Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Предмет:

Вовед во науката за податоци

Тема на проектот:

Прибирање на податоци за различни производи од повеќе е-продавници, нивно претпроцесирање и стандардизација.

Изработен од:

Андреј Трајковски 223255

Ментор/Асистент:

Димитар Пешевски

Линк до Github репозиториумот:

https://github.com/AndrejT03/VNP-Proekt

Јуни 2025

Содржина:

1. Вовед, цели и очекувани резултати на проектот

1.1 Вовед

Во современата дигитална ера, е-трговијата се развива со забрзано темпо, што резултира со огромни количини на податоци за производи распределени низ различни платформи. Овој проект се фокусира на создавање на автоматизиран систем за собирање и стандардизација на информации за производи од домен електроника (технологија).

Проектот претставува комплексен систем кој комбинира различни техники за веб скрепинг, обработка на податоци и статистичка анализа. Целиот код е организиран во логички секции кои постепено градат кон конечниот резултат - унифициран dataset со производи од три различни извори.

1.2 Главни цели на проектот

- ✓ Автоматизација на процесот на собирање податоци од повеќе веб страници.
- ✓ Стандардизација на структурата на собраните податоци за полесна анализа.
- ✓ Развој на скалабилен систем кој може да се прошири на дополнителни извори.
- ✓ Демонстрација на техники за справување со различни веб архитектури.

1.3 Очекувани резултати

Проектот за прибирање податоци од е-продавници треба да резултира со создавање на унифициран dataset што содржи информации за производи од три различни е-продавници, со стандардизирана структура погодна за понатамошна анализа и споредба.

2. Методологија и имплементација на кодот

2.1 Избор на извори на податоци

1. Мобеликс (mobelix.com.mk)
   - Специјализирана продавница за мобилни телефони и технологија
   - Фокус на премиум брендови и најнови модели
   - Добро структурирани веб страници погодни за скрепинг

2. Нептун (neptun.mk)
   - Најголемиот малопродажен ланец за бела техника и електроника во Македонија
   - Основан во 1998 година со 25 продавници
   - Богат асортиман од различни категории производи

3. Сетек (setek.com.mk)
   - Воспоставена е-продавница со широк спектар на електронски производи
   - Конкурентни цени и промоции
   - Модерна веб платформа

2.2 Имплементација на кодот

2.2.1 Дел 1: Иницијализација и поставување на околината

```
!pip install selenium
```

Selenium е моќна библиотека за автоматизација на веб прелистувачи која е неопходна за интеракција со динамички веб страници . За разлика од традиционалните HTTP барања, Selenium овозможува извршување на JavaScript код и интеракција со елементи што се генерираат динамички.

```
!apt-get install -y chromium-browser chromium-chromedriver
```

Chrome Driver служи како мост меѓу Selenium и прелистувачот, преведувајќи ги Selenium командите во конкретни акции во прелистувачот . Параметарот -у автоматски одговара позитивно на сите системски прашања за време на инсталацијата.

```
import re, requests
import json, pathlib
import urllib.parse
import time
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver
from IPython.display import display
```

Секоја од овие библиотеки има специфична улога во проектот :

- ✓ requests за HTTP комуникација со веб серверите
- ✓ BeautifulSoup за парсирање на HTML структури
- ✓ pandas за манипулација на табеларни податоци
- ✓ selenium.webdriver за контрола на веб прелистувач
- ✓ time за управување со временски паузи

2.2.2 Дел 2: Скрејпинг на Мобеликс

```
response = requests.get( "https://mobelix.com.mk/" )
soup = BeautifulSoup(response.text, "html.parser" )
```

Основно барање до серверот. Функцијата requests.get() праќа HTTP GET барање и враќа Response објект кој содржи status код, заглавја и HTML содржина . BeautifulSoup потоа го парсира HTML кодот и создава навигабилна DOM структура.

```
product_containers = soup.select( 'section.deal-of-the-day div.deal-slider > div' )
```

