Subject

Information Assurance and Security

Second Deliverable Of The Project

Market Assessment of Electric Vehicles Using Web Scraping

Authors

Salma Azouzi   Eya Maalej

Sarra Jebali   Meriem Houissa

Kmar Abessi

Professor

Mrs. Manel Abdelkader

Institution : Tunis Business School

Major: Business Analytics **Minor:** IT

Academic Year : 2024-2025

Contents

1 System Design

1.1 Brief Description

This project provides a practical, data-driven assessment of electric vehicle (EV) options for potential buyers. It leverages automated data collection and sentiment analysis to guide decision-making.

1.2 System Layers

Our solution follows a modular architecture with four main stages:

- Data Collection Layer
- Preprocessing Layer
- Analysis Layer (Sentiment Analysis & Feature Benchmarking)
- Storage and Visualization Layer
- Alert system

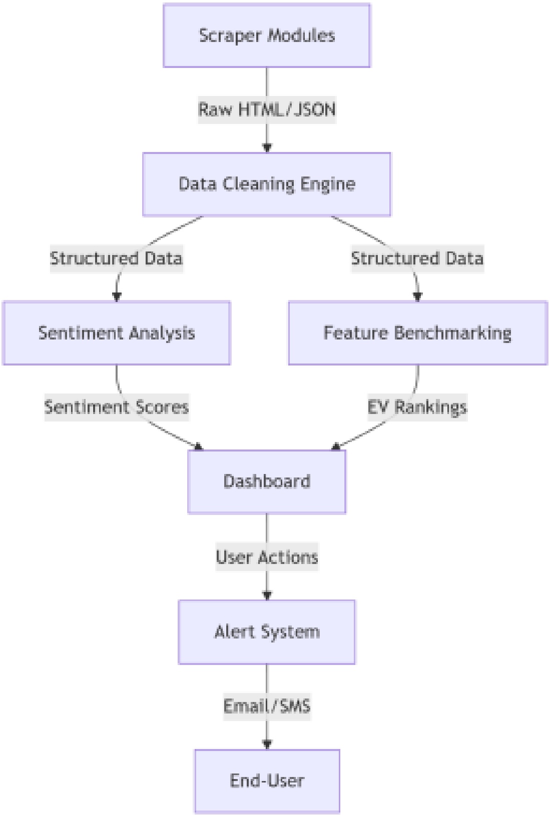

Figure 1: System Architecture Diagram showing the four main layers and their interactions

2 Communication

- Components communicate using JSON-formatted messages passed via internal APIs.
- Scheduler sends requests to the scraping module with a list of URLs.
- Scraping output is sent to the NLP module via message queues.

3 Components

Component	**Subcomponent**	**Function**
Scraper Modules	Static Site Scraper	Extracts data from static HTML pages using BeautifulSoup/ixml.
	Dynamic Content Scraper	Captures JavaScript-rendered content via Selenium.
	API Integrator	Fetches structured data from public APIs (Twitter, Reddit, etc.).
	Proxy/CAPTCHA Handler	Avoids IP bans and solves CAPTCHAs using proxy rotation and 2CAPTCHA.
Data Cleaner	Deduplicator	Removes exact and near-duplicate entries using hashing and fuzzy matching.
	Unit Normalizer	Converts units (km→mi, €→$), standardizes dates and formats.
	Text Preprocessor	Cleans text (stopwords, emojis), lemmatizes for NLP readiness.
	Missing Value Handler	Imputes or flags missing values (e.g., prices, text fields).
Sentiment Analyzer	Text Vectorization	Converts text to numerical form using TF-IDF or Word Embeddings.
	Sentiment Classifier	Labels text as Positive/Neutral/Negative using VADER or BERT.
	Trend Analyzer	Tracks sentiment changes over time and compares brands/models.
Benchmark Engine	Data Normalization	Scales features for fair comparison (Min-Max, Z-Score).
	Feature Weighting	Assigns importance to features (user-defined or PCA).
	Similarity Scoring & Ranking	Ranks EVs based on feature similarity to preferences.
	Clustering	Groups similar EVs using K-means (optional).
Alert System	Trigger Conditions	Detects events (price drop, new model, sentiment spike, etc.).
	Notification Channels	Sends alerts via Email (smtplib), SMS (Twilio), In-App (Firebase).
	User Preferences	Supports custom alert types, frequencies, and mute options.
Dashboard (Output)	Live Data Overview	Displays real-time market/sentiment stats.
	EV Comparison Tool	Allows side-by-side feature comparisons.
	Alert Center	Highlights unread alerts and provides links.
	User Customization	Saves filters, manages watchlists, and adjusts preferences.

