısını belirlemek için etiketler kullanır ve tarayıcılara bu sayfaların nasıl görüntüleneceğini söyler. Temel amacı, yazı, görüntü, video gibi çeşitli verileri ve bunları içeren sayfaları birbirine basitçe bağlamak ve web tarayıcıları tarafından düzgün olarak görüntülenmesi için gerekli kuralları belirlemektir (İnt. Kyn.20).

Temel düzeyde bir web sayfası oluşturabilmek için HTML yeterlidir. Bu işaretleme dili kullanılarak web tarayıcılarının yorumlayabileceği dokümanlar oluşturulabilir. İnternetin ilk yıllarında HTML ile oluşturulmuş statik web sayfaları yeterli görülürken, günümüzde görsel açıdan zengin ve dinamik işleve sahip web siteleri oluşturmak amacıyla CSS stil dili ve JavaScript, PHP gibi programlama dilleri ile beraber kullanılmaktadır. (İnt. Kyn.19).

CSS, web sayfalarının stil ve görünümünü belirlemek için kullanılan bir stil dilidir. CSS, HTML'de kullanılan başlık, paragraf gibi elementlerin nasıl görüneceğini bildirir. Yazı tipi, renk, boyut, arka plan ve dokular gibi ögelerde arzu edilen değişiklikleri tanımlamanızı sağlar. CSS ile düşen karlar, uçan kuşlar, yükleniyor simgesi gibi animasyonlar yaparak web sitelerine şık detaylar eklemek mümkündür. Ayrıca, CSS kullanarak bir sitenin cep telefonu, masaüstü ve tablet gibi farklı cihazların ekranlarında nasıl görüneceğini kontrol etmek mümkündür. CSS, web sayfanızın her bir öğesine ayrı bir stil vermenizi sağlar ve web sayfanız üzerinde tümüyle kontrol sahibi olmanıza olanak vermektedir. (İnt. Kyn.21).

Web kazıma yaparken HTML ve CSS bilgisi oldukça önemlidir. HTML, web sayfalarının yapısını ve içeriklerin nasıl düzenlendiğini belirlerken, CSS bu içeriklerin

nasıl görüneceğini tanımlar. Web kazıma işlemi sırasında, hedef web sayfasının HTML yapısını anlamak, gerekli verileri doğru bir şekilde çıkarmak için kritik öneme sahiptir. CSS ise, belirli öğelerin stil ve konumlandırmasını anlamaya yardımcı olur, bu da kazıma işlemini daha verimli hale getirir. HTML ve CSS bilgisi, web kazıma sürecinde verilerin doğru bir şekilde tanımlanması ve çıkarılması için gereklidir. Bu bilgiler, kazıma işleminin doğruluğunu ve etkinliğini artırarak, elde edilen verilerin daha anlamlı ve kullanılabilir olmasını sağlar.

3.1.1 HTML Etiketleri

HTML, küçüktür (<) ve büyüktür (>) işaretleri arasına girilen HTML etiketleri kullanılarak yazılır. İşaretlenen metnin başını ve sonunu belirtmek için çoğunlukla çift olarak kullanılırlar. (Örnek: <h1>Selam</h1>). Fakat işaretlemek yerine metnin bir yerine bir işaret konacaksa, tek olarak da kullanılabilirler (Örnek: <img>). Bir web sitesinde en yaygın olarak kullanılan öğeler metin, görsel, linklerdir. En sık kullanılan HTML etiketlerine örnek olarak <html>, <title>, <h1>, <p>, <a>, <img>, etiketleri verilebilir. Örnek bir HTML kodu, Resim 3.1'de görülmektedir.

Resim 3.1 Örnek bir HTML kodu

HTML de kullanılan tüm etiketler ve açıklamaları EK-2'de görülebilir.

3.1.2 CSS HTML'ye Nasıl Eklenir

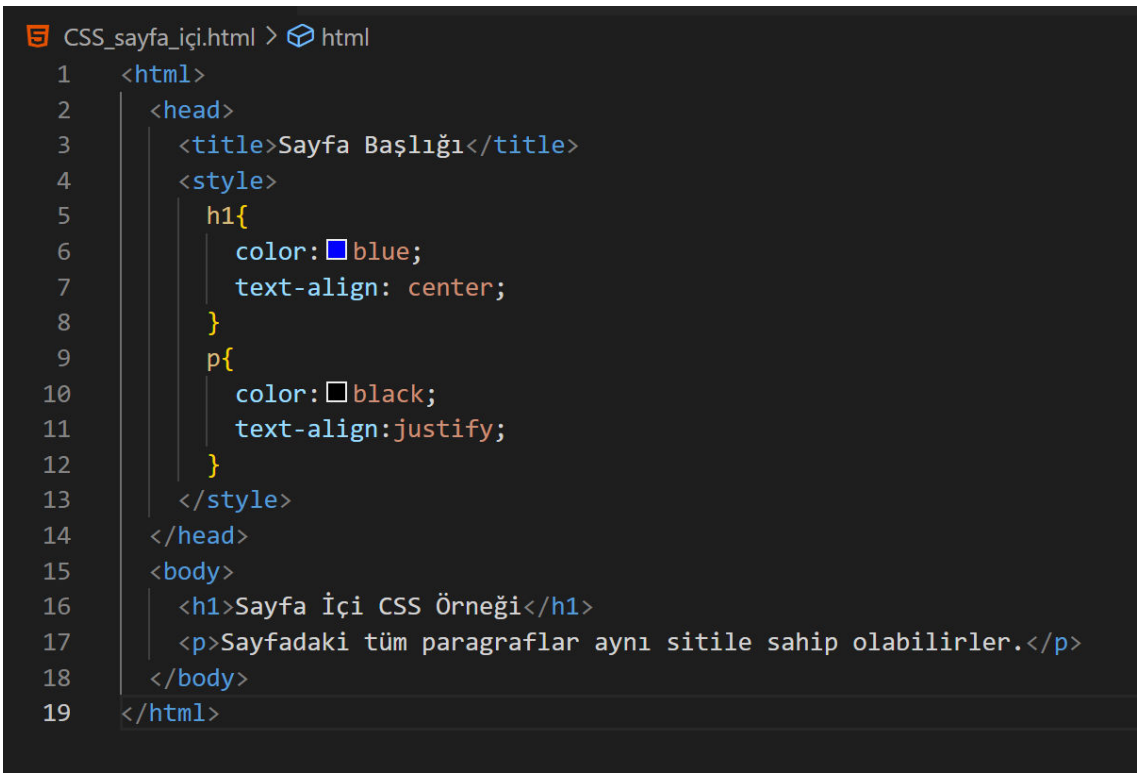
CSS stilleri HTML şablonunuzun her öğesine ayrı ayrı eklenebilir ama CSS birden çok stil bir öğeye ya da bir stil birden çok sayfaya uygulamak gibi daha pratik özellikleriyle ünlüdür . Örneğin; tüm H2 başlıklarının aynı renkte gösterilmesi sağlanabilmektedir.

CSS kodları HTML'ye üç farklı şekilde eklenebilir.

Satır İçi (inline) / Yerel CSS: Satır içi CSS ile sadece o etiketi etkileyecek biçimler verilebilir. Satır içi CSS kodlamasına bir örnek Resim 3.2'de görülebilir.

Resim 3.2 Örnek bir satır içi CSS kodu

Sayfa İçi / Genel CSS: Sayfa içinde belirli bir etiketi gördüğü her yerde, tanımlanmış özellikleri uygular. Genel CSS kodları, <style></style> etiketlerinin arasına yerleştirilir. Sayfalarınızın geç yüklenmesine neden olabilir. Resim 3.3'teki gibi bir CSS koduyla sayfadaki tüm paragrafların ya da başlıkların aynı renk olması sağlanabilmektedir.



Resim 3.3 Örnek bir sayfa içi CSS kodu

Harici (External) CSS: Head etiketleri içinde harici bir CSS dosyasına link vererek, bir CSS dosyasının HTML dosyasına eklenmesi yöntemidir. Birden fazla sayfada aynı .css dosyasının kullanılması mümkündür. Harici CSS dosyanız yüklenene kadar sayfalar düzgün görünmeyebilir. Harici CSS, HTML sayfalarının boyutunu küçülttüğü için ve sayfaların hızlı yüklenmesine neden olduğu için web sayfalarında sıklıkla kullanılan bir CSS ekleme yöntemidir. Resim 3.4'teki gibi bir CSS koduyla sayfada harici bir dosyadaki css kodları yüklenebilir.

CSS_harici.html > html

```
1   <html>
2     <head>
3       <title>Sayfa Başlığı</title>
4       <link rel="styleshhet" type="text/css" href="stil.css">
5
6     </head>
7     <body>
8       <h1>Harici CSS Örneği</h1>
9       <p>Sayfadaki tüm paragraflar aynı stile sahip olabilirler.</p>
10    </body>
11  </html>
```

Resim 3.4 Örnek harici CSS kodu eklenmesi

3.1.3 CSS Seçiciler

CSS seçicileri, biçimlendirmek istediğiniz HTML öğelerini "bulmak" (veya seçmek) için kullanılır (İnt. Kyn. 22). Basit seçiciler, öğeleri öğe adına, kimliğe ve sınıfa göre seçer. Ek olarak, evrensel seçici (*) vardır. CSS seçicileri Çizelge 3.1'de görülmektedir.

Çizelge 3.1 Basit CSS Seçicileri

Seçici	Örnek	Açıklama
öğe	p	Sayfadaki tüm p öğelerini seçer.
#id	#ilksefer	Sayfadaki kimliği id=ilksefer olarak belirtilen tüm öğeleri seçer.
.class	.intro	Sayfadaki sınıfı class=”intro” olarak belirlenmiş tüm öğeleri seçer.
*	*	Sayfadaki tüm öğeleri seçer.

CSS Öznitelik (attribute) Seçicileri, belirli bir öznitelik kümesine sahip HTML öğelerini seçerler. Çizelge 3.2'de öznitelik CSS seçicileri açıklanmıştır.

Çizelge 3.2 Öznitelik CSS Seçicileri

Seçici	Örnek	Açıklama
[attribute]	[lang]	Lang niteliğine sahip tüm öğeleri seçer
[attribute=value]	[lang="tr"]	Lang niteliği lang="tr"olan tüm öğeleri seçer.
[attribute~=value]	[title~="okul"]	Title niteliğine sahip olup içinde "okul" ifadesi geçen tüm öğeleri seçer.
[attribute|=value]	[lang|="en"]	Lang niteliğine sahip olup "en" ifadesine eşit olan ya da "en" ifadesi ile başlayan tüm öğeleri seçer.
[attribute^=value]	[href^="https"]	Href niteliğine sahip olan ve değeri "https" ile başlayan tüm öğeleri seçer.
[attribute$=value]	[href$=".pdf"]	Href niteliğine sahip olan ve değeri ".pdf" ile biten tüm öğeleri seçer.
[attribute*=value]	[href*="akü"]	Href niteliğine sahip olan ve içinde "akü" ifadesi geçen tüm öğeleri seçer.

3.2 Python Programlama Dili

Python, yüksek seviyeli, genel amaçlı, yorumlanan ve nesneye dayalı bir programlama dilidir. Guido van Rossum tarafından 1991 yılında geliştirilmiştir. Python, web geliştirme, veri bilimi, yapay zeka, makine öğrenmesi, oyun geliştirme, sistem yönetimi ve daha birçok alanda yaygın olarak kullanılmaktadır.

Python, İngilizce'ye yakın bir sözdizimi ile basit ve öğrenmesi kolay bir dildir. Python, her türlü programlama ihtiyacını karşılayan geniş bir kütüphane yelpazesine sahiptir. Python, ücretsiz ve açık kaynaklı bir dildir. Bu sayede herkes tarafından kullanılabilir ve geliştirilebilir. Python, Windows, Linux ve macOS gibi farklı işletim sistemlerinde çalıştırılabilir (İnt. Kyn.2).

3.2.1 Python Programlama Dilinin Avantajları

Okuması, anlaması ve yazması basittir. Python, İngilizceye benzer sözdizimine sahip üst düzey bir programlama dilidir. Bu, kodun okunmasını ve anlaşılmasını kolaylaştırır. Python'un öğrenilmesi ve kullanılması son derece basittir, bu yüzden birçok kişi onu yeni başlayanlar için ideal olarak görmektedir. C/C++ ve Java gibi öne çıkan diğer dillerle karşılaştırıldığında, aynı amacı gerçekleştirmek için daha az kod satırına ihtiyacınız vardır.

Python verimliliği artıran bir dildir. Python'un basitliği, geliştiricilerin mevcut soruna konsantre olmalarını sağlar. Programlama dilinin sözdizimini veya davranışını öğrenmek için çok fazla zaman harcamalarına gerek yoktur. Daha az kod yazılır ve daha fazlasını başarılır.

Python yorumlanan bir yazılım dilidir. Yani kod satır satır yürütülür. Bir hata oluştuğunda programı sonlandırır ve hatayı bildirir. Programın çok sayıda hatası olsa bile Python yalnızca birini görüntüler. Bu hata ayıklamayı kolaylaştırır.

Python'da değişkenler dinamik olarak kullanılır. Kodu çalıştırana kadar Python'un ne tür bir değişken kullanıldığını bilmez. Yürütme sırasında veri türünü otomatik olarak atar. Programcının değişkenleri veya onların veri türlerini bildirmesi gerekmez.

Python, OSI (Open Systems Interconnection - Açık Sistemler Ara Bağlantısı) tarafından tanınan bir açık kaynak lisansı altında yayınlanmıştır. Kullanımı ve dağıtımı tamamen ücretsizdir. Kaynak kodunu alabilir, değiştirebilir ve hatta kendi Python sürümünüzü paylaşabilirsiniz.

Python'un standart kütüphanesi çok detaylıdır ve işiniz için gerekli olan hemen hemen tüm fonksiyonları içerir. Ayrıca son derece büyük bir üçüncü taraf kütüphanelerine sahiptir. 200.000'in üzerinde ek paket kullanım için hazır bulunmaktadır.

C/C++ gibi birçok dil, diğer platformlarda çalışabilmesi için kodunuzu değiştirmenizi gerektirir. Python'da durum böyle değildir. Yalnızca bir kez kodlamanız yeterlidir ve her yerde çalışabilir (İnt. Kyn.3).

3.2.2 Python Programlama Dilinin Dezavantajları

Python, yorumlanan ve dinamik olarak yazılan bir dildir. Satır satır gerçekleştirilen kod yürütme bazı uygulamalar için yavaş olabilir. Python'un zayıf hızı, kodu çalıştırırken ek iş yapmasını gerektiren dinamik yapısından kaynaklanmaktadır. Python, hızın kritik olduğu projeler için önerilmez. Python programlama dili belleği geniş ölçüde kullanır. Uygulamalarımızda bellek optimizasyonu gerektiğinde Python verimsiz kalabilir. Python, sunucu tarafı geliştirme için popüler bir dildir. Python, istemci tarafında veya mobil uygulamalarda bellek açısından verimsizdir ve yavaş işleme kapasitesine sahip olduğu için tercih edilmemektedir (İnt. Kyn.3).

3.3 Selenium

Selenium açık kaynaklı bir web otomasyon çerçevesidir (framework). Yazılım geliştiricilerin tarayıcı etkileşimlerini otomatikleştirmesine ve işlevsel testler gerçekleştirmesine olanak tanır. WebDriver gibi araçlarla çeşitli programlama dillerini destekler. Desteklediği diller arasında JavaScript (Node.js), C#, Groovy, Java, Perl, PHP, Python, Ruby ve Scala bulunmaktadır. Selenium Windows, Linux ve macOS işletim sistemlerinde kullanılabilir (İnt. Kyn.4). Selenium resmi web sitesi adresi : "https://www.selenium.dev/" 'dir. Selenium, her biri farklı roller üstlenen çeşitli bileşenlerden oluşur. Bunlar WebDriver, IDE (Integrated Development Environment - Entegre Geliştirme Ortamı) ve Grid'dir.

3.3.1 Selenium WebDriver

WebDriver, tarayıcıyı kontrol etmek ve testleri çalıştırmak için tarayıcı yazılım üreticileri tarafından sağlanan tarayıcı otomasyon API'lerini (Application Programming Interface – Uygulama Programlama Arayüzü) kullanır. Bu sanki tarayıcıyı gerçek bir

kullanıcının çalıştırması gibidir. Geliştirici WebDriver kullanarak yayına aldığı uygulamanın aynısını test edebilecektir (İnt. Kyn.5). WebDriver ile Web sayfalarına gitmek, Formlara veri girmek, Butonlara tıklamak, Linke tıklamak, Sayfadaki elementleri bulmak, Web sayfalarının ekran görüntülerini almak gibi işlemler yapılabilir.

3.3.2 Selenium IDE

IDE, Selenium test senaryolarını geliştirmek için kullanılan bir araçtır. Kullanımı kolay bir Chrome ve Firefox uzantısıdır ve genellikle test senaryoları geliştirmenin en etkili yoludur. Mevcut Selenyum komutlarını kullanarak, o öğenin bağlamı tarafından tanımlanan parametrelerle, kullanıcıların tarayıcıdaki eylemlerini sizin için kaydeder. Bu sadece zaman tasarrufu sağlamakla kalmaz, aynı zamanda Selenyum komut dosyası sözdizimini öğrenmenin mükemmel bir yoludur (İnt. Kyn.5).

3.3.3 Selenium Grid

Selenium Grid, test senaryolarını farklı platformlardaki farklı makinelerde çalıştırmanıza olanak tanır. Test senaryolarının tetiklenmesinin kontrolü yerel uçtadır ve test senaryoları tetiklendiğinde uzak uç tarafından otomatik olarak yürütülür. WebDriver testlerinin geliştirilmesinden sonra testlerinizi birden fazla tarayıcı ve işletim sistemi kombinasyonunda çalıştırma ihtiyacıyla karşılaşabilirsiniz. Grid bu noktada devreye girmektedir (İnt. Kyn.5).

3.3.4 Selenium Kütüphanesinin Web Kazıma İçin Kullanımı

Selenium kurulumu için cmd.exe ya da benzeri bir terminal programı kullanılarak bir komut istemi başlatılmalı ve “pip install selenium” komutu çalıştırılmalıdır. Komutun Windows PowerShell aracılığı ile çalıştırılması Resim 3.5’de gösterilmiştir.

Resim 3.5 PowerShell aracılığı ile selenium kütüphanesinin yüklenmesi.

Selenium, seçilen tarayıcıyla arayüz oluşturmak için bir sürücü gerektirir. Kullanılmak istenen tarayıcının sürücüsü indirilerek yüklenmeli ve sistemin PATH'inde olduğundan emin olunmalıdır. Bu adıma uyulmaması "Selenium.common. Exceptions.WebDriverException" hatasını verecektir.

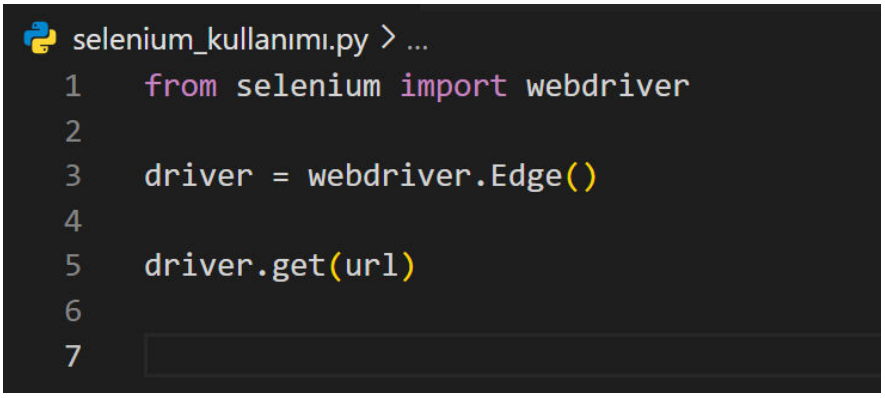
Selenium'da kullanılmak üzere popüler tarayıcıların sürücüleri bulunmaktadır. Çizelge 3.3'te bu sürücüler ve erişim adresleri belirtilmiştir.

Çizelge 3.3 Selenium ile kullanılabilecek popüler tarayıcılar

Tarayıcı	URL
Chrome	https://sites.google.com/chromium.org/driver/
Edge	https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
Firefox	https://github.com/mozilla/geckodriver/releases
Safari	https://webkit.org/blog/6900/webdriver-support-in-safari-10/

3.3.4.1 Selenium Kullanarak Bir Web Adresine Ulaşmak

Bir web adresine ulaşabilmek için öncelikle SeleniumWebDriver içe aktarılmalıdır. İçe aktarma işlemi Resim 3.6'da gösterilmiştir. Bunun için "from selenium import webdriver" komutu kullanılır. Daha sonra WebDriver nesnesi yaratılarak kullanılacak olan sürücü belirtilmelidir. Bunun için "driver = webdriver.Edge()" komutu kullanılır. Get metodu kullanılarak istenen web adresine erişim sağlanır. Bu konutun kullanımı da "driver.get(url)" şeklindedir (İnt. Kyn.6).



Resim 3.6 Selenium kütüphanesi içe aktarma ve bir adrese ulaşma.

3.3.4.2 Selenium Kullanarak Bir Web Adresindeki Öğelere Ulaşmak

Selenium ile erişilen web adresindeki öğelerle etkileşim kurabilmek için farklı yöntemler kullanılabilir. Tek bir öğeye erişebilmek için "By" sınıfı içe aktarılmalıdır. Bu "from selenium.webdriver.common.by importBy" komutuyla gerçekleştirilir (İnt. Kyn.7). Selenium sekiz farklı yöntem ile bir öğeyi belirleyebilir. Bu sekiz yöntem Çizelge 3.4'te listelenmiştir.

Çizelge 3.4 Selenium ile öğeleri belirleme yöntemleri

Konumlandırıcı	Açıklama
By.ID	Konumla eşleşen id öznitelik değerine sahip ilk öğe döndürülecektir.
By.NAME	Konumla eşleşen ad öznitelik değerine sahip ilk öğe döndürülecektir.
By.XPATH	Konumla eşleşen xpath sözdizimine sahip ilk öğe döndürülecektir.
By.LINK_TEXT	Konumla eşleşen bağlantı metni değerine sahip ilk öğe döndürülecektir.
By.PARTIAL_LINK_TEST	Konumla eşleşen kısmi bağlantı metni değerine sahip ilk öğe döndürülecektir.
By.TAG_NAME	Belirtilen etiket adına sahip ilk öğe döndürülecektir.
By.CLASS_NAME	Eşleşen sınıf öznitelik adına sahip ilk öğe döndürülecektir.
By.CSS_SELECTOR	Eşleşen CSS seçiciye sahip ilk öğe döndürülecektir.

Seleniun ile birden çok öğeyi aynı anda belirlemek te mümkündür (İnt. Kyn.8). Selenium yedi farklı yöntem ile çoklu öğeleri belirleyebilir. Çizelge 3.5'te bu yöntemler listelenmiştir.

