Aptech
COMPUTER EDUCATION
Unleash your potential

Insight Bot

Prepared BY:

Data Minds

Project Documentation

Automated News
Extraction & Visualization System

Table of Contents

1. Project Overview

Project Title: InsightBot
Domain: Data Science Dominion
Theme: Daily News Simplified

Description:
InsightBot is a data science-driven project that automates the process of collecting, cleaning, and extracting news articles from multilingual websites (English, Arabic, and Russian). The system uses web scraping and pattern mining to simplify raw news content into clean summaries, which are then displayed in a user-friendly interface with interactive dashboards. This solution helps readers and professionals save time, avoid information overload, and stay updated with trustworthy and relevant news.

2. Problem Definition

Current challenges or gaps:

People are overloaded with hundreds of news articles and blog posts every day across multiple platforms.

It is difficult to filter out trustworthy and relevant news because articles are often long, filled with ads, or structured differently from site to site.

Readers waste time switching between multiple websites and languages (English, Arabic, Russian) to stay updated.

Existing tools are either too complex or not designed to handle multilingual, real-time simplification of news.

Why is this solution needed?
There is a growing demand for personalized, real-time delivery of news content. Readers want quick access to clean and simplified information without going through unnecessary details. By applying data science techniques (web scraping, pattern

3. Objectives

Automate news data ingestion:
Develop a system that automatically scrapes news articles from multiple multilingual websites (English, Arabic, Russian) on a regular basis without manual intervention.

Apply preprocessing and cleaning:
Implement preprocessing steps to remove unnecessary HTML tags, ads, and scripts while normalizing text for consistency, ensuring that only relevant content (titles and article bodies) is retained.

Train models for article classification:
Use rule-based and pattern mining techniques to accurately identify article headlines and main bodies, and validate this extraction logic on unseen test websites to ensure generalization.

Store data in MongoDB/CSV:
Save the cleaned and structured news data in multiple formats (MongoDB, CSV) for future analysis, scalability, and integration with other tools.

Provide access through Streamlit & Power BI:
Design a user-friendly interface using Streamlit to browse articles, along with interactive dashboards (via Power BI) that visualize trends such as article count, language distribution, and popular topics.

4. System Architecture

Data Flow Diagram:
The following diagram illustrates the overall architecture of InsightBot, showing how data flows from news websites through preprocessing, extraction, and storage, and finally to the user interface and dashboard.

Block Explanations (brief):
Data Ingestion - Collects HTML pages from multilingual news/blog websites (training + testing).

Preprocessing - Cleans and normalizes raw HTML, removing ads/scripts and isolating useful tags.

Pattern Matching & Extraction - Applies rule-based logic to identify article titles and main bodies.

Content Storage - Saves structured outputs in CSV format for further analysis and visualization.

Model Training & Testing - Uses 40 websites for training and 10 unseen sites for testing generalization.

User Interface (UI) - Displays simplified news content in a clean, user-friendly format with filtering.

Dashboard Visualization - Provides insights (article trends, language distribution) using Tableau/Power BI.

5. Functional Requirements

The system must provide the following main functionalities:

Web Scraping with Python - Collect articles automatically from multiple news sources.

Data Cleaning & Preprocessing - Remove noise, normalize text, and prepare structured datasets.

Pattern Mining & Modeling - Extract patterns (titles, headlines, context, language) and support classification or sentiment analysis.

Database Management (MongoDB) - Store articles and metadata in a scalable NoSQL database.

User Authentication - Secure login and registration for users and admins.

Admin Dashboard - Interface for managing users, monitoring data, and controlling the system.

Power BI Visualization - Generate dashboards and visual reports (trends, sources, languages).

6. Non-Functional Requirements

Scalability – The system should handle increasing amounts of news sources and data without major redesign.

Performance – Data scraping, cleaning, and visualization should run efficiently with minimal delays.

Reliability – The pipeline must work consistently, recover from errors, and ensure data integrity.

Usability – Interfaces (Streamlit, Power BI dashboards) should be intuitive and user-friendly.

Maintainability – Codebase and architecture should be modular and easy to update or extend.

7. Tools & Technologies

The following tools and technologies were used to develop the system:

Python 3.13 – Core programming language for scraping, preprocessing, and modeling.

Jupyter Notebook – For experimentation, prototyping, and testing.

MongoDB – NoSQL database to store cleaned articles and metadata.

Streamlit – Interactive web app to browse and filter scraped articles.

Power BI – Dashboard visualization of article trends, sources, and languages.

