Science, Engineering & Education, 11, (1), 2026, 109-116

Advanced Methods for Data Retrieval and Analysis through Internet APIs, Web Scraping, and Machine Learning: Development of a Software Solution for Comparative Price Analysis

Ivan Pirinski[1], Nikolay Voenkinov[1] , Georgi Hristov[1, 2, *]

[1]*Second English Language School „Thomas Jefferson", 26 Trayanova Vrata St., Sofia 1408 Strelbishte Residential Complex,*
[2]*Sofia University "St. Kliment Ohridski", 15 Tsar Osvoboditel Blvd., Sofia 1504*

Received 14 September 2025, Accepted 09 January 2026
DOI: 10.59957/see.v11.i1.2026.15

ABSTRACT

The report examines the use of Internet APIs, web scraping, and machine learning for automated data extraction and processing. It presents the principles of REST APIs, tools for web scraping such as Scrapy and Selenium, as well as machine learning algorithms, including neural networks and classification models. The report discusses the challenges of handling dynamic content, managing large data volumes, and the ethical aspects of web scraping. It also analyzes the prospects for data automation and the enhancement of analytical models. The conclusion emphasizes the importance of integrating these technologies for more efficient information management across various industries.

During the preparation of the report, a software solution was developed to automate the comparative analysis of prices for identical products across multiple online retailers.

Keywords: APIs, web scraping, machine learning, data automation and ethics.

INTRODUCTION

Automated information extraction systems play an increasingly important role in today's digital environment, where the volume of available data is growing exponentially. The data used in information systems can generally be divided into structured and unstructured categories [1]. Structured data is typically organized in databases, making it easy to search, filter, and analyze, while unstructured data - such as text from web pages, social media, or news portals - requires more advanced processing methods.

Internet APIs provide a standardized way to access structured data and facilitate integration between different systems. In cases where such interfaces are unavailable, web scraping serves as an effective alternative for extracting information from unstructured sources [2]. This technique enables the automated collection of data from

**Correspondence to: Georgi Hristov, Second English Language School „Thomas Jefferson " Trayanova Vrata St., Sofia 1408, Strelbishte Residential Complex, and Sofia University "St. Kliment Ohridski", 15 Tsar Osvoboditel Blvd., Sofia 1504, e-mail: georgi.hristov@2els.com*

websites and has broad applications - from monitoring prices in e-commerce to analyzing market trends and gathering research resources.

A key element in modern information processing is machine learning (ML). It provides powerful tools for pattern recognition, trend prediction, and process automation, relying on advanced algorithms capable of analyzing large volumes of data [3]. By combining web scraping and ML, it becomes possible to create systems that not only extract data but also analyze it in depth, uncovering patterns invisible to traditional methods.

In recent years, the integration of APIs, web scraping, and machine learning has found widespread application across diverse fields such as business, marketing, healthcare, science, and automation. These technologies enable the creation of intelligent solutions that optimize processes and support informed decision-making. One practical example of such an approach is the development of price comparison programs, which combine automated data extraction, database storage, and analysis to help users identify the most cost-effective offers.

API and data access

APIs (Application Programming Interfaces) provide a standardized way to exchange data between systems. The most popular approach is REST, which uses standard HTTP methods. However, other options exist, such as SOAP for more secure communication, GraphQL for requesting only the exact data needed, and gRPC for fast and efficient operations in microservice environments. Depending on organizational needs, APIs can be public, private, or partner oriented.

Web scraping

When an API is unavailable, web scraping allows automated extraction of data from websites. The process typically involves downloading the web page, parsing the HTML code, and structuring the extracted information into formats such as CSV or JSON. For dynamic websites, tools like Selenium are used to emulate user actions. This technique is widely applied-for example, for price monitoring in e-commerce, analyzing market trends, collecting customer reviews, or building datasets for machine learning. It is crucial to follow ethical and legal standards during web scraping, such as respecting the rules defined in a website's robots.txt file.

EXPERIMENTAL

Machine learning and data analysis

Machine Learning (ML) is a key technology for analyzing large datasets and making predictions.

Types of learning:

- Supervised learning - uses labeled data to find relationships between inputs and outputs. It is applied in classification (spam detection, medical diagnostics) and regression (financial forecasting) [4].

Unsupervised learning - detects hidden patterns in unlabeled data, such as in market segmentation.

Applications

ML is used in healthcare for analyzing medical images, in finance for fraud detection and forecasting, in marketing for recommendation systems, and in autonomous vehicles for processing sensor data in real time.