Çizelge 3.5 Selenium ile çoklu öğeleri belirleme yöntemleri

Konumlandırıcı	Açıklama
find_elements(By.NAME, "name")	Ad öznitelik değeri konumla eşleşen tüm öğeler döndürülecektir.
find_elements(By.XPATH, "xpath")	Konumla eşleşen xpath sözdizimine sahip tüm öğeler döndürülecektir.
find_elements(By.LINK_TEXT, "link text")	Bağlantı metni değeri konumla eşleşen tüm öğeler döndürülecektir.
find_elements(By.PARTIAL_LINK_TEXT, "partial link text")	Konumla eşleşen kısmi bağlantı metni değerine sahip tüm öğeler döndürülecektir.
find_elements(By.TAG_NAME, "tag name")	Belirtilen etiket adına sahip tüm öğeler döndürülecektir.
find_elements(By.CLASS_NAME, "class name")	Eşleşen sınıf öznitelik adına sahip tüm öğeler döndürülecektir.
find_elements(By.CSS_SELECTOR, "css selector")	Eşleşen CSS seçiciye sahip tüm öğeler döndürülecektir.

3.2.4.3 Selenium'daki Aksiyon Zincirleri

Aksiyon zincirleri, fare hareketleri, fare düğmesi eylemleri, tuşa basma ve bağlam menüsü etkileşimleri gibi düşük düzeyli etkileşimleri otomatikleştirmenin bir yoludur. Bu, üzerine gelme ve sürükleyip bırakma gibi daha karmaşık eylemleri gerçekleştirmek için kullanışlıdır. Eylem zinciri yöntemleri, bir öğeyi sürüklememiz, bir öğeye tıklamamız gereken gelişmiş komut dosyaları tarafından kullanılır.

Aksiyon zinciri nesnesi oluşturmak için, aksiyon zinciri sınıfı dokümanlardan içe aktarılmalı ve sürücü anahtar argüman olarak iletilmelidir. Bundan sonra bu nesne aksiyon zincirlerinin tüm işlemlerini gerçekleştirmek için kullanabilir. Bunun için "from selenium import web driver" , "from selenium.webdriver.common.action_chains import ActionChains", "driver = webdriver.Egde()" ve "action = ActionChains(driver)"

komutları kullanılır (İnt. Kyn.9). Selenium aksiyon zincirlerinde çeşitli metotlar kullanabilir. Çizelge 3.6'da bu metotların açıklamaları verilmiştir.

Çizelge 3.6 Selenium aksiyon zincirleri metotları

Metot	Açıklama
click	Bir öğeye tıklama.
click_and_hold	Sol fare tuşu ile bir öğeye tıklama ve basılı tutma.
context_click	Bir öğeye fare sağ tuşu ile tıklama.
double_click	Bir öğeye çift tıklama.
drag_and_drop	Bir öğeye farenin sol tuşu ile tıklayıp basılı tutarak bir başka alana sürüklemek ve bırakmak.
drag_and_drop_by_offset	Bir öğe üzerinde sol fare düğmesini basılı tutup, ardından hedef uzaklığa hareket edip fare düğmesini
key_down	Bir tuşa basmak.
key_up	Basılı bir tuşu bırakmak.
move_by_offset	Fareyi geçerli fare konumundan bir uzaklığa taşımak.
move_to_element	Fareyi bir öğenin ortasına taşımak.
move_to_element_with_offset	Fareyi belirtilen öğenin uzaklığı kadar hareket ettirmek. Uzaklıklar, öğenin sol üst köşesine göredir.
perform	Kaydedilen tüm eylemleri gerçekleştirir.
pause	Tüm girişleri saniye cinsinden belirtilen süre boyunca duraklatmak.
release	Bir öğe üzerinde tutulan fare düğmesinin serbest bırakılması.
reset_actions	Halihazırda yerel olarak ve uzak uçta depolanan eylemleri temizleme.
send_keys	Bir öğeye bir dizi tuş vuruşu göndermek.

3.4 BeautifulSoup

BeautifulSoup, HTML ve XML dosyalarından veri çekmek için kullanılan bir Python kütüphanesidir. BeautifulSoup, web kazıma ve veri işleme için güçlü ve kullanımı kolay bir araçtır. BeautifulSoup , Kullanımı Kolay: Basit ve anlaşılır bir API'ye sahiptir. Karmaşık HTML ve XML yapılarını ayrıştırabilir. Farklı ayrıştırıcılarla birlikte kullanılabilir. Verileri hızlı bir şekilde işleyebilir. Web kazıma, veri işleme, metin

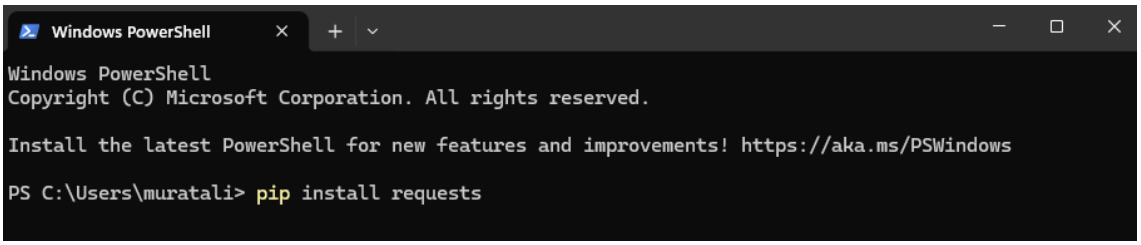
analizi gibi birçok alanda kullanılabilir (İnt. Kyn.10). Kütüphanenin resmi sitesi : “https://www.crummy.com/software/BeautifulSoup/” adresindedir.

3.4.1 BeautifulSoup Kütüphanesinin Web Kazıma İçin Kullanımı

BeautifulSoup kurulumu için cmd.exe ya da benzeri bir terminal programı kullanılarak bir komut istemi başlatılmalı ve “pip install beautifulsoup4” komutu çalıştırılmalıdır. Komutun Windows PowerShell aracılığı ile çalıştırılması Resim 3.7’de gösterilmiştir.

Resim 3.7 PowerShell aracılığı ile BeautifulSoup kütüphanesinin yüklenmesi.

BeautifulSoup kullanılarak çevrimiçi bir web sayfasından bilgi kazınabileceği gibi, bilgisayarımızda kayıtlı bir html ya da xml dosyasından da kazıma yapılabilir. Çevrimiçi bir web sayfasına erişebilmek için requests kütüphanesinin kurulması gereklidir. Bunun için “pip install requests” komutu çalıştırılmalıdır (İnt. Kyn.11). Komutun Windows PowerShell aracılığı ile çalıştırılması Resim 3.8’de gösterilmiştir.

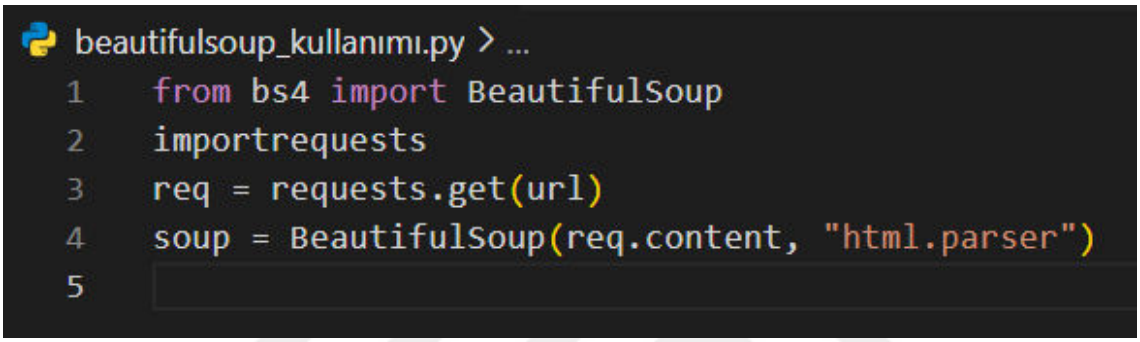


Resim 3.8 PowerShell aracılığı ile Requests kütüphanesinin yüklenmesi.

3.4.2 BeautifulSoup Kullanarak Bir Web Adresine Ulaşmak

Bir web adresine ulaşabilmek için öncelikle BeautifulSoup ve Requests kütüphaneleri içe aktarılmalıdır. Bunun için "from bs4 import BeautifulSoup" ve “import requests”

komutları kullanılır. Daha sonra request.get() metodu ile istenen web adresine erişilir ve BeautifulSoup ve standart python kütüphanesinde bulunan HTML parser aracılığı ile kazıma işlemi yapılabilir. Bunun için "req = requests.get(url)" ve “soup = BeautifulSoup(req.content, "html.parser")” komutları kullanılır. requests.get(url) komutundaki url değişkeni, erişmek istediğimiz web sitesinin adresini içermelidir. Ya da bu adres komutun içine doğrudan metin olarak yazılabilir. BeautifulSoup ve Requests kütüphanelerinin içe aktarılması ve bir adrese ulaşma Resim 3.9’da gösterilmiştir.



Resim 3.9 BeautifulSoup ve Requests kütüphaneleri içe aktarma.

Bir BeautifulSoup yapıcısına bir html belgesini veya dizesini aktardığımızda, BeautifulSoup temel olarak karmaşık bir html sayfasını farklı python nesnelerine dönüştürür. Bs4 paketinde dört ana nesne türü tanımlanmıştır. Bunlar Tag (Etiket), NavigableString (Gezinilebilir Metin), BeautifulSoup ve Comment(Yorum) türleridir (İnt. Kyn.12).

3.4.2.1 Tag (Etiket) Nesnesi

Çeşitli içerik türlerini tanımlamak için HTML etiketleri kullanılır. BeautifulSoup'taki bir Tag nesnesi, gerçek sayfa veya belgedeki bir HTML veya XML etiketine karşılık gelir. Etiketler birçok özellik ve metot içerir. Her bir etiketin bir adı ve özellikleri mevcuttur.

3.4.2.2 NavigableString (Gezinilebilir Metin) Nesnesi

Genellikle belirli bir türdeki açılış ve kapanış etiketine belirli bir dize yerleştirilir. Tarayıcının HTML motoru, öğeyi oluştururken dize üzerinde amaçlanan efekti uygular. NavigableString nesnesi bir etiketin içeriğini temsil eder. bs4.element.NavigableString sınıfının bir nesnesidir. İçeriklere erişmek için ".string" etiketi kullanılır.

3.4.2.3 BeautifulSoup Nesnesi

BeautifulSoup nesnesi, ayrıştırılan nesnenin tamamını temsil eder. Bir web kaynağını kazımaya çalıştığımızda oluşturulan nesnedir. Tag nesnesine benzediğinden belge ağacını ayrıştırmak ve aramak için gereken işlevselliği destekler.

3.4.2.4 Comment (Yorum) Nesnesi

HTML ve XML belgesinde <!-- ve --> arasında yazılan her türlü metin yorum olarak değerlendirilir. BeautifulSoup bu tür yorumlu metni bir Yorum nesnesi olarak algılayabilir.

3.4.3 BeautifulSoup Kullanarak Bir HTML Etiketine Ulaşmak

Herhangi bir html belgesindeki önemli öğe parçalarından biri html etiketleridir. Html etiketlerinin de kendi altında alt etiketleri bulunabilir. BeautifulSoup, etiketin alt öğelerinde gezinmek ve onları yinelemek için farklı yollar sağlar. Bir html belgesinde arama yapmanın en temel yolu etiketi adına göre aramaktır. Etiketin adı belirtilerek içeriğine ulaşılabilir. Soup.head, soup.title, soup.p bunlardan bazılarıdır. Tag.children özelliği kullanılarak o etikete ait tüm alt etiketlere ulaşılabilir. Tag.find_all() metodu ile bağımsız değişken etiketiyle eşleşen tüm etiketlerin içeriklerinin sonuç kümesi elde edilir (İnt. Kyn.13).

3.4.3.1 Bir Öğenin İçeriğini ID'ye (kimliğine) Göre Bulmak

Bir HTML belgesinde genellikle her öğeye bir ID (kimlik) atanır. BeautifulSoup ile belirli bir öğenin içeriği ID'ye (kimliğine) göre bulunabilir. find() metodu, BeautifulSoup nesnesinin argüman olarak verilen kriterleri karşılayan ilk öğeyi arar. Find_all() metodu verilen ID'ye (kimliğe) sahip tüm öğelerin bir listesini döndürür. Bu işlemler için "obj = soup.find(id = 'nm')" ve "obj = soup.find_all(id = 'nm')" komutları kullanılabilir (İnt. Kyn.14).

3.4.3.2 Belli Bir CSS Sınıfı İle Stillendirilmiş Etiketleri Bulmak

BeautifulSoup'ta belli bir CSS sınıfı ile stillendirilmiş etiketleri bulmak mümkündür. Bunun için yine find() ve find_all() metodları sınıf (class) adı belirtilerek kullanılır.Bu işlem için "obj = soup.find_all(attrs={"class": "mainmenu"})" komutu kullanılabilir (İnt. Kyn.15).

3.4.3.3 Metotlara Aktarılan Argümanlara Göre Etiketleri Bulmak

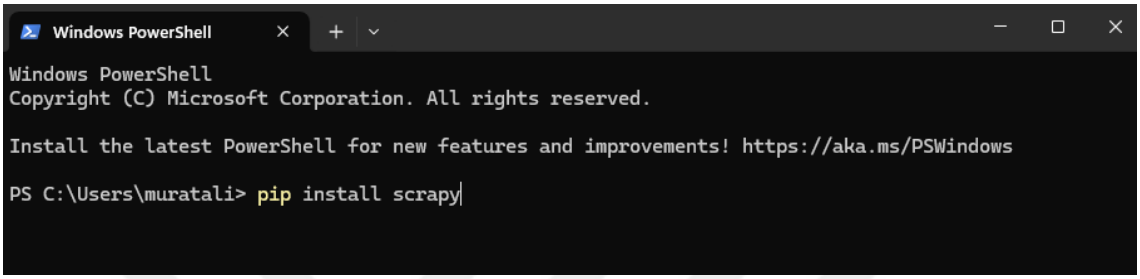
Hem find() hem de find_all() metotları, bu metotlara aktarılan argümanlara göre belgedeki etiketlerden birini veya tamamını bulmayı amaçlamaktadır. attrs parametresi ile bu metotların belli bir özelliği aramalarını isteyebiliriz. Bu işlem için "obj = soup.find(attrs={"name":'marks'})" komutu kullanılabilir (İnt. Kyn.16).

3.5 Scrapy

Scrapy, web sitelerini taramak ve sayfalarından yapılandırılmış verileri çıkarmak için kullanılan hızlı, üst düzey bir web tarama ve web kazıma çerçevesidir. Açık kaynaklı bir yazılımdır. Veri madenciliğinden izleme ve otomatik testlere kadar çok çeşitli amaçlar için kullanılabilir. Scrapy, HTTP bağlantılar kurabilir. CSS seçicileri Xpath (XML Path Language - XML Yol Dili) ifadelerini destekler. Elde edilen veri CSV, JSON veya XML formatında dışa aktarılabilir. Otomatik tekrar bağlantılar sağlayabilir (İnt. Kyn.17). Kütüphanenin resmi sitesi : "https://scrapy.org/" adresindedir.

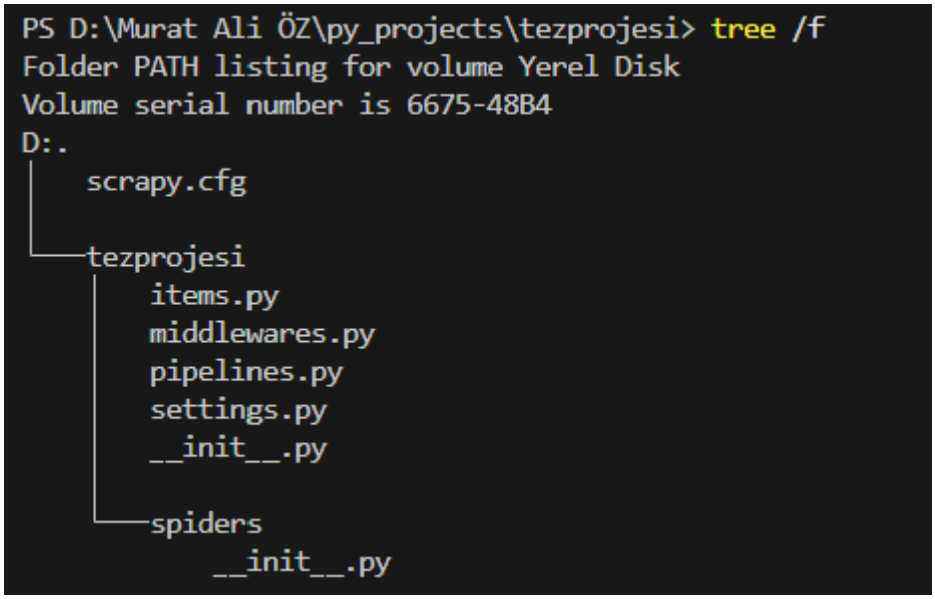
3.5.1 Scrapy Kütüphanesinin Web Kazıma İçin Kullanımı

Scrapy kullanabilmemizin ön şartı Python 3 versiyonuna sahip olmaktır. Scrapy kurulumu için cmd.exe ya da benzeri bir terminal programı kullanılarak bir komut istemi başlatılmalı ve "pip install scrapy" komutu çalıştırılmalıdır. Komutun Windows PowerShell aracılığı ile çalıştırılması Resim 3.10'da gösterilmiştir.



Resim 3.10 Windows PowerShell aracılığı ile Scrapy kütüphanesinin yüklenmesi.

Bir web kazıma işlemine başlanmadan önce yeni bir scrapy projesi oluşturmak gereklidir. Bilgisayarda projenin oluşturulması istenen dizine girilip "scrapy startproject <proje_adı>" komutu çalıştırılır. Bu komut <proje_adı> ile ismi verilen dizini ve Resim 3.11'de görülen dizin ve dosyaları oluşturur (İnt. Kyn.17).



Resim 3.11 Tezprojesi adı ile oluşturulan scrapy projesi dosya ve dizin yapısı

3.5.2 Scrapy Proje Dizini Öğeleri

Scrapy projesi oluşturulduktan sonra içinde bulunulan dizinde bazı dosya ve dizinler oluşur. Bu dosya ve dizinlerin farklı görevleri vardır.

3.5.2.1 scrapy.cfg Dosyası

Scrapy yapılandırma parametrelerini içerir. Birçok Scrapy projesi bu dosyayı paylaşabilir.

3.5.2.2 items.py Dosyası

Scrapy'nin kazıma sırasında dolduracağı öğe veri yapısını tanımlar. Öğeler, web sitelerinden almak istediğiniz yapılandırılmış verileri temsil eder. Her öğe scrapy.item'den miras alan bir sınıftır ve çeşitli veri alanlarından oluşur.

3.5.2.3 middlewares.py Dosyası

Ara yazılım bileşenlerini tanımlar. Bunlar istekleri ve yanıtları işleyebilir, hataları yönetebilir ve diğer görevleri gerçekleştirebilir. En alakalı olanlar Kullanıcı Aracısı rotasyonu, çerez yönetimi ve proxy yönetimi ile ilgilidir.

3.5.2.4 pipelines.py Dosyası

Alıntılanan öğeler üzerindeki işleme sonrası işlemleri belirtir. Bu, kazınmış verileri çeşitli formatlarda ve çeşitli hedeflerde temizlemeye, işlemeye ve saklamaya olanak tanır. Scrapy bunları tanım sırasına göre yürütür.

3.5.2.5 settings.py Dosyası

Çok çeşitli konfigürasyonlar da dahil olmak üzere proje ayarlarını içerir. Bu parametreleri değiştirerek kazıyıcının davranışı ve performansı özelleştirilebilir.

3.5.2.6 spiders Dizini

Python sınıfları tarafından temsil edilen spider'ların (örümcek) ekleneceği dizindir. Scrapy kütüphanesinde spider, belirli bir web sitesinden veya bir grup web sitesinden veri kazımak için kullanılan sınıfları ifade eder.

3.5.3 Scrapy'de Spider (Örümcek)

Scrapy'de spider (örümcek), belirli bir site veya site grubu için kazıma işlemini belirten bir Python sınıfıdır. Sayfalardan yapılandırılmış verilerin nasıl çıkarılacağını ve tarama için bağlantıların nasıl takip edileceğini tanımlar. Scrapy, farklı amaçlar için çeşitli spider (örümcek) türlerini destekler. En yaygın olanları şunlardır:

1-Spider: Önceden tanımlanmış bir URL listesinden başlar ve bunlara veri ayrıştırma mantığını uygular.
2-CrawlSpider: Bağlantıları takip etmek ve birden fazla sayfadan veri çıkarmak için önceden tanımlanmış kuralları uygular.
3-SitemapSpider: Web sitelerini site haritalarına göre kazımak için kullanılır.

Yazılan bir spider kodu .py uzantılı bir dosyaya kaydedilip spiders dizininin içine yerleştirilmelidir. Örnek bir spider kodu EK-1'de yer almaktadır.

3.5.3.1 Spider'ı (Örümcek) Çalıştırmak

Spider'ın (Örümcek) çalıştırılması için projenin üst dizin seviyesinde "scrapy crawl spider_adı" komutu çalıştırılır. Bu komut adı belirtilen spider'ı çalıştırır. Belirtilen web adreslerine erişir, içeriğini alır. Spider içinde parse() metodunda belirtilen işlemleri gerçekleştirir.

3.5.4 Selector'ler (Seçiciler)

Scrapy, veri çıkarmak için kendi mekanizmasıyla birlikte gelir. Bunlara seçici denir. Seçiciler HTML belgesinin XPath veya CSS ifadeleriyle belirtilen belirli bölümlerini

"seçerler". XPath, XML belgelerindeki düğümleri seçmek için kullanılan ve HTML ile de kullanılabilen bir dildir. CSS, HTML belgelerine stil uygulamak için kullanılan bir dildir.

3.5.5 Kazınan Verinin Kaydedilmesi

Scrapy kazınan verinin bir dosya olarak dışarıya aktarılmasını sağlayabilir. Elde edilen veri JSON, XML, ya da CSV formatında dışarıya aktarılabilir. Çalıştırılan spider (örümcek)'ın sonuçları dosyaya kaydetmesi için "scrapy crawl <spider_adı> -O dosya_adı.json" komutu kullanılabilir.

3.6 Örnek Sayfaların Belirlenmesi

Web kazıma, forumlardan, sosyal medya platformlarından, e-ticaret sitelerinden, haber sitelerinden, finansal analiz sayfalarından, iş ilanları, emlak siteleri, akademik yayınlardan yapılabilir. Bu sitelerin hepsinin yapısı, içeriği, kullanılan teknoloji farklıdır. Sitenin türüne, içeriğine göre doğru kütüphaneyi kullanmak, araştırmacıya öncelikle kodlama kolaylığı, daha sonrasında ise data toplama hızı, elde edilen verilerin düzenliliği gibi avantajlar sağlayacaktır. İçeriğe göre yanlış bir kütüphane kullanımı, veri toplamayı zorlaştıracağı gibi, sonucu analiz ederken de problem yaşatacaktır.