Libraries (BeautifulSoup, Requests, Pandas, NLP & Transformers, Sumy) –
- BeautifulSoup & Requests for web scraping.
- Pandas for data cleaning and manipulation.
- NLP & Transformers + Sumy for text processing, summarization, and analysis.

Workflow & Pipeline Tools – Used to manage data flow and ensure smooth end-to-end processing.

Perfect – For workflow reliability and orchestration.

Note: Other supporting libraries are included in the requirements.txt file and not all are listed here individually.

8. Implementation

8.1 Data Ingestion – Collect articles via Python (Requests, BeautifulSoup) using RSS feeds or HTML parsing.

8.2 Data Preprocessing – Clean text (remove ads, stopwords, duplicates) and structure into title, body, metadata.

8.3 Modeling & Pattern Extraction – Apply NLP, Transformers, and Sumy for headlines, summaries, and multilingual support (EN/AR/RU).

8.4 Storage – Save articles in MongoDB and export to CSV (UTF-8-SIG).

8.5 Application Layer (Streamlit) – Interactive interface to browse, search, and filter news.

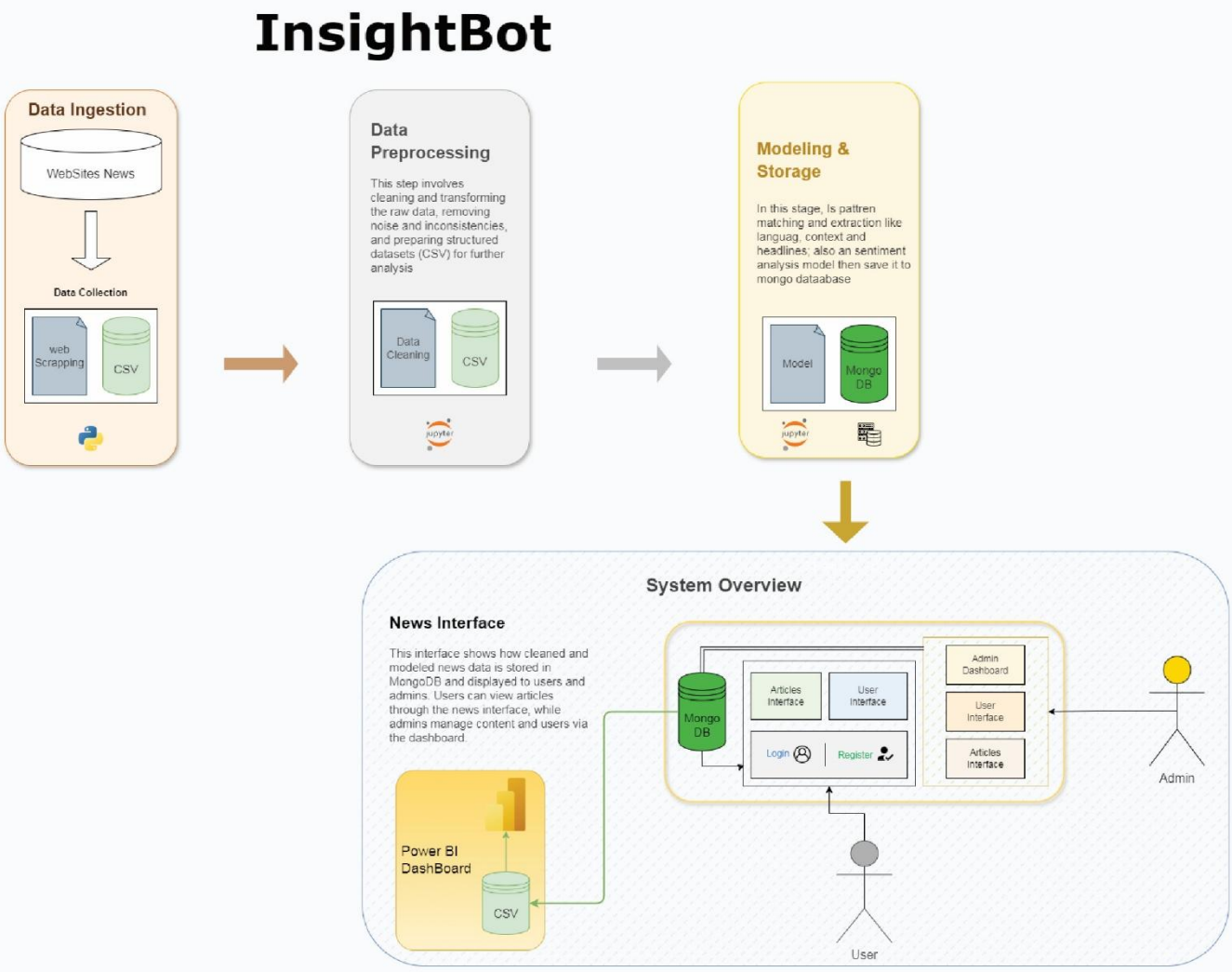
8.6 Visualization (Power BI) – Dashboards for trends, sources, and language insights.

