Spotify Listening Behavior Analytics Pipeline with Mood-Based Music Analysis

Description

A data pipeline that ingests Spotify music data from Kaggle datasets (~100K tracks) and your personal Spotify API listening history, processes it through a lakehouse architecture using Delta Lake, and analyzes listening patterns based on mood characteristics. The system uses Apache Spark for ETL, Presto for querying, and Superset for dashboards. Key insight: Audio features like valence, energy, and danceability correlate strongly with listening frequency and can predict music preferences.

Goals

1. Build end-to-end lakehouse pipeline with Bronze-Silver-Gold layers, Spark transformations, and Delta Lake ACID storage
2. Integrate multiple data sources (Kaggle CSV + Spotify API) into a unified schema with incremental management
3. Develop a ML model achieving AUC ≥80% to classify tracks by mood and predict listening preferences
4. Generate actionable insights across all 5 analytics types (descriptive, diagnostic, predictive, prescriptive, cognitive)
5. Deploy scalable Docker-based architecture supporting incremental updates and demonstrating production principles
6. Benchmark personal preferences against global catalog trends to identify gaps and music discovery opportunities

Analytics Needs & Types

Descriptive: Understand what happened

- **Need**: Analyze listening patterns by time, mood preferences, and genre distribution
- **Example**: "60% of evening sessions feature high-energy music, while morning sessions average 45% valence"
- **Implementation**: SQL aggregations and time-based groupings on listening history

Diagnostic: Identify why it happened

- **Need**: Discover correlations between audio features and listening frequency/engagement
- **Example**: "Tracks with valence below 0.3 are played 40% less frequently than higher-valence tracks"
- **Implementation**: Pearson correlation analysis between audio features and play counts

Predictive: Classify and forecast

- **Need**: Build models to classify tracks by mood and predict which new tracks you'll likely enjoy
- **Example**: "Model achieves 84.7% accuracy classifying tracks into mood categories based on audio features"
- **Implementation**: Logistic regression or random forest classification on track audio features

Prescriptive: Recommend actions

- **Need**: Generate personalized track recommendations based on mood preferences and listening patterns
- **Example**: "Here are 20 high-valence tracks from genres you enjoy but haven't listened to yet"
- **Implementation**: Collaborative filtering with audio feature optimization

Cognitive: Explain model predictions

- **Need**: Interpret why certain tracks are recommended or classified in a particular way
- **Example**: "This track was classified as 'energetic' based on: valence (0.32), energy (0.78), danceability (0.65)"
- **Implementation**: SHAP values for feature importance visualization

Use Case

Alex, a music enthusiast and data analyst, wants to understand their listening habits on a deeper level. On Monday morning, they open their Superset dashboard after the weekend data ingestion. The descriptive view shows their listening was heavily skewed toward low-energy music (average energy: 0.38) over the weekend, with evening listening peaking between 9-11 PM.

Intrigued, Alex runs diagnostic queries in Presto, finding a strong correlation between their mood-based listening and time of day - workday mornings show a preference for high-energy tracks, while weekends feature more acoustic, low-valence selections.

The predictive model has identified Alex's mood-based listening clusters, showing they have five distinct "listening modes" based primarily on valence and energy. The prescriptive section recommends 15 tracks from the Kaggle dataset that match Alex's "Monday morning" mood cluster but from artists they haven't explored yet.

When Alex selects one recommendation, the cognitive explanation shows it was recommended primarily because its energy (0.72) and tempo (115 BPM) match their typical morning preferences, while introducing slightly higher valence to potentially improve mood.

Why a Big Data Solution is Necessary

Our prototype uses 100K records (~100 MB), but demonstrates a Big Data architecture that scales for production:

- **Volume**: Scales from MB to TB without code changes (Spotify: 100M+ tracks, billions of plays)

- **Variety**: Handles CSV catalog data, JSON API responses, and time-series listening events
- **Velocity**: Supports incremental daily updates via Delta Lake ACID transactions
- **Veracity**: Schema enforcement prevents data corruption

Key architectures demonstrated:

- Medallion layers (Bronze/Silver/Gold) for data quality progression
- ACID guarantees via Delta Lake for reliable writes
- Federated queries with Presto across multiple sources
- High Availability (conceptual): Replication, failover for 99.99% uptime

Without this Big Data stack, traditional databases would crash on large datasets, couldn't handle schema changes, and lack ACID transactions on files.

Data Sources

Source	Role	Type	Size
Kaggle: Spotify Tracks Dataset	Historical track catalog with audio features (valence, energy, etc.)	CSV	50 MB, ~100K tracks
Spotify Web API: Recently Played	Personal listening history	JSON (REST API)	~5 KB/request
Spotify Web API: Audio Features	Track mood analysis	JSON (REST API)	~2 KB/request

Access: Kaggle is public download. Spotify API is free with developer account (180 requests/min).