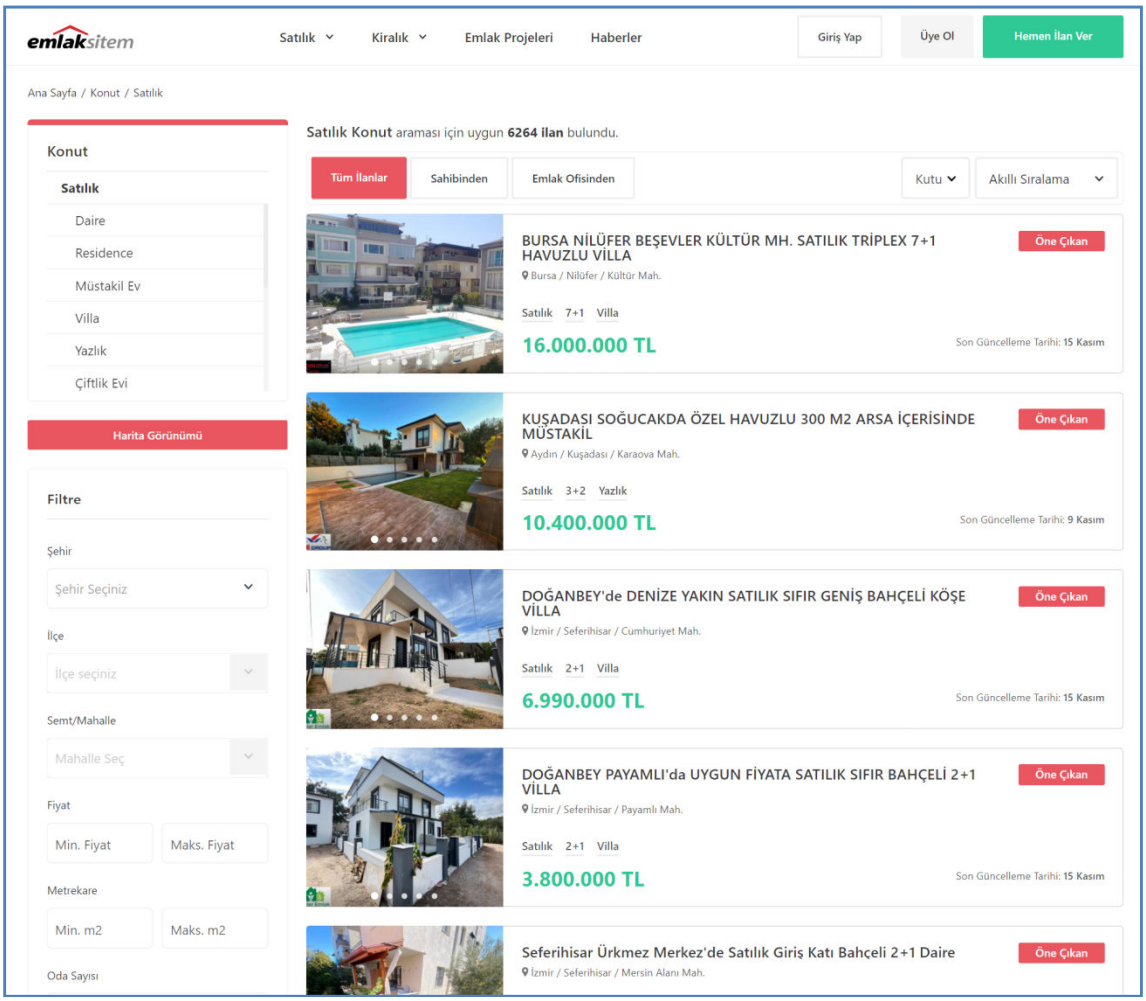
Web taramasında örnek olarak birer emlak ilan sitesi, haber sitesi, online market, e-ticaret sitesi kaynakları kullanılacaktır. Kullanılacak siteler ve adresleri Çizelge 3.7'de belirtilmiştir.

Çizelge 3.7 Web kazımada kullanılacak örnek siteler.

Tür	Erişim Adresi (URL)
Emlak İlan Sitesi	https://emlaksitem.com/satilik
Haber Sitesi	https://www.unyekent.com/
Online Market	https://www.onurmarket.com/meyve-sebze
E-Ticaret Sitesi	https://www.bizimyoresel.com/tum-yoresel-urunler

3.6.1 Emlak İlan Sitesi

Emlak ilan sitesi olarak kullanacağımız site emlaksitem.com olacaktır. Bu siteden satılık emlakları listeleyen https://emlaksitem.com/satilik adresindeki sayfa dan bilgi kazınacaktır. Sayfanın ekran görüntüsü Resim 3.12'de görülmektedir.



Resim 3.12 https://emlaksitem.com/satilik sayfasının ekran görüntüsü

Listelenen emlaklardan kazınacak bilgiler Resim 3.13'te işaretlenmiştir.

Resim 3.13 Listelenen emlaklardan kazınacak bilgiler

HTML kodunun içinde emlak ilanları "item-wrapper featured-4" sınıfından bir DIV öğesi kullanılarak listelenmiştir. Bu DIV öğeleri Resim 3.14'te peşpeşe görünmektedir. "item-wrapper featured-4" sınıfından DIV öğesinin içeriği Resim 3.15'te görünmektedir. Emlak hakkında bilgi içeren başlık "A" (link) öğesinin metin içeriğinden alınacaktır. Emlağın türünü belirten kısa bilgi "item-properties" sınıfından bir "DIV" öğesinin metin içeriğinden alınacaktır. Emlağın fiyat bilgisi "STRONG" öğesi ile belirtilen metinden alınacaktır.

```
▶<div class="item-wrapper featured-4">…</div> == $0
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
▶<div class="item-wrapper featured-4">…</div>
```

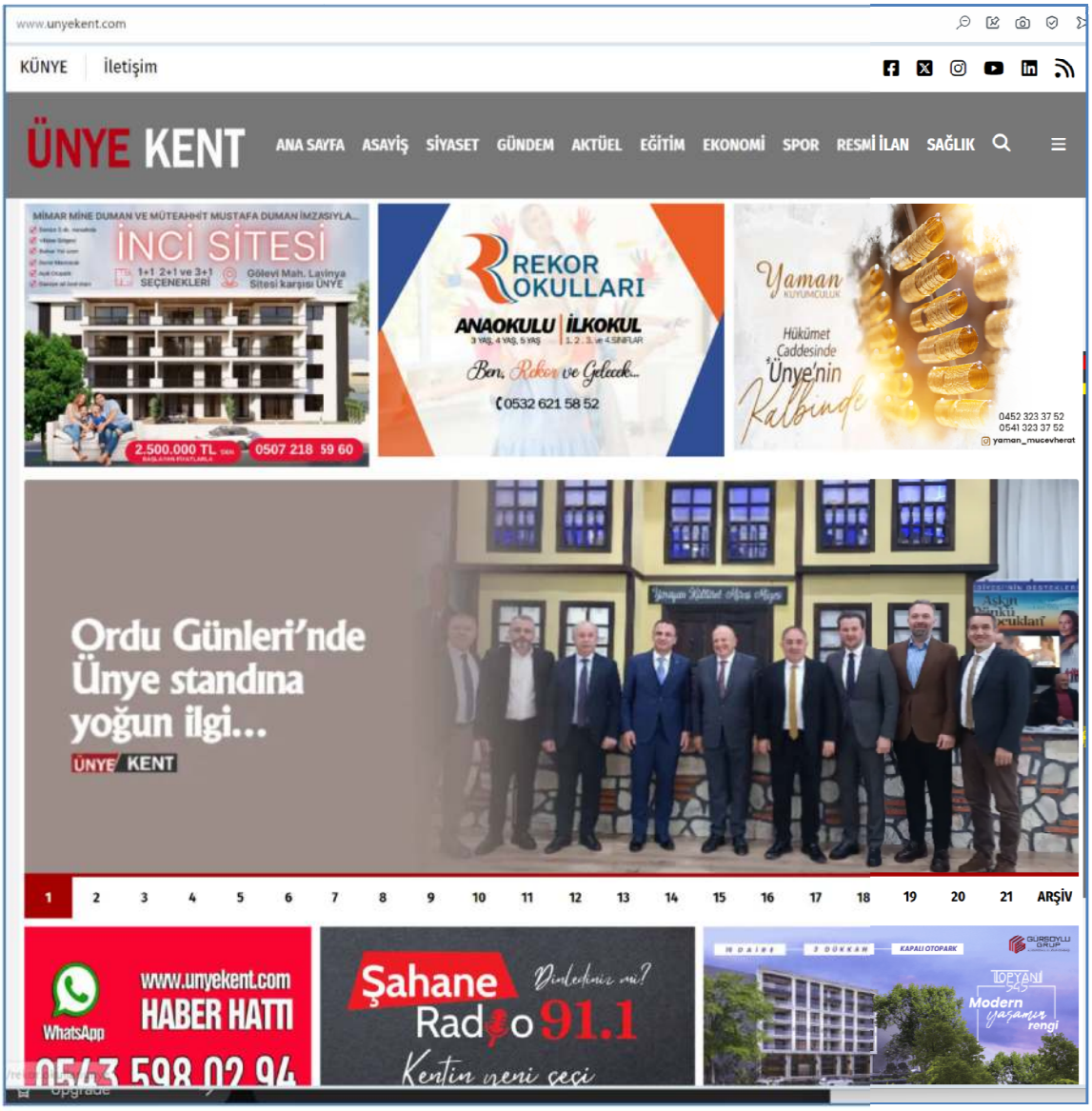
Resim 3.14 item-wrapper featured-4" sınıfından DIV öğeleri

```
▼<div class="result-item lazy-loaded-container" style="display:;"> flex
  ▶<div class="card-image-slider" data-v-0c37103e>…</div>
  ▼<div class="item-content">
    ▼<h2>
      ▼<a href="/ilan/kusadasi-sogucakda-ozel-havuzlu-300-m2-arsa-icerisinde-mustakil-579303" class="classified
      -title"> == $0
         " KUŞADASI SOĞUCAKDA ÖZEL HAVUZLU 300 M2 ARSA İÇERİSİNDE MÜSTAKİL "
       </a>
      ▶<div>…</div>
     </h2>
    ▶<small class="property-categories">…</small> flex
    ▼<div class="item-properties">
       <span> Satılık </span>
       <span> 3+2 </span>
       <span> Yazlık </span>
       <!---->
       <!---->
     </div>
     <div class="item-actions"></div> flex
    ▼<footer> flex
       <strong>10.400.000 TL </strong>
      ▶<small>…</small>
     </footer>
   </div>
 </div>
</div>
```

Resim 3.15 item-wrapper featured-4" sınıfından DIV öğesinin içeriği

3.6.2 Haber Sitesi

Haber sitesi olarak kullanacağımız site unyekent.com olacaktır. Ünye Kent Gazetesi Ordu ili Ünye ilçesinde yayınlanmakta olan bir yerel gazetedir. Gazetenin internet sitesi olan https://www.unyekent.com/ adresi ana sayfasında yayınlanan yerel haberler kazınacaktır. Sayfanın ekran görüntüsü Resim 3.16'da görülmektedir.

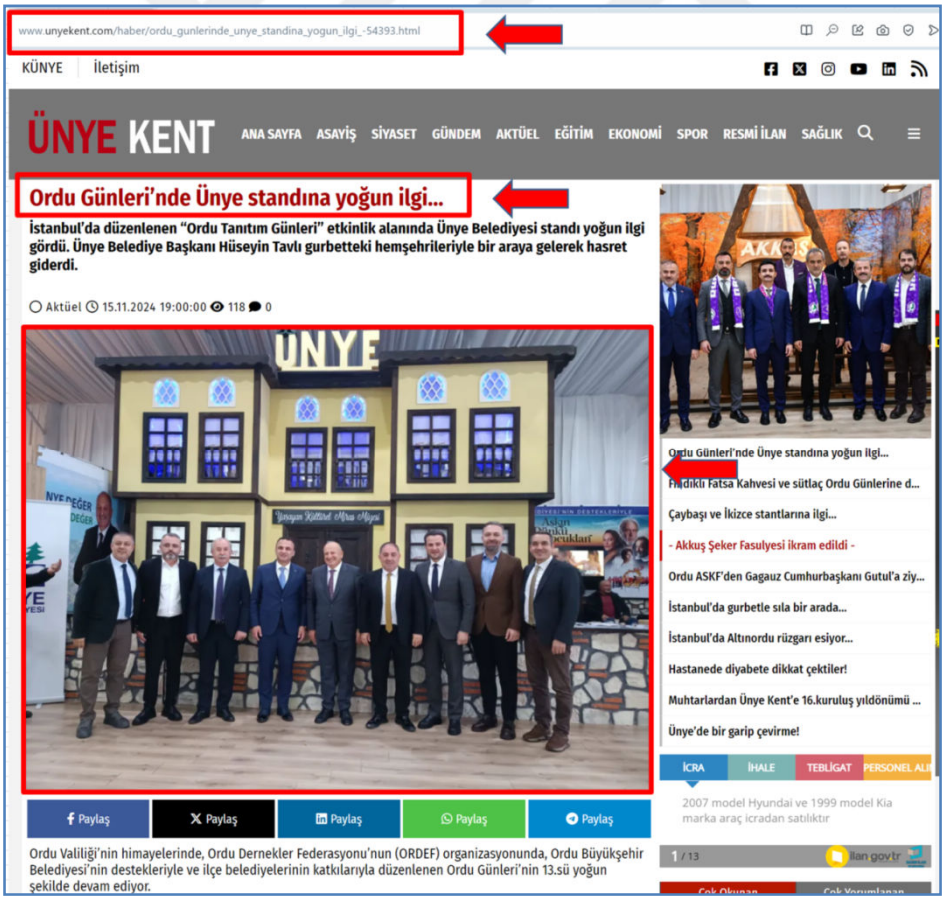


Resim 3.16 https://www.unyekent.com/ ana sayfasının ekran görüntüsü

Ana sayfada tıklandığında ayrı bir sayfa açan 21 adet manşet haber listelenmektedir. Resim 3.17'de işaretlenmiş olan linklere tıklandığında ilgili haber yeni bir sayfada

açılmaktadır. Açılan haber sayfası ve bu sayfadan kazınacak bilgiler Resim 3.18'de görülmektedir.

Resim 3.17 Ünyekent Gazetesi manşet haberleri



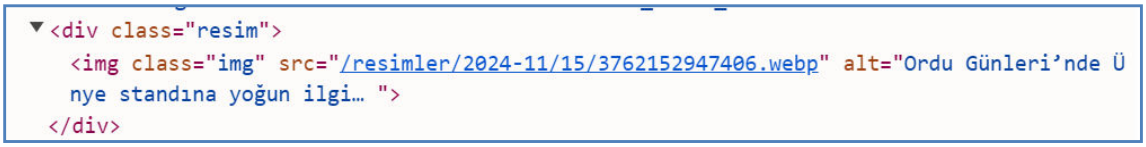
Resim 3.18 Ünyekent Gazetesi haber sayfası ekran görüntüsü ve kazınacak bilgiler

www.unyekent.com ana sayfasındaki 21 adet manşet haberinin link URL'leri HTML kodunun içinde emlak ilanları "ulwrap" sınıfından bir "DIV" öğesi içinde "LI" öğeleri kullanılarak listelenmiştir. "LI" öğeleri içinde "A" öğesi ile belirtilen linkler Resim 3.19'da peş peşe görünmektedir. Kazınan linklerdeki haber detay sayfalarından haberin başlık bilgisi ve haber fotoğrafının URL'si kazınacaktır. Haberin başlığı "H1" öğesi kullanılarak belirtilmiştir. Resim 3.20'de haberin başlığını HTML kodu içinde görmekteyiz. Haberin fotoğrafının URL'sini ise "resim" sınıfından bir "DIV" öğesi içindeki "IMG" öğesinin "SRC" özelliği içinden almaktayız. Resim 3.21'de HTML kodu içindeki fotoğraf linki bilgisini görmekteyiz.

```
<hr class="active wf">
▼<div class="ulwrap">
  ▼<ul class="flx fnwr moduleBackground"> flex
    ▼<li class="Numarali_Manset_6_p_0 active">
       <a target="_blank" href="/haber/ordu_gunlerinde_unye_standina_yogun_ilgi_-5439
       3.html">1</a>
     </li>
    ▼<li class="Numarali_Manset_6_p_0"> == $0
       <a target="_blank" href="/haber/findikli_fatsa_kahvesi_ve_sutlac_ordu_gunlerine
       _damga_vurdu_-54392.html">2</a>
     </li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
    ▶<li class="Numarali_Manset_6_p_0">…</li>
   </ul>
 </div>
```

Resim 3.19 Ünyekent Gazetesi manşet haberlerin linklerini içeren "LI" öğeleri

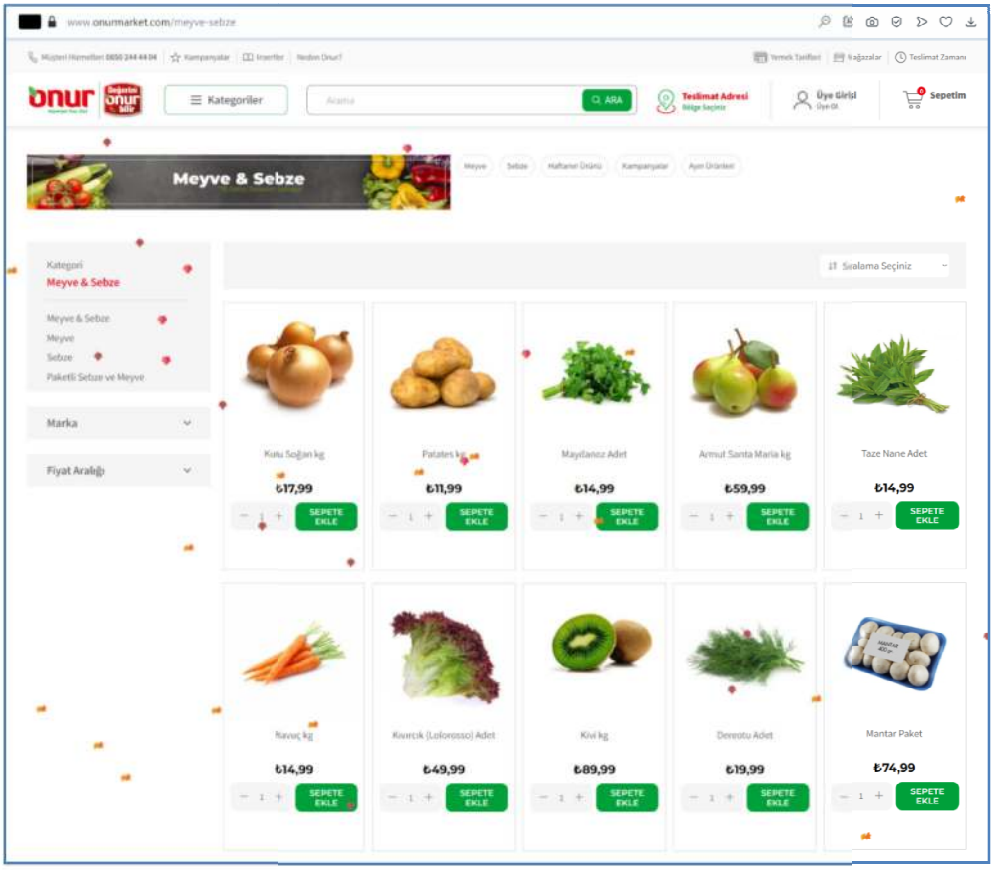
Resim 3.20 Ünyekent Gazetesi haber başlığını içeren H1 öğesi



Resim 3.21 Ünyekent Gazetesi haber fotoğrafı URL'sini içeren DIV öğesi

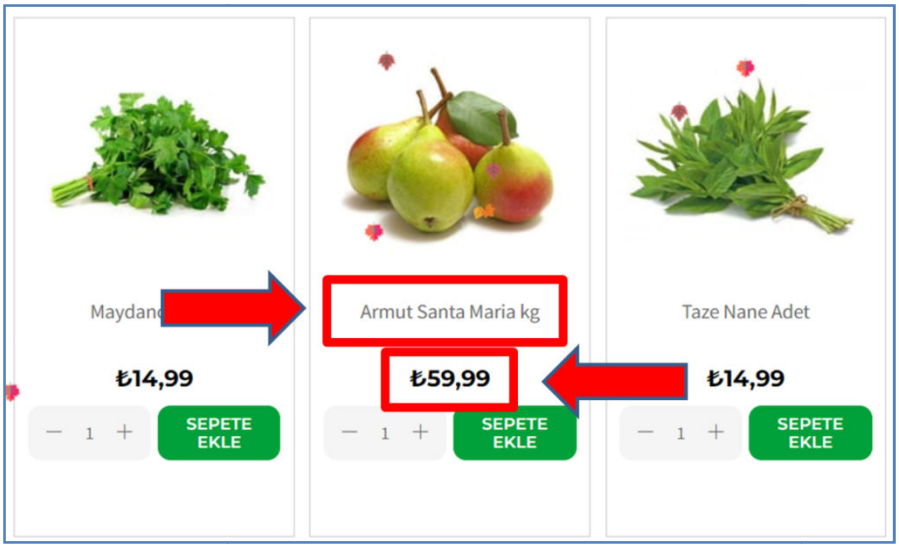
3.6.3 Online Market

Online Market sitesi olarak kullanacağımız site onurmarket.com olacaktır. Onur Market Türkiye çapında birçok ilde şubeleri olan bir süpermarket zinciridir. www.onurmarket.com sitesi vasıtasıyla internet üzerinden satış ta yapmaktadır. Onur Market sitesinden https://www.onurmarket.com/meyve-sebze adresindeki meyve ve sebze ürünlerinin bilgileri kazınacaktır. Sayfanın ekran görüntüsü Resim 3.22'de görülmektedir.



Resim 3.22 www.onurmarket.com meyve-sebze sayfası

Listelenen ürünlerden kazınacak bilgiler Resim 3.23'te işaretlenmiştir.