Овој CSS селектор е прецизно дизајниран за структурата на Мобеликс .
Селекторот section.deal-of-the-day бара секција со класа за дневни промоции, потоа се спушта до контејнерот div.deal-slider, и конечно со > div бара само директни деца елементи.

```
for i, container in enumerate(product_containers):
    if i >= 3:
        break
    product_link_element = container.find('a', class_='d-flex')

    if product_link_element:
        name_element = product_link_element.find('h5', class_='h2')
        product_name = name_element.text.strip() if name_element else "Не е достапно име"
        price_element = product_link_element.find('p', class_='h4 price')
        price = price_element.text.strip() if price_element else "Не е достапна цена"
        extracted_products_mobelix.append({
            "name": product_name,
            "regular_price": price + "."
        })
```

Овој циклус демонстрира неколку важни принципи :

- ✓ enumerate() обезбедува и индекс и вредност за секој елемент
- ✓ Ограничувањето if i >= 3: break спречува преоптоварување на серверот
- ✓ Conditional expressions обезбедуваат robusnost против променети структури
- ✓ Секој производ се зачувува како dictionary со конзистентни клучеви

2.2.3 Дел 3: Скрејпинг на Нептун

```
base_url = "https://www.neptun.mk"

response = requests.get(base_url)

soup = BeautifulSoup(response.text, "html.parser")

print(soup.prettify())
```

Функцијата prettify() ги форматира HTML тага за полесно читање и анализа на структурата . Оваа анализа е клучна за разбирање како Нептун ги организира производите на својата платформа.

```
list_url=[1,8,18]

anchors = soup.select('ul.dropdown-menu > li > a[href$=".nspx"]')

wanted = [anchors[i-1] for i in list_url if i-1 < len(anchors)]

full_urls = [urllib.parse.urljoin(base_url, a['href']) for a in wanted]
```

- ✓ list_url=[1]: Ова е листа на индекси. Целта е да се изберат првиот, осмиот и осумнаесеттиот линк од менито за навигација.
- ✓ anchors = soup.select(...): Ова е CSS селектор кој ги наоѓа сите линкови (<a> тагови) кои се наоѓаат во паѓачкото мени (ul.dropdown-menu > li). Клучниот дел е [href$=".nspx"], кој ги филтрира само оние линкови чија href адреса завршува на .nspx. Ова е специфичен шаблон за линковите до категории во Нептун.
- ✓ wanted = [...]: Оваа линија код ги филтрира најдените линкови (anchors) според индексите од list_url. i-1 се користи бидејќи листите во Python се 0-индексирани. if i-1 < len(anchors) е заштита за да се избегне грешка ако се побара индекс кој не постои.
- ✓ full_urls = [...]: Ги претвора релативните URL адреси (пр. /categories/...) во целосни, апсолутни адреси (пр. https://www.neptun.mk/categories/...) со помош на urllib.parse.urljoin.

Помошната функција (**format_price**) има за цел да ја стандардизира цената. Таа прима цена (која може да биде број или текст), ја чисти од непотребни знаци (како валута или

запирки), ја претвора во број и на крајот ја форматира во читлив формат како 11,509.00 ден.. Ова осигурува конзистентност на податоците.

Главниот код:

- ✓ for idx, url in enumerate(full_urls, 1):: Овој циклус поминува низ секоја од генерираните URL адреси. enumerate дава и реден број (idx) и вредност (url).
- ✓ try...except: Блокот за справување со грешки осигурува дека програмата нема да прекине ако има проблем со мрежата (пр. timeout).
- ✓ match = re.search(...): **Ова е клучната линија.** Таа користи *регуларен израз* за да го пребара целиот HTML текст (html) и да ја најде линијата каде што се дефинира JavaScript променливата shopCategoryModel.
    - r'var\s+shopCategoryModel\s*=\s*...': Го бара текстот "var shopCategoryModel = " со флексибилни празни места (\s*).
    - ({.*?}|"(?:\\.|[^"\$$)*"): Овој дел е дизајниран да го "фати" (capture) JSON објектот, без разлика дали е директно вметнат ({...}) или е внатре во наводници ("...").
    - re.DOTALL: Овозможува точката (.) во изразот да совпаѓа и со нови редови, што е важно за повеќередични JSON објекти.