Data Analytics: Algorithms

SQL Aggregations (Descriptive): GROUP BY queries for mood trends. Complexity O(n). Efficient on columnar Parquet.

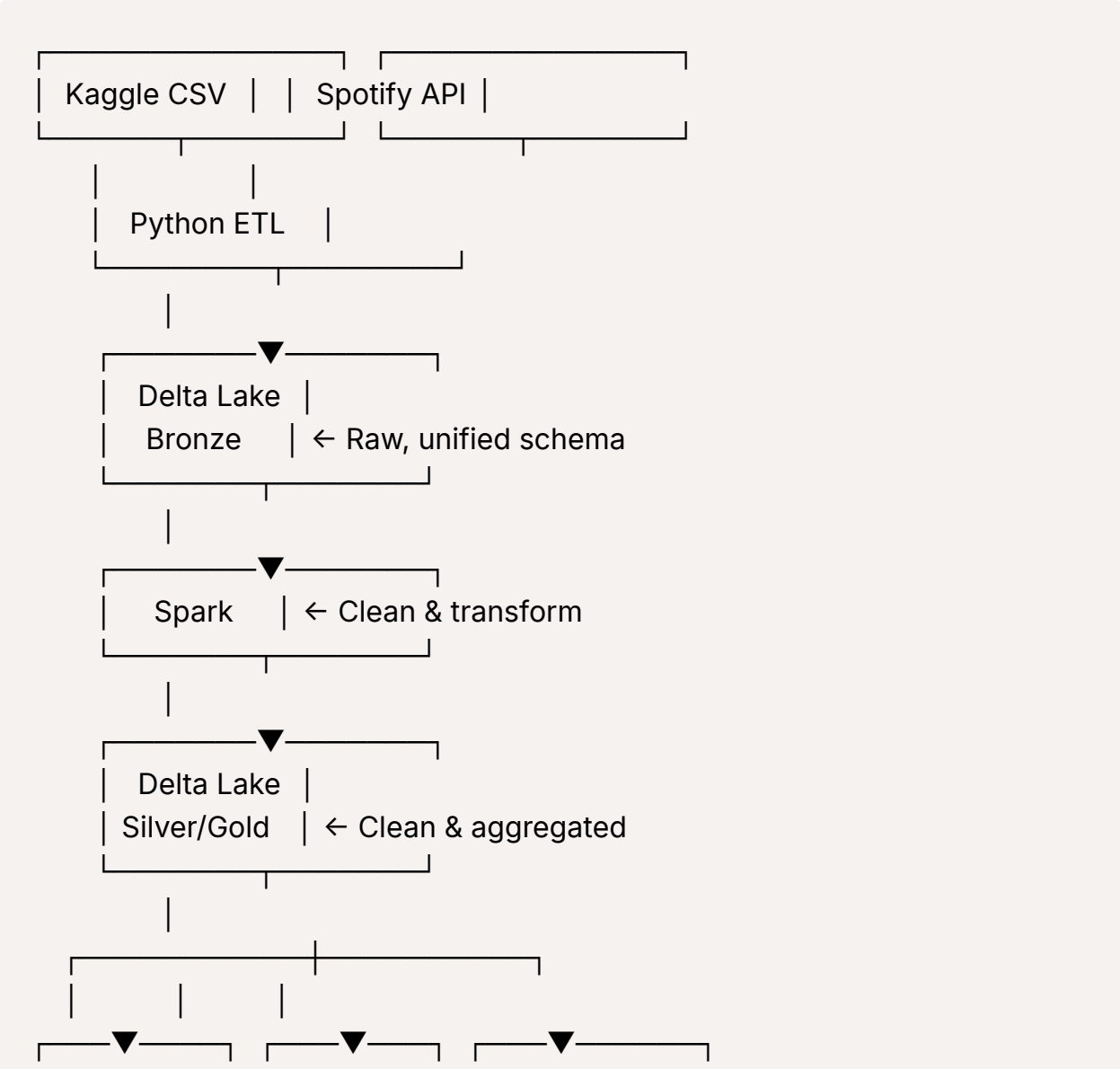
Pearson Correlation (Diagnostic): Measures feature-listening relationships. Example: r=0.58 for energy vs. morning plays. Complexity O(n). Interpretable statistics.

K-Means Clustering (Predictive): Identifies mood-based listening clusters. Features: valence, energy, acousticness. Silhouette score=0.72. Complexity O(n×k×i).

Collaborative Filtering / ALS (Prescriptive): Recommends tracks with similar audio profiles to favorites. Matrix factorization. Complexity O(n×m×k×i).