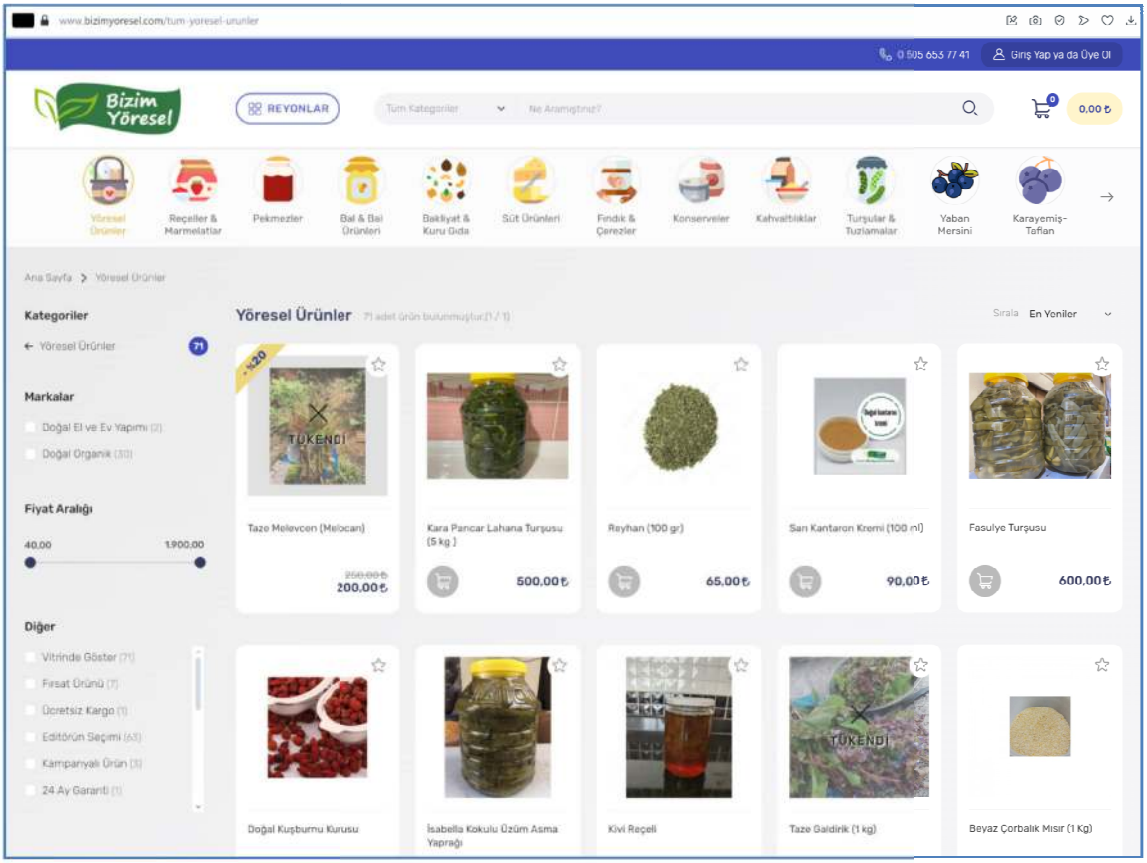
Resim 3.23 Onur Market meyve sebze ürünlerinden kazınacak bilgiler

HTML kodunun içinde ürünler “productItem” sınıfından DIV öğeleri kullanılarak listelenmiştir. Bu öğelerin içinde “productNamedetailUrl” sınıfından bir “DIV” öğesi ürünün adını içermektedir. “discountPrice” sınıfından bir “DIV” öğesi de ürün fiyat bilgisini içermektedir. Bilgilerin kazınacağı HTML öğeleri Resim 3.24’te görünmektedir.

Resim 3.24 Onur Market meyve sebze ürünleri bilgilerini içeren HTML öğeleri

3.6.4 E-Ticaret Sitesi

E-Ticaret sitesi olarak kullanacağımız site bizimyoresel.com olacaktır. Bu site Ordu ilinden yöresel ürünlerin satışını yapan bir sitedir. Bu sitede satışta olan yöresel ürünlerin bilgileri https://www.bizimyoresel.com/tum-yoresel-urunler adresinden kazınacaktır. Sayfanın ekran görüntüsü Resim 3.25’te görülmektedir.



Resim 3.25 www.bizimyoresel.com yöresel ürünler sayfası

Listelenen ürünlerden kazınacak bilgiler Resim 3.26'da işaretlenmiştir.

Resim 3.26 Listelenen yöresel ürünlerden kazınacak bilgiler

HTML kodunun içinde ürünler “product-item” sınıfından DIV öğeleri kullanılarak listelenmiştir. Bu öğelerin içinde “detail” sınıfından bir “DIV” öğesi ürünün adını içermektedir. “last-price” sınıfından bir “DIV” öğesi de ürün fiyat bilgisini içermektedir. Bilgilerin kazınacağı HTML öğeleri Resim 3.27’de görünmektedir.

```
<div class="product-item" data-urun-id="161">
  <div class="image">...</div>
  <div class="badges kampanya" data-value="0">Kampanya</div>
  <div class="badges ucretsiz-kargo" data-value="0">Ücretsiz Kargo</div>
  <div class="detail">
    <a href="https://www.bizimyoresel.com/kivi-receli" class="name">Kivi Reçeli</a>
    <div class="price" data-minimum="1"> flex
      <div class="add-to-cart">
        <div class="pamount" data-birim="Adet">...</div>
        <button class="SepeteEkle" data-urun-id="161">
          <svg>
            <use xlink:href="#cart"></use>
          </svg>
        </button>
      </div>
      <div class="price-inner">
        <div class="last-price" data-text="Adet">
          "80,00"
          <i class="fa fa-try">...</i>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">
  <div class="product-item" data-urun-id="157">...</div>
</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">
  <div class="product-item" data-urun-id="152">...</div>
</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div> == $0
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
<div class="col-xs-4 col-sm-3 col-md-3 col-lg-3">...</div>
```

Resim 3.27 Yöresel ürünlerin bilgilerini içeren HTML öğeleri

3.7 Yazılan Kodlar ve Bilgi Kazıma

BeautifulSoup kullanarak yazılmış "Emlak İlan Sitesi" kazıma kodu EK-3'te, "Haber Sitesi" kazıma kodu EK-4'te, "Online Market Sitesi" kazıma kodu EK-5'te, "E-Ticaret Sitesi" kazıma kodu EK-6'da verilmiştir. Selenium kullanarak yazılmış "Emlak İlan Sitesi" kazıma kodu EK-7'de, "Haber Sitesi" kazıma kodu EK-8'te, "Online Market Sitesi" kazıma kodu EK-9'da, "E-Ticaret Sitesi" kazıma kodu EK-10'da verilmiştir. Scrapy kullanılarak örnek siteler için birer proje oluşturulmuştur. Oluşturulan projelerde spiders dizininde bir spider.py dosyası oluşturulmuş ve kazıma için kullanılmıştır. "Emlak İlan Sitesi" için oluşturulan spider EK-11'de, "Haber Sitesi" için oluşturulan spider EK-12'de, "Online Market Sitesi" için oluşturulan spider EK-13'te, "E-Ticaret Sitesi" için oluşturulan spider EK-14'da verilmiştir.

Oluşturulan kodlar internet bağlantı hız farklılıklarından fazla etkilenmemesi amacıyla Çizelge 3.8'de belirtilen sıra ile 10 tekrar şeklinde çalıştırılmış, çalışma süreleri kaydedilmiştir.

Çizelge 3.8 Kodların çalıştırılma sırası

Sıra	Kütüphane	Erişim Adresi (URL)
1	BeautifulSoup	https://emlaksitem.com/satilik
2	Selenium	https://emlaksitem.com/satilik
3	Scrapy	https://emlaksitem.com/satilik
4	BeautifulSoup	https://www.unyekent.com/
5	Selenium	https://www.unyekent.com/
6	Scrapy	https://www.unyekent.com/
7	BeautifulSoup	https://www.onurmarket.com/meyve-sebze
8	Selenium	https://www.onurmarket.com/meyve-sebze
9	Scrapy	https://www.onurmarket.com/meyve-sebze
10	BeautifulSoup	https://www.bizimyoresel.com/tum-yoresel-urunler
11	Selenium	https://www.bizimyoresel.com/tum-yoresel-urunler
12	Scrapy	https://www.bizimyoresel.com/tum-yoresel-urunler

Oluşturulan kodlar yukarıda belirtilen sıra ve tekrar sayılarıyla farklı platformlarda çalıştırılmış, çalışma süreleri kaydedilmiştir. Kodlar Windows kişisel bilgisayar, Windows üzerinde sanal makine olarak çalışan Linux sistemi ve bulut sunucuda çalışan Linux sistemi olmak üzere üç farklı platformda çalıştırılmıştır.

3.7.1 Windows Kişisel Bilgisayar Sistem Özellikleri

Sistem Modeli : HP ZBookFirefly 14 inch G8 Mobile Workstation.
İşletim sistemi : Microsoft Windows 11 Pro
İşlemci : 11. nesil Intel(R) Core(TM) i7-1165G7 @ 2.80 Ghz 4 Core(s)
RAM (Random Access Memory - Rastgele Erişimli Bellek) : 16 GB
Python Versiyonu : Python 3.10.11
Requests : 2.27.1
beautifulsoup4 : 4.10.0
Scrapy : 2.11.1
Selenium : 4.16.0

3.7.2 Linux Sanal Makine Sistem Özellikleri

Sanal makine kullanımı için Windows kişisel bilgisayar üzerinde Oracle VM (Virtual Machine - Sanal Makine) Virtualbox yazılımı çalıştırılmıştır. Oracle VM Virtualbox ilgili ekran görüntüsü Resim 3.28'de görülmektedir.

Resim 3.28 Oracle VM Virtualbox

İşletim Sistemi : Ubuntu (64-bit) 22.04 Server
Sanal İşlemci Sayısı : 4
Ayrılan RAM : 4 GB
Python Versiyonu : Python 3.10.12
Requests : 2.32.3
beautifulsoup4 : 4.12.3
Scrapy : 2.11.2
Selenium : 4.26.1

3.7.3 Linux Bulut Sunucu Sistem Özellikleri

Bulut sunucu olarak Almanya'da bir veri merkezindeki sanal sunucu kullanılmıştır.

Veri Merkezi : Berlin Almanya
İşletim Sistemi : Ubuntu (64-bit) 18.04 Server
Sanal İşlemci Sayısı : 2 vCore
RAM : 2 GB
Python Versiyonu : Python 3.6.9
Requests : 2.27.1
beautifulsoup4 : 4.12.3
Scrapy : 2.6.3
Selenium : 3.141.0

Kullanılan bilgisayar sistemlerinin özelliklerinin karşılaştırmalı tablosu Çizelge 3.9'da belirtilmiştir.

Çizelge 3.9 Kullanılan bilgisayarların özellikleri

Özellik	Kişisel Bilgisayar	Sanal Makine	Bulut Sunucu
İşletim Ssitemi	Microsoft Windows 11 Pro	Ubuntu (64-bit) 22.04 Server	Ubuntu (64-bit) 18.04 Server
Hafıza (RAM)	16 GB	4 GB	2 GB
İşlemci	11. nesil Intel(R) Core i7-1165G7		
Sanal İşlemci Sayısı		4	2 vCore
Lokasyon	Ev	Ev	Veri Merkezi

3.7.3 Kazınan Bilgilerin Ekranda Gösterilmesi

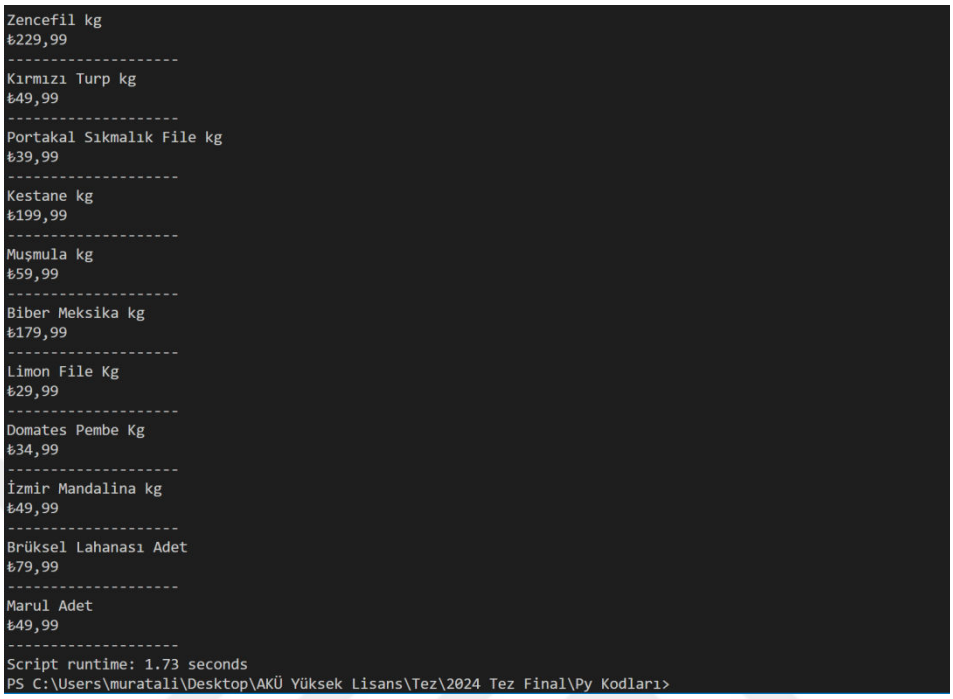
Kodların çalıştırılması sonucu, kazınan bilgiler, ekrana metin olarak yazdırılmaktadır. Kodun çalışması sonlandığında çalışma süresi hesaplanıp saniye olarak ekrana yazdırılmıştır.

Emlaksitem sitesi BeautifulSoup ile kazıma sonuçları Resim 3.29'da görülmektedir.

```
BURSA NİLÜFER BEŞEVLER KÜLTÜR MH. ACİL SATILIK 2+1 GİRİŞ KATI DAİRE 2.400.000 TL
Satılık   2+1   Daire
2.400.000 TL
--------------------
ES EMLAK'TAN TORBALI CUMHURİYET MAH.150 m2 3+1 D.GAZLI, BAKIMLI DAİREMİZ SATILIKTIR
Satılık   3+1   Daire
2.750.000 TL
--------------------
ES EMLAK'TAN TORBALI CUMHURİYET MAH. SİTE İÇİ, 145m2 3+1 D.GAZLI, ARAKAT, BAKIMLI DAİREMİZ SATILIKTI
Satılık   3+1   Daire
2.300.000 TL
--------------------
KUŞADASI MARİNA BÖLGESİNDE ÖZEL HAVUZLU 6+2 AKILLI EV SİSTEMİ
Satılık   6+2   Villa
21.500.000 TL
--------------------
KUŞADASI MARİNADA HAVUZLU SİTEDE MASRAFSIZ 3+1 BAHÇE KATI
Satılık   3+1   Daire
3.975.000 TL
--------------------
Didim Akbük'te Muhteşem Panaromik Deniz Manzaralı 3+1 Havuzlu Villa
Satılık   3+1   Villa
6.100.000 TL
--------------------
DİDİM ALTIN EKMEK MEVKİSİNDE OKULLARA YAKIN 2+1 ARAKAT DAİRE
Satılık   2+1   Daire
3.250.000 TL
--------------------
Doğanbey de Deniz Çok Yakın Ultra Lükx Satılık 4+1 Villa
Satılık   4+1   Villa
7.500.000 TL
--------------------
MUĞLALILAE. EMLAKTAN YOĞURTCULAR MAH AYRANCILARDA BAHCE SATILIK
Satılık   2+2   Çiftlik Evi
25.000.000 TL
--------------------
Script runtime: 0.89 seconds
PS C:\Users\muratali\Desktop\AKÜ Yüksek Lisans\Tez\2024 Tez Final\Py Kodları>
```

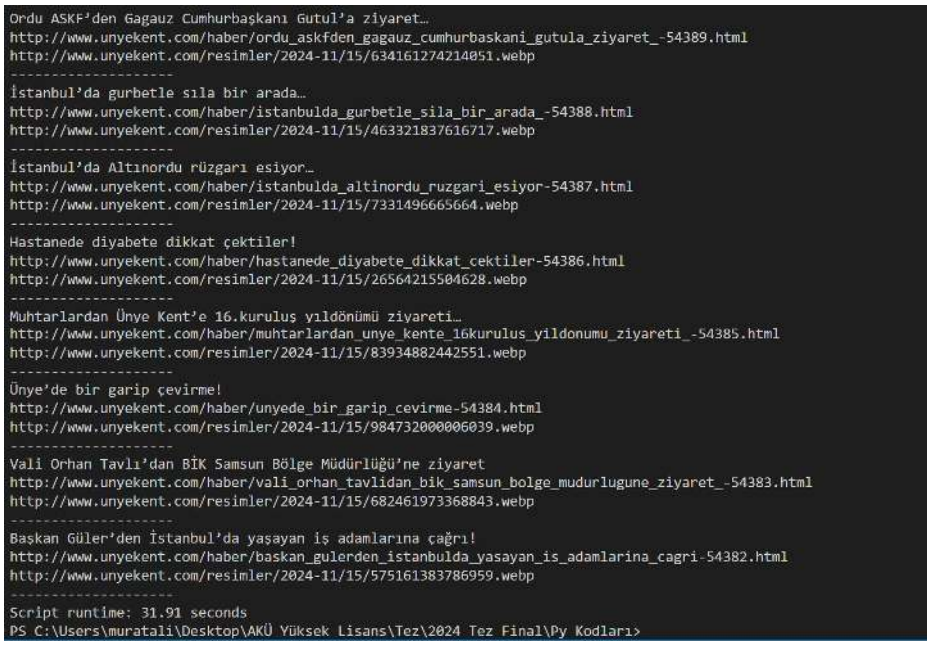
Resim 3.29 Emlaksitem BeautifulSoup kazıma sonuçları

Onurmarket sitesi BeautifulSoup ile kazıma sonuçları Resim 3.30'da görülmektedir.



Resim 3.30 Onurmarket BeautifulSoup kazıma sonuçları

Ünyekent sitesi BeautifulSoup ile kazıma sonuçları Resim 3.31'de görülmektedir.



Resim 3.31 Ünyekent BeautifulSoup kazıma sonuçları

Bizim Yöresel sitesi BeautifulSoup ile kazıma sonuçları Resim 3.32'de görülmektedir.

```
Fırın Mısır Unu (1 Kg)
150,00
--------------------
Akkuş Kuru Fasulyesi Çangal  (1 Kg)
350,00
--------------------
İsabella Kokulu Üzüm Şirası (660cc )
150,00
--------------------
İncir Reçeli ( 850 Gr. )
200,00
--------------------
Melevcen - Diken Ucu
250,00
--------------------
Dağ Çileği Reçeli (390 gr)
250,00
--------------------
Sızma Soğuk Sıkım Zeytinyağı
300,00
--------------------
Armut Pekmezi (1Kg)
400,00
--------------------
Kuşburnu Marmelatı
150,00
--------------------
Trabzon (Cennet) Hurması Pekmezi (1 kg)
400,00
--------------------
ELMA SİRKESİ (500 ML)
100,00
--------------------
Script runtime: 1.11 seconds
PS C:\Users\muratali\Desktop\AKÜ Yüksek Lisans\Tez\2024 Tez Final\Py Kodları>
```

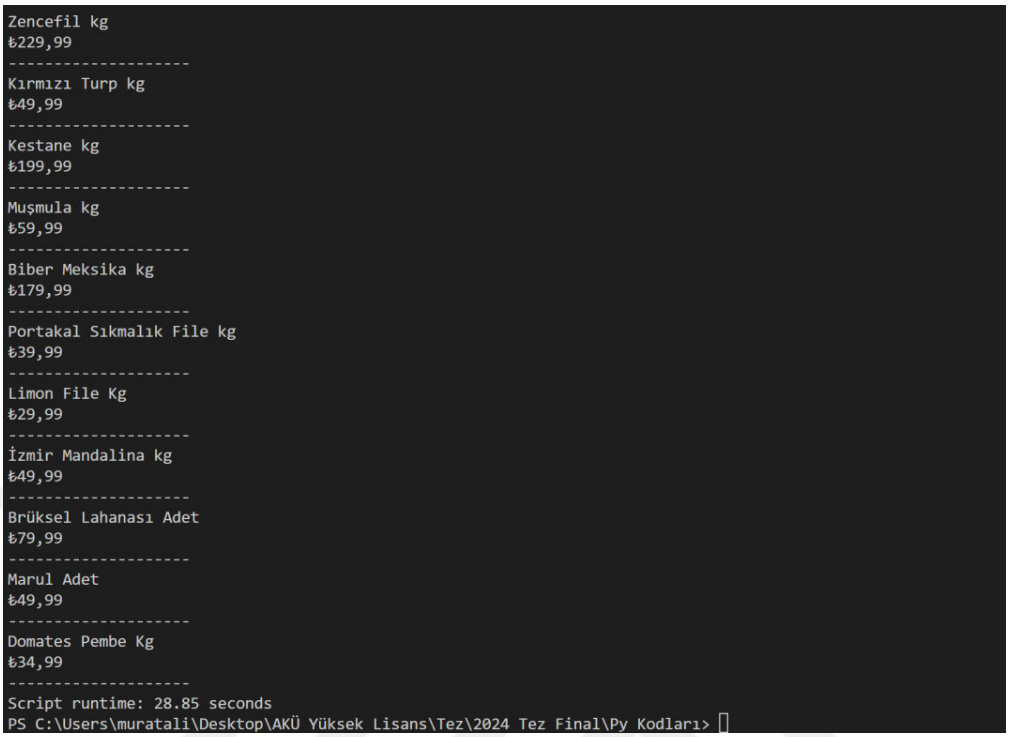
Resim 3.32 Bizim Yöresel BeautifulSoup kazıma sonuçları

Emlaksitem sitesi Selenium ile kazıma sonuçları Resim 3.33'da görülmektedir.

```
BURSA NİLÜFER BEŞEVLER KÜLTÜR MH. ACİL SATILIK 2+1 GİRİŞ KATI DAİRE 2.400.000 TL
Satılık 2+1 Daire
2.400.000 TL
--------------------
ES EMLAK'TAN TORBALI CUMHURİYET MAH.150 m2 3+1 D.GAZLI, BAKIMLI DAİREMİZ SATILIKTIR
Satılık 3+1 Daire
2.750.000 TL
--------------------
ES EMLAK'TAN TORBALI CUMHURİYET MAH. SİTE İÇİ, 145m2 3+1 D.GAZLI, ARAKAT, BAKIMLI DAİREMİZ SATILIKTI
Satılık 3+1 Daire
2.300.000 TL
--------------------
KUŞADASI MARİNA BÖLGESİNDE ÖZEL HAVUZLU 6+2 AKILLI EV SİSTEMİ
Satılık 6+2 Villa
21.500.000 TL
--------------------
KUŞADASI MARİNADA HAVUZLU SİTEDE MASRAFSIZ 3+1 BAHÇE KATI
Satılık 3+1 Daire
3.975.000 TL
--------------------
Didim Akbük'te Muhteşem Panaromik Deniz Manzaralı 3+1 Havuzlu Villa
Satılık 3+1 Villa
6.100.000 TL
--------------------
DİDİM ALTIN EKMEK MEVKİSİNDE OKULLARA YAKIN 2+1 ARAKAT DAİRE
Satılık 2+1 Daire
3.250.000 TL
--------------------
Doğanbey de Deniz Çok Yakın Ultra Lükx Satılık 4+1 Villa
Satılık 4+1 Villa
7.500.000 TL
--------------------
Script runtime: 19.96 seconds
PS C:\Users\muratali\Desktop\AKÜ Yüksek Lisans\Tez\2024 Tez Final\Py Kodları>
```

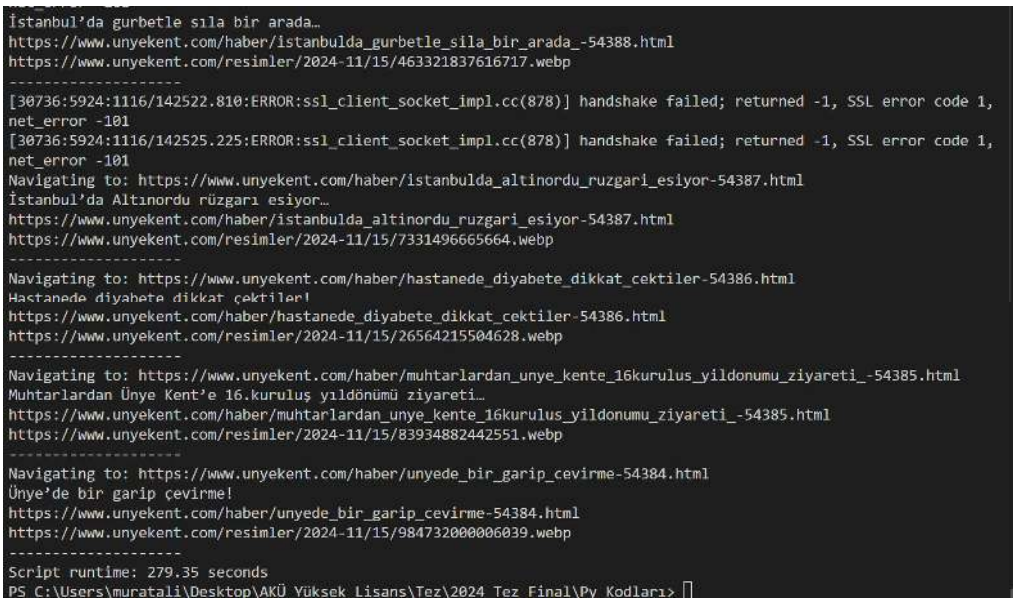
Resim 3.33 Emlaksitem Selenium kazıma sonuçları

Onurmarket sitesi Selenium ile kazıma sonuçları Resim 3.34'ta görülmektedir.



