СТАНДАРДЕН ПРОЕКТ ПО ПРЕДМЕТОТ:

ВОВЕД ВО НАУКАТА ЗА ПОДАТОЦИ

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

Линк до GitHub репозиториум: https://github.com/antonnedelkoski/VNP_Standarden_Proekt_Tema6

Изработил:Антон Неделкоски 193101

Асистент: Димитар Пешевски

Професор: Игор Мишковски

1. Вовед

Во модерниот свет на е-комерцијата, голем број на онлајн продавници нудат безброј производи, често со разлики во брендирањето, спецификациите и цените. Еден таков пример се лаптопите, каде што еден ист лаптоп може да биде прикажан во целосно различни формати на различни платформи. Овие неконзистентности претставуваат сериозен предизвик за аналитичарите и истражувачите кои сакаат да прават споредби, анализи или да градат модели врз основа на реални податоци за производи.

Целта на овој проект е да собере податоци за продукти(лаптопи) од некој од сајтовите за е-комерција во Македонија, односно, Анхоч, ТехноМаркет, Жирафа50 и Нептун и да ги трансформираат необработените податоци во стандардизирано податочно множество кое поддржува конзистентна анализа. Главната цел на овој проект е да се имплементира целосно автоматизиран процес кој:

- *Собира податоци за лаптопи со помош на Selenium и BeautifulSoup во Python.*
- *Извлекува клучни атрибути како име на производ, бренд, процесор, RAM, складирање и цени.*
- *Чисти и претпроцесира податоци за да се справи со недостиг или нерегуларни формати.*
- *Стандардира вредности во унифицирана структура погодна за понатамошна анализа или моделирање.*

Крајниот резултат е чист датасет во CSV формат што дава реален приказ на понудата за лаптопи на овие четири онлајн продавници. Овој датасет може да се искористи за анализа на пазарни цени, истражување на карактеристики или како основа за системи за препорака. Во овој документ ќе биде даден детален преглед на користените технологии, чекорите при прибирање на податоци, предизвиците со кои се соочив (на пр. кирилична нотација, неусогласени формати за складирање и различни имиња за процесори), како и применетите техники за претпроцесирање — вклучувајќи го и користењето на регуларни изрази (regex) за извлекување структуирани податоци од наслови со слободен формат.

2. Собирање на податоци и scraping логика

Првиот важен чекор во овој проект претставува само ваѓање необработени податоци од сајотви за е-комерција. Избраните сајтови, ги претставуваат информациите за лаптопите во различна HTML структура, честопати со неконзистентност на пример во имињата или во форматот на цените. За да се надминат овие, т.н препреки, користев Selenium и BeautifulSoup, кои овозможуваат динамичко рендерирање на страниците и флексибилно парсирање на HTML.

Користени технологии:

- **Selenium** – за автентикација на пребарувач и справување со содржина која е рендерирана со JavaScript.
- **BeautifulSoup** – за парсирање и навигирање низ HTML структурите.
- re(regex) – за структурирано добивање на компоненти од самите наслови на продуктите.
- **Pandas** – за градење и експортирање на финалното податочно множество.

Веб scraping стратегија:

Сите сајтови, поради разликите во DOM(*Domain Object Model*) структурата, "бараат" да се користи различна scraping логика. Во овој проект се применети следниве пристапи:

- За **Анхоч**:
  - Насловите на продуктот беа сместени во ***title*** атрибутот на секоја карта на продуктот.
  - Цените се наоѓаа во ***.product-price*** блокот
  - Насловите најчесто го имаа овој формат: "**Notebook Brand Model Processor / RAM / Storage / GPU / OS**"
- За **ТехноМаркет**:
  - Лаптопите беа листани во ***.span4*** контејнери
  - Варијациите на цените се наоѓаа во ***smart-products*** или како "Редовна Цена"
  - Некои продукти немаа бренд или имаа скратен формат.
- За **Gjirafa50**:
  - Сајтот е рендериран динамички со JavaScript, но пристаплив со Selenium

- Насловите започнуваа во “Лаптоп...“, често со запирки како “граница“ за секоја компонента.
- Сајтот имаше најконзистентна структура и беше најлесен да се парсира со regex.

- **Нептун**
  - Нептун ја поседува познатата промотивна понуда/клуб цена, т.н “HaPPy цена“.
  - Регуларната и хепи цената се рендерираат условно
  - “Специјално“ внимание беше дадено на парсирањето на ***product-price__amount—value*** и ***HappyCard*** елементите.

