Workshop No. 2

Melisa Maldonado Melenge - 20231020110

Jean Pierre Mora Cepeda - 20231020105

Juan Diego Martínez Beltrán - 20231020131

Luis Felipe Suárez Sánchez - 20231020033

Universidad Distrital Francisco José de Caldas
Computer Engineering Program
School of Engineering

Systems Analysis & Design
Eng. Carlos Andrés Sierra, M.Sc.

2025

Contents

1. Summary of Systems Analysis Findings

During Workshop 1, we analyzed the Kaggle competition ACM SF Chapter Hackathon, Big Data (Best Buy Mobile Web Site), which focuses on predicting the product category a mobile visitor will most likely choose based on search queries and browsing behavior. The dataset is extensive (7 GB) and contains two years of clickstream and textual data, making scalability and preprocessing critical components of the system.

The systems analysis identified several key constraints: the high volume and heterogeneity of data, imbalanced product categories, and noisy, short search queries. These issues make accurate prediction and generalization difficult. Additionally, temporal dependencies between user actions introduce complexity, since intent evolves throughout a browsing session. The system must therefore process sequential patterns efficiently while handling sparse and inconsistent data.

From a complexity perspective, small variations in user behavior can lead to completely different outcomes, for example, a single typo or accidental click may alter recommendations significantly. Such chaotic dynamics require robust models that can manage uncertainty and reduce bias in feedback loops.

To address these challenges, the design should emphasize modular data pipelines, feature engineering for temporal and textual data. It must also include scalable infrastructure to handle large scale ingestion and retraining, and continuous monitoring to adapt to evolving user behavior. These design principles directly respond to the findings of Workshop 1, ensuring a system that is both adaptive and reliable under the constraints of Big Data and the chaotic nature of user behavior.

2. System Requirements Definition

2.1. Functional Requirements

Data Processing & Storage

- The system must handle and process 7GB of user behavior data.
- Must support both cloud scale and PC scale implementations.
- Must parse and integrate multiple data sources: `train.csv`, `test.csv`, and `product_data.tar.g`
- Must handle temporal data with `click_time` and `query_time` relationships.

Prediction Capabilities

- Must predict top 5 SKUs (stock keeping units) that users are likely to click for given queries.
- Must process user search queries and match them to product SKUs.
- Must account for uncertainty in query-click relationships (clicks may not directly result from searches).
- Must generate predictions in the required format (space delimited SKU lists).

Analysis Features

- Must analyze user behavior patterns across categories.
- Must process product metadata including categories, reviews, and attributes.
- Must identify popular SKUs per category as baseline.
- Must handle time series analysis (5 minute window between query and click).

2.2. Non-Functional Requirements

Performance

- Scalability: Must scale from PC sized sample to full 7GB cloud dataset.
- Response time: Predictions should be generated efficiently for 1.2M user queries.
- Memory efficiency: Must optimize for large scale data processing.

Reliability

- Data integrity: Must handle missing or inconsistent data gracefully.
- Fault tolerance: System should recover from processing errors without data loss.
- Robustness: Must handle edge cases (unusual queries, temporal anomalies).

Maintainability

- Code modularity: Separate data ingestion, feature engineering, model training, and prediction components.
- Documentation: Clear documentation of data preprocessing steps and model choices.
- Versioning: Track model iterations and performance metrics.
- Extensibility: Design should allow easy integration of additional features or algorithms.

Usability (User Centric Needs)

Interpretability:

- Provide explainable predictions showing why certain SKUs were recommended.
- Visualize user behavior patterns and category trends.
- Track feature importance for model decisions.

Ease of Use:

- Clear input/output formats matching competition requirements.
- Automated data validation and preprocessing pipelines.

Security:

- Protect user privacy.
- Secure handling of Best Buy product data.

- No exposure of proprietary algorithms or training data.

Compatibility

- Must integrate with Best Buy's product API structure.
- Must support cloud computing platforms.

Robustness

- Maintain consistent recommendations even when user behavior is noisy, incomplete, or unusual.

2.3. Quality Attributes

Accuracy & Precision

- Primary metric: Optimize for MAP@5 evaluation metric.
- Baseline: Must exceed simple popularity based benchmark.
- Validation: Cross validation strategy to prevent overfitting.