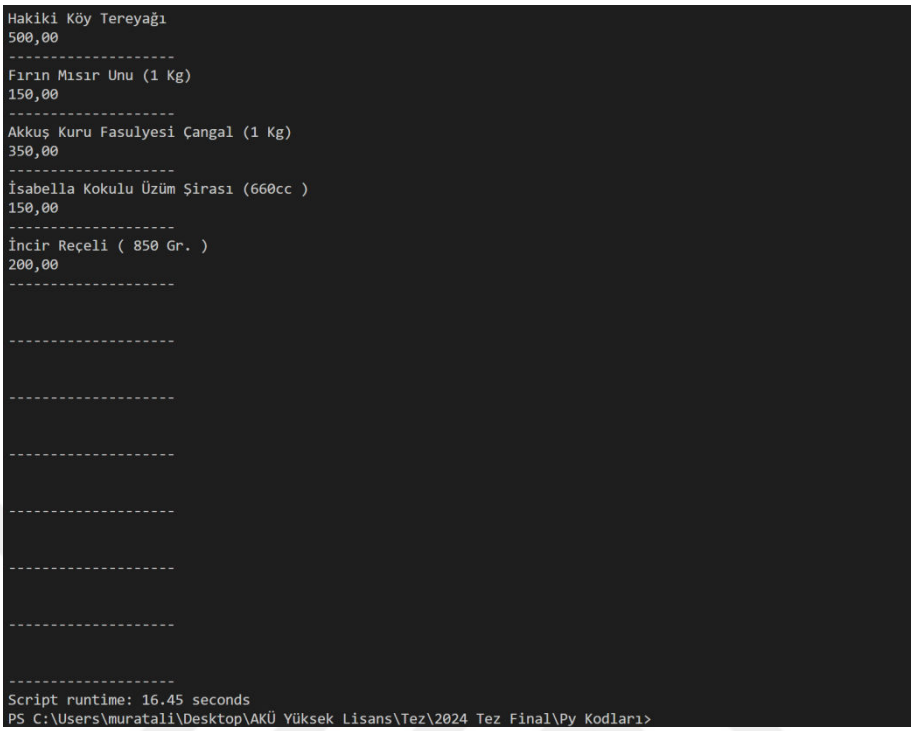
Resim 3.34 Onurmarket Selenium kazıma sonuçları

Ünyekent sitesi Selenium ile kazıma sonuçları Resim 3.35'te görülmektedir.



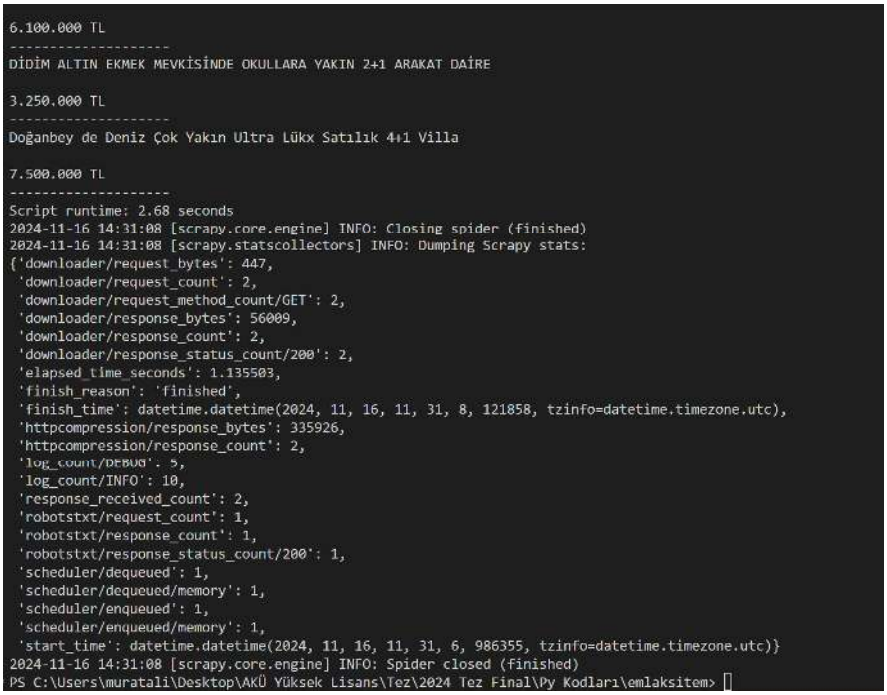
Resim 3.35 Ünyekent Selenium kazıma sonuçları

Bizim Yöresel sitesi Selenium ile kazıma sonuçları Resim 3.36'da görülmektedir.



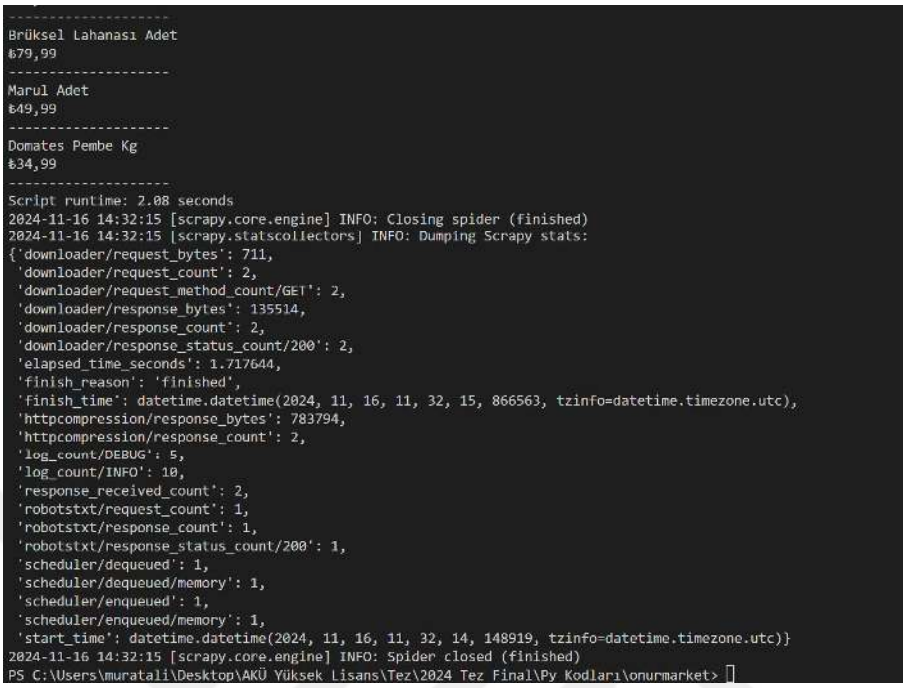
Resim 3.36 Bizim Yöresel Selenium kazıma sonuçları

Emlaksitem sitesi Scrapy ile kazıma sonuçları Resim 3.37'de görülmektedir.



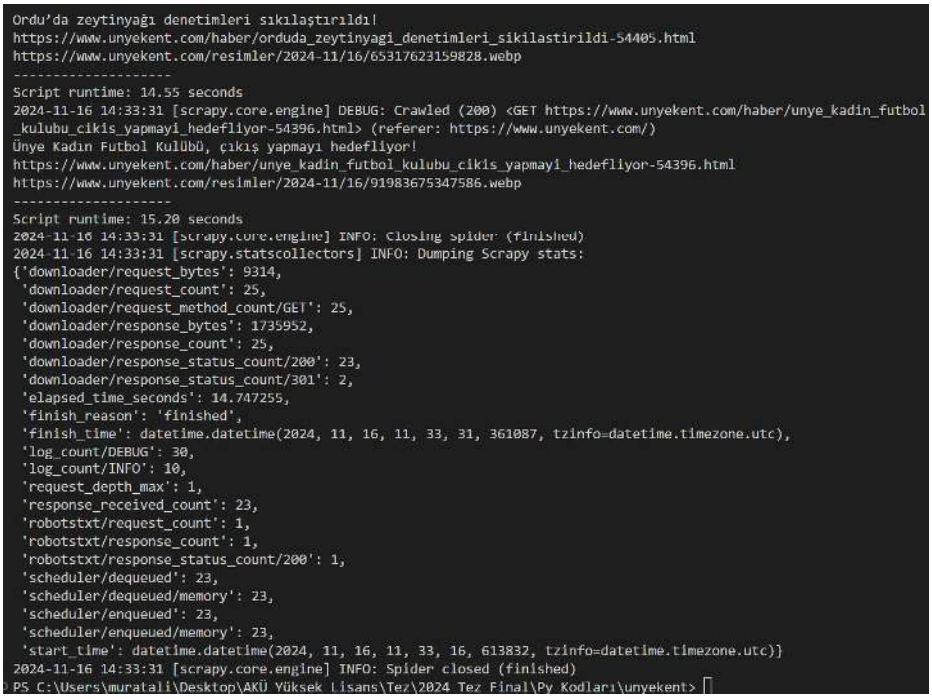
Resim 3.37 Emlaksitem Scrapy kazıma sonuçları

Onurmarket sitesi Scrapy ile kazıma sonuçları Resim 3.38'de görülmektedir.



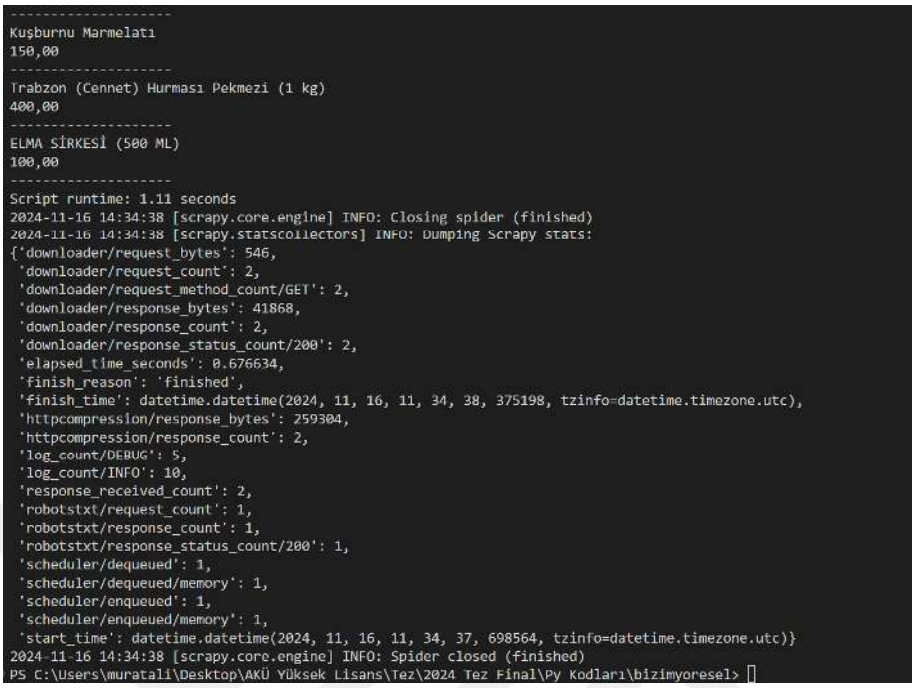
Resim 3.38 Onurmarket Scrapy kazıma sonuçları

Ünyekent sitesi Scrapy ile kazıma sonuçları Resim 3.39'da görülmektedir.



Resim 3.39 Ünyekent Scrapy kazıma sonuçları

Ünyekent sitesi Scrapy ile kazıma sonuçları Resim 3.40'ta görülmektedir.



Resim 3.40 Bizim Yöresel Scrapy kazıma sonuçları

4. BULGULAR

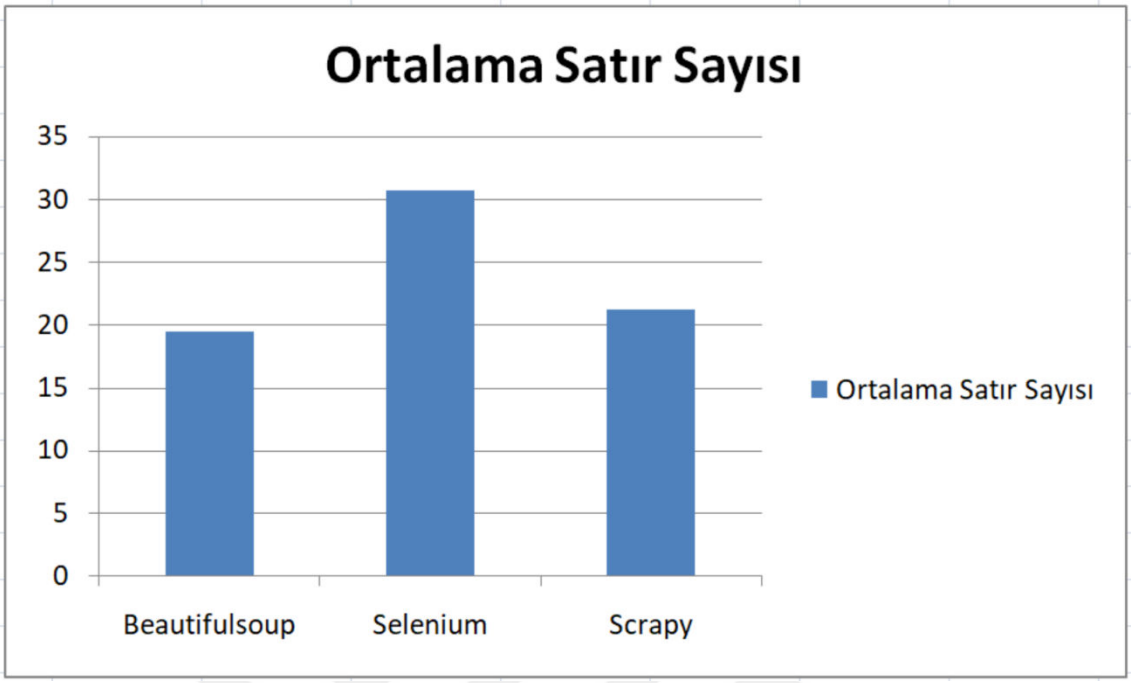
Bu tez çalışmasında, BeautifulSoup, Selenium ve Scrapy kütüphanelerini kullanarak aynı kazıma işlemini yapan kodlar yazılmıştır. Kodların verimliliğini ve performansını belirlemek için kodun karakter sayısı, satır sayısı ve ayrı ayrı platformlardaki hızı metrikleri kullanılmıştır.

4.1 Yazılan Kodların Uzunluğu

BeautifulSoup, Selenium ve Scrapy kütüphanelerini karşılaştırmak için yazılan, örnek sitelerden kazıma yapan kodların satır ve karakter sayıları Çizelge 4.1 ve Çizelge 4.2'de gösterilmiştir. Kütüphaneler ile yazılan kodların ortalama karakter ve satır sayılarını gösteren grafik Şekil 4.1'de yer almaktadır. BeautifulSoup ve Selenium için bağımsız bir kod dosyası yazılırken, Scrapy için yaratılan projenin içine bir spider dosyası yazılmıştır.

Çizelge 4.1 Yazılan Kodların Satır Sayısı

Kütüphane	**Beautifulsoup**	**Selenium**	**Scrapy**
Emlaksitem	19	28	19
Onurmarket	17	26	25
Ünyekent	24	40	23
Bizim Yöresel	18	29	18

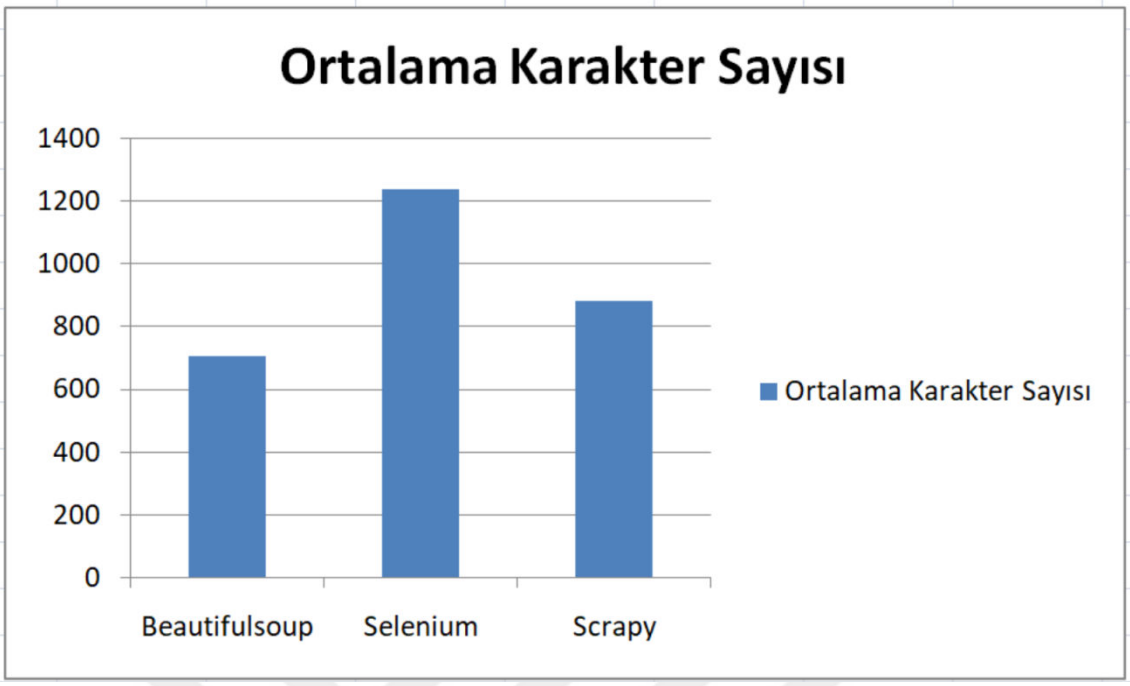


Şekil 4.1 Ortalama satır sayıları grafiği

Çizelge 4.2 Yazılan Kodların Karakter Sayısı

Kütüphane	BeautifulSoup	Selenium	Scrapy
Emlaksitem	709	1141	773
Onurmarket	643	1059	1025
Ünyekent	837	1634	947
Bizim Yöresel	635	1109	776

Kütüphaneler ile yazılan kodların ortalama satır sayılarını gösteren grafik Şekil 4.2'de yer almaktadır.



Şekil 4.2 Ortalama karakter sayıları grafiği

Selenium ile yazılan kodların, diğer kütüphaneler ile yazılan kodlara göre hem satır sayısı hem de karakter sayısı olarak daha fazla olduğu gözlemlenmiştir.

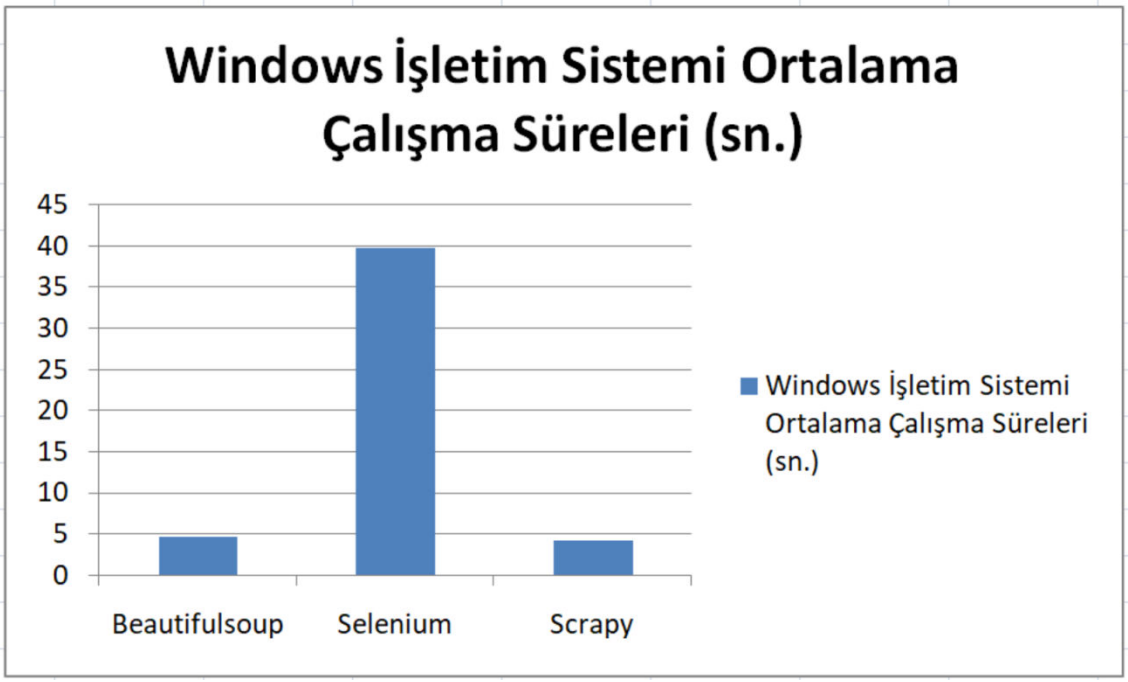
4.2 Yazılan Kodların Çalışma Süreleri

BeautifulSoup, Selenium ve Scrapy kütüphaneleri ile yazılan kodlar Çizelge 3.8'de belirtilen sıra ile 10 tekrar şeklinde çalıştırılmış, çalışma süreleri kaydedilmiştir. Kaydedilen değerlerin saniye cinsinden ortalamaları farklı işletim sistemleri için Çizelge 4.3, Çizelge 4.4 ve Çizelge 4.5'te görülmektedir. Kütüphanelere göre ortalama çalışma sürelerinin grafikleri Resim 4.3, Resim 4.4 ve Resim 4.5'te gösterilmiştir.

Selenium ile yazılan kodlarda, sayfanın yüklenmesi için 3 saniye beklenmesi kodun içinde belirtilmiştir. Süreler değerlendirilirken bu göz önüne alınmalıdır.