4 Data Sources

Data is collected from diverse online platforms:

Category	Examples	Key Data
Official EV Websites	Tesla, Rivian, BYD	Technical specs, pricing, and new releases.
Online Marketplaces	CarGurus, Autotrader, EV.com	Price trends, dealer offers, and availability.
Review Platforms	Edmunds, Trustpilot, Kelley Blue Book	Customer reviews, ratings, and complaints.
Social Media	Twitter/X, Reddit (r/electricvehicles), YouTube	Sentiment trends, buzz metrics, and engagement.
Deal Trackers	Electrek, InsideEVs	Promotions, industry news, and model launches.

5 Workflows

5.1 Data Collection Workflow

Objective: Automatically extract relevant EV-related data from official sites, online marketplaces, review platforms, and social media.

Steps:

1. **Target Selection**: Identify key data sources such as Tesla.com, Autotrader.com, Trustpilot, Reddit, and Twitter.
2. **Scraping Strategy**
   - Use Requests + BeautifulSoup for static pages.(HTML)
   - Use Selenium for JavaScript-heavy and dynamic content.
3. **Legal Compliance**: We will check each website's robots.txt to verify allowed access.
4. **Proxy & Identity Handling**:
   - Rotate IPs using proxy pools.
   - Randomize user-agent headers to mimic real users.
5. **Extraction**:
   - Retrieve product names, prices, specs (battery, range, charging), and user reviews.
6. **Data Storage**:
   - Save raw output in JSON or CSV format for preprocessing.

Output:

```
{
    "model": "Tesla Model 3",
    "battery_kwh": "60",
    "price": "$39,999",
    "review": "Great car but charging is slow"}
```

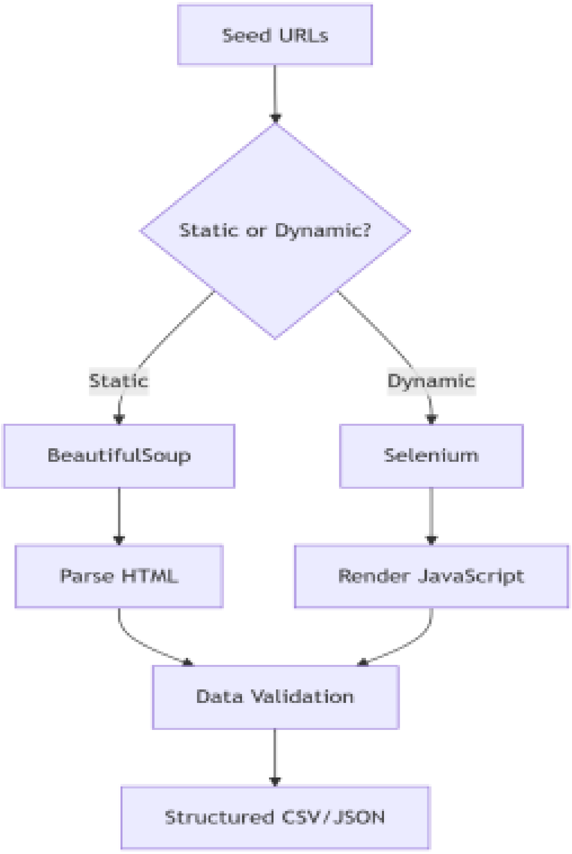

Figure 2: Data Collection Workflow Diagram

5.2 Data Processing Workflow

Objective: Clean and prepare the scraped data for analysis.