Scalability Guidelines

- Horizontal scaling: Distribute processing.
- Vertical scaling: Optimize memory usage for single machine processing.
- Data partitioning: Implement efficient data chunking strategies.

2.4. Visualization Requirements (for Visualization Contest)

- Category level click distribution analysis.
- Temporal patterns in search and click behavior.
- Model performance metrics and comparison visualizations.

2.5. Constraints & Dependencies

Technical Constraints

- 5 minute maximum window between `query_time` and `click_time`.

- No guarantee of direct query click causation.
- Category based product organization.
- Space delimited output format requirement.

Data Dependencies

- Best Buy Product API integration.
- Product attributes dictionary.
- Historical click and query data.
- Product reviews and metadata.

3. High level architecture

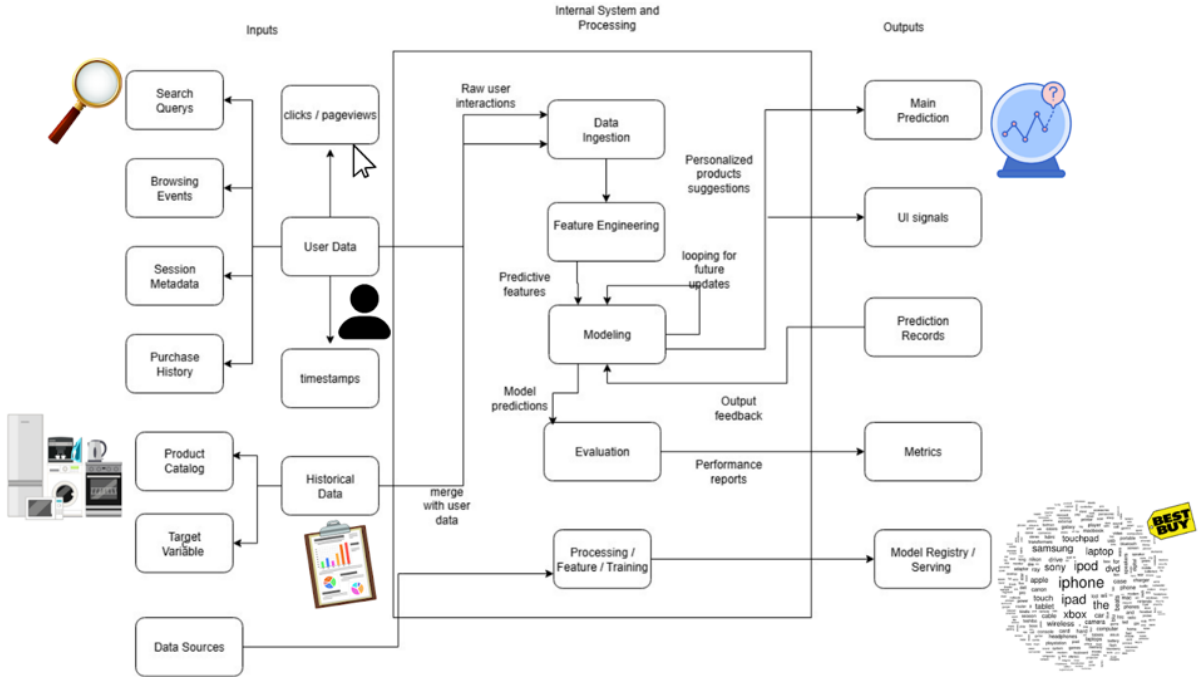

Figure 3.1: High level architecture diagram for the system

MODULES AND THEIR RESPONSIBILITIES

Inputs (Data Sources)

Search Queries / Browsing Events / Session Metadata / Purchase History / Product Catalog / Target Variable: These represent the data sources. They provide raw information about user behavior, product information, and labels used for model training.

Responsibility: Data extraction.

User Data

Aggregates user interactions such as searches, clicks, and browsing history, organizing them by session and user.

Responsibility: Data collection and structuring.

Historical Data

Stores two years of behavioral logs and product metadata. It merges new data with past records for long term trend analysis.

Responsibility: Historical storage and merging.

Data Ingestion

Collects and imports raw user interaction data from multiple sources (mobile logs, catalog, and sessions) into the processing pipeline.