Çizelge 4.3 Windows İşletim Sisteminde Çalışma Süreleri Ortalamaları

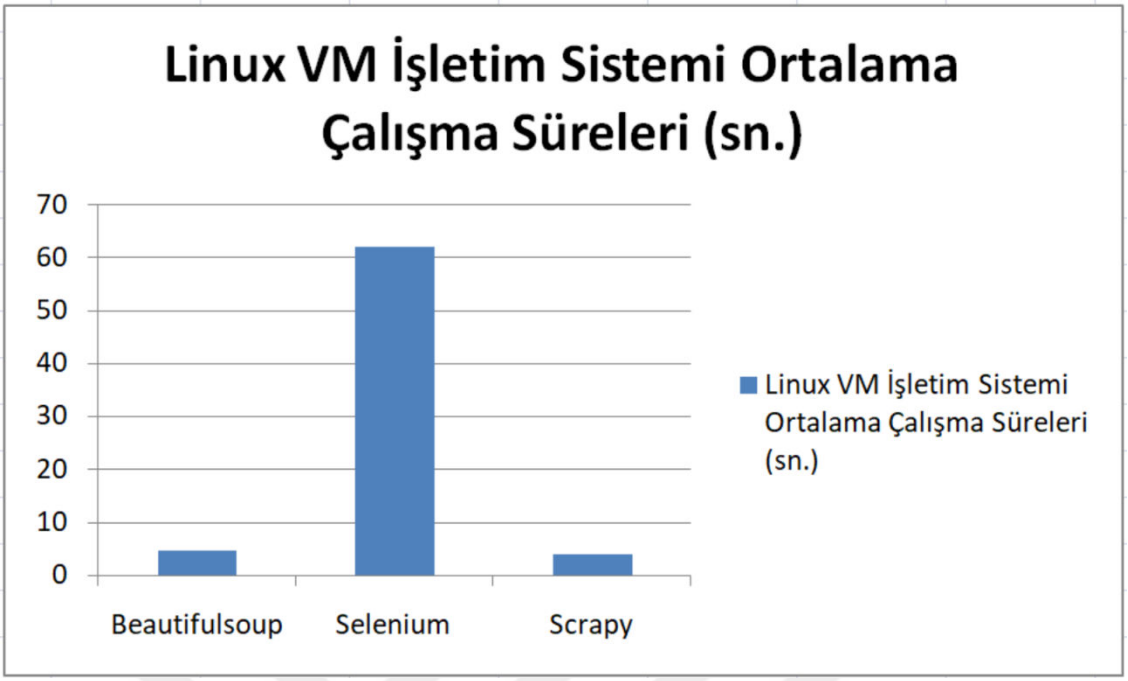
Kütüphane	BeautifulSoup	Selenium	Scrapy
Emlaksitem	0,53	10,79	0,96
Onurmarket	1,32	15,69	2,02
Ünyekent	16,60	120,59	13,27
Bizim Yöresel	0,27	11,88	0,79



Şekil 4.3 Windows işletim sistemi ortalama çalışma süreleri grafiği

Çizelge 4.4 Linux VM İşletim Sisteminde Çalışma Süreleri Ortalamaları

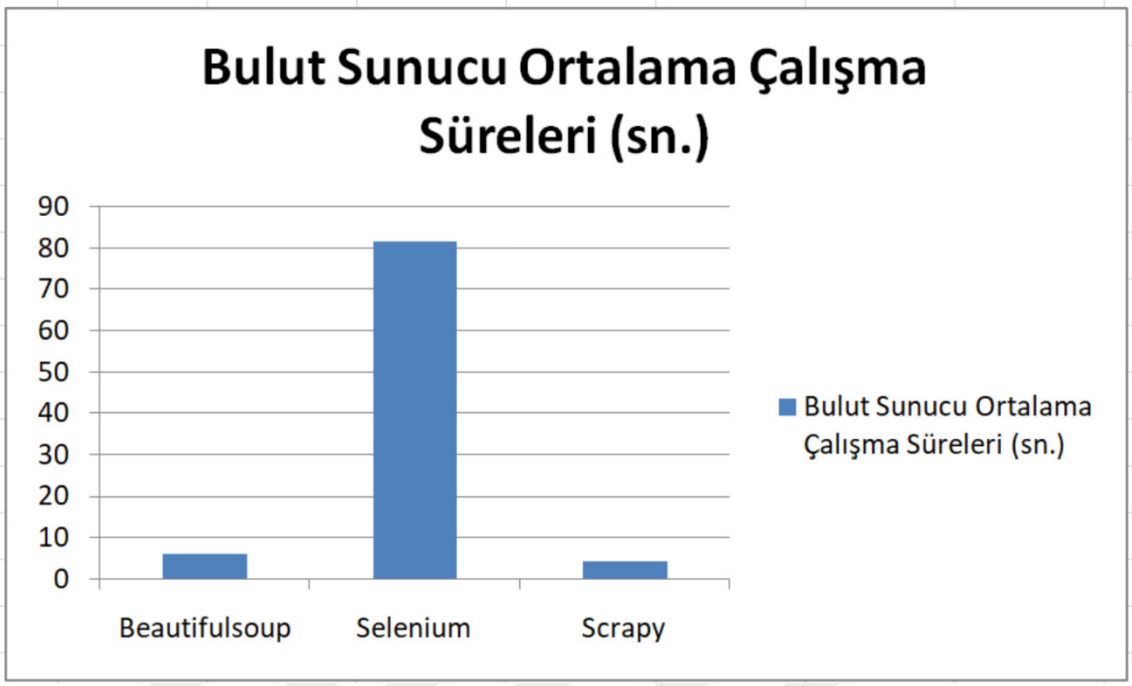
Kütüphane	BeautifulSoup	Selenium	Scrapy
Emlaksitem	0,61	17,80	0,81
Onurmarket	1,36	34,49	1,64
Ünyekent	16,23	173,45	13,03
Bizim Yöresel	0,31	21,56	0,57



Şekil 4.4 Linux VM işletim sistemi ortalama çalışma süreleri grafiği

Çizelge 4.5 Bulut Sunucuda Çalışma Süreleri Ortalamaları

Kütüphane	**BeautifulSoup**	**Selenium**	**Scrapy**
Emlaksitem	0,78	24,02	1,06
Onurmarket	1,31	39,66	1,47
Ünyekent	22,87	234,02	14,09
Bizim Yöresel	0,48	29,10	0,67



Şekil 4.5 Bulut sunucuda ortalama çalışma süreleri grafiği

Kaydedilen tüm ölçüm değerleri EK-15, EK-16 ve EK-17'de belirtilmiştir.

4.3 İşletim Sistemlerine Göre Süre Ortalamaları

BeautifulSoup, Selenium ve Scrapy kütüphaneleri ile yazılan kodların işletim sistemlerine göre saniye cinsinden ortalama çalışma süreleri Çizelge 4.6'da belirtilmiştir.

Çizelge 4.6 İşletim Sistemlerine Göre Çalışma Süreleri Ortalamaları

Site	Kütüphane	Windows	Linux VM	Bulut Sunucu
Emlaksitem	BeautifulSoup	0,53	0,61	0,78
Onurmarket	BeautifulSoup	1,32	1,36	1,31
Ünyekent	BeautifulSoup	16,60	16,23	22,87
Bizim Yöresel	BeautifulSoup	0,27	0,31	0,48
Emlaksitem	Selenium	10,79	17,80	24,02
Onurmarket	Selenium	15,69	34,49	39,66
Ünyekent	Selenium	120,59	173,45	234,02

Çizelge 4.6 (Devam) İşletim Sistemlerine Göre Çalışma Süreleri Ortalamaları

Site	Kütüphane	Windows	Linux VM	Bulut Sunucu
Bizim Yöresel	Selenium	11,88	21,56	29,10
Emlaksitem	Scrapy	0,96	0,81	1,06
Onurmarket	Scrapy	2,02	1,64	1,47
Ünyekent	Scrapy	13,27	13,03	14,09
Bizim Yöresel	Scrapy	0,79	0,57	0,67

4.4 Yazılan Kodlar ve Çalışma Sürelerinin Değerlendirilmesi

BeautifulSoup, Selenium ve Scrapy kütüphaneleri kullanarak yazılan kodlar arasında, BeautifulSoup en az satır ve karakter kullanılarak aynı sonucu vermiştir. Selenium ise üç kütüphane arasında daha fazla satır ve karakter kullanarak aynı sonuca ulaşabilmiştir. BeautifulSoup, Emlaksitem, Onur Market ve Bizim Yöresel sitelerinde en hızlı çalışan kütüphane olmuştur. Bu üç sitede de taranan tüm bilgiler tek bir html sayfasından alınmaktadır.

Ünyekent sitesi için en hızlı çalışan kütüphane Scrapy olarak gözlemlenmiştir. Bu haber sitesi taranırken, önce ana sayfadaki haberlerin linkleri belirlenip, bu linklere tek tek erişilip bilgi alınmaktadır. Birden çok sayfa erişimi gerektiren taramada Scrapy en hızlı kütüphane olarak öne çıkmıştır.

Selenium her senaryoda en yavaş çalışan kütüphane olmuştur. Selenium bilgi taramak için bir web tarayıcı simüle etmektedir. Bu sebeple öncelikler sayfanın ve içinde mevcutsa tüm dinamik içeriğin tamamen yüklenmesini beklemektedir. Bu da veri kazıma hızını yavaşlatmaktadır.

İşletim sistemi kodların çalışma sürelerini fazla etkilememektedir. Windows ve Linux işletim sisteminde çalıştırılan kodlar yakın sonuçlar vermiştir. Bulut sunucuda çalışan kodlar biraz daha yavaş sonuca ulaşmıştır. Bulut sunucu yurtdışında bulunduğu için test için belirlenen sitelere erişiminin daha yavaş olması beklenen bir sonuçtur.

5. TARTIŞMA ve SONUÇ

5.1 BeautifulSoup Kütüphanesinin Avantajları

BeautifulSoup, yeni başlayanlar için bile kolayca kurulabilen ve anlaşılabilen, kullanıcı dostu bir kütüphanedir. HTML veya XML içeriklerini işleme konusunda hızlıdır. Çeşitli belge yapılarını, hatta hatalı biçimlendirilmiş işaretlemeleri bile işleyebilir. Requests gibi diğer Python kütüphaneleriyle iyi çalışır. Kapsamlı dokümantasyonu ve destekleyici bir topluluğu vardır (İnt. Kyn.23).

HTML ve XML belgelerinden kolayca veri çıkarmak için ayrıştırma ağaçları oluşturur.find(), find_all(), select() ve select_one() gibi güçlü arama ve filtreleme yöntemleri sunar. Tarayıcılara ihtiyaç duymadan çalışır. Statik veya az dinamik web sitelerinden veri çekmek için uygundur. Web sitelerinden veri çekmek için hızlıca kod oluşturmanızı sağlar (İnt. Kyn.24).

Hafiftir ve az kaynak tüketir. Bozuk HTML'yi düzeltebilir ve sayfa kodlamasını otomatik olarak algılayabilir. HTML içeriğini yinelemek, aramak ve değiştirmek için basit yöntemler sunar (İnt. Kyn.25).

5.2 BeautifulSoup Kütüphanesinin Dezavantajları

Dinamik içerikleri işlemekte sınırlıdır. JavaScript ile dinamik olarak oluşturulan web sayfalarını ayrıştırmada yetersiz kalır. Web sayfalarıyla etkileşim kuramaz. Örneğin, butonlara tıklama, form doldurma gibi işlemleri yapamaz (İnt. Kyn.26).

Büyük miktarda veri ayrıştırmada diğer bazı kütüphanelere göre daha yavaş olabilir. Sadece Python'da çalışır, diğer programlama dillerini desteklemez. Web sayfalarından veri indirmek için 'Requests' veya 'HTTPx' gibi ek kütüphanelere ihtiyaç duyar (İnt. Kyn.27).

5.3 BeautifulSoup Kütüphanesi Hangi Tür Sitelerden Bilgi Kazımada Kullanılmalı

BeautifulSoup kütüphanesi, statik veya az dinamik web sitelerinden bilgi kazımak için idealdir. Bu kütüphane, HTML ve XML belgelerini ayrıştırma konusunda oldukça başarılıdır ve bu tür belgelerden veri çıkarmak için basit ve kullanıcı dostu bir yol sunar (İnt. Kyn.28).

İçeriği genellikle değişmeyen statik web siteleri ve sunucu tarafından doğrudan gönderilen HTML sayfaları için uygundur. Bu tür sitelerde, veriler HTML kodunda doğrudan bulunur ve kolayca çekilebilir.XML formatında sunulan verileri ayrıştırmak ve işlemek için de kullanılabilir. Küçük ve orta ölçekli web kazıma projelerinde kullanımı kolay ve etkilidir (İnt. Kyn.30).

JavaScript kullanılarak dinamik olarak oluşturulan veya sürekli değişen içeriklere sahip web sitelerinde yetersiz kalır. Bu tür sitelerde, veriler genellikle tarayıcıda JavaScript kodları çalıştırıldıktan sonra ortaya çıkar ve BeautifulSoup bu içeriğe doğrudan erişemez.Butonlara tıklama, form doldurma gibi kullanıcı etkileşimlerini gerektiren web sayfalarında kullanılamaz. Bu tür durumlarda Selenium gibi tarayıcı otomasyon araçları kullanmanın daha uygun olacağı değerlendirilmektedir (İnt. Kyn.31).

5.4 Selenium Kütüphanesinin Avantajları

Selenium, Python, Java, C#, Ruby ve JavaScript gibi birçok programlama dilini destekler. Bu, farklı geliştiricilerin kendi tercih ettikleri dilde web otomasyon ve kazıma işlemleri yapabilmesini sağlar. Chrome, Firefox, Safari, Edge gibi tüm büyük web tarayıcılarını destekler. Bu sayede farklı platformlarda ve tarayıcılarda web uygulamalarını test etmek ve veri çekmek mümkün olur. JavaScript ile dinamik olarak yüklenen içerikleri işleyebilir. Bu, özellikle modern web sitelerinde veri çekmek için çok önemlidir (İnt. Kyn.29).

Web sayfalarıyla kullanıcı gibi etkileşim kurabilir; butonlara tıklayabilir, formları doldurabilir ve sayfalar arasında gezinebilir. Web elementleri ile etkileşim kurmak için bir arayüz sağlar. Bu arayüz sayesinde web sayfalarındaki öğelere erişmek, tıklamak,

değerlerini değiştirmek gibi işlemler kolaylaşır.Tarayıcıyı görünür bir pencere açmadan çalıştırabilir. Bu, sunucu ortamlarında veya arka planda web kazıma işlemleri yapmak için idealdir (İnt. Kyn.33).

Büyük bir topluluğa ve kapsamlı dokümantasyona sahiptir. Bu sayede sorunlara çözüm bulmak ve yeni özellikler öğrenmek kolaylaşır. Web sayfalarının ekran görüntülerini alabilir. Hem web kazıma hem de web uygulamalarını test etme amacıyla kullanılabilir (İnt. Kyn.35).

5.5 Selenium Kütüphanesinin Dezavantajları

Özellikle WebDriver kullanımı, yeni başlayanlar için daha zor olabilir. Programlama kavramlarına aşina olmak gerekir. Kendi içinde raporlama özelliği yoktur, bu nedenle üçüncü taraf araçlara veya frameworklere ihtiyaç duyulur. Sürekli değişen dinamik web sayfalarında, testler veya kazıma işlemleri kararsız olabilir ve ek çaba gerektirebilir. Tarayıcı ile etkileşim kurduğu için bazı durumlarda daha yavaş çalışabilir (İnt. Kyn.36).

Tarayıcı sürücülerine (WebDriver) bağımlıdır ve tarayıcı güncellemeleri, sürücülerin de güncellenmesini gerektirebilir. Bazı web siteleri Selenium tarafından kontrol edildiğini tespit edebilir, bu da kazıma işlemlerini zorlaştırabilir (İnt. Kyn.37).

5.6 Selenium Kütüphanesi Hangi Tür Sitelerden Bilgi Kazımada Kullanılmalı

Selenium kütüphanesi, özellikle dinamik içeriğe sahip ve kullanıcı etkileşimi gerektiren web sitelerinden bilgi kazımak için idealdir. JavaScript kullanılarak oluşturulan ve içeriği sürekli değişen web sitelerinde etkilidir. Bu tür sitelerde, veriler genellikle tarayıcıda JavaScript kodları çalıştırıldıktan sonra ortaya çıkar ve Selenium, bu dinamik içeriğe erişebilir (İnt. Kyn.38).

Butonlara tıklama, form doldurma, kaydırma gibi kullanıcı etkileşimleri gerektiren web sayfalarında kullanılabilir. Selenium, bu etkileşimleri simüle ederek gerekli verilerin elde edilmesini sağlar. Web uygulamalarındaki verileri çekmek için kullanışlıdır. Özellikle, web uygulamalarının test edilmesi ve otomasyonu için yaygın olarak

kullanılır. Birden fazla etkileşim ve dinamik içeriğin olduğu karmaşık web kazıma projelerinde başarılıdır. Bazı siteler sık sık sayfa yapısını değiştirebilir ve bu durumda, Selenium, sayfa yapısındaki değişikliklere uyum sağlayarak kazıma işlemini devam ettirebilir (İnt. Kyn.39).

Selenium, İçeriği değişmeyen basit HTML sayfalarından veri çekmek için uygun değildir. Bu durumlarda BeautifulSoup gibi daha basit ve hızlı kütüphaneler tercih edilebilir. Küçük ölçekli ve statik verilerin çekileceği projelerde Selenium yerine daha hafif ve hızlı çözümler kullanılabilir (İnt. Kyn.40).

5.7 Scrapy Kütüphanesinin Avantajları

Scrapy, büyük ölçekli web kazıma projeleri için yüksek verimlilik sunar. Asenkron (eşzamansız) çalışma özelliği sayesinde, isteklerin işlenmesi için beklemeden daha hızlı veri toplama imkanı sağlar. Middleware ve pipeline gibi özellikleri sayesinde kazıma sürecini özelleştirme ve genişletme imkanı sunar. İhtiyaçlarınıza göre farklı veri işleme adımları ve veri depolama seçenekleri ekleyebilirsiniz. Çerezler (cookies), oturumlar (sessions) ve kullanıcı-aracı (user-agent) taklit etme gibi özellikleri yerleşik olarak destekler. Bu, web sitelerinin engelleme önlemlerini aşmaya yardımcı olur.Veri çıkarma ve depolama için yerleşik destek sunar. Çıkarılan veriyi temizleme, doğrulama ve veri tabanına kaydetme gibi işlemleri kolayca yapabilirsiniz. Veri çıkarmak için CSS ve XPath seçicilerini kullanma imkanı sunar. Bu, web sayfalarındaki belirli öğelere kolayca ulaşmanızı sağlar. Web sitelerini nasıl gezileceğini, hangi verilerin çıkarılacağını ve linkleri nasıl takip edeceğini tanımlayan "spiders" sınıflarını kullanır. Bu sayede, farklı siteler için özel kazıma stratejileri oluşturulabilir (İnt. Kyn.32).

Asenkron işlemleri destekler, bu da performansı artırır. Bir isteğin tamamlanmasını beklemeden diğer isteklere geçebilir. Veriyi CSV, JSON, XML gibi çeşitli formatlarda kaydetme ve FTP (File Transfer Protocol - Dosya Transfer Protokolü), S3 veya yerel dosya sistemine depolama imkanı sunar. Hatalı istekler için otomatik yeniden deneme mekanizması sunar. Bu, kazıma işleminin kesintisiz devam etmesine yardımcı olur. Scrapy Shell, spider'ı çalıştırmadan kazıma kodunu test etmeye yarayan bir araçtır. Bu,

hataları ayıklamak ve doğru seçicileri bulmak için kullanışlıdır.Web sitelerinde linkleri takip ederek sayfaları tarayabilen crawlspider özelliğini destekler (İnt. Kyn.34).

5.8 Scrapy Kütüphanesinin Dezavantajları

Özellikle yeni başlayanlar için daha zor öğrenilir. Scrapy'nin yapısını ve özelliklerini anlamak zaman alabilir. Basit web kazıma görevleri için aşırı karmaşık ve fazla özellikli olabilir. Küçük projeler için daha basit araçlar (örneğin, BeautifulSoup) daha uygun olabilir. JavaScript ile oluşturulan dinamik içerikleri işleme konusunda Selenium kadar güçlü değildir. Kurulumu ve proje yapısı, basit kütüphanelere göre daha karmaşıktır (İnt. Kyn.41).

5.9 Scrapy Kütüphanesi Hangi Tür Sitelerden Bilgi Kazımada Kullanılmalı

Scrapy, çok sayıda sayfadan veri çekmek gerektiğinde, asenkron yapısı sayesinde hızlı ve verimli çalışır. Bu özelliği, büyük miktarda veriyi kısa sürede toplamak isteyenler için ideal kılar. Scrapy, birden fazla sayfayı takip etme, linkleri analiz etme ve verileri çıkarma gibi karmaşık işlemleri yönetebilir. Bu özelliği sayesinde, web sitelerinin derinliklerine inerek gerekli verileri toplama imkanı sunar. Scrapy, web sitelerinin API'leri ile etkileşim kurarak veri toplama işlemlerini gerçekleştirebilir. Scrapy, çıkarılan verileri temizleme, doğrulama ve farklı formatlara dönüştürme gibi işlemleri kolaylaştıran pipeline (işlem hattı) özelliğine sahiptir. Bu, veriyi analiz için hazırlama sürecini kolaylaştırır. Scrapy, farklı web sitelerinden veri toplama ve bu verileri birleştirme imkanı sunar. Farklı kaynaklardan veri çekmek ve bunları tek bir yerde toplamak isteyen projeler için uygundur. Scrapy, web sitelerindeki linkleri takip ederek sayfaları tarama yeteneğine sahiptir. Bu özellik, belirli bir web sitesindeki tüm sayfaları veya belirli bir kategorideki sayfaları taramak isteyenler için idealdir. Eğer belirli aralıklarla aynı web sitelerinden veri çekmek gerekiyorsa, Scrapy'nin otomatik yeniden deneme özelliği ve zamanlanabilir görevleri sayesinde bu süreç kolaylaşır. Scrapy’nin, büyük ölçekli, karmaşık ve dinamik olmayan web sitelerinden veri kazımak için daha uygun bir seçenek olduğu değerlendirilmektedir (İnt. Kyn.42).

Eğer sadece birkaç sayfadan statik veri çekmeniz gerekiyorsa, BeautifulSoup gibi daha basit kütüphaneler daha hızlı ve yeterli olabilir. Eğer web sitesi yoğun olarak JavaScript kullanıyorsa ve dinamik içerikler oluşturuyorsa, Selenium gibi tarayıcı otomasyon araçları daha uygun olabilir. Scrapy, bu tür sitelerde Selenium kadar etkili olmayabilir (İnt. Kyn.43).

5.10 BeautifulSoup, Selenium ve Scrapy Kütüphanelerinin Karşılaştırmalı Değerlendirmesi

Bu üç kütüphane, web kazıma (web scraping) alanında farklı ihtiyaçlara yönelik çözümler sunar. Aşağıda, bu kütüphanelerin özelliklerini, avantajlarını, dezavantajlarını ve kullanım alanlarını içeren bir karşılaştırma Çizelge 5.1'de görülebilir.