Tools and technologies

Different platforms are used for developing and implementing ML models:

- Scikit-learn - for classic algorithms and rapid prototyping.
- TensorFlow and PyTorch - for deep learning and complex neural networks.
- Pandas and NumPy - for data processing and preparation.

Metrics such as accuracy, precision, recall, and F1-score help evaluate models and improve their performance.

Challenges and opportunities

Key challenges include limited data accessibility, the need for process optimization, and adherence to ethical principles.

Nevertheless, the combination of APIs, web scraping, and machine learning opens up significant opportunities for automation, scientific research, and the development of intelligent systems.

Methods

This report is grounded in an in-depth review and analysis of numerous scientific publications and university lecture materials related to web scraping and machine learning [5]. The sources were critically examined to identify key principles, current methodologies, and emerging trends. Insights from these materials were synthesized to build a comprehensive understanding of the subject and to present well-structured conclusions that integrate both theoretical foundations and practical applications [6].

Data extraction through Web scraping

Web scraping is a technique for extracting information from web pages automatically, which is beneficial when data is not available through official APIs. The web scraping process typically involves several key steps:

- Downloading the web page: The HTML content of the page is retrieved through HTTP requests.
- Parsing the HTML page: After the content is downloaded, tools such as BeautifulSoup or lxml are used to analyze and structure the HTML code.
- Extracting and structuring data: Once the desired elements are identified, the data is extracted and converted into structured formats like JSON or CSV, making it easier for further analysis.
- Handling dynamic content: Many modern websites load content dynamically through JavaScript. For such cases, tools like Selenium are employed to emulate user behavior within a browser.
- Optimization and ethical considerations: Effective web scraping requires managing large data volumes through parallel requests and caching to reduce server load. It is also critical to adhere to ethical and legal standards while scraping.

Machine learning and data analysis

Machine learning (ML) is a field within artificial intelligence focused on developing algorithms capable of analyzing data and making predictions. Its primary goal is to enable computers to learn from data without being explicitly programmed for specific tasks. Advancements in natural language processing (NLP) and self-learning models have further expanded the range of machine learning applications.

Application of machine learning in data analysis

Once data is extracted via web scraping or APIs, it can be processed and analyzed using various machine learning algorithms:

- Supervised learning - uses labeled datasets where each input has a known output. During training, the model identifies patterns between inputs and outputs, using optimization techniques such as gradient descent to minimize errors. This method is used in image recognition, medical diagnostics, and financial forecasting.
- Unsupervised learning - works with unlabeled data, detecting hidden structures or clusters. It is applied in market segmentation, customer behavior analysis, and anomaly detection.

Tools and technologies for machine learning

Developing and deploying machine learning models involves multiple libraries and platforms:

- Scikit-learn - ideal for classic algorithms like linear and logistic regression, decision trees,

random forests, and clustering.

- TensorFlow and PyTorch - used for deep learning tasks, enabling the creation of complex neural networks with GPU acceleration and regularization techniques such as Dropout and L1/L2 normalization [7].
- NumPy and Pandas - essential for data preprocessing and manipulation. NumPy provides efficient multidimensional array operations, while Pandas supports powerful data filtering, grouping, and aggregation [8].

Real-world applications

The integration of web scraping, APIs, and machine learning provides practical solutions across multiple industries:

- E-commerce: Monitoring competitor prices, analyzing customer reviews, and building recommendation systems.
- Finance: Detecting fraudulent transactions, forecasting stock trends, and evaluating credit risks.
- Healthcare: Analyzing medical images, predicting disease risks, and optimizing patient treatment plans.
- Marketing and advertising: Identifying customer preferences, segmenting audiences, and personalizing campaigns.
- Transportation: Enhancing autonomous driving systems by processing real-time sensor data and traffic information.
- Academic research: Collecting and processing large datasets for trend analysis, behavioral studies, or AI training [8].

Software development

In the initial stage of development, a working prototype of the program was created and tested. This first version functioned entirely through the terminal without a graphical user interface. The execution of commands and interaction with the system were performed via the command line, where the results were displayed in a structured text format. The usage instructions were included as comments at the beginning of the source code to guide users through the available functions and workflow, as shown in Fig. 1.

The early implementation of the system is further illustrated in Fig. 2, which presents the terminal-based environment in which the program operated. This version successfully demonstrated the ability to extract product information, calculate price differences, and determine potential savings. Although lacking visual interactivity, it validated the core functionalities of web scraping, data storage, and basic price comparison.

The experimental evaluation of this version focused on ensuring stable data collection from e-commerce websites, consistent integration with the database, and reliable calculation of price differences. Selenium WebDriver was used to simulate user actions and access dynamically loaded content, while SQLite was employed to store and manage the extracted data. These components formed the foundation for further development of the program into a graphical application.