Steps: **Steps:**

1. **Load Raw Data**: Import CSV/JSON using Pandas.
2. **Deduplication**: Remove identical or repeated listings.
   - We will be using pandas.drop_duplicates() with subset=['model', 'price', 'source'].
3. **Unit Normalization**:
   - Convert currencies to USD.
   - Convert distance units to kilometers.
   - We will be using the ‘pint‘ library for unit conversions.
4. **Missing Value Handling**:
   - Fill missing fields where possible or remove incomplete entries.
5. **Data Structuring**:
   - Organize data into tables: Features, Reviews, Prices.
   - These tables are sent to Sentiment Analysis and Benchmarking Workflows.

Output:

```
{
    "model": "Tesla Model 3",
    "battery_kwh": 60,
    "range_km": 420,
    "charging_time_min": 45,
    "price_usd": 39999,
    "review": "Great car, but charging is slow"
}
```

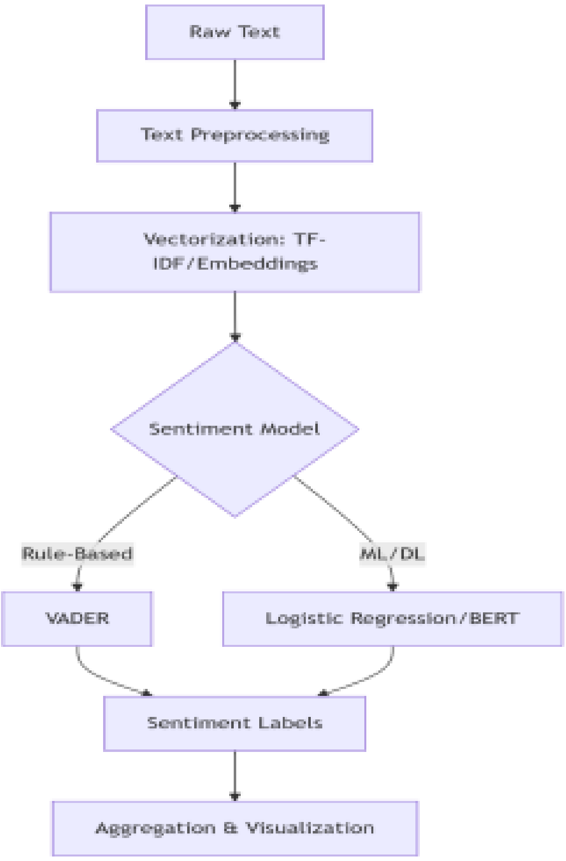

Figure 3: Data Processing Workflow

5.3 Sentiment Analysis Workflow

Objective: Analyze customer reviews to determine public perception of different EV models.

Steps:

1. **Text Preprocessing**: Remove punctuation, uppercase, emojis, and tokenize.
2. **Sentiment Scoring**:
   - Apply VADER for short-form reviews. It gives us a score from -1 (very bad) to +1 (very good).
   - Optionally use BERT, a deep learning model for nuanced context
3. **Classification**:
   - Classify reviews as Positive, Neutral, or Negative based on polarity scores.

- > 0.05: Positive
- -0.05 to 0.05: Neutral
- < -0.05: Negative

4. **Output Storage**: Append sentiment class and score to each review in the dataset.

Output:

```
{
    "review": "Great car, but charging is slow",
    "compound_score": 0.15,
    "sentiment": "neutral"
}
```