9. Results & Screenshots

This section presents the main outputs of the system, including workflow diagrams, application interfaces, and dashboards.

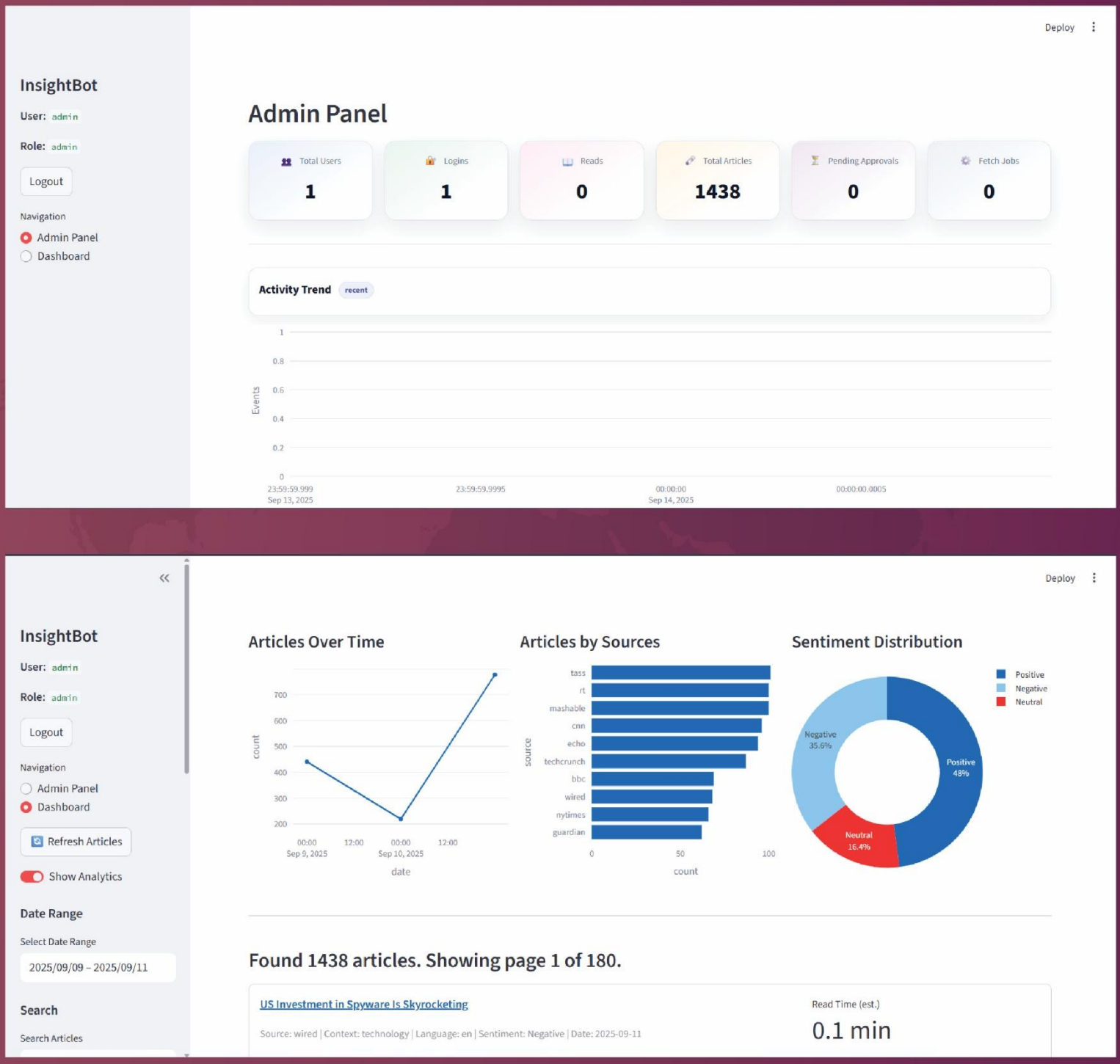
System Workflow (draw.io)

Figure 9.1: Data Flow Diagram showing the pipeline from data ingestion to visualization.