- ✓ for idx, url in enumerate(full_urls, 1):: Овој циклус поминува низ секоја од генерираните URL адреси. enumerate дава и реден број (idx) и вредност (url).
- ✓ try...except: Блокот за справување со грешки осигурува дека програмата нема да прекине ако има проблем со мрежата (пр. timeout).
- ✓ match = re.search(...): Ова е клучната линија. Таа користи регуларен израз за да го пребара целиот HTML текст (html) и да ја најде линијата каде што се дефинира JavaScript променливата shopCategoryModel.
- ✓ r'var\s+shopCategoryModel\s*=\s*...': Го бара текстот "var shopCategoryModel = " со флексибилни празни места (\s*).
- ✓ ({.*?}|"(?:\\.|[^"\$$)*"): Овој дел е дизајниран да го "фати" (capture) JSON објектот, без разлика дали е директно вметнат ({...}) или е внатре во наводници ("...").
- ✓ re.DOTALL: Овозможува точката (.) во изразот да совпаѓа и со нови редови, што е важно за повеќередични JSON објекти.

- ✓ raw_json = match.group(1).strip(): Од пронајдениот текст (match), ја зема само групата во загради - чистиот JSON податок.
- ✓ data = json.loads(raw_json): Ја користи библиотеката json за да го претвори текстуалниот JSON во Python речник (dictionary), што го прави лесен за работа.
- ✓ if isinstance(data, str): data = json.loads(data): Ова е многу важна проверка. Понекогаш, JSON податоците се "двојно енкодирани" (JSON внатре во JSON стринг). Оваа линија проверува дали резултатот од првото парсирање е повторно стринг и, ако е, го парсира уште еднаш.

- ✓ if data and data.get("Products"):: Проверува дали JSON податоците се успешно вчитани и дали содржат клуч "Products".
- ✓ first_product = data["Products"]: Од листата на сите производи во таа категорија, го зема само првиот производ. Ова е свесна одлука во кодот за да се земе само еден примерок од секоја категорија.
- ✓ product_name = first_product.get("Title", "N/A"): Од податоците за производот, безбедно го чита клучот "Title". .get() се користи за да се избегне грешка ако клучот не постои.
- ✓ product_info = { ... }: Ги организира извлечените податоци во стандардизиран речник.
- ✓ extracted_products_neptun.append(product_info): Го додава речникот во финалната листа со производи од Нептун.

2.2.4 Дел 4: Скрејпинг на Сетек со Selenium

Конфигурација на Chrome опции:

chrome_options = webdriver.ChromeOptions()

chrome_options.add_argument("--headless")

chrome_options.add_argument("--no-sandbox")

chrome_options.add_argument("--disable-dev-shm-usage")

chrome_options.add_argument("--disable-gpu")

chrome_options.add_argument("--window-size=1920,1080")

Секоја од овие опции има специфична цел :

- ✓ --headless овозможува извршување без графички интерфејс
- ✓ --no-sandbox е потребно за контејнериски околини
- ✓ --disable-dev-shm-usage спречува проблеми со меморијата
- ✓ --disable-gpu намалува сложеност во headless режим
- ✓ --window-size поставува димензии важни за responsive дизајн

```
driver = webdriver.Chrome(options=chrome_options)

base_url = "https://setek.com.mk/categories/64/outlet"

driver.get(base_url)

time.sleep(10)
```

Паузата од 10 секунди е критична за овозможување на JavaScript кодот да се изврши целосно и на асинхронните AJAX повици да завршат . Ова осигурува дека сите динамички елементи се рендерираат пред извлекувањето на податоци.

```
html_content = driver.page_source

soup = BeautifulSoup(html_content, 'html.parser')

driver.quit()
```

driver.page_source ја враќа целосната HTML содржина како што е по сите JavaScript промени . Ова претставува финална DOM структура која потоа може да се парсира со BeautifulSoup.