Собрани атрибути

За сите продукти, пребарувањето имаше за цел да ги извади следниве атрибути:

ИМЕ НА КОЛОНА	ОПИС
Prodavnica	*Име на онлајн продавницата*
Ime	*Целосен наслов за продуктот на сајтот*
Brend	*Име на бренд, добиено со помош на regex*
Procesor	*Информации за процесорот*
RAM_Memorija	*RAM меморија во GB*
Kapacitet	*Мемориски капацитет(512гб пример)*
Cena	*Стандардна цена*
Klub/Smart/Happy cena	*Попустна/клуб цена, ако ја има*

Scraper-от е напишан со функции за извлекување бренд, процесор, RAM и мемориски капцитет, односно за секој сајт поединечно, користи редовни изрази, како и патерни за откривање и извлекување структурирани информации од насловите со слободен текст. Секоја локација беше прегледана низ (обично 5 страници), а податоците беа додадени во заедчники ***pandas.DataFrame.***

```python
#------ВЕБ СКРЕЈПИНГ ЗА САЈТ:

# 1-------- АННОСН --------
for page in range(1, 6):
    driver.get(f"https://www.anhoch.com/categories/site-laptopi/products?brand=&attribute=&toPrice=324980&inStockOnly=2&sort=latest&perPage=20&page={page}")
    time.sleep(5)
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    products = soup.find_all("div", class_="product-card")
    for product in products:
        title = product.get("title")
        price_tag = product.find("div", class_="product-price")
        price = price_tag.get_text(strip=True).split("ден")[0].strip() + " ден." if price_tag else "No price"
        discount_tag = price_tag.find("span", class_="previous-price") if price_tag else None
        discount_price = discount_tag.get_text(strip=True) if discount_tag else None
        p = normalize_price(price)
        dp = normalize_price(discount_price)
        if p and dp and dp > p:
            price, discount_price = discount_price, price
        results.append({
            "Prodavnica": "Anhoch",
            "Ime": title,
            "Brend": extract_brand(title),
            "Procesor": extract_cpu(title),
            "RAM_Memorija": extract_ram(title),
            "Kapacitet": extract_storage(title),
            "Cena": price,
            "Klub/Smart/Happy cena": discount_price
        })
```

СЛИКА: ПРИМЕР ЗА БЛОК НА КОД ЗА ЛОГИКАТА ЗА СОБИРАЊЕ ПОДАТОЦИ(SCRAPING)

3. ПРЕТПРОЦЕСИРАЊЕ И СТАНДАРДИЗАЦИЈА

Откако сите податоци за лаптопите беа “собрани“ и споени во едно податочно множество, следниот важен чекор беше претпроцесирањето, односно конвертирањето на неконзистентните текстуални податоци во чист и анализирачки формат. Ова вклучуваше средување на вредностите кои фалат, прочистување на нумеричките формати, како и трансфромирањето на стринговите во стандардизирани репрезентации.

3.1 ЧИСТЕЊЕ НА КОЛОНИТЕ ЗА ЦЕНИ

- Оригиналните колони за цени беа текстуални и во себе содржеа симболи или текст за денарска противвредност, како и точки или запирки.

Чистење на цените

```
[40]: def clean_price(value):
          if pd.isna(value):
              return None
          value = str(value).replace("ден.", "").replace("MKD", "").replace(",", "").replace(".", "").strip()
          try:
              return float(value)
          except ValueError:
              return None

      df1["Cena"] = df1["Cena"].apply(clean_price)
      df1["Klub/Smart/Happy cena"] = df1["Klub/Smart/Happy cena"].apply(clean_price)
```

```
[41]: df1
```

[41]:

	Prodavnica	Ime	Brend	Procesor	RAM_Memorija	Kapacitet	Cena	Klub/Smart/Happy cena
0	Anhoch	Notebook HP 15 i5-1334U 16GB/512GB/15.6" FHD I...	Hp	I5 1334U	16 GB	512 GB SSD	2598000.0	NaN
1	Anhoch	Notebook HP Pavilion 15 Ryzen7 7730U 16GB/512G...	Hp	RYZEN7 7730U	16 GB	512 GB SSD	2798000.0	2698000.0

СЛИКА: Ф-ЈА ЗА ЧИСТЕЊЕ НА КОЛОНИТЕ ЗА ЦЕНИ СО ПОНАТАМОШЕН ИЗЛЕЗ НА ПОДАТОЧНОТО МНОЖЕСТВО

3.2 Стандардизирање на RAM

- RAM вредностите вариира во форматите (пример: “16GB”,”16 GB”, “8 GB”). Целта беше да се екстрактираат само нумеричките делови и да се претворат во интеџер.

