💿 2026 Music Taste Model: The Master Retooling Guide (v2.0)

This guide consolidates our strategic plan to evolve your New Music Friday (NMF) model into a high-variance, research-backed regression engine that doubles as a "First Impression" memory bank.

---

1. The "First Impression" Memory Bank

Your 2026 Gut Scores (30–100) now serve two purposes:

1. **Training Data:** Teaching the model the intensity of your taste.
2. **Discovery Insurance:** A way to flag albums you gave an **85+** on first listen so you don't forget them when ranking your Top 100 in November.

---

2. The Tiered Scoring System (2021–2025 + 2026)

We maximize variance where it matters (the music you like) and use stable bins for the rest.

Category	Score Range	Mental Effort
Elite (Top 100 / 85+ Gut)	**85.0 to 100.0**	**High:** Rank them precisely.
Great (Top 100 / 70-84 Gut)	**70.0 to 84.9**	**High:** Rank them precisely.
Good (Honorable Mention / 61-69 Gut)	**61.0 to 69.9**	**Medium:** Distinguish from "just okay."
Honorable Mention Floor	**60.0**	**Low:** Flat score for "enjoyed it."
Mid (Historical & 2026)	**45.0**	**Zero:** Flat bin for "fine/forgettable."
Nope (Historical & 2026)	**30.0**	**Zero:** Flat bin for "didn't like it."

---

3. Code Update: Data Preparation & Scoring

You will need to update the section of your notebook where you assign the `liked` target variable.

A. New Scoring Logic (Python)

Python

```python
# 1. Load your 2026 Gut Scores
df_2026 = pd.read_csv("data/2026_gut_scores.csv") # Columns: Artist, Album, Gut_Score

# 2. Load Historical Top 100s (Example for one year)
df_top_100 = pd.read_csv("data/top_100_2024.csv")
# Apply the 0.4-point-per-rank sliding scale (100 to 60.4)
df_top_100['liked'] = 100 - ((df_top_100['Rank'] - 1) * 0.4)

# 3. Combine with Binned Mids/Nopes
df_mid['liked'] = 45.0
df_not_liked['liked'] = 30.0

# 4. Final Training Set (Excluding Liked Songs to avoid bias)
df_train = pd.concat([df_2026, df_top_100, df_mid, df_not_liked])
```

---

4. Code Update: The "Network vs. Training" Split

To keep the **PageRank** benefits without the **Disco Bias**, split your logic:

B. Building the Graph (Broad Discovery)

Python

```python
# Build the graph using EVERYTHING (including Liked Songs)
# This ensures "Artist Centrality" remains a broad discovery feature.
G = build_graph(df_liked_songs, df_train, df_liked_similar)
df, centrality_scores = calculate_centrality_scores(G, df)
```

C. Training the Model (Focused Intensity)

Python

```python
# Train ONLY on the high-variance album data
# This prevents single-track outliers from skewing the regression.
features = ['Genres_encoded', 'Artist Centrality', 'Record Label Frequency
```

```
Encoded', 'mood_score']
X = df_train[features]
y = df_train['liked']

model.fit(X, y)
```

---

5. The 80/20 Exploration Strategy

To keep the model's "BS Detector" sharp, follow this weekly NMF workflow:

- **80% Exploitation:** Listen to the top-ranked albums (Predicted > 70).
- **20% Exploration:** Pick 1–2 albums from the **bottom** of the list (Predicted < 40).
- **Calibration:** If you hate the "Exploration" albums, give them a **30**. This confirms the model's negative filters are working.

---

6. Summary of 2026 Improvements

1. **Memory Bank:** Gut scores act as a "don't forget" list for year-end rankings.
2. **High Resolution:** The 0.4-point sliding scale allows the model to learn the nuance of "Elite" music.
3. **Bias Control:** Liked songs power the **Network** (PageRank) but don't skew the **Training** (Regression).
4. **Efficiency:** Binning Mids (45) and Nopes (30) saves mental energy for the music that matters.

Retooling Strategy developed by Manus AI (2025)