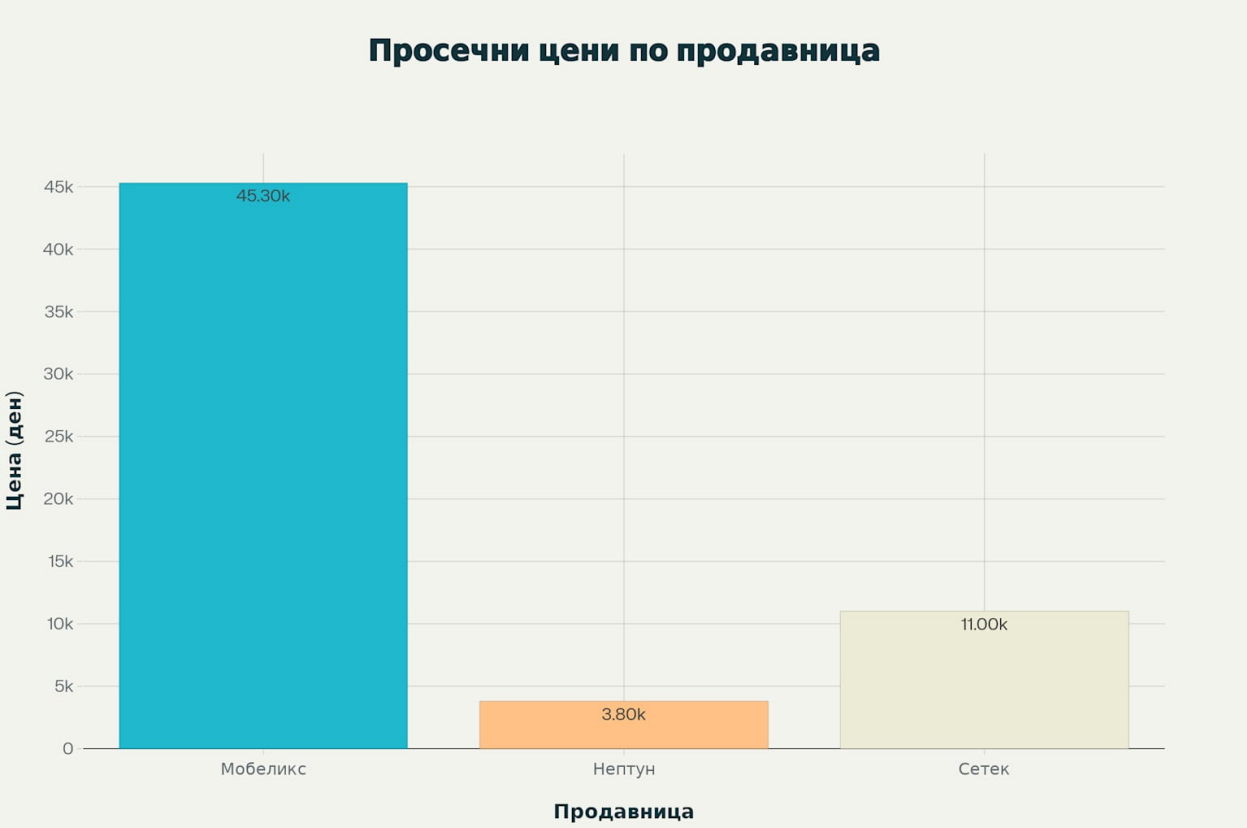
3. Резултати и кратка анализа

3.1 Резултати од Скрејпингот

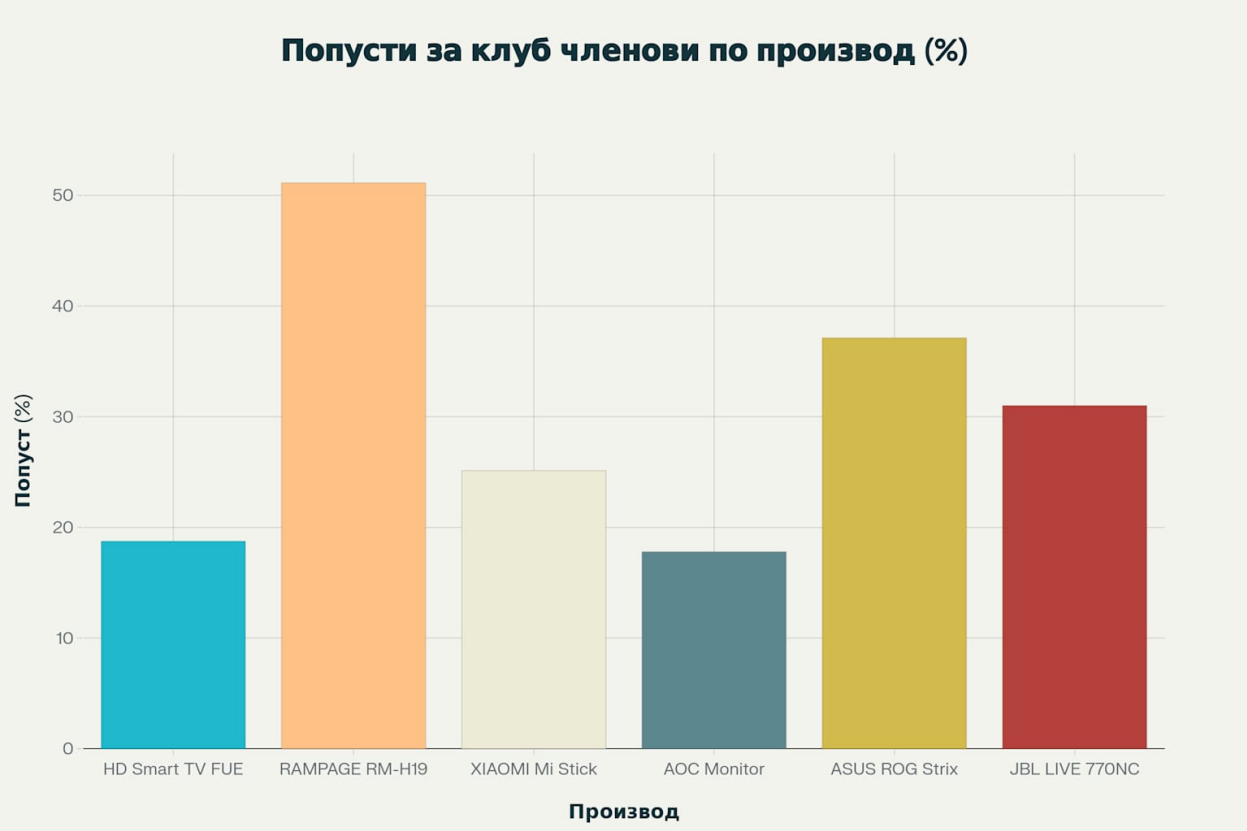
Проектот успешно собра 9 производи од трите е-продавници (односно по 3 производи од секоја една е-продавница), со 100% успешност во извлекување на имиња и редовни цени и 66,5% успешност во прибирање на клуб цените (Мобеликс е единствената е-продавница од трите која што не работи на истиот принцип – да има редовна цена и клуб цена која се добива со лојалти картичка). Со тоа што клуб цените се достапни за две од трите продавници, укажува на различни бизнис модели кај продавниците.

3.2 Кратка анализа на резултати од Скрејпингот

Податоците покажуваат значителна варијација во цените и стратегиите за попусти меѓу различните продавници:



Споредбата на просечните цени покажува дека Мобеликс се фокусира на премиум сегментот со значително повисоки цени, додека Нептун предлага поприфатливи опции. Сетек заема средна позиција во ценовниот спектар.



Анализата на попустите за клуб членови покажува дека RAMPAGE RM-H19 слушалките имаат највисок попуст од над 50%, што укажува на агресивна ценовна стратегија за одредени производи. Ваквите попусти можат да бидат дел од стратегија за зголемување на лојалноста на клиентите.

4. Заклучок и линкови

4.1 Проектот успешно демонстрира

- ✓ Техничка компетентност во користење на различни веб скрепинг техники
- ✓ Способност за адаптација на различни веб архитектури
- ✓ Ефективна стандардизација на хетерогени податоци
- ✓ Практична примена на науката за податоци во е-трговијата

1.2 Практична применливост

Развиениот систем може да биде проширен за:

- ✓ Мониторинг на цени во реално време
- ✓ Анализа на пазарни трендови во електрониката
- ✓ Споредбена анализа на конкурентските цени
- ✓ Автоматизирани извештаи за промени во асортиманот

1.3 Користени/Корисни линкови при креирање на проектот

https://mobelix.com.mk/

https://www.neptun.mk/

https://setek.com.mk/

https://www.crummy.com/software/BeautifulSoup/

https://selenium-python.readthedocs.io/

https://pandas.pydata.org/docs/

https://realpython.com/python-web-scraping-practical-introduction/

https://www.jcchouinard.com/web-scraping/

https://research.aimultiple.com/python-web-scraping-libraries/

https://www.zenrows.com/blog/python-web-scraping-library

https://github.com/pxxthik/Advanced-web-scraping

https://www.youtube.com/watch?v=RJc6wgzEYIY