Step 0: Practical Web Scraping for Data Science

Tutor: Haythem Ghazouani

1 Introduction

Web Scraping is the process of extracting unstructured data from websites and transforming it into structured data. In this module, scraping is the first step toward building a specialized dataset for Machine Learning.

Implementation Note

The source code for this step is located in `code/scraping.py`. The resulting datasets are stored in the `data/` directory.

2 Aligning Scraping with Data Science Objectives

Before writing a single line of code, you must ensure your data "makes sense" for your target objective. Ask yourself:

The "Alignment" Checklist

1. **Target Variable Identification:** Does the website provide the label you want to predict (e.g., Sentiment, Price, Category)?
2. **Feature Richness:** Are there enough attributes (features) to build a model (e.g., Text, Date, Author, Metadata)?
3. **Data Volume:** Is the site large enough to provide thousands of examples? ML models thrive on data.
4. **Temporal Relevance:** Is the data historical or real-time? For stock prediction, you need time-series alignment.

3 Concrete Use Cases

- **Classification:** Scraping product reviews to classify them as "Positive" or "Negative".
- **Regression:** Scraping real estate listings (surface, location, amenities) to predict house prices.
- **Clustering:** Scraping news articles to group them into topics (Economy, Politics, Sports) using unsupervised learning.

4 Methodology: The Scraping Pipeline

1. **Inspection:** Use Chrome DevTools (F12) to identify CSS selectors (classes and IDs).
2. **Requests:** Use the `requests` library to fetch HTML content. Use headers to simulate a real browser.
3. **Parsing:** Use `BeautifulSoup` to navigate the DOM tree and extract text/attributes.
4. **Normalization:** Clean the scraped text (remove HTML tags, extra spaces).

5 Implementation Code

Below is a detailed example for scraping financial news from CNBC.

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_cnbc_finance():
    url = "https://www.cnbc.com/finance/"
    # Headers are crucial to avoid being blocked
    headers = {"User-Agent": "Mozilla/5.0"}

    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        return "Error fetching page"

    soup = BeautifulSoup(response.content, "html.parser")
    data = []

    # Locate all article containers
    containers = soup.find_all("div", class_="Card-textContent")

    for item in containers:
        headline = item.find("a", class_="Card-title")
        timestamp = item.find("time")

        if headline:
            data.append({
                "headline": headline.text.strip(),
                "link": headline["href"],
                "date": timestamp.text.strip() if timestamp else "
    N/A"
            })

    return pd.DataFrame(data)

# Execution
df = scrape_cnbc_finance()
```

```python
df.to_csv("data/financial_news.csv", index=False)
```

Data Reference

After execution, you will find the final data in `data/financial_news.csv`. Ensure this file is present before moving to the sentiment analysis sub-task.

6 Self-Validation Exercise

Task: Find a second financial news site (e.g., Reuters, Bloomberg). Scrape the same day's headlines. **Question:** How would you merge these two datasets to create a "Unified Financial Sentiment" feature for your Churn model?