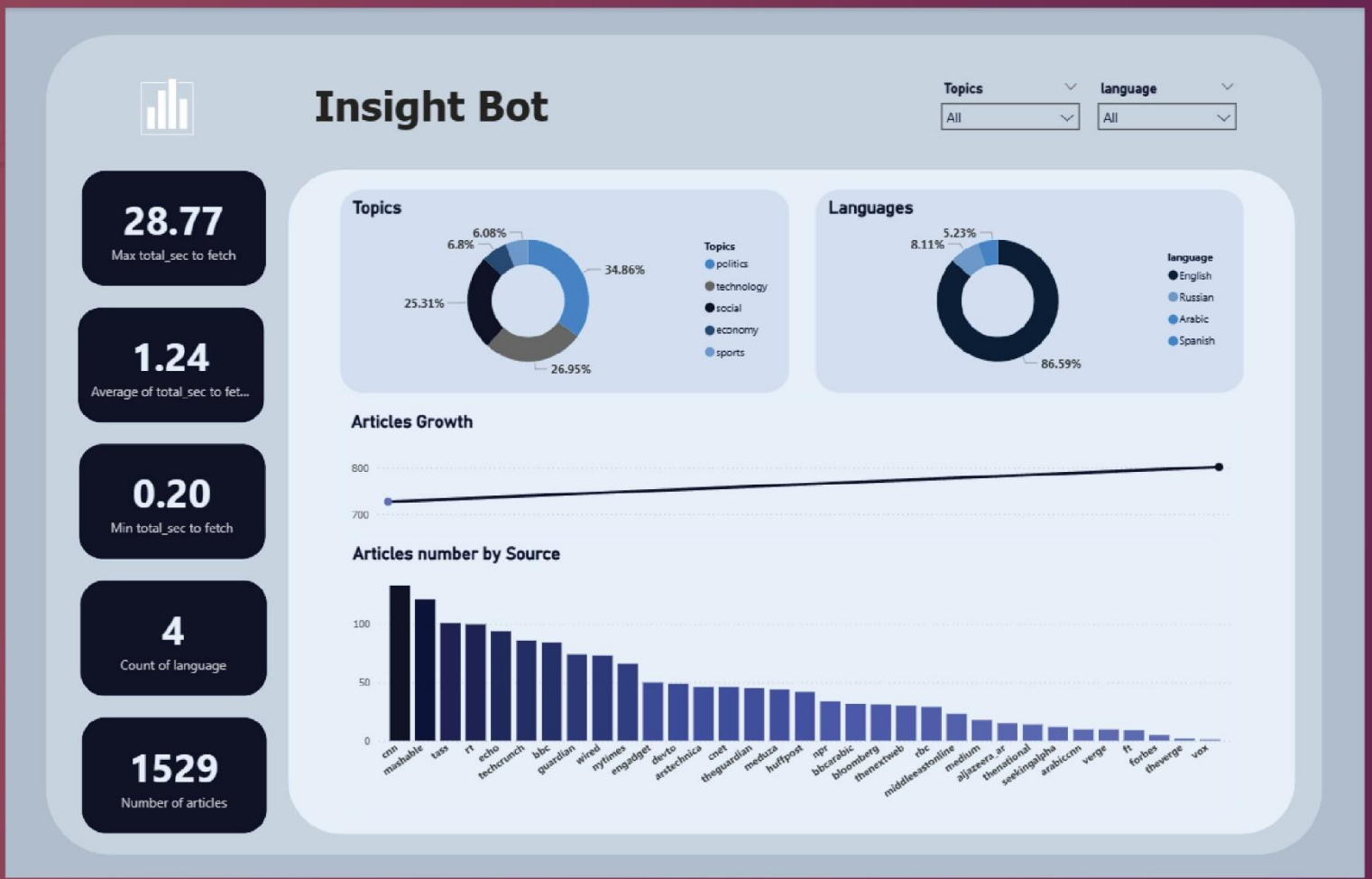
Streamlit Application

Figure 9.2: Screenshot of the Streamlit interface for browsing and filtering articles.



Power BI Dashboard

Figure 9.3: Visualization of article distribution by source, language, and time.