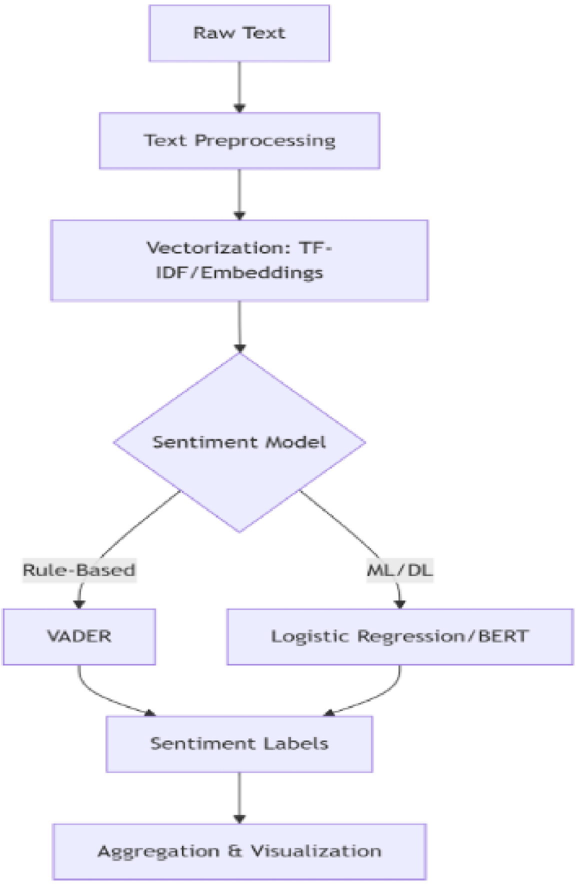

Figure 4: Sentiment Analysis Workflow

5.4 Benchmarking Workflow

Objective: Rank EVs using normalized feature comparisons.

Step	**Action**	**Formula/Model**	**Example Output**
1. Normalization	Min-Max scale: $\frac{x-\min}{\max-\min}$	sklearn.MinMaxScaler	300 mi → 0.75 (if max=400).
2. Weighted Scoring	User-defined weights: Score = Σ(weight * norm_value).	Dynamic weights in config.	Score = 0.8*range + 0.2*price.
3. TOPSIS Ranking	Rank by proximity to ideal solution.	scipy.spatial.distance.	[”model”: ”Tesla”, ”rank”: 1].
4. Clustering	Group similar EVs using K-means ($k = 5$).	sklearn.cluster.	Cluster 3

Output Example:

```
{
    "model": "Tesla Model 3",
    "score": 0.86,
    "rank": 1,
    "similar_to": ["BYD Atto 3", "Hyundai Ionig 5"]
}
```

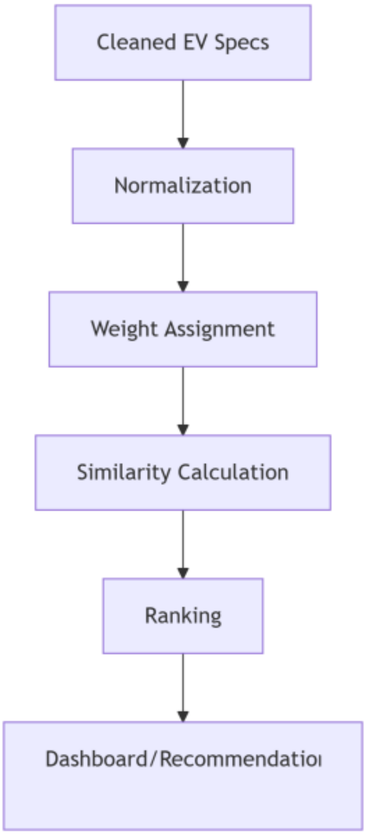

Figure 5: Benchmarking Workflow

5.5 Alert Generation Workflow

Objective: Notify users when predefined conditions are met (e.g., price drops, new models).

Steps:

1. **User Preference Setup**:
   - Users define price thresholds or keywords (e.g., "Model Y under $40K").
2. **Trigger Check**:
   - After each scraping cycle, check if new data matches any condition.
   - Trigger checks run after each scheduled scrape (every 6 hours) using the schedule or external cron jobs.
3. **Alert Formatting**:
   - Format alert with key info (model, price, source, timestamp).
4. **Notification Delivery**:
   - Display on the dashboard or send via email (smtplib)

Output:

```
{
    "alert": "Tesla Model 3 dropped below $40,000",
    "price": 39999,
    "link": "https://autotrader.com/tesla-model3"
}
```

