Automated ETL Pipeline with Machine Learning Insights

This project presents the design and implementation of an Automated Extract–Transform–Load (ETL) pipeline that collects data from multiple heterogeneous sources, including CSV files, REST APIs, and web scraping, processes the data through cleaning, feature engineering, anomaly detection, clustering, and statistical analysis, and generates interactive reports.

The pipeline supports automation via Windows Task Scheduler and includes email notifications to alert stakeholders of the pipeline's execution status

Project overview and objective

In modern data-driven organizations, the Extract–Transform–Load (ETL) process forms the backbone of analytics and decision-making. Traditional manual pipelines are often prone to delays, errors, and inconsistencies, which can lead to unreliable insights and delayed reporting. Automating the ETL process addresses these challenges by ensuring timely data availability, error-free transformations, and consistent reporting, thereby enabling stakeholders to make informed decisions efficiently.

This project presents a modular, Python-based ETL pipeline that integrates machine learning techniques for deeper insights and can be automated to run at scheduled intervals without manual intervention. The system is designed to handle heterogeneous data sources, including CSV files, APIs, and web scraping, and transforms raw data into structured formats suitable for analysis and reporting.

The main objectives of the project are:

1. Develop a modular ETL system capable of handling multiple data sources efficiently.
2. Implement robust data transformation logic, including cleaning, anomaly detection, clustering, and feature engineering.
3. Automatically generate processed datasets and analytical reports in multiple formats.
4. Enable automation and scheduling of the pipeline to run periodically without manual intervention.

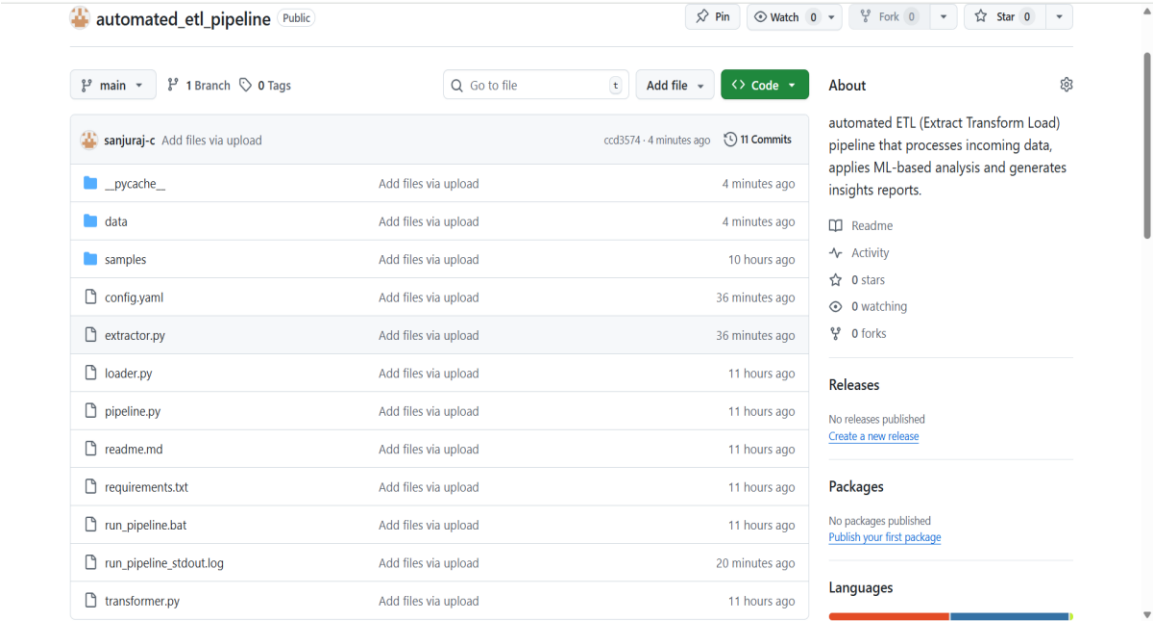
5. Provide real-time notifications via email, informing stakeholders of pipeline execution status and potential issues.

This unified approach ensures that the pipeline is not only scalable and extensible, but also capable of delivering actionable insights with minimal human intervention, serving as a blueprint for enterprise-level ETL solutions.

Environmental Setup

The project The project is developed using Python, which provides a flexible and powerful platform for data processing and machine learning. To ensure consistency and avoid dependency conflicts, a dedicated virtual environment is created using tools such as venv or. All required Python libraries are specified in the requirements.txt file and can be installed using the standard pip package manager. These libraries include essential packages such as pandas, numpy, scikit-learn, matplotlib, seaborn, requests, beautifulsoup4, pyyaml, and yagmail. The project follows a modular folder structure, with separate directories for raw and processed data, sample datasets, and core scripts like extractor.py, transformer.py, loader.py, and pipeline.py. Additionally, Windows Task Scheduler is configured to automate the execution of the pipeline at regular intervals, ensuring smooth and timely data processing. This environment setup provides a reproducible and scalable foundation for running the ETL pipeline efficiently across different systems.