Çizelge 5.1 Üç Kütüphanenin Karşılaştırılması

Özellik	**BeautifulSoup**	**Selenium**	**Scrapy**
Temel İşlevi	HTML/XML ayrıştırma ve veri çekme	Web tarayıcı otomasyonu ve etkileşim	Büyük ölçekli web kazıma ve tarama
Kullanım Alanları	Statik siteler, basit projeler	Dinamik siteler, etkileşim gerektiren projeler	Büyük ölçekli projeler, karmaşık yapılı siteler
Dinamik İçerik Desteği	Yok	Var	Sınırlı
Tarayıcı Etkileşimi	Yok	Var	Yok
Hız	Hızlı	Yavaş	Hızlı
Öğrenme Kolaylığı	Kolay	Orta	Zor
Özelleştirilebilirlik	Sınırlı	Orta	Yüksek
Veri İşleme	Basit	Basit	Gelişmiş
Uygun Olduğu Projeler	Küçük/orta ölçekli, statik veri çekme projeleri	Dinamik içerik, etkileşimli siteler	Büyük ölçekli, karmaşık, veri işleme gerektiren projeler
Ek Kütüphane İhtiyacı	requests veya httpx (veri indirme için)	Yok	Yok

BeautifulSoup, hızlı ve basit web kazıma projeleri için idealdir. Statik web sitelerinden veri çekmek ve temel HTML/XML ayrıştırma işlemleri için uygundur.

Selenium, dinamik içerik gerektiren ve kullanıcı etkileşimlerinin önemli olduğu projeler için en iyi seçenektir. Web uygulamalarını test etmek ve otomatikleştirmek için de kullanılabilir.

Scrapy, büyük ölçekli ve karmaşık web kazıma projelerinde, yüksek performans ve özelleştirilebilirlik gerektiren durumlarda tercih edilebilir. Çok sayıda sayfayı tarama, verileri işleme ve farklı kaynaklardan veri toplama gibi ihtiyaçlar için Scrapy uygun bir çözüm olarak düşünülebilir.

Hangi kütüphanenin kullanılacağı, projenin gereksinimlerine ve web sitesinin yapısına bağlıdır. Basit projeler için BeautifulSoup yeterli olabilirken, daha karmaşık ve dinamik siteler için Selenium veya Scrapy tercih edilebilir.

5.11 Öneriler

Web kazıma, günümüzde büyük veri analizi, makine öğrenimi ve veri madenciliği gibi çeşitli alanlarda kritik bir rol oynamaktadır. Yapay zeka (Artificial Intelligence) teknolojilerinin gelişimiyle birlikte, web kazıma süreçlerinde de önemli iyileştirmeler ve yeni kullanım alanları ortaya çıkmıştır. AI, web kazıma süreçlerini daha akıllı, verimli ve otomatik hale getirme potansiyeline sahiptir.

AI algoritmaları, web kazıma ile toplanan verilerin kalitesini artırmada kullanılabilir. Örneğin, NLP (Natural Language Processing – Doğal Dil İşleme) teknikleri, web sayfalarındaki metin verilerini analiz ederek anlamlı bilgileri ayıklayabilir, gürültülü verileri filtreleyebilir ve veri setlerini daha tutarlı hale getirebilir. AI destekli web kazıma araçları, JavaScript ve diğer dinamik teknolojilerle oluşturulan web sayfalarındaki içerikleri daha etkin bir şekilde işleyebilir. Bu sayede, dinamik web sitelerinden veri toplama zorluğu aşılabilir. AI, web kazıma araçlarının web sitelerinin yapısını otomatik olarak anlamasına ve buna göre veri çekmesine yardımcı olabilir. Bu,

manuel olarak yapılandırma ihtiyacını azaltarak kazıma süreçlerini daha esnek ve uyarlanabilir hale getirebilir. AI algoritmaları, web kazıma ile toplanan verileri analiz ederek anlamlı sonuçlar çıkarabilir. AI, web kazıma süreçlerinde etik ve yasal konulara daha fazla dikkat edilmesini sağlayabilir. AI algoritmaları, web kazıma ile elde edilen verilerden otomatik olarak içerik üretebilir. Örneğin, farklı kaynaklardan haberler toplanarak özet haber bültenleri oluşturulabilir veya e-ticaret sitelerindeki ürün bilgileri otomatik olarak düzenlenerek karşılaştırma tabloları oluşturulabilir.

Bu tez çalışmasında incelenen BeautifulSoup, Selenium ve Scrapy gibi web kazıma kütüphaneleri, AI teknolojileriyle entegre edilerek daha güçlü hale getirilebilir. Özellikle, Selenium'un dinamik içerik işleme yeteneği ve Scrapy'nin büyük ölçekli veri toplama performansı, AI uygulamaları için önemli bir temel oluşturabilir. Gelecekteki çalışmalar, web kazıma süreçlerinde AI'ın daha etkin bir şekilde nasıl kullanılabileceğine odaklanmalıdır. AI destekli web kazıma araçları, sadece veri toplama süreçlerini hızlandırmakla kalmayacak, aynı zamanda daha anlamlı, analiz edilebilir ve değerli bilgiler elde etmemizi sağlayacaktır.

6. KAYNAKLAR

Abdillah A, Rahmatulloh A, Widiyasono N, Rizal R, 2024, Designing A Telegram Bot With Web Scraping Capabilities For Automating Scientific Article Downloads, Journal of Advanced Computing Technology and Application, 6(1), 37-49.

Agüero-Torales, M M, Cobo M J, Herrera-Viedma E, López-Herrera A G, 2019, A Cloud-Based Tool For Sentiment Analysis In Reviews About Restaurants On TripAdvisor, Procedia Computer Science, 162, 392-399.

Akbar A S I, Ramadhani C, 2024, Marketplace Data Analysis Using Web Scraping, Dielektrika, 11(2), 154-158.

Al-Madhhachi Z T, Mahmood S A, 2024, Web Scraping Scientific Repositories: Springerand Nature for University of Basrah, Journal of Al-Qadisiyah for Computer Science and Mathematics, 16(1), 1-8.

Almaqbali I S H, Al Khufairi F M A, Khan M S, Bhat A Z, Ahmed I, 2019, Web Scrapping: Data Extraction from Websites, Journal of Student Research.

Altamimi M, Alayba A M, 2023, ANAD: Arabic news article dataset, Data in Brief, 50, 109460.

Anbu A, Miriam D D H, Robin C R, 2024, A Comprehensive Web Scraping of IMDb's Top 50 Movies using Beautiful Soup, 2024 International Conference on Communication Computing and Internet of Things (IC3IoT) (pp. 1-9), IEEE.

Brown M A, Gruen A, Maldoff G, Messing S, Sanderson Z, Zimmer M, 2024, Web Scraping for Research: Legal Ethical Institutional and Scientific Considerations, arXivpreprint arXiv:2410, 23432.

Cavallo A, Rigobon R, 2016, The Billion Prices Project: Using Online Prices for Measurement and Research, Journal of Economic Perspectives, 30(2), 151-178.

Chandrawat A S, Kumar N, Bohra V, 2024, Optimizing E-Commerce Decision Making Using Web Scraping, Authorea Preprints.

Chataut S, Dangi N, Pakka N, Nepal N, Rauniyar K, 2024, Generative AI-Driven Automated News Content Generation: Integrating Web Scraping Media Creation

and Social Media Optimization, International Journal of Science Engineering and Technology, 12-6.

Davenport T H, Prusak L, 1998, Workingknowledge: How Organizations Manage What They Know, Harvard Business Press.

Farias W A, Melo D M, dos Santos L M, de Oliveira Â A, Medeiros R L, Silva Y K, 2024, Web Scraping as a Sientific Tool for Theoretical Reference, In Press.

Galvez-Hernandez P, Gonzalez-Viana A, Gonzalez-de Paz L, Shankardass K, Muntaner C, 2024, Generating Contextual Variables From Web-Based Data for Health Research: Tutorial on Web Scraping Text Mining and Spatial Overlay Analysis, JMIR Public Health and Surveillance, 10, e50379.

Gheorghe M, Mihai F C, Dârdală M, 2018, Modern Techniques of Web Scraping for Data Scientists, International Journal of User-System Interaction, 11(1), 63-75.

Goulas S, Karamitros G, 2024, How to Harness The Power of Web Scraping for Medical and Surgical Research: An Application in Estimating International Collaboration, World Journal of Surgery, 48(6), 1297-1300.

Hajba G L, 2018, Website Scraping with Python, Berkeley: Apress.

Khder M A, 2021, Web Scrapingor Web Crawling: State of Art Techniques Approaches and Application, International Journal of Advances in Soft Computing & Its Applications, 13(3).

Kumar V V, TamizhSelvan C, Manoharan L, Sekhar R V, Kumar V V, 2024, Utilizing Web Scraping and Machine Learning for Improving Accuracy of Smartphone Price Prediction, International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, p.p. 1-5.

Kuyucuk T, Çallı L, 2022, Using Multi-Label Classification Methods to Analyze Complaints Against Cargo Services During the COVID-19 Outbreak: Comparing Survey-Based and Word-Based Labeling, Sakarya University Journal of Computer and Information Sciences, 5(3), 371-384.

Lawson R, 2015, Web Scraping with Python, Packt Publishing Ltd.

Meyberg C, Rendtel U, Leerhoff H, 2024, Flat Rent Price Prediction in Berlin with Web Scraping, AStAWirtschafts-und Sozialstatistisches Archiv, 1-34.

Naing I, Aung S T, Wai K H, Funabiki N, 2024, A Reference Paper Collection System Using Web Scraping, Electronics, 13(14), 2700.

Özen İ A, Karadeniz G, Zaro E, 2022, Kadın Girişimci Restoranlar ve Turist Tercihlerinin Belirlenmesi: Kapadokya Örneği.

Pandey P, Jo H, Tseng A, 2024, Adaptingto AI: Approaches for Digital Publishers in Managing Web Scraping, Available at SSRN 4714744.

Pant S, Yadav E N, Sharma M, Bedi Y, Raturi A, 2024, Web Scraping Using Beautiful Soup, International Conference on Knowledge Engineering and Communication Systems (ICKECS), (Vol. 1, pp. 1-6), IEEE.

Pillai P, Amin D, 2020, Understanding the Requirements of The Indian IT Industry Using Web Scrapping, Procedia Computer Science, 172, 308-313.

Plotnikov A, Shcheludyakov A, Cherdantsev V, Bochkarev A, Zagoruiko I, 2020, Data on Post Bank Customer Reviews from Web, Data in Brief, 32, 106152.

Prasetyani M, Isnanto R R, Widodo C E, 2024, Analyzing Road Users' Behavior: A Data Mining Approach Using Google Maps Popular Time and Web Scraping for Rest Area Visitation Patterns on Highways and Toll Roads, Ingénieriedes Systèmes d'Information, 29(5).

Rodrigo A, 2024, Flight Price Prediction Website: Development of a Flight Prediction Website Using Web Scraping and Neural Networks.

Seliverstov Y, Seliverstov S, Malygin I, Korolev O, 2020, Traffic Safety Evaluation in Northwestern Federal Districtusing Sentiment Analysis of Internet Users' Reviews, Transportation research procedia, 50, 626-635.

Sharma K, Borkar G M, 2023, Comparative Analysis of Dynamic Web Scraping Strategies: Evaluating Techniques for Enhanced Data Acquisition, Advancements in Communication and Systems, 241-252.

Sirisuriya D S, 2015, A Comparative Study on Web Scraping, Proceedings of 8th International Research Conference, KDU

Syed A A, Gaol F L, Boediman A, Matsuo T, Budiharto W, 2023, A Data Package for Abstractive Opinion Summarization Title Generation and Rating-based Sentiment Prediction for Airline Reviews, Data in Brief, 50, 109535.

Üzümcü A, Eligüzel N, 2023, Predictive Analysis Using Web Scraping for the Real Estate Market in Gaziantep, Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 12(1), 17 - 24.

VandenBroucke S, Baesens B, 2018, Practical Web Scraping for Data Science, (pp. 3-5), New York, NY: Apress.

Wibawa E G, Dewa P K, 2024, Boarding House Property Market Trendsand Investor Preferences in Boarding House Development: A Comparative Study with Web Scraping, Jurnal Rekayasa Industri (JRI), 6(1), 85-99.

Yılmaz S, Selvi İ H, 2023, Price Prediction Using Web Scraping and Machine Learning Algorithms in the Used Car Market, Sakarya University Journal of Computer and Information Sciences, 6(2), 140-148.

Yüksel D, 2023, Çevrimiçi Müşteri Yorumları Üzerine Bir İçerik Analizi: E-Ticaret Müşteri Şikâyetleri, Üçüncü Sektör Sosyal Ekonomi Dergisi, 58(3), 2573-2587.

Zhekova M, Yumer E, 2024, JavaScript Web Scraping Tool for Extraction Information from Agriculture Websites, In BIO Web of Conferences, (Vol. 102, p. 03008), EDP Sciences.

Zoszak K, Batterham M, Simpson-Yap S, Probst Y, 2024, Web Scraping of User-simulated Online Nutrition in Formation for People with Multiple Sclerosis, Multiple Sclerosis and Related Disorders, 105746.

İnternet Kaynakları

1- https://www.sas.com/en_ca/insights/articles/analytics/using-big-data-to-predict-suicide-risk-canada.html#/ , 07.02.2024
2- https://en.wikipedia.org/wiki/Python_(programming_language) , 28.10.2024
3- https://python.plainenglish.io/pythons-advantages-disadvantages-72fefe70dd9e, 28.10.2024
4- https://en.wikipedia.org/wiki/Selenium_(software) , 28.10.2024
5- https://www.selenium.dev/documentation/overview/ , 28.10.2024
6- https://www.geeksforgeeks.org/navigating-links-using-get-method-selenium-python/, 29.10.2023
7- https://www.geeksforgeeks.org/locating-single-elements-in-selenium-python/, 29.10.2024
8- https://www.geeksforgeeks.org/locating-multiple-elements-in-selenium-python/, 29.10.2024
9- https://www.geeksforgeeks.org/action-chains-in-selenium-python/, 29.10.2024
10- https://www.crummy.com/software/BeautifulSoup/bs4/doc/, 30.10.2024
11- https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_installation.htm, 30.10.2024
12-https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_kinds_of_objects.htm , 30.10.2024
13-https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_navigating_by_tags.htm, 30.10.2024
14-https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_find_elements_by_id.htm, 30.10.2024
15-https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_find_elements_by_class.htm, 30.10.2024
16-https://www.tutorialspoint.com/beautiful_soup/beautiful_soup_find_elements_by_attribute.htm, 30.10.2024
17- https://docs.scrapy.org/en/latest/, 31.10.2024
18- https://docs.scrapy.org/en/latest/intro/tutorial.html, 1.11.2024
19- https://www.techcareer.net/dictionary/html, 2.11.2024
20- https://tr.wikipedia.org/wiki/HTML, 2.11.2024

21- https://www.turhost.com/blog/css-nedir-css-kodlari-nelerdir/, 2.11.2024
22- https://www.w3schools.com/cssref/css_selectors.php, 3.11.2024
23- https://medium.com/python-in-plain-english/15-python-based-web-scraping-libraries-you-should-know-about-08ebdf0f52d9, 20.12.2024
24- https://medium.com/latinxinai/5-essential-python-libraries-for-web-scraping-in-2024-e955ef64694a, 15.11.2024
25- https://medium.com/@JailynUpdates/do-web-scraping-data-mining-automation-python-programming-2024-c9c06fa68665, 20.12.2024
26- https://blog.stackademic.com/mastering-web-scraping-with-beautifulsoup-a-practical-guide-for-python-developers-32f75af5544d, 20.12.2024
27- https://medium.com/@udofiaetietop/webscrapping-beautifulsoup-or-selenium-3467edb3c0d9, 20.12.2024
28- https://python.plainenglish.io/build-a-web-scraper-in-python-c0ae9d2f86de, 20.12.2024
29- https://python.plainenglish.io/how-to-scrape-a-website-step-by-step-3217bba095ef, 20.12.2024
30- https://medium.com/@fatihazir/beautifulsoup-kütüphanesi-ile-python-kullanarak-web-scraping-uygulamasi-65755f1da27e, 20.12.2024
31- https://python.plainenglish.io/library-of-the-week-7-beautifulsoup-4-e904c7eebaca, 20.12.2024
32- https://medium.com/@pankaj_pandey/web-scraping-using-python-for-dynamic-web-pages-and-unveiling-hidden-insights-8dbc7da6dd26, 20.12.2024
33- https://medium.com/@thoren.lederer/automate-your-web-scraping-with-docker-schedule-python-selenium-scripts-on-cron-and-watch-the-a15511701a75, 20.12.2024
34- https://medium.com/@abdur-rahman/top-10-python-libraries-for-web-scraping-you-should-master-in-2025-3f1dc7b5ff01, 20.12.2024
35- https://medium.com/pythoneers/10-web-scraping-projects-using-selenium-and-python-22fa7a85804f, 20.12.2024
36- https://eodhd.medium.com/scraping-financial-data-with-python-c906b226ca90, 20.12.2024
37- https://medium.com/@megh.gala765/web-scraping-with-selenium-beautifulsoup-and-csv-in-python-3153f338ee93, 20.12.2024

38- https://python.plainenglish.io/dynamic-web-scraping-with-selenium-in-python-a-guided-demonstration-d5947c02a796, 21.12.2024
39- https://medium.com/@pawaraniket94334/web-scraping-using-selenium-3c8dadece905, 21.12.2024
40- https://ynevet.medium.com/web-scrapping-using-python-store-in-parquet-file-24756f6770e9, 21.12.2024
41- https://medium.com/pythons-gurus/python-web-scraping-best-libraries-and-practices-86344aa3f703, 21.12.2024
42- https://adzic-tanja.medium.com/a-web-scraping-project-with-scrapy-bba1e2037c4d, 21.12.2024
43- https://www.zenrows.com/blog/web-crawler-python, 21.12.2024

EKLER

EK 1. Örnek Bir Spider Kodu

```
from pathlib import Path
import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            "https://quotes.toscrape.com/page/1/",
            "https://quotes.toscrape.com/page/2/",
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = f"quotes-{page}.html"
        Path(filename).write_bytes(response.body)
        self.log(f"Saved file {filename}")
```

EK 2. HTML Etiketleri Listesi

<!--...--> Bir yorumu tanımlar
<!DOCTYPE> Belge türünü tanımlar
<a> Bir köprü metni tanımlar
<abbr> Bir kısaltma veya baş harf kısaltması tanımlar
<acronym> HTML5'te desteklenmez. Bunun yerine <abbr> kullanılmalıdır. Bir baş harf kısaltması tanımlar
<address> Bir belgenin yazarı/sahibi için iletişim bilgilerini tanımlar
<applet> HTML5'te desteklenmez. Bunun yerine <embed> veya <object> kullanılmalıdır. Gömülü bir baş harf tanımlaması yapar
<area> Bir resim haritasının içindeki bir alanı tanımlar
<article> Bir makale tanımlar
<aside> Sayfa içeriğinden ayrı bir içerik tanımlar
<audio> Gömülü ses içeriğini tanımlar
<b> Kalın metni tanımlar
<base> Bir belgedeki tüm göreli URL'ler için temel URL/hedefi belirtir
<basefont> HTML5'te desteklenmez. Bunun yerine CSS kullanılmalıdır. Bir belgedeki tüm metinler için varsayılan bir renk, boyut ve yazı tipi belirtir
<bdi> Metnin, dışındaki diğer metinden farklı bir yönde biçimlendirilebilecek bir bölümünü izole eder
<bdo> Geçerli metin yönünü geçersiz kılar
<big> HTML5'te desteklenmez. Bunun yerine CSS kullanılmalıdır. Büyük metni tanımlar
<blockquote> Başka bir kaynaktan alıntılanan bir bölümü tanımlar
<body> Belgenin gövdesini tanımlar

 Tek bir satır sonu tanımlar
<button> Tıklanabilir bir düğme tanımlar
<canvas> Komut dosyası (genellikle JavaScript) aracılığıyla anında grafik çizmek için kullanılır
<caption> Bir tablo başlığı tanımlar

EK 2. (Devam) HTML Etiketleri Listesi

 HTML5'te desteklenmez. Bunun yerine CSS kullanılmalıdır. Ortalanmış metni tanımlar
<cite> Bir eserin başlığını tanımlar
<code> Bir bilgisayar kodu parçasını tanımlar
<col> Bir <colgroup> öğesindeki her sütun için sütun özelliklerini belirtir
<colgroup> Biçimlendirme için bir tabloda bir veya daha fazla sütun grubunu belirtir
<data> Belirli bir içeriğin makine tarafından okunabilir bir çevirisini ekler
<datalist> Giriş denetimleri için önceden tanımlanmış seçeneklerin bir listesini belirtir
<dd> Bir açıklama listesindeki bir terimin açıklamasını/değerini tanımlar
<del> Bir belgeden silinen metni tanımlar
<details> Kullanıcının görüntüleyebileceği veya gizleyebileceği ek ayrıntıları tanımlar
<dfn> İçerik içinde tanımlanacak bir terimi belirtir
<dialog> Bir iletişim kutusu veya pencere tanımlar
<dir> HTML5'te desteklenmez. Bunun yerine <ul> kullanılır. Bir dizin listesi tanımlar
 Bir belgedeki bölümü tanımlar
<dl> Bir açıklama listesi tanımlar
<dt> Bir açıklama listesindeki bir terimi/adı tanımlar
 Vurgulanan metni tanımlar
<embed> Harici bir uygulama için bir kapsayıcı tanımlar
<fieldset> Bir formdaki ilgili öğeleri gruplandırır
<figcaption> Bir <figure> öğesi için bir başlık tanımlar
<figure> Bağımsız içeriği belirtir
<font> HTML5'te desteklenmez. Bunun yerine CSS kullanılır.Metin için yazı tipi, renk ve boyutu tanımlar
<footer> Bir belge veya bölüm için bir altbilgi tanımlar
<form> Kullanıcı girişi için bir HTML formu tanımlar
<frame> HTML5'te desteklenmez.Bir çerçeve kümesinde bir pencere (çerçeve) tanımlar
<frameset> HTML5'te desteklenmez.Bir çerçeve kümesini tanımlar
<h1> ila <h6> HTML başlıklarını tanımlar
<head> Belge için meta veri/bilgi içerir
<header> Bir belge veya bölüm için bir başlık tanımlar

EK 2. (Devam) HTML Etiketleri Listesi

<hgroup> Bir başlık ve ilgili içerik tanımlar
<hr> İçerikte tematik bir değişikliği tanımlar
<html> Bir HTML belgesinin kökünü tanımlar
<i> Metnin bir bölümünü alternatif bir ses veya ruh haliyle tanımlar
<iframe> Satır içi bir çerçeve tanımlar
<img> Bir resmi tanımlar
<input> Bir giriş denetimini tanımlar
<ins> Bir belgeye eklenen bir metni tanımlar
<kbd> Klavye girişini tanımlar
<label> Bir <input> öğesi için bir etiket tanımlar
<legend> Bir <fieldset> öğesi için bir başlık tanımlar
 Bir liste öğesini tanımlar
<link> Bir belge ile harici bir kaynak arasındaki ilişkiyi tanımlar (çoğunlukla stile bağlanmak için kullanılır) sayfalar)
<main> Bir belgenin ana içeriğini belirtir
<map> Bir görüntü haritasını tanımlar
<mark> İşaretli/vurgulanmış metni tanımlar
<menu> Sıralanmamış bir listeyi tanımlar
<meta> Bir HTML belgesi hakkında meta verileri tanımlar
<meter> Bilinen bir aralıktaki skaler bir ölçümü tanımlar (bir gösterge)
<nav> Gezinme bağlantılarını tanımlar
<noframes> HTML5'te desteklenmez. Çerçeveleri desteklemeyen kullanıcılar için alternatif bir içerik tanımlar
<noscript> İstemci tarafı betiklerini desteklemeyen kullanıcılar için alternatif bir içerik tanımlar
<object> Harici bir uygulama için bir kapsayıcı tanımlar
 Sıralı bir liste tanımlar
<optgroup> Açılır listede ilgili seçenekler grubunu tanımlar
<option> Açılır listede bir seçeneği tanımlar
<output> Bir hesaplamanın sonucunu tanımlar
<p> Bir paragraf tanımlar