Responsibility: Data acquisition and loading.

Feature Engineering

Transforms and enriches the ingested data by generating predictive variables — such as query embeddings, click frequencies, and session durations.

Responsibility: Data transformation and feature extraction.

Modeling

Uses the engineered features to train a machine learning model that predicts the most relevant product categories for each user session.

Responsibility: Model training and prediction generation.

Evaluation

Assesses model performance using metrics such as NDCG, Recall@K, and MAP. Produces reports that determine whether the model meets deployment criteria.

Responsibility: Performance evaluation and validation.

Processing / Feature / Training

Handles periodic retraining, feature updates, and batch processing to ensure the model remains accurate and adapts to changing user behavior.

Responsibility: Retraining and continuous improvement.

Outputs

- **Main Prediction:** Provides the system's primary output, the predicted product or category the user is most likely to engage with. Responsibility: Final recommendation output.
- **UI Signals:** Displays personalized product suggestions within the Best Buy mobile interface, improving user experience and engagement. Responsibility: User interaction and recommendation delivery.
- **Prediction Records:** Stores past predictions and corresponding user responses for feedback loops and retraining purposes. Responsibility: Data logging and feedback collection.
- **Metrics:** Tracks performance indicators such as model accuracy, latency, and ranking quality for continuous monitoring. Responsibility: Performance measurement and monitoring.
- **Model Registry / Serving:** Manages the storage, versioning, and deployment of trained models. Ensures that the best performing version is used for live predictions. Responsibility: Model deployment and version control.

HOW SYSTEMS ENGINEERING PRINCIPLES SHAPED THE STRUCTURE

Separation of Concerns

Each module has a single clear responsibility (extraction, transformation, modeling, evaluation).

Pipeline with Feedback

The process flows sequentially but includes feedback loops for updates.

Integration of Multiple Data Sources

User, product, and historical data are merged.

Scalability and Reusability

New data sources or models can be added without redesigning the whole pipeline.

Design Rationale

Designing the system through modular components makes it scalable, since each module can be developed, modified, or replaced independently without affecting the rest of the architecture. For instance, if a new feature extraction method such as embedding based representations were introduced to replace TF-IDF, the other modules such as data ingestion, model training, or evaluation—would remain unaffected.

This modularity ensures minimal coupling between components, as they interact only through structured data formats such as DataFrames or serialized feature sets. Each stage—data preprocessing, feature engineering, modeling, and serving, has a single, well defined responsibility, which enhances maintainability and testing.

The system also follows the principle of efficiency, as it is designed to handle large scale (7 GB) behavioral data using optimized data pipelines, lightweight models, and batch processing strategies. This allows it to minimize resource consumption and maximize performance during both training and inference.

Simplicity is maintained by prioritizing clear data flow and intuitive pipeline structure, avoiding unnecessary complexity in feature transformations or model selection. This simplifies debugging and enables other developers or analysts to understand and extend the system easily.

Finally, abstraction is applied by separating the logical layers of the system data handling, model training, and prediction serving, through clearly defined interfaces. This abstraction allows the use of different modeling techniques or data sources without altering the overall architecture, ensuring adaptability and long term sustainability of the system design.

4. Addressing Sensitivity and Chaos

To manage the sensitivity and chaotic behavior present in user interactions and data variability, the system incorporates several design strategies to ensure stability, reliability, and adaptability over time.

Chaos Mitigation

- **Feedback Loop Control:** User interactions (clicks, queries, and purchases) are reintegrated into the training process through controlled, periodic retraining cycles instead of continuous updates. This prevents short term fluctuations or random behaviors from destabilizing the model.
- **Temporal Modeling:** Session based modeling captures user intent over time, reducing the influence of isolated or accidental actions (such as single clicks or one time searches).
- **Ensemble Stability:** The system supports ensemble or hybrid models that integrate textual and behavioral features, improving robustness to outliers and abrupt behavioral changes.
- **Data Regularization:** Feature engineering applies normalization and smoothing techniques to highly variable signals (e.g., query frequency, session duration) to prevent overfitting to extreme values.