Чистење/Стандардизирање на колоната за RAM меморија

```
[42]: def standardize_ram(ram):
          if pd.isna(ram):
              return None
          match = re.search(r'(\d{1,2})', str(ram))
          return int(match.group(1)) if match else None

      df1["RAM_Memorija"] = df1["RAM_Memorija"].apply(standardize_ram)
```

СЛИКА:Ф-ЈА ЗА СТАНДАРДИЗИРАЊЕ НА RAM КОЛОНАТА

3.3 Стандардизирање на меморискиот капацитет

- За да се овозможат нумерички компарации со меморискиот капацитет, оваа трансформација ја пресметува вкупната меморија во **гигабајти**, притоа вклучувајќи конверзија од TB->GB.

```python
def extract_total_storage_gb(val):
    if not isinstance(val, str):
        return None

    total = 0
    parts = val.upper().split("/")
    for part in parts:
        match = re.search(r'(\d+(?:\.\d+)?)\s*(TB|GB)', part)
        if match:
            size, unit = match.groups()
            gb = float(size) * (1024 if unit == "TB" else 1)
            total += gb
    return int(total) if total > 0 else None


df1["Kapacitet"] = df1["Kapacitet"].apply(extract_total_storage_gb)
```

СЛИКА: ПРЕТХОДНОНАВЕДЕНАТА Ф-ЈА ЗА СТАНДАРДИЗИРАЊЕ НА МЕМОРИСКИОТ КАПАЦИТЕТ

3.4 Стандардизирање на имињата на процесорите, почисто и порубстно

```python
def clean_procesor(cpu):
    if pd.isna(cpu) or cpu.strip().lower() == "unknown":
        return None
    cpu = cpu.strip().upper()

    # Стандардизирање на празни места
    cpu = re.sub(r'\s+', ' ', cpu)

    # Нормализирање на чести патерни
    cpu = cpu.replace("INTEL CORE ", "I").replace("CORE ", "I")
    cpu = cpu.replace("AMD RYZEN", "RYZEN")
    cpu = cpu.replace("RYZEN R", "RYZEN ")
    cpu = cpu.replace("M1", "M1").replace("M2", "M2").replace("M3", "M3")

    cpu = cpu.replace("HX ", "HX")

    return cpu.strip().title()

df1["Procesor"] = df1["Procesor"].apply(clean_procesor)
```

СЛИКА: Ф-ЈА ЗА СТАНДАРДИЗИРАЊЕ НА ИМИЊАТА НА ПРОЦЕСОРИТЕ

Овие чекори за претпроцесирање и стандардизација овозможија податочното множество да стане структурирано, нумеричко каде што треба, и пред се, конзистентно. Со самото тоа, тоа доби уште поголема “верност“ за понатамошни и идни анализи.

4. Regex логика и екстракција на патерни

Regex исказите беа од круцијално значење, бидејќи со нив беа екстрактирани структурираните информации за лаптопите, бидејќи сите информации за лаптопите се наоѓаа во самите наслови т.е имиња на продуктите.