RESULTS AND DISCISSION

This project represents the development of a tool designed to automate the process of extracting and analyzing product price information from online stores. The primary objective is to offer users a convenient means of comparing prices and identifying the most cost-effective offers, while also storing the collected information in a structured database for future use. At the very beginning, the program was implemented as a terminal-based tool, where users interacted only through the command line. This version focused on the core functionality: scraping product data, storing it in a database, and displaying results in a structured text-based format. The tool could show the best and worst prices, the difference between them, and potential savings. Although effective, this approach was

```
"""
Price Comparison Tool

This script scrapes product prices from multiple Bulgarian e-commerce websites
and allows you to compare prices to find the best deal.

How to use:

1. First run will:
   - Create a database to store product information
   - Scrape initial product data (this may take a few minutes)
   - Allow you to search for products

2. Subsequent runs will:
   - Use the existing database
   - Give you the option to update prices or just search

3. When searching:
   - Enter keywords like "keyboard", "mouse", "Logitech", "Razer", etc.
   - View detailed price comparisons to see how much you can save

"""
```

Fig. 1. Instructions for using the program, included as comments at the beginning of the source code.

not user-friendly for people unfamiliar with working in a console environment. Because of this limitation, we decided to extend the project by adding a graphical user interface (GUI). The GUI version allowed users to search for products more intuitively, update prices with a single click, and view results in a visually organized layout. This step significantly improved accessibility and usability, making the tool suitable for a wider range of users.

Technologies used

The tool was developed in Python, chosen for its flexibility and rich ecosystem of libraries.

- Selenium: Used for automated interaction with websites, particularly where prices are dynamically loaded via JavaScript. Unlike traditional approaches such as *requests* or *BeautifulSoup*, Selenium is capable of simulating user actions like clicking, entering text, and navigating through pages, which makes it essential for handling modern e-commerce websites.
- SQLite: A lightweight relational database, ideal for small- to medium-scale projects. It stores information about products, including name, price, link, and the time of scraping. Python's built-in *sqlite3* module was used for seamless integration.
- Logging: A logging mechanism was implemented to record program activity, which greatly helped in identifying errors and debugging issues during development.

Functionality and workflow

- Database initialization: On the first run, the tool creates a database (if none exists) and defines the tables necessary for storing product attributes.
- Data extraction: Selenium WebDriver loads the configured websites, parses the content, and extracts relevant price information for predefined products.
- Processing and storage: The collected data is stored in the database, ensuring duplicate entries are avoided while existing records are updated.
- Search and comparison: Users can search for

```
Price Comparison Tool
============================================================
All products have been scraped successfully!

Enter product to search (or 'q' to quit): MX keys

Found 3 products matching 'MX keys':
============================================================
------------------------------------------------------------

==============================
PRICE COMPARISON SUMMARY
==============================

⏳ Best price: ardes.bg - 189.00 лв
💰 Most expensive: desktop.bg - 239.00 лв
💸 Maximum savings: 50.00 лв (26.5%)
   by choosing ardes.bg over desktop.bg

Detailed price comparison:
------------------------------------------------
Site            Price           Savings
------------------------------------------------
ardes.bg        189.00 лв       BEST PRICE
ozone.bg        189.99 лв       0.99 лв (0.5%)
desktop.bg      239.00 лв       50.00 лв (26.5%)
------------------------------------------------
Total price range: 189.00 лв - 239.00 лв
Price difference: 50.00 лв

============================================================

Enter product to search (or 'q' to quit):
```

Fig. 2. Terminal-based environment showing the operation of the first working version of the program.

products using keywords, and the program returns a formatted list of results with prices and their sources.

- Price updates: The tool provides the option to update all prices in the database by running a new scraping process.
- Price analysis: Beyond simple price listing, the system calculates the potential savings by selecting the cheapest available offer and comparing it against more expensive ones.

Technical challenges

One of the major challenges was handling websites with dynamically loaded content, where prices appear only after JavaScript execution. Traditional scraping methods failed to capture this data, so Selenium was used to simulate a real browser session and load all hidden elements.

Another difficulty was dealing with anti-scraping mechanisms on some websites. These required techniques, such as changing the user

agent, managing cookies, and introducing delays between requests to prevent the program from being blocked. Additionally, ensuring that data remained consistent across multiple sources required careful database design and error handling.

During the transition from the terminal version to the GUI version, we also faced issues related to data presentation. While the console version worked well for developers, designing a clear and intuitive visual interface required additional effort in layout organization and interaction logic. This was a valuable learning experience, as it combined both backend (scraping and database) and frontend (user interface) development.