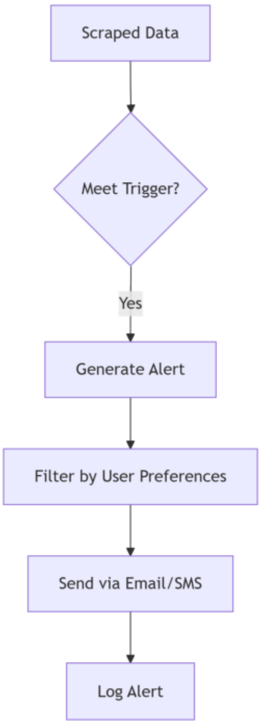

Figure 6: Alert Generation Workflow

5.6 Visualization and Reporting Workflow

Objective: Present data insights through an interactive, real-time dashboard.

Steps:

1. **Data Retrieval**:
   - Load processed data (pricing, sentiment, specs) from the database.
2. **Chart Generation**:
   - Plot pricing trends (line chart), sentiment distribution (pie chart), and feature scores (bar chart).
3. **User Interaction**:
   - Add filters for price, brand, feature, or sentiment level.
4. **Report Exporting**:
   - Provide download options (e.g., "Top EVs under 40K").

Output:

- Web dashboard with:
  - Graphs
  - Filters
  - Alerts
  - Export buttons

Each of these workflows is interconnected, forming a secure, modular pipeline, from raw data scraping to real-time insights and alerting. By integrating these stages, the system empowers users with up-to-date EV comparisons, crowd sentiment, and customized deal alerts on a centralized dashboard.

6 Users

User Type	Purpose	Key Interactions	Tools Used
End Users (Buyers)	Make informed EV purchases using real-time insights.	View dashboards, set alerts.	Power BI, Twilio/smtplib
Data Analysis	Extract insights to improve recommendations.	Analyze sentiment, refine benchmarks.	VADER/BERT, scikit-learn, PyMCDM
System Admins	Manage infrastructure, ensure security/compliance.	Deploy updates, monitor health.	AWS EC2, Docker, Power BI Service

7 Tools & Technologies

Category	**Tools/Libraries**	**Purpose**
Data Collection		
Web Scraping	BeautifulSoup/lsml	Parse static HTML content.
	Selenium	Extract dynamic content (JavaScript-heavy sites like YouTube, CarGurus).
	Scrapy	Large-scale, modular scraping.
APIs	Tweepy (Twitter), PRAW (Reddit)	Fetch structured social media data.
Anti-Blocking	Scrapy-ProxyPool	Rotate IPs to avoid bans.
	2CAPTCHA	Solve CAPTCHAs automatically.
Data Processing		
Cleaning	pandas	Deduplicate, normalize units/currencies, handle missing values.
Storage	spaCy/nltk	Preprocess text (lemmatization, stopword removal).
	PostgreSQL	Store structured data (specs, prices).
	MongoDB	Handle unstructured data (reviews, social media posts).
Analysis		
Sentiment	VADER	Rule-based sentiment scoring (fast for MVP).
	BERT	Deep learning for nuanced sentiment (sarcasm/context detection).
Benchmarking	scikit-learn	Feature clustering (K-means), dimensionality reduction (PCA).
	PyMCDM (TOPSIS)	Rank EVs based on weighted criteria (price vs. range).
Alerts		
Notifications	smtplib	Send email alerts (price drops).
	Twilio	SMS notifications for urgent updates.
Queueing	Celery + Redis	Schedule and prioritize alert tasks.
Dashboard		
Visualization	Power BI	

8 Development Phases

Phase	**Dates**	**Key Tasks**	**Phase**
Scraping Bot Development	Apr 30 - May 6	Build static/dynamic scrapers with proxy/IP rotation	Core Scraping
Data Processing	May 4 - May 9	Database setup and data cleaning	Data Pipeline
Analysis	May 7 - May 14	Implement sentiment and benchmark analysis	Analysis
Alert System	May 10 - May 14	Configure email/SMS notifications	Alert System
Testing	May 13 - May 16	Conduct anti-blocking and stress tests	Testing
Deployment	May 17 - May 18	Containerize and deploy to production	Deployment