- CPU екстрактирањето со помош на **extract_cpu** функцијата, користеше патерни за детектирање и стандардизирање, а некои од нив беа:
  - **INTEL CORE I[3579] \d{3,5}[A-Z]*:** ги фаќа процесорите на *Intel Core i*-сериите
  - **RYZEN \d \d{3,5}:** ги фаќа AMD Ryzen моделите

```
def extract_cpu(title):
    if not isinstance(title, str):
        return "Unknown"

    title = title.upper().replace("®", "").replace("™", "").replace("-", " ")
    title = re.sub(r"\s+", " ", title)

    patterns = [
        r'INTEL CORE ULTRA \d+ \d{3,5}[A-Z]*',
        r'INTEL CORE I[3579] \d{3,5}[A-Z]*',
        r'INTEL CORE I[3579]',
        r'INTEL I[3579] \d{3,5}[A-Z]*',
        r'I[3579] \d{3,5}[A-Z]*',
        r'I[3579]-\d{3,5}[A-Z]*',
        r'INTEL \d',
        r'CORE I[3579] \d{3,5}[A-Z]*',
        r'CORE [3579]',
        r'INTEL [3579]',
        r'AMD RYZEN \d \d{3,5}[A-Z]*',
        r'RYZEN \d \d{3,5}[A-Z]*',
        r'RYZEN\d \d{3,5}[A-Z]*',
        r'RYZEN[3579] \d{3,5}[A-Z]*',
        r'RYZEN [3579]',
        r'RYZEN[3579]',
        r'R[3579] \d{3,5}[A-Z]*',
        r'R[3579]-\d{3,5}[A-Z]*',
        r'APPLE M[1234]',
        r'M[1234]',
        r'INTEL UHD',
        r'INTEL XE',
        r'VEGA \d+',
        r'ADRENO \d+',
        r'SNAPDRAGON\s+[A-Z]*\s?X?[A-Z]*[\w\-]+',
        r'CORE\s+ULTRA\s+[Uu]?\d[- ]?\d{3,5}[A-Z]*',
        r'ULTRA\s+\d\s+\d{3,5}[A-Z]*',
        r'U\d-\d{3,5}[A-Z]*',
        r'ULTRA\s+\d\s+\d{3,5}[A-Z]*',
        r'RYZEN\s+AI\s+\d\s+HX\s+\d{3}',
        r'R\d-AI\s+\d{3}',
        r'R\d\s+AI\s+\d{3}',
        r'PENTIUM\s?\d{0,5}[A-Z]*',
        r'CELERON\s?\d{0,5}[A-Z]*'

    ]
```

- **extract_ram** функцијата го детектираше RAM-от користејќи ја следнава шема:
  - **(?<!\d)(\d{1,2})\s?GB(?!\s?(SSD|HDD)) –** осигурувајќи дека секаде само споменувањето на RAM-от ќе биде фатено без спецификации како “512GB SSD“.

```python
def extract_ram(title):
    if not isinstance(title, str):
        return None
    match = re.search(r'(?<!\d)(\d{1,2})\s?GB(?!\s?(SSD|HDD))', title, re.IGNORECASE)
    return match.group(1) + " GB" if match else None
```

СЛИКА: Ф-ЈА ЗА ЕКСТРАКТИРАЊЕ НА RAM

- Меморијата претставуваше предизвик заради различните формати и неконзистентното лабелирање, но финалната **regex** логика:
  - Фаќа патерни како **(\d+(?:\.\d+)?\s*(?:TB|GB))(?:\s*(SSD|HDD))?**
  - Филтрира помали вредности од 128 како RAM спецификации.

```python
def extract_storage(title):
    if not isinstance(title, str):
        return None

    title = title.upper()
    title = title.replace(" / ", " ").replace("+", " ")
    title = title.replace("ТБ", "TB").replace("ГБ", "GB")  # за секој случај за случајна појава на кирлица
    title = re.sub(r'([A-Z])(\d+GB)', r'\1 \2', title)     #  за зборовите кои се споени или се без празно место меѓу нив.
    title = re.sub(r'(\d+GB)(\d+(?:GB|TB))', r'\1 \2', title)

    matches = re.findall(r'(\d+(?:\.\d+)?)(?:\s*)(TB|GB)(?:\s*(SSD|HDD)?)?', title)

    results = set()
    for size, unit, dtype in matches:
        try:
            num = float(size)
            if unit == "TB" or (unit == "GB" and num >= 128):
                disk_type = dtype.strip() if dtype else "SSD"
                results.add(f"{int(num) if num.is_integer() else num} {unit} {disk_type}")
        except:
            continue

    return " / ".join(sorted(results)) if results else None
```

СЛИКА: Ф-ЈА ЗА ЕКСТРАКТИРАЊЕ НА МЕМОРИСКИ КАПАЦИТЕТ

5. Заклучок

Овој проект го прикажа целиот процес на прибирање, претпроцесирање и стандардизација на податоци за производи од повеќе македонски е-продавници. Главната цел беше да се извлечат значајни и унифицирани карактеристики — како што се бренд, процесор, RAM и мемориски капацитет — од неструктурирани наслови на производи на различни веб-страници (Anhoch, Tehnomarket, Gjirafa50 и Neptun).

Најголемиот технички предизвик беше самата неконзистентност во форматите. За надминување на ова, беа искористени регуларни изрази, техники за нормализација и сопствено дефинирани функции за извлекување на карактеристики кои овозможија подобро структурирање на податоците.

Добиеното податочно множество може да се искористи за дополнителни анализи поврзани со науката за податоци,од визуелизација па се до тренирање на модели.