Spotify Music Analytics Valence Explorer

Course:
CSc 47400
Instructor:
Yunhua Zhao

Group Members:
Saanavi Goyal
Ezzeldin Moussa
Nirath Hussan

GitHub Repo: https://github.com/saanavig/Spotify-Visualization

Table of Contents

Project Overview

Build an interactive visual analytics dashboard to analyze valence (musical positivity) across audio features and genres

Approach

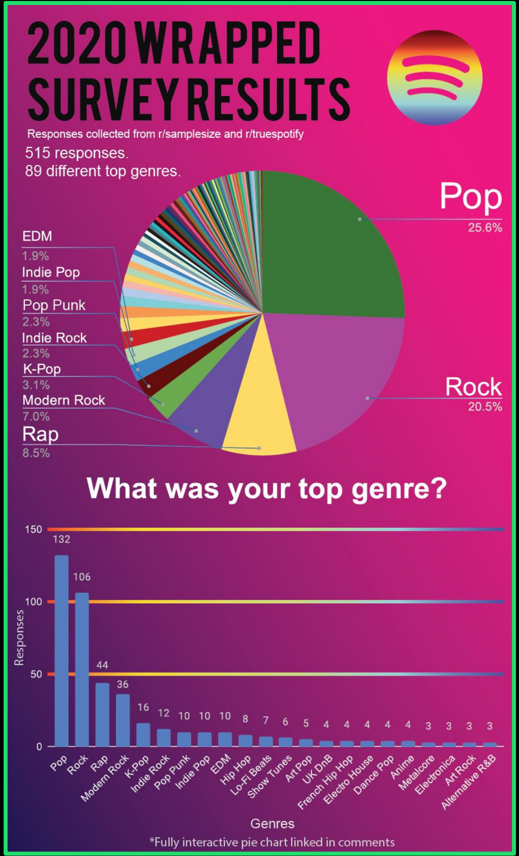
1. Clean and preprocess Spotify track-level dataset
2. Visualize patterns using scatter plots, heatmaps, and parallel coordinates
3. Build an interactive dashboard with filters