GitHub Project Setup:



Project Structure

```
automated_etl_pipeline/
│
├── config.yaml              # Central configuration
├── extractor.py             # Extraction logic
├── transformer.py           # Transformation & ML
├── loader.py                # Data loading & reporting
├── pipeline.py              # Orchestration
├── run_pipeline.bat         # For Windows Task Scheduler
│
├── data/
│   ├── staging/             # Raw extracted data
│   └── processed/
│       ├── final_processed_data/
│       └── reports/
│
├── samples/                 # Example input datasets
├── requirements.txt         # Dependencies
└── readme.md                # Documentation
```

System Architecture

Key Components

1. Extractor: Responsible for ingesting raw data from various sources and storing it in a staging layer for intermediate processing. Supports CSV, Excel, API JSON, and web scraping.

2. Transformer: Performs data cleaning, feature engineering, and applies machine learning models such as clustering and anomaly detection.

3. Loader: Saves cleaned and enriched data into multiple formats (CSV, Parquet, JSON) and generates visual reports for stakeholders.
4. Pipeline Orchestrator: Manages execution, logs errors, and ensures smooth operation of the ETL stages.
5. Automation Layer: Enables scheduled execution and sends email notifications to stakeholders regarding pipeline success or failure.

Data Sources

1. CSV Files

- AirPassengers.csv – Airline passenger data for time series analysis. (https://github.com/sanjuraj-c/automated_etl_pipeline/blob/main/samples/AirPassengers.csv)
- Retail_Transaction_Dataset.csv – Transactional sales data for clustering and anomaly detection. (https://github.com/sanjuraj-c/automated_etl_pipeline/blob/main/samples/Retail_Transaction_Dataset.csv)
- PowerConsumption.csv – Electricity consumption data for trend analysis. (https://github.com/sanjuraj-c/automated_etl_pipeline/blob/main/samples/powerconsumption.csv)

2. API Sources

- CoinGecko API – Cryptocurrency market data, including price, volume, and market cap. (https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd)

3. Web Scraping Sources

- IMDb Top 250 Movies – Web scraping used to collect movie rankings, titles, release years, and ratings for analytics. (https://www.imdb.com/chart/top)

Implementation Details

1. Extractor Module

- Handles reading multiple formats: CSV, Excel, SQLite, JSON (from APIs), and HTML (from web scraping).
- Schema validation ensures data integrity by checking row/column counts and mandatory fields.
- Extracted data is stored as timestamped Parquet files in the staging layer for efficient processing.

2. Transformer Module

- Data Cleaning: Handles missing values, removes duplicates, and detects outliers using Interquartile Range (IQR).
- Feature Engineering: Derives new features such as year, month, day, and day-of-week from date columns.
- Normalization: StandardScaler normalizes numerical features.
- Anomaly Detection: Isolation Forest identifies rare events in datasets.
- Clustering: KMeans divides data into meaningful clusters for pattern analysis.
- Statistical Analysis: Computes descriptive statistics, correlations, and distribution visualizations.
- Time Series Analysis: Detects trends, seasonality, and anomalies in temporal datasets.

3. Loader Module

- Saves processed data into CSV, Parquet, and JSON formats.
- Generates HTML reports containing:
    - Summary statistics
    - Anomaly and cluster visualizations
    - Distribution plots and correlation heatmaps
    - Time trends and insights

4. Orchestrator Module

- Orchestrates the complete ETL process.
- Logs execution details at each stage for reproducibility and debugging.
- Sends automated email notifications with execution summaries.

5. Automation Layer

- Uses Windows Task Scheduler to execute run_pipeline.bat at fixed intervals (e.g., every 6 hours).
- Log files include:
    - run_pipeline_stdout.log – captures standard output messages
    - pipeline.log – detailed execution logs per ETL stage

Results

- Fully automated ETL pipeline executed successfully on schedule.
- Processed datasets generated in multiple formats for downstream applications.
- HTML reports with visualizations provide actionable insights for stakeholders.
- Email notifications confirm successful pipeline execution or alert failures.
- Logging enables traceability, debugging, and reproducibility.

Examples of report generated:

1.Report of power consumption data(csv format)

Pipeline Dashboard: samplepowercons_csv

Summary Insights

Total Rows: 1228
Total Columns: 4

Missing Values

Column	Missing
Unnamed: 0	0
TxnDate	0
TxnTime	0
Consumption	0

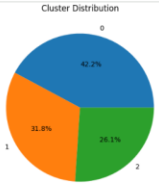
ML Analysis

Anomaly Detection

Type	Count
1	1166
-1	62

Cluster Distribution

Cluster	Count
0	518
1	390
2	320

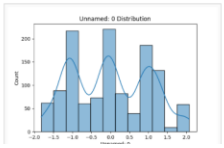


Summary Statistics

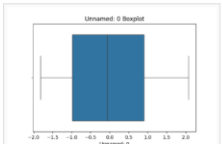
Column	Mean	Std	Min	Max
Unnamed: 0	4.166048613577134 5e-16	1.0004074149690316	-1.8212668791688909	2.0748742649472347
Consumption	-5.786178629968243e-17	1.0004074149690316	-0.774189864303472	3.3124328426154555

