Personalized Product Recommendation

EURON

Use Case: Personalized Product Recommendation

A recommendation engine suggests products to users based on their browsing history, purchase behavior, and preferences. It enhances customer engagement, increases sales, and improves user experience.

Functional Architecture

Key Components:

1. **User Interaction Layer**
    - Web & Mobile Applications
    - APIs for communication
    - Search and browsing behavior tracking
2. **Data Ingestion Layer**
    - Streaming platforms (Kafka/NiFi) to capture real-time user events
    - Batch ingestion (ETL pipelines) for historical data
    - APIs for collecting product details, reviews, and inventory
3. **Data Processing & Storage Layer**
    - User profile database (NoSQL: MongoDB, DynamoDB)
    - Product catalog (SQL or NoSQL depending on scale)
    - Behavioral data store (Clickstream, purchase history, ratings, etc.)
    - Feature store for machine learning models
4. **Recommendation Engine**
    - **Collaborative Filtering** (User-based, Item-based)
    - **Content-Based Filtering** (Text similarity, product attributes)
    - **Hybrid Models** (Ensemble, Deep Learning-based embeddings)
    - **Real-Time Personalization** (Session-based recommendations)
5. **Model Training & Deployment**
    - Feature Engineering (Spark, Databricks, Pandas)
    - Model Training (TensorFlow, PyTorch, MLflow)
    - Model Deployment (SageMaker, Databricks ML, TensorFlow Serving)
    - Online learning & retraining mechanisms
6. **Serving Layer**
    - API Gateway (GraphQL/REST)
    - Caching (Redis, Memcached)
    - Response Time Optimization (Precomputed recommendations, Approximate nearest neighbors)
7. **Monitoring & Feedback Loop**
    - A/B Testing (Experimentation platform)
    - Model Performance Monitoring (MLOps, Drift detection)
    - Continuous Feedback (Reinforcement learning, user ratings)

Technical Architecture

Technology Stack

Layer	Tools & Technologies
Frontend	React.js, Angular, Vue.js (Web), Flutter, React Native (Mobile)
Backend	Node.js, Python (Flask, FastAPI)
Data Streaming	Confluent Kafka, Apache NiFi
Storage	MongoDB (User Data), PostgreSQL/MySQL (Product Catalog), Elasticsearch (Search Index), S3 (Raw Data)
Big Data Processing	Apache Spark (Databricks), Apache Flink (Real-Time Processing)
ML & AI	Scikit-learn, TensorFlow, PyTorch, MLflow, Hugging Face Transformers
Recommendation Algorithms	Collaborative Filtering, Deep Learning (Neural Networks), Hybrid Models (Transformer-based)
Model Deployment	AWS SageMaker, Azure ML, Databricks MLflow
Caching & Search	Redis, Memcached, Elasticsearch
A/B Testing & Analytics	Google Optimize, Apache Superset, Power BI, Tableau
MLOps & Monitoring	Kubeflow, MLflow, Prometheus, Grafana

End-to-End Data Flow

1. **User Interaction:** Users browse products, interact with listings, and make purchases.
2. **Data Collection:** Kafka streams user behavior, clickstream data, and purchase history into storage.
3. **Feature Engineering:** Spark jobs process raw user interactions and transform them into model-ready features.
4. **Model Training:** ML models train on historical data using collaborative filtering, deep learning, and hybrid approaches.
5. **Model Deployment:** The trained model is deployed via an API, serving real-time recommendations.
6. **Recommendation Delivery:** The API fetches recommendations from Redis (cached) or generates new suggestions using the ML model.
7. **Continuous Improvement:** The system monitors user engagement, collects feedback, and retrains models dynamically.

Why This Design Works?

✅ **Scalable:** Kafka & Spark handle real-time and batch processing.
✅ **Real-time & Batch:** Supports both instant recommendations (Redis + real-time models) and batch processing (precomputed ML models).
✅ **Flexible:** Hybrid models optimize accuracy by combining deep learning and traditional ML.
✅ **Optimized for Latency:** Caching, indexing, and efficient model inference reduce response times.
✅ **Feedback Loop:** Improves over time with A/B testing and real-time model retraining.