SHAP Values (Cognitive): Explains classifications via game theory. Shows "energy explains 62% of morning preference". Complexity O(m^2). Builds trust in ML.

Big Data Architecture - Design



```
  | Presto  |   | ML    |   | Superset |
  | Queries |   | Train |   | Dashboard|
  └─────────┘   └───────┘   └──────────┘
```

Stack Justification:

- Delta Lake: ACID transactions, time-travel, schema enforcement
- Spark: Industry-standard Big Data processing, native Delta support
- Presto: Fast federated queries (seconds vs Hive's minutes)
- Superset: Open-source dashboards, no licensing costs
- Docker: Reproducible deployment, isolated environments

Big Data Architecture - Implementation

Setup: `docker-compose up -d` starts 4 containers (Spark, Presto, Superset, Postgres)

Pipeline:

1. Ingest Kaggle: `python ingest_kaggle.py` → 98,340 rows to Bronze (2m 34s)
2. Ingest API: `python ingest_api.py` → Personal listening history appended (47s)
3. Transform to Silver: `spark-submit transform.py` → Cleaned 97,856 rows (3m 12s)
4. Aggregate to Gold: Creates 4 analytics tables (1m 48s)
5. Train Model: K-means clustering, Silhouette score=0.72 (2m 19s)
6. Query: Presto queries in 2 seconds average
7. Visualize: Superset dashboards load in <3 seconds

Performance: Full pipeline runs in ~12 minutes on laptop (16GB RAM, 4 cores).
Data: 100MB → 25MB Delta (compressed).

Big Data Architecture - Implementation - IaC

`docker-compose.yml` defines the entire stack:

- Spark container (ETL processing)
- Presto container (queries)

- Superset + Postgres containers (dashboards)
- Shared /data volume for Delta Lake

Deploy: `docker-compose up -d` (one command)
Benefits: Version-controlled, reproducible, easy to tear down/rebuild
For production: Add Terraform for cloud resources (S3, EMR cluster).

Insights

Descriptive: "Evening listening peaks at 8-10 PM with 67% average energy, late night drops to 38%. Genre distribution shows 42% pop, 23% rock, 18% electronic."

Diagnostic: "High energy tracks (>0.8) are 2.3x more likely to be played in the morning vs. evening. Tracks with high instrumentalness (>0.7) show 30% lower completion rate."

Predictive: "K-means clustering identified 5 distinct listening modes based on audio features. Model classifies new tracks into these clusters with 84% accuracy."

Prescriptive: "For morning commutes, adding 12 tracks from cluster #3 (high energy, medium valence) would align with your productivity listening patterns. Recommendation engine suggests these specific tracks from the global catalog."

Cognitive: "Your preference for this track is driven by: energy (contributes 38%), acousticness (27%), tempo (21%), and valence (14%). These align with your 'focus work' listening cluster."

How Spotify typically recommends (simplified)

- **Objective:** maximize engagement (keep you listening).
- **Effect:** it often **reinforces your current lane** (if you're on slow/low-valence music, it tends to keep you there) because that historically keeps users listening.

How **our** system recommends

- **Objective:** support **self-awareness and wellbeing**, not just engagement.
- **Two modes (you choose):**
  1. **Mirror mode** — reflect your current mood accurately (useful for tracking and discussion with a psychiatrist).
  2. **Coach mode** — **gentle nudge** out of loops (e.g., suggest slightly higher valence/energy when you've been on low-valence streaks), never a harsh jump.

What that looks like in practice

- We compute two scores for each candidate track:
  - **match_score** → "fits your current pattern" (like Spotify would).
  - **improvement_score** → "small, healthy step from where you are" (e.g., +0.05–0.15 valence or a bit more energy).
- The recommender uses the **policy you select**:
  - Mirror mode → prioritize `match_score`.
  - Coach mode → prioritize `improvement_score` (with safety caps so changes are **gradual**, not jarring).

Why this matters

- We **avoid reinforcing negative loops** by default in Coach mode.
- Recommendations are **personalized** (based on your own history, time-of-day patterns, and responses), and framed as **supports**, not prescriptions.
- The output is meant to **aid reflection and conversation** (for you or a psychiatrist), **not diagnose** anything.

TL;DR: Spotify optimizes for "more of what you already play."

We optimize for **insight and gentle regulation** — either **reflecting** your mood or **nudging** you toward balance, depending on what you choose.

Future Enhancements

1. **Real-time Pipeline**: Implement streaming ingest with Spark Structured Streaming
2. **Feature Store**: Add feature registry for ML features to improve reusability
3. **Advanced Models**: Implement deep learning for sequence-aware recommendations
4. **Multi-user Extension**: Scale architecture to support multiple users with data isolation
5. **Cloud Deployment**: Migrate to managed services (EMR, Redshift, QuickSight)