10. Challenges & Limitations

Website Structure Changes

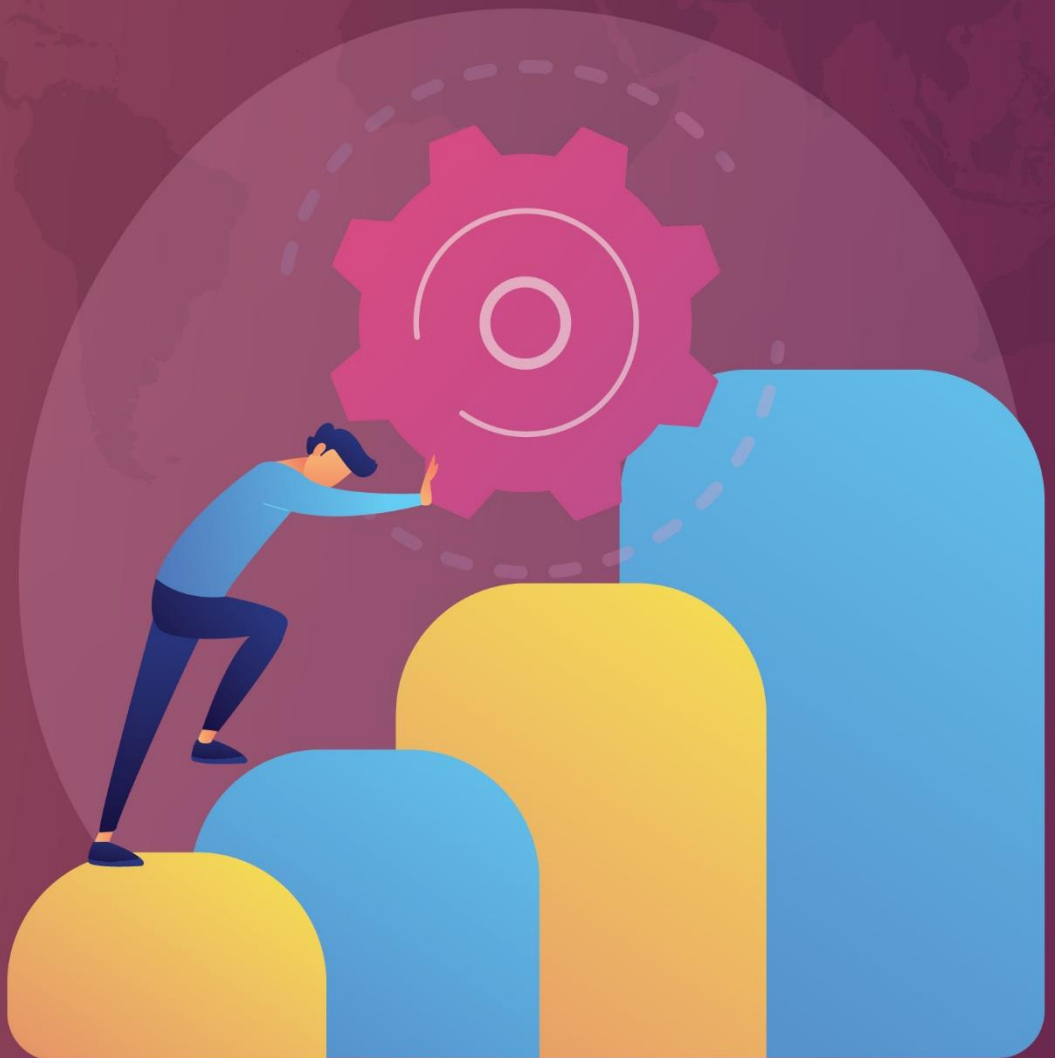
Many news sites update their HTML structure frequently, which can break scraping rules and require constant maintenance.

Multilingual Handling

Supporting English, Arabic, and Russian increases complexity in preprocessing, encoding, and NLP tasks.

Power BI Integration Pending

Full integration of automated pipelines with Power BI dashboards is still in progress and not fully completed.



11. Future Scope

Sentiment Analysis – Analyze articles to detect positive, negative, or neutral tone.

Fake News Detection – Integrate models to identify misinformation across sources.

Real-Time Deployment – Enable live scraping and instant dashboard updates.

Extended Multilingual Support – Improve coverage for more languages beyond English, Arabic, and Russian.

12. Deliverables

- **- Source code (Python + Notebooks).**
- **- Project report (this documentation).**
- **- GitHub repository.**
- **- Data flow diagram.**
- **- Video demo.**
- **- Blog article.**

13. Conclusion

This project successfully demonstrates a complete pipeline for automated news collection, preprocessing, modeling, and visualization. By combining web scraping, NLP techniques, and interactive dashboards, the system provides valuable insights into multilingual news data. Future improvements such as sentiment analysis, fake news detection, and real-time deployment will make the system even more powerful and practical.