Data Distributions

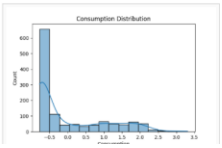
Unnamed: 0 Distribution



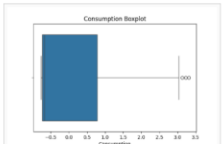
Unnamed: 0 Boxplot



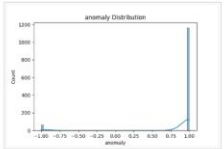
Consumption Distribution



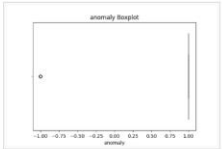
Consumption Boxplot



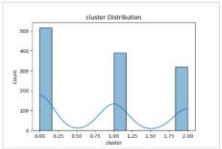
anomaly Distribution



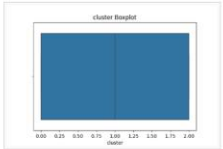
anomaly Boxplot



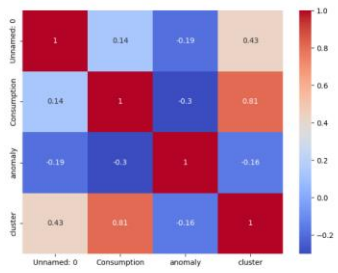
cluster Distribution



cluster Boxplot



Correlation Heatmap



2.Report of imdb data(API)

Pipeline Dashboard: imdb_top

Summary Insights

Total Rows: 25
Total Columns: 5

Missing Values

Column	Missing
rank	0
title	0
year	0
rating	0
link	0

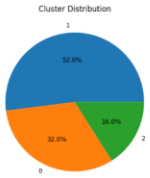
ML Analysis

Anomaly Detection

Type	Count
1	23
-1	2

Cluster Distribution

Cluster	Count
1	13
0	8
2	4

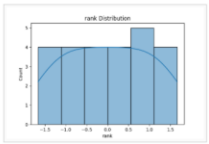


Summary Statistics

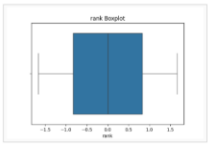
Column	Mean	Std	Min	Max
rank	8.881784197001253e-18	1.0206207261596576	-1.6641005886756874	1.6641005886756874
year	3.597122599785507e-15	1.0206207261596574	-2.3874286731002807	1.4487043555107622
rating	-3.9701575360595596e-15	1.0206207261596574	-1.1059630927007091	2.478176559570095

Data Distributions

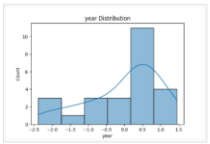
rank Distribution



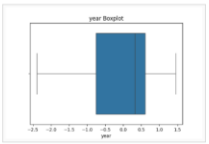
rank Boxplot



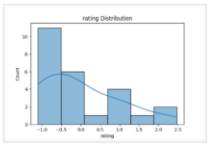
year Distribution



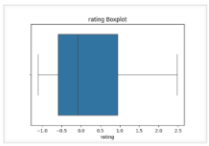
year Boxplot



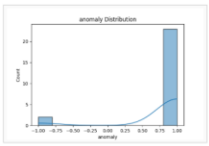
rating Distribution



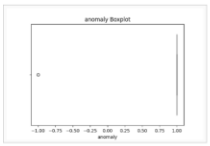
rating Boxplot



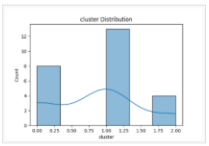
anomaly Distribution



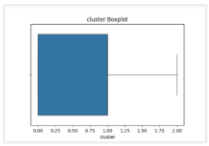
anomaly Boxplot



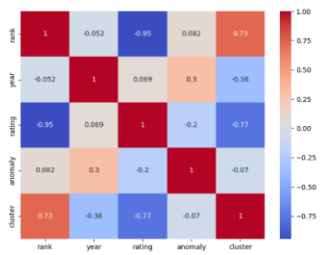
cluster Distribution



cluster Boxplot



Correlation Heatmap



Pipeline Technologies

Technologies Used

- Programming Language: Python 3.
- Python Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn, requests, beautifulsoup4, yagmail, pyyaml
- Storage Formats: Parquet, CSV, JSON
- Automation Tools: Windows Task Scheduler
- Reporting: HTML dashboards and visual summaries

Advantages

- Fully automated, requiring minimal manual intervention.
- Modular design allows easy addition of new data sources and transformation steps.
- Machine learning integration enables deeper insights like anomaly detection and clustering.
- Supports multiple storage formats for flexibility in downstream applications.
- Scalable and extensible, suitable for enterprise deployment.

Conclusion

This project demonstrates the design and implementation of a production-ready automated ETL pipeline with modular architecture, machine learning-based insights, and comprehensive reporting. The pipeline ensures timely and accurate data processing, reduces manual workload, and provides actionable insights for decision-makers. It serves as a blueprint for enterprise-level ETL solutions applicable across domains like retail, finance, energy, and entertainment.

By combining data engineering best practices, ML algorithms, and automation, this system highlights the potential of intelligent pipelines in modern data-driven organizations.