Expected Outcome

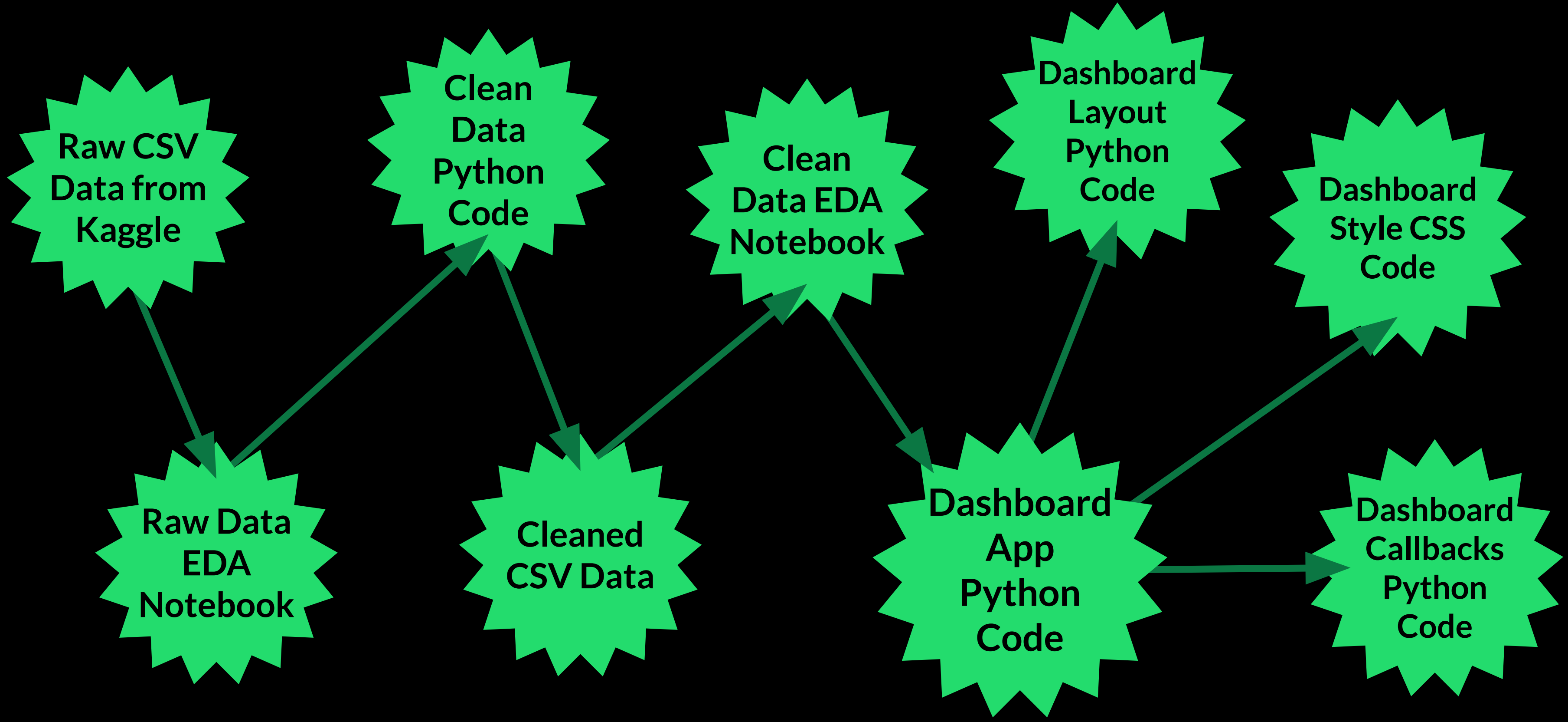
1. Identify trends across audio features and genres
2. Reveal correlations and detect anomalies in popular artists/tracks

Project Significance

- People love **Spotify Wrapped** because it makes data personal + visual
- But Wrapped is once a year while our dashboard gives **year-round insights**
- Turns huge, messy datasets into **fun, interactive stories**, just like Wrapped
- Helps users explore trends in energy, mood, popularity, and genres **on demand**
- Makes music data **easy to understand, playful, and engaging.**