Following the validation of the terminal-based version, the program was further developed into a graphical user interface (GUI) application. This extension significantly improved usability by allowing users to interact with the system in a more intuitive and accessible way. Instead of working exclusively through command-line inputs, users could now perform searches, update prices, and view results with a single click.

The developed interface is presented in Fig. 3. It displays a structured and user-friendly environment where product queries can be entered and executed. Compared to the first version, this interface reduces the learning curve for new users and increases the overall efficiency of interaction with the program.

The results obtained after creating user queries are illustrated in Fig. 4. The program successfully collected and processed product data from the selected online stores, displaying the extracted information in a clear and organized format. In addition to listing current offers, the system automatically highlighted the lowest and highest prices, calculated the differences between them, and identified potential savings.

This version of the program confirmed the stability and reliability of the underlying web scraping and database mechanisms, while also addressing the limitation of accessibility observed in the terminal-based prototype. The GUI-based implementation, therefore, represents a more complete and practical solution for automated price comparison, serving as a basis for further improvements such as expanding the range of supported websites and integrating predictive machine learning models.

Future development

The tool has significant potential for future improvements:

- Expanding the number of websites supported for scraping.
- Optimizing the scraping speed by reducing unnecessary delays.
- Adding more advanced filters and sorting options in the GUI.
- Implementing additional use cases, such as collecting educational resources, analyzing documents, or comparing prices for tickets and travel services.
- Incorporating machine learning techniques to analyze historical data and predict future price trends.

Fig. 3. Interface of the developed author's software product.

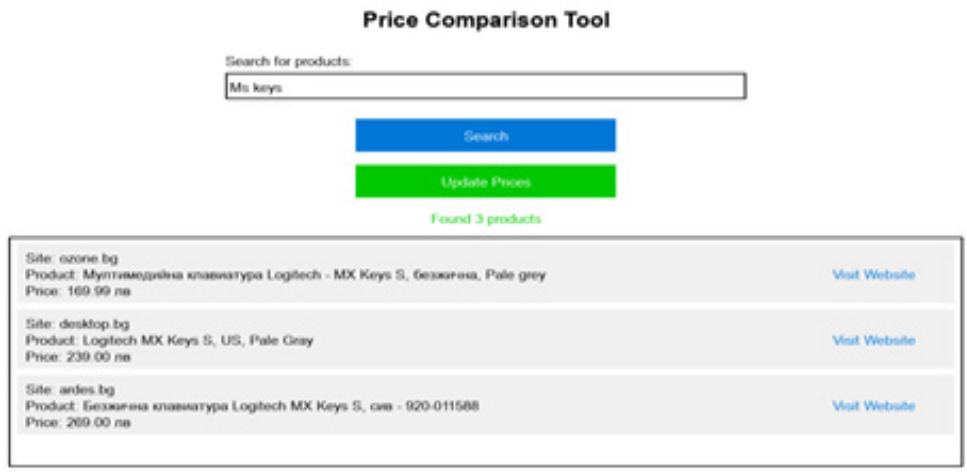

Fig. 4. Results obtained after creating user queries.

CONCLUSIONS

The development of this tool demonstrates how the combination of web scraping, databases, and user interfaces can provide a practical and accessible solution to a real-world problem. Starting from a simple terminal-based prototype and evolving into a fully functional graphical application, the project reflects both the technical challenges of working with dynamic websites and the importance of usability in software design.

By integrating Internet APIs, web scraping, and machine learning in the future, the system could become even more powerful-capable not only of presenting the cheapest option today but also of forecasting the best time to make a purchase.

REFERENCES

1. H. Tarale, Web Scraping using Python, An Online Peer Reviewed, 3, 1, 2025, 1025.
2. A. Jeyaraj, Scaffolding in Business Analytics Education: Using Python for Web Scraping, Journal of Information Systems Education, 35, 4, 2024, 438-450.
3. G. Kamingu, Python vs R for Data Scientists: Procedures and Functions, Optimall series on Python vs. R for Data Scientists (English), 001, 003, 2023.
4. An end-to-end platform for machine learning. https://www.tensorflow.org, Accessed 24 08 2025.
5. Scikit-learn, Machine Learning in Python. https://scikit-learn.org/stable/index.html
6. M. Valcheva, 3 main approaches to mashine education,. SoftUni, https://softuni.bg/blog/three-types-of-machine-learning (in Bulgarian)
7. Scikit-learn, Machine Learning in Python. https://scikit-learn.org/stable/documentation.html, Accessed 24 08 2025
8. NumPy user guide. https://numpy.org/doc/2.2/user/index.html.
9. F1_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html, Accessed 24 08 2025.