African Institute for Mathematical Sciences (AIMS Senegal)

Web Scraping, Data Cleaning and Interactive Dashboard Development using Streamlit

Submitted by

TIKOME NANA Ismaila

December 2025

Abstract

This project focuses on building a complete data collection pipeline that involves web scraping, data cleaning, database storage, interactive visualization, and user evaluation through online forms. A Streamlit web application was developed to automate data extraction from the Coinafrique platform, store raw and cleaned datasets in a SQLite database, visualize key insights through dashboards, and collect evaluations from users via both Google Forms and KoboCollect. The system provides an integrated and efficient solution for handling semi-structured web data and deploying a functional analytics interface.

Contents

Introduction

With the exponential growth of online classified platforms across Africa, extracting relevant information from websites has become essential for market analysis, research, and automation. This project aims to develop a fully functional data collection and analytics ecosystem for animal listings on the Coinafrique Senegal platform.

Using Python, BeautifulSoup, SQLite, and Streamlit, we designed a pipeline capable of:

- Scraping datasets across multiple web pages.
- Cleaning and preprocessing extracted data.
- Storing information in a structured relational database.
- Visualizing insights through statistical dashboards.
- Collecting user feedback through two evaluation systems.

This report presents the methodology, system architecture, implementation details, results, and conclusion.

Objectives

The main objectives of this project are:

1. Extract data across multiple pages of Coinafrique using BeautifulSoup.
2. Clean and standardize scraped data (e.g., price formatting).
3. Store raw and cleaned data in a SQLite database.
4. Build an interactive Streamlit application to:
   - launch the scraper,
   - download uncleaned datasets,
   - explore cleaned data via a dashboard,
   - fill out an evaluation form.
5. Provide a dual evaluation system using Google Forms and KoboCollect.

Methodology

0.1 Web Scraping

The scraping phase uses the BeautifulSoup library to collect animal advertisement data (name, price, location, and image). The scraper works over several pages using incremental URLs:

```
https://sn.coinafrique.com/categorie/chiens?page=1
```

Each page is parsed and transformed into a structured Python dictionary.

0.2 Data Cleaning

The raw price values contain non-numerical characters (e.g., “250.000 FCFA”). A cleaning function extracts digits to convert prices into integers:

```
import re
digits = re.sub(r"[^0-9]", "", price_raw)
```

Missing or invalid values are handled safely.

0.3 Database Storage

A SQLite database was selected for simplicity and local integration with Streamlit. Three tables were created:

- **raw_ads**: unprocessed scraped data
- **cleaned_ads**: cleaned and standardized listings
- **evaluations**: stored user feedback (if needed)

Data is inserted via the pandas `to_sql` method.

0.4 Streamlit Application

The application includes the following modules:

- Scraper module with progress bar.
- CSV download module for uncleaned data.
- Dashboard with filtering, metrics, and charts.
- Evaluation page integrating Google Forms and KoboCollect.
- Database viewer for inspection.

0.5 Evaluation System

Two evaluation platforms were integrated:

0.5.1 Google Forms

A Google Apps Script automatically creates a form based on project requirements. Responses are stored in a Google Sheet.

0.5.2 KoboCollect

An XLSForm was generated to deploy a web-based KoboToolbox form. Responses are collected through the Kobo online dashboard.

Results

0.6 Scraped Data

The scraper successfully extracted:

- Names of animal listings
- Raw and cleaned prices
- Addresses
- Image URLs

These datasets were saved as:

- `category_raw.csv`
- `category_cleaned.csv`

0.7 Dashboard Summary

Several visual analytics were produced:

- Histogram of price distribution
- Bar chart of top locations by number of listings
- KPI metrics (average price, categories, total ads)

0.8 Application Features

The final Streamlit app is:

- visually appealing (custom theme + gradient background)
- interactive (filters, metrics, download buttons)
- complete (scraping + visualization + evaluation)

Discussion

The system performs well for moderately sized datasets and provides a full workflow from collection to analysis. Some challenges included:

- Website HTML changes requiring updates in scraping classes.
- Managing missing values in price or address.
- Slow scraper performance when pages contain many images.

Conclusion

This project successfully demonstrates the development of a complete data-processing pipeline that includes web scraping, cleaning, database storage, and interactive visualization. The Streamlit application provides an accessible platform for end users to launch scrapers, inspect data, explore dashboards, and submit evaluations.

Future improvements may include:

- Deploying the app online (Streamlit Cloud, Render, or Azure).
- Adding machine learning modules for price prediction.
- Improving scraping robustness with Selenium for dynamic content.

Appendix: Important Code Excerpts

.1 Scraper Function

```
def scrape_category(url_base, category_key, max_pages=10):
    ...
```

.2 Streamlit Dashboard

```
st.header("Dashboard - Cleaned Data")
...
```

Appendix: Directory Structure

```
app/
 |- data/
 |- images/
 |- .streamlit/
 |      |- config.toml
 |- my_app.py
 |- requirements.txt
```