Project Architecture



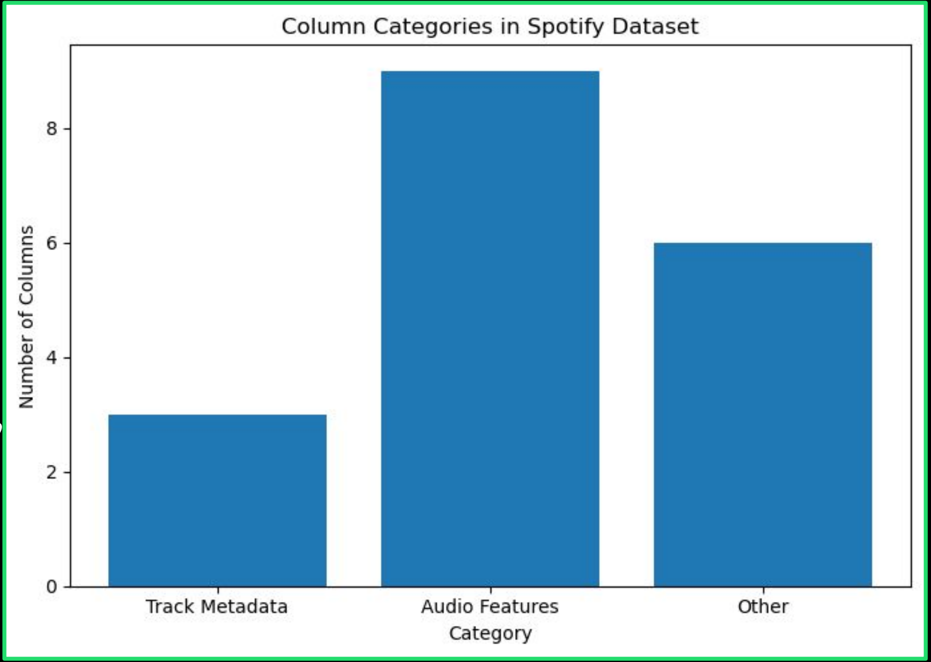
Project Architecture

- *Data:*
  - SpotifyFeatures.csv → raw dataset (Kaggle + custom API pulls)
  - cleanDataset.csv → cleaned dataset (produced by cleandata.py)
- *Src/Cleaning:*
  - cleandata.py → cleaning functions and clean_spotify_dataset() entry
- *Notebooks:*
  - EDA.ipynb → exploratory analysis of raw dataset quick checks (nulls, data types)
  - cleanDataEDA.ipynb → post-cleaning EDA (summary stats, distribution checks, feature selection)
- *Dashboard:*
  - app.py → app bootstrap, app.layout and register_callbacks()
  - layout.py → all UI components and dropdowns (feature, genre, etc.)
  - callbacks.py → plotting logic and callbacks for each chart
  - styles.css → styles for spacing and fonts for the app

Understanding the Data

- *Source:* Ultimate Spotify Tracks DB (Kaggle Dataset)
    - Total Tracks: 232, 725

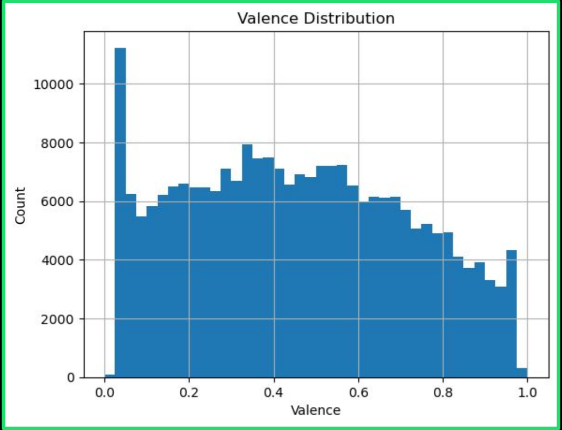
- *Key Columns:*
    - **Track Information**: track_name, artist_name, genre
    - **Audio Features**: valence, danceability, energy, acousticness, instrumentalness, speechiness, loudness, tempo



Notebook#1: Understanding the Data (EDA.ipynb)

- *Purpose:* initial diagnosis of raw CSV — check nulls, types, and distributions

- *Key findings that motivated cleaning:*
  - Several columns had missing numeric values (filled later with column mean).
  - Column names inconsistent (spaces & capitals) → normalized to snake_case.
  - Duplicates present (same track across rows) → removed by track_name + artist_name.