Sensitivity Management

- **Robust Evaluation Metrics:** Ranking metrics such as NDCG, Recall@K, and MAP@5 are employed to ensure that minor variations in user data do not disproportionately affect performance evaluation.
- **Retraining Routine:** A scheduled retraining pipeline refreshes the model with new data while preserving historical balance, thereby mitigating concept drift.
- **Uncertainty Handling:** Model outputs include probability distributions or confidence scores to quantify uncertainty and flag unreliable predictions.

- **Monitoring and Alerts:** A continuous monitoring subsystem tracks data distributions, latency, and prediction accuracy. When anomalies or drifts are detected, alerts are triggered, enabling inspection, retraining, or rollback to a stable model version stored in the Model Registry.

5. Technical Stack and Implementation Sketch

For the development and implementation of the proposed system, the Python programming language was selected as the fundamental foundation due to its clarity, modularity, and extensive ecosystem for data science and machine learning. Python is widely used in data mining competitions such as Kaggle, which ensures compatibility with the evaluation format and facilitates experimentation with multiple frameworks. This choice supports the management of large structured datasets, statistical analysis, and the integration of scalable machine learning models within a transparent and maintainable architecture.

A. Technology Stack

- **Language:** Python (chosen for its versatility, scalability, and rich AI/ML ecosystem).
- **Key Libraries:**
  - **NumPy:** Used for numerical operations, including matrix manipulations, vector transformations, and statistical computations necessary for feature generation and model input preparation.
  - **Pandas:** Employed for data ingestion, validation, and transformation of the large `train.csv`, `test.csv`, and product metadata files. It enables grouping, filtering, and aggregation operations that prepare data for efficient processing.
  - **Dask:** Integrated to scale Pandas workflows for large datasets (7 GB), enabling distributed computation and parallel execution on multi core or cloud environments.
  - **Scikit-learn:** Provides core machine learning utilities for preprocessing (scaling, encoding, TF-IDF vectorization) and baseline model implementation (e.g., logistic regression, random forest).
  - **LightGBM / XGBoost:** Gradient boosting algorithms that efficiently handle large scale data and provide accurate ranking predictions for identifying

top product recommendations.

- **Matplotlib and Seaborn:** Used for visualizing model performance metrics, category distributions, and temporal behavior patterns.

B. Modeling Strategy

The modeling strategy is based on a modular pipeline architecture, emphasizing scalability, maintainability, and abstraction. Each component is developed as an independent module that can be replaced or improved without affecting the rest of the system.

Data Ingestion Module

Responsible for loading and validating raw behavioral data from multiple sources (clickstream logs, search queries, and product metadata). It manages data formats, ensures integrity, and partitions data for efficient processing.

Preprocessing and Feature Engineering Module

Applies data cleaning, normalization, and feature extraction techniques. This includes transforming textual queries using TF-IDF or contextual embeddings (BERT), aligning query and click timestamps, and generating time based and categorical features that represent user behavior.

Modeling and Evaluation Module

Trains and evaluates ranking or classification models such as LightGBM or XGBoost. Evaluation is based on the MAP@5 metric (used in the competition), complemented by additional measures of accuracy and recall to ensure robustness. Cross validation techniques are applied to reduce overfitting and improve generalization.

Prediction and Submission Module

Generates ranked SKU lists (top 5 predictions) and formats them according to Kaggle's submission requirements (space delimited format).

Monitoring and Optimization Module

Tracks data drift, evaluates feature importance, and identifies potential bias in model predictions. This ensures that changes in user behavior over time do not degrade model performance.

Bibliography

[1] Glider, Greg; Makowski, Neeral Beladia; and Nick Kolegraff. *Data Mining Hackathon on BIG DATA (7GB) Best Buy mobile web site.* Kaggle, 2012. https://kaggle.com/competitions/acm-sf-chapter-hackathon-big.

[2] Martínez Beltrán, J.D., Suárez Sánchez, L.F., Maldonado Melenge, M., Mora Cepeda, J.P. (2025). *Workshop No. 1 — Kaggle Systems Engineering Analysis* [Workshop report]. GitHub. https://github.com/MelisaMelenge/Data-Mining-Hackathon-on-BIG-DATA-7GB-Best-Buy-mobile-web-site/tree/main/Workshop%201.

[3] BestBuy, "Best Buy Official Online Store," BestBuy.com. [Online]. Available: https://www.bestbuy.com/.