EK 2. (Devam) HTML Etiketleri Listesi

<param> Bir nesne için bir parametre tanımlar
<picture> Birden fazla resim kaynağı için bir kapsayıcı tanımlar
<pre> Önceden biçimlendirilmiş metni tanımlar
<progress> Bir görevin ilerlemesini temsil eder
<q> Kısa bir alıntı tanımlar
<rp> Ruby açıklamalarını desteklemeyen tarayıcılarda ne gösterileceğini tanımlar
<rt> Karakterlerin açıklamasını/telaffuzunu tanımlar (Doğu Asya tipografisi için)
<ruby> Ruby açıklamasını tanımlar (Doğu Asya tipografisi)
<s> Artık doğru olmayan metni tanımlar
<samp> Bir bilgisayar programından örnek çıktıyı tanımlar
<script> Bir istemci tarafı betiğini tanımlar
<search> Bir arama bölümünü tanımlar
<section> Bir belgedeki bir bölümü tanımlar
<select> Bir açılır listeyi tanımlar
<small> Daha küçük metni tanımlar
<source> Medya öğeleri için birden fazla medya kaynağı tanımlar (<video> ve <audio>)
 Bir belgedeki bir bölümü tanımlar
<strike> HTML5'te desteklenmez. Bunun yerine <del> veya <s> kullanılır. Üstü çizili metni tanımlar
 Önemli metni tanımlar
<style> Bir belge için stil bilgilerini tanımlar
<sub> Alt simgeli metni tanımlar
<summary> Bir <details> öğesi için görünür bir başlık tanımlar
<sup> Üst simgeli metni tanımlar
<svg> SVG grafikleri için bir kapsayıcı tanımlar
<table> Bir tabloyu tanımlar
<tbody> Bir tablodaki gövde içeriğini gruplandırır
<td> Bir tablodaki bir hücreyi tanımlar
<template> Sayfa yüklendiğinde gizlenmesi gereken içerik için bir kapsayıcı tanımlar
<textarea> Çok satırlı bir giriş denetimini (metin alanı) tanımlar

EK 2. (Devam) HTML Etiketleri Listesi

<tfoot> Bir tablodaki alt bilgi içeriğini gruplandırır
<th> Bir tablodaki bir üst bilgi hücresini tanımlar
<thead> Bir tablodaki üst bilgi içeriğini gruplandırır
<time> Belirli bir saati (veya tarih saatini) tanımlar
<title> Belge için bir başlık tanımlar
<tr> Bir tablodaki bir satırı tanımlar
<track> Medya öğeleri için metin parçalarını tanımlar (<video> ve <audio>)
<tt> HTML5'te desteklenmez. Bunun yerine CSS kullanılır.Teletype metnini tanımlar
<u> Normal metinden farklı şekilde biçimlendirilmiş ve ifade edilmemiş bazı metinleri tanımlar
 Sıralanmamış bir liste tanımlar
<var> Bir değişkeni tanımlar
<video> Gömülü video içeriğini tanımlar
<wbr> Olası bir satır sonunu tanımlar

EK 3. BeautifulSoup kullanarak yazılmış “Emlak İlan Sitesi” kazıma kodu

```
import time
import requests
from bs4 import BeautifulSoup

# Start time to measure script runtime
start_time = time.time()

URL = "https://emlaksitem.com/satilik"
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html.parser')

ilanlar = soup.findAll('div', attrs={'class': 'item-wrapper featured-4'})

for ilan in ilanlar:
    emlak = ilan.find('h2').find('a', attrs={'class': 'classified-title'}).text.strip()
    emlak_turu = ilan.find('div', attrs={'class': 'item-properties'}).text.strip()
    fiyat = ilan.find('strong').text.strip()
    print(emlak)
    print(emlak_turu)
    print(fiyat)
    print("--------------------")

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 4. BeautifulSoup kullanarak yazılmış “Haber Sitesi” kazıma kodu

```
import time
import requests
from bs4 import BeautifulSoup
# Start time to measure script runtime
start_time = time.time()
URL = "http://www.unyekent.com/"
# page_record = []
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html.parser')
haberler = soup.find('div', attrs={'class': 'ulwrap'}).find("ul").find_all("li")
i = 0
for item in range(21):
    haber = haberler[i]
    haber_url = "http://www.unyekent.com" + haber.find("a")["href"]
    r = requests.get(haber_url)
    soup = BeautifulSoup(r.content, 'html.parser')
    resim_url = "http://www.unyekent.com" + soup.find('div', attrs={'class':
'resim'}).find("img")["src"]
    baslik = soup.find('h1').text.lstrip()
    print(baslik)
    print(haber_url)
    print(resim_url)
    print("--------------------")
    i += 1
# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 5. BeautifulSoup kullanarak yazılmış “Online Market Sitesi” kazıma kodu

```
import time
import requests
from bs4 import BeautifulSoup

# Start time to measure script runtime
start_time = time.time()

URL = "https://www.onurmarket.com/meyve-sebze"
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html.parser')

urunler = soup.find_all("div", attrs={'class': 'productItem'})

for urun in urunler:
    urun_adi = urun.find('div', {'class': 'productNamedetailUrl'}).text
    urun_fiyati = urun.find('div', attrs={'class': 'discountPrice'}).find('span').text.strip()
    print(urun_adi)
    print(urun_fiyati)
    print("--------------------")

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 6. BeautifulSoup kullanarak yazılmış “E-Ticaret Sitesi” kazıma kodu

```
import time
import requests
from bs4 import BeautifulSoup

# Start time to measure script runtime
start_time = time.time()

URL = "https://www.bizimyoresel.com/tum-yoresel-urunler"
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html.parser')

urunler = soup.find_all("div", attrs={'class': 'product-item'})

for urun in urunler:
    print(urun.find('div', {'class': 'detail'}).find('a').text)
    try:
        print(urun.find('div', {'class': 'detail'}).find('div', {'class': 'last-price'}).text)
    except:
        pass
    print("--------------------")

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 7. Selenium kullanarak yazılmış "Emlak İlan Sitesi" kazıma kodu

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options

# Start time to measure script runtime
start_time = time.time()

# Setting up options
options = Options()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

# Initialize the WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)
URL = "https://emlaksitem.com/satilik"
driver.get(URL)

# Allow time for the page to load
time.sleep(3)

# Locate all listings with the 'item-wrapper featured-4' class
ilanlar = driver.find_elements(By.CLASS_NAME, 'item-wrapper.featured-4')

# Extract details from each listing
for ilan in ilanlar:
```

EK 7. (Devam) Selenium kullanarak yazılmış "Emlak İlan Sitesi" kazıma kodu

```
    # Find the title
    emlak = ilan.find_element(By.CSS_SELECTOR, 'h2 a.classified-title').text.strip()
    # Find the property type
    emlak_turu = ilan.find_element(By.CLASS_NAME, 'item-properties').text.strip()
    # Find the price
    fiyat = ilan.find_element(By.TAG_NAME, 'strong').text.strip()
    # Print the results
    print(emlak)
    print(emlak_turu)
    print(fiyat)
    print("--------------------")

# Close the WebDriver
driver.quit()

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 8. Selenium kullanarak yazılmış "Haber Sitesi" kazıma kodu

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options

# Start time to measure script runtime
start_time = time.time()

# Setting up options
options = Options()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

# Initialize the WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)

# Open the URL
URL = "http://www.unyekent.com/"
driver.get(URL)

# Allow time for the page to load
time.sleep(3)

# Find the elements
haberler = driver.find_elements(By.CSS_SELECTOR, 'div.ulwrap ul li')
```

EK 8. (Devam) Selenium kullanarak yazılmış "Haber Sitesi" kazıma kodu

```
for item in range(21):
    haber = haberler[item]
    partial_haber_url = haber.find_element(By.TAG_NAME, 'a').get_attribute("href")
    # Ensure the URL is correctly formed
    if not partial_haber_url.startswith("http"):
        haber_url = "http://www.unyekent.com" + partial_haber_url
    else:
        haber_url = partial_haber_url

    # Continue with the rest of your code here...

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 9. Selenium kullanarak yazılmış "Online Market Sitesi" kazıma kodu

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options

# Start time to measure script runtime
start_time = time.time()

# Setting up options
options = Options()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

# Initialize the WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)

URL = "https://www.onurmarket.com/meyve-sebze"
driver.get(URL)

# Allow time for the page to load completely
time.sleep(3)

# Locate all product items with the class 'productItem'
urunler = driver.find_elements(By.CLASS_NAME, 'productItem')

# Loop through each 'urun' (product) element to get product name and price
```

EK 9. (Devam) Selenium kullanarak yazılmış "Online Market Sitesi" kazıma kodu

```
for urun in urunler:
    # Find the product name within the product item
    urun_adi = urun.find_element(By.CLASS_NAME, 'productName.detailUrl').text
    # Find the product price within the product item
    urun_fiyati = urun.find_element(By.CSS_SELECTOR, 'div.discountPrice span').text.strip()
    # Print the results
    print(urun_adi)
    print(urun_fiyati)
    print("--------------------")

# Close the WebDriver
driver.quit()

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 10. Selenium kullanarak yazılmış "E-Ticaret Sitesi" kazıma kodu

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options

# Start time to measure script runtime
start_time = time.time()

# Setting up options
options = Options()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

# Initialize the WebDriver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)

URL = "https://www.bizimyoresel.com/tum-yoresel-urunler"
driver.get(URL)

# Allow time for the page to load completely
time.sleep(3)

# Locate all product items with the specified class
urunler = driver.find_elements(By.CLASS_NAME, 'product-item')

for urun in urunler:
```

EK 10. (Devam) Selenium kullanarak yazılmış "E-Ticaret Sitesi" kazıma kodu

```
    # Extract product name
    urun_adi = urun.find_element(By.CSS_SELECTOR, 'div.detail a').text
    print(urun_adi)
    # Attempt to extract the last price if available
    try:
        urun_fiyati = urun.find_element(By.CSS_SELECTOR, 'div.detail div.last-price').text
        print(urun_fiyati)
    except:
        pass  # If last price is not available, continue
    print("--------------------")

# Close the WebDriver
driver.quit()

# End time and calculate duration
end_time = time.time()
runtime = end_time - start_time
print(f"Script runtime: {runtime:.2f} seconds")
```

EK 11. Scrapy kullanarak yazılmış "Emlak İlan Sitesi" kazıma spider kodu

```
import scrapy
import time

class EmlakSpider(scrapy.Spider):
    name = "emlak"
    start_urls = ["https://emlaksitem.com/satilik"]

    # Start time to measure script runtime
    start_time = time.time()

    def parse(self, response):
        # Select all listings with the specified class
        ilanlar = response.css('div.item-wrapper.featured-4')
        for ilan in ilanlar:
            # Extract the title, type, and price
            emlak = ilan.css('h2 a.classified-title::text').get().strip()
            emlak_turu = ilan.css('div.item-properties::text').get().strip()
            fiyat = ilan.css('strong::text').get().strip()
            # Print the extracted data
            print(emlak)
            print(emlak_turu)
            print(fiyat)
            print("--------------------")

        # End time and calculate duration
        end_time = time.time()
        runtime = end_time - self.start_time
        print(f"Script runtime: {runtime:.2f} seconds")
```

EK 12. Scrapy kullanarak yazılmış “Haber Sitesi” kazıma spider kodu

```
import scrapy
import time

class MySpider(scrapy.Spider):
    name = "unyekent"
    start_urls = ["http://www.unyekent.com/"]
    # Start time to measure script runtime
    start_time = time.time()

    def parse(self, response):
        haberler = response.css('div.ulwrap ul li')
        for i, haber in enumerate(haberler[:21]):
            haber_url = haber.css('a::attr(href)').get()
            if haber_url:
                haber_url = response.urljoin(haber_url)
                yield scrapy.Request(haber_url, callback=self.parse_haber)

    def parse_haber(self, response):
        baslik = response.css('h1::text').get().strip()
        resim_url = response.urljoin(response.css('div.resim img::attr(src)').get())
        print(baslik)
        print(response.url)
        print(resim_url)
        print("--------------------")

        # End time and calculate duration
        end_time = time.time()
        runtime = end_time - self.start_time
        print(f"Script runtime: {runtime:.2f} seconds")
```

EK 13. Scrapy kullanarak yazılmış "Online Market Sitesi" kazıma spider kodu

```
import scrapy
import time
class MySpider(scrapy.Spider):
    name = "onur"
    start_urls = ["https://www.onurmarket.com/meyve-sebze"]
    # Start time to measure script runtime
    start_time = time.time()
    def parse(self, response):
        urunler = response.css('div.productItem')
        for urun in urunler:
            urun_adi = urun.css('div.productName::text').get()
            if not urun_adi:
                urun_adi = urun.css('div.productName.detailUrl::text').get()
            if not urun_adi:
                urun_adi = urun.css('div.productName a::text').get()
            urun_fiyati = urun.css('div.discountPrice span::text').get()
            if urun_adi:
                urun_adi = urun_adi.strip()
            if urun_fiyati:
                urun_fiyati = urun_fiyati.strip()
            print(urun_adi)
            print(urun_fiyati)
            print("--------------------")
        # End time and calculate duration
        end_time = time.time()
        runtime = end_time - self.start_time
        print(f"Script runtime: {runtime:.2f} seconds")
```

EK 14. Scrapy kullanarak yazılmış “E-Ticaret Sitesi” kazıma spider kodu

```
import scrapy
import time

class YoreselSpider(scrapy.Spider):
    name = "yoresel"
    start_urls = ["https://www.bizimyoresel.com/tum-yoresel-urunler"]

    # Start time to measure script runtime
    start_time = time.time()

    def parse(self, response):
        # Select all product items with the specified class
        urunler = response.css('div.product-item')
        for urun in urunler:
            # Extract product name
            urun_adi = urun.css('div.detail a::text').get().strip()
            # Extract last price, handling possible absence
            urun_fiyati = urun.css('div.detail div.last-price::text').get()
            urun_fiyati = urun_fiyati.strip() if urun_fiyati else "Price not available"
            # Print results
            print(urun_adi)
            print(urun_fiyati)
            print("--------------------")

        # End time and calculate duration
        end_time = time.time()
        runtime = end_time - self.start_time
        print(f"Script runtime: {runtime:.2f} seconds")
```

EK 15. Windows İşletim Sisteminde Kodların Çalışma Süresi Ölçümleri (saniye)

	BeautifulSoup	Selenium	Scrapy
Emlaksitem Ölçüm 1	0,56	10,56	0,96
Emlaksitem Ölçüm 2	0,52	11,3	0,97
Emlaksitem Ölçüm 3	0,53	11,7	0,95
Emlaksitem Ölçüm 4	0,52	10,85	0,95
Emlaksitem Ölçüm 5	0,51	10,62	0,96
Emlaksitem Ölçüm 6	0,52	10,45	0,95
Emlaksitem Ölçüm 7	0,51	10,88	0,96
Emlaksitem Ölçüm 8	0,5	10,53	0,98
Emlaksitem Ölçüm 9	0,57	10,54	0,96
Emlaksitem Ölçüm 10	0,52	10,46	0,94
Onurmarket Ölçüm 1	1,32	16,18	2,07
Onurmarket Ölçüm 2	1,31	17,07	2,11
Onurmarket Ölçüm 3	1,35	14,18	2,02
Onurmarket Ölçüm 4	1,38	15,81	2.00
Onurmarket Ölçüm 5	1,24	16,9	1,91
Onurmarket Ölçüm 6	1,33	16,47	1,99
Onurmarket Ölçüm 7	1,36	14,65	2.00
Onurmarket Ölçüm 8	1,28	14,61	2,05
Onurmarket Ölçüm 9	1,38	14,28	2.00
Onurmarket Ölçüm 10	1,27	16,71	2,08
Ünyekent Ölçüm 1	16,91	114,91	12,57
Ünyekent Ölçüm 2	17,97	116,44	13,48
Ünyekent Ölçüm 3	16,38	107,75	12,67
Ünyekent Ölçüm 4	17,23	113,02	13,48
Ünyekent Ölçüm 5	16,55	114,98	13,38
Ünyekent Ölçüm 6	15,94	120,75	13,18
Ünyekent Ölçüm 7	15,89	129,05	12,98
Ünyekent Ölçüm 8	16,57	131,33	13,8
Ünyekent Ölçüm 9	16,1	132,02	13,4
Ünyekent Ölçüm 10	16,44	125,68	13,73
Bizim Yöresel Ölçüm 1	0,31	12,29	0,73
Bizim Yöresel Ölçüm 2	0,25	12,8	0,71
Bizim Yöresel Ölçüm 3	0,27	11,26	0,72
Bizim Yöresel Ölçüm 4	0,3	11,01	0,75
Bizim Yöresel Ölçüm 5	0,28	12,65	0,77
Bizim Yöresel Ölçüm 6	0,29	12,91	0,79
Bizim Yöresel Ölçüm 7	0,25	12,12	0,91
Bizim Yöresel Ölçüm 8	0,26	12,04	0,81
Bizim Yöresel Ölçüm 9	0,27	10,83	0,86
Bizim Yöresel Ölçüm 10	0,26	10,86	0,82

EK 16. Linux İşletim Sisteminde Kodların Çalışma Süresi Ölçümleri (saniye)

	BeautifulSoup	Selenium	Scrapy
Emlaksitem Ölçüm 1	0,65	17,9	0,86
Emlaksitem Ölçüm 2	0,59	17,85	0,84
Emlaksitem Ölçüm 3	0,59	18,1	0,78
Emlaksitem Ölçüm 4	0,54	17,72	0,75
Emlaksitem Ölçüm 5	0,57	17,61	0,77
Emlaksitem Ölçüm 6	0,67	17,76	0,78
Emlaksitem Ölçüm 7	0,56	18,02	0,8
Emlaksitem Ölçüm 8	0,64	17,42	0,81
Emlaksitem Ölçüm 9	0,77	17,44	0,85
Emlaksitem Ölçüm 10	0,56	18,15	0,83
Onurmarket Ölçüm 1	1,36	31,4	1,62
Onurmarket Ölçüm 2	1,43	35,56	1,62
Onurmarket Ölçüm 3	1,33	33,53	1,67
Onurmarket Ölçüm 4	1,38	33,37	1,69
Onurmarket Ölçüm 5	1,39	31,63	1,72
Onurmarket Ölçüm 6	1,36	37,00	1,67
Onurmarket Ölçüm 7	1,3	35,88	1,66
Onurmarket Ölçüm 8	1,33	35,54	1,58
Onurmarket Ölçüm 9	1,38	35,98	1,58
Onurmarket Ölçüm 10	1,35	35,05	1,63
Ünyekent Ölçüm 1	16,53	183,82	13,2
Ünyekent Ölçüm 2	16,56	182,55	13,43
Ünyekent Ölçüm 3	15,94	174,26	12,92
Ünyekent Ölçüm 4	14,95	171,15	12,79
Ünyekent Ölçüm 5	16,13	167,24	12,64
Ünyekent Ölçüm 6	16,74	173,13	13,23
Ünyekent Ölçüm 7	16,25	161,37	13,74
Ünyekent Ölçüm 8	16,81	186,95	12,67
Ünyekent Ölçüm 9	16,26	169,38	13,16
Ünyekent Ölçüm 10	16,09	164,63	12,48
Bizim Yöresel Ölçüm 1	0,33	21,21	0,56
Bizim Yöresel Ölçüm 2	0,31	21,18	0,59
Bizim Yöresel Ölçüm 3	0,29	20,79	0,6
Bizim Yöresel Ölçüm 4	0,33	22,11	0,57
Bizim Yöresel Ölçüm 5	0,33	21,52	0,55
Bizim Yöresel Ölçüm 6	0,28	21,87	0,56
Bizim Yöresel Ölçüm 7	0,32	21,95	0,55
Bizim Yöresel Ölçüm 8	0,28	21,57	0,59
Bizim Yöresel Ölçüm 9	0,31	21,95	0,52
Bizim Yöresel Ölçüm 10	0,28	21,4	0,57

EK 17. Bulut Sunucuda Kodların Çalışma Süresi Ölçümleri (saniye)

	BeautifulSoup	Selenium	Scrapy
Emlaksitem Ölçüm 1	0,8	24,44	1,04
Emlaksitem Ölçüm 2	0,77	24,41	1,02
Emlaksitem Ölçüm 3	0,75	24,26	1,06
Emlaksitem Ölçüm 4	0,78	24,00	1,13
Emlaksitem Ölçüm 5	0,76	23,86	0,96
Emlaksitem Ölçüm 6	0,91	24,08	1,19
Emlaksitem Ölçüm 7	0,75	23,83	0,99
Emlaksitem Ölçüm 8	0,82	23,75	1,06
Emlaksitem Ölçüm 9	0,69	23,84	1,02
Emlaksitem Ölçüm 10	0,8	23,74	1,1
Onurmarket Ölçüm 1	1,38	39,06	1,45
Onurmarket Ölçüm 2	1,23	37,36	1,5
Onurmarket Ölçüm 3	1,26	42,07	1,46
Onurmarket Ölçüm 4	1,32	41,03	1,45
Onurmarket Ölçüm 5	1,23	41,99	1,38
Onurmarket Ölçüm 6	1,43	37,86	1,49
Onurmarket Ölçüm 7	1,24	36,17	1,4
Onurmarket Ölçüm 8	1,26	41,03	1,56
Onurmarket Ölçüm 9	1,4	40,64	1,57
Onurmarket Ölçüm 10	1,34	39,38	1,47
Ünyekent Ölçüm 1	22,36	237,22	13,97
Ünyekent Ölçüm 2	22,67	225,73	14,19
Ünyekent Ölçüm 3	22,41	251,35	14,25
Ünyekent Ölçüm 4	24,26	233,2	14,8
Ünyekent Ölçüm 5	22,66	241,41	13,95
Ünyekent Ölçüm 6	22,73	220,51	13,6
Ünyekent Ölçüm 7	22,55	240,43	14,38
Ünyekent Ölçüm 8	22,53	219,81	13,52
Ünyekent Ölçüm 9	23,12	246,21	14,36
Ünyekent Ölçüm 10	23,36	224,32	13,89
Bizim Yöresel Ölçüm 1	0,5	30,03	0,64
Bizim Yöresel Ölçüm 2	0,48	28,64	0,66
Bizim Yöresel Ölçüm 3	0,47	29,66	0,64
Bizim Yöresel Ölçüm 4	0,49	28,44	0,68
Bizim Yöresel Ölçüm 5	0,48	28,28	0,74
Bizim Yöresel Ölçüm 6	0,46	28,38	0,67
Bizim Yöresel Ölçüm 7	0,49	29,81	0,66
Bizim Yöresel Ölçüm 8	0,48	29,88	0,7
Bizim Yöresel Ölçüm 9	0,47	29,75	0,65
Bizim Yöresel Ölçüm 10	0,48	28,13	0,66