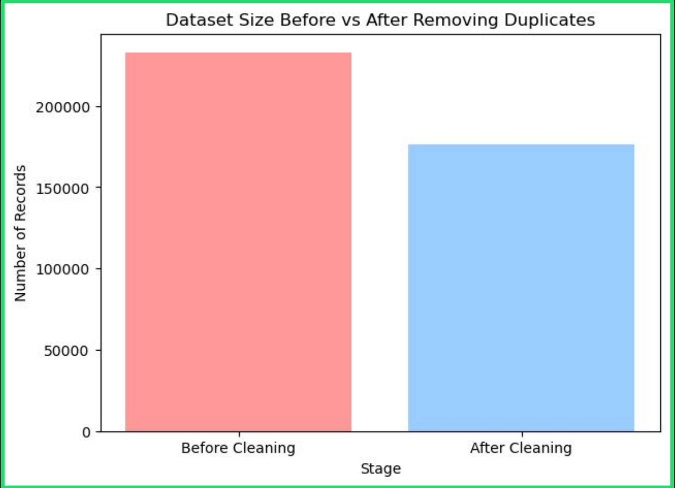



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   genre             232725 non-null  object
 1   artist_name       232725 non-null  object
 2   track_name        232725 non-null  object
 3   track_id          232725 non-null  object
 4   popularity        232725 non-null  int64
 5   acousticness      232725 non-null  float64
 6   danceability      232725 non-null  float64
 7   duration_ms       232725 non-null  int64
 8   energy            232725 non-null  float64
 9   instrumentalness  232725 non-null  float64
 10  key               232725 non-null  object
 11  liveness          232725 non-null  float64
 12  loudness          232725 non-null  float64
 13  mode              232725 non-null  object
 14  speechiness       232725 non-null  float64
 15  tempo             232725 non-null  float64
 16  time_signature    232725 non-null  object
 17  valence           232725 non-null  float64
dtypes: float64(9), int64(2), object(7)
memory usage: 32.0+ MB
```

Dataset Cleaning Process

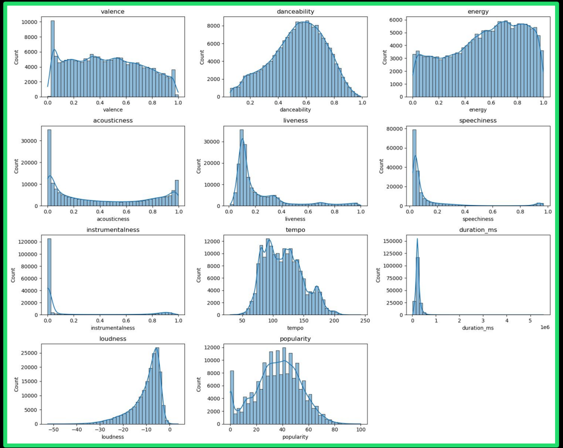
- *Normalize column names:*
  - lowercase + underscores (clean_column_names())
- *Remove duplicates:*
  - track_name, artist_name (remove_duplicates())
- *Handle missing values: (handle_missing())*
  - numeric → fill with column mean
  - categorical → fill with "Unknown"
- *Save cleaned file:*
  - data/cleanDataset.csv (save_clean_data())
- *Scripts & notebooks:*
  - src/cleaning/cleandata.py
  - notebooks/cleanDataEDA.ipynb

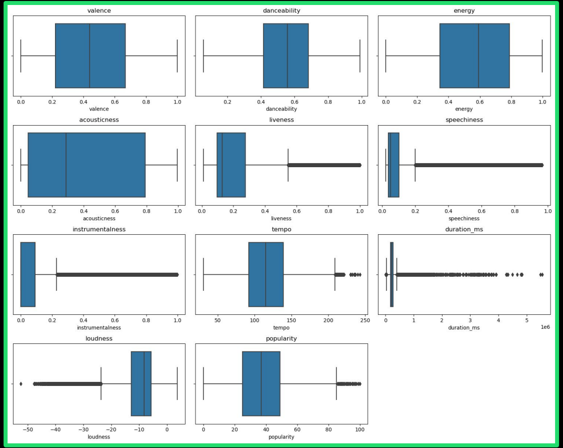
```
def clean_spotify_dataset():
    df = load_raw_data()
    df = clean_column_names(df)
    df = remove_duplicates(df)
    df = handle_missing(df)
    save_clean_data(df)
```

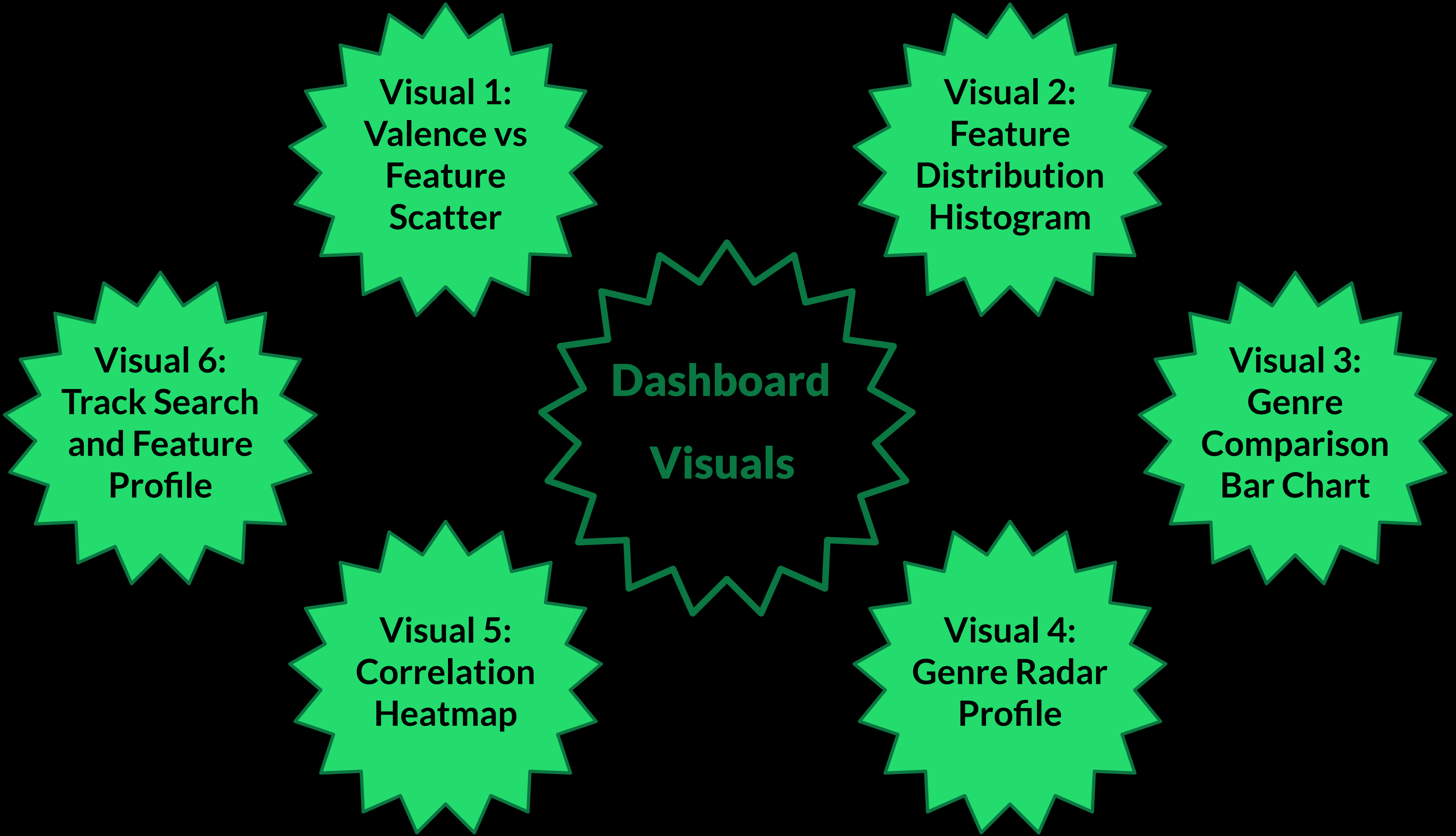


Notebook #2: Post-Cleaning (cleanDataEDA.ipynb)

- *Purpose:* validate cleaning and compute summary stats used by dashboard
- *Example outputs:*
  - **valence** mean = **0.451642**, std = **0.267853**
  - **loudness** observed range ≈ **[-52.457, 3.744]** → we scale loudness for radar charts.
  - **duration_ms** max = **5552917** → used to normalize track radars
- *Actionable decisions from EDA:*
  - **Keep features:** danceability, energy, acousticness, instrumentalness, speechiness, loudness, tempo, duration_ms, popularity.
  - Use sampling for scatter (3,000 points) to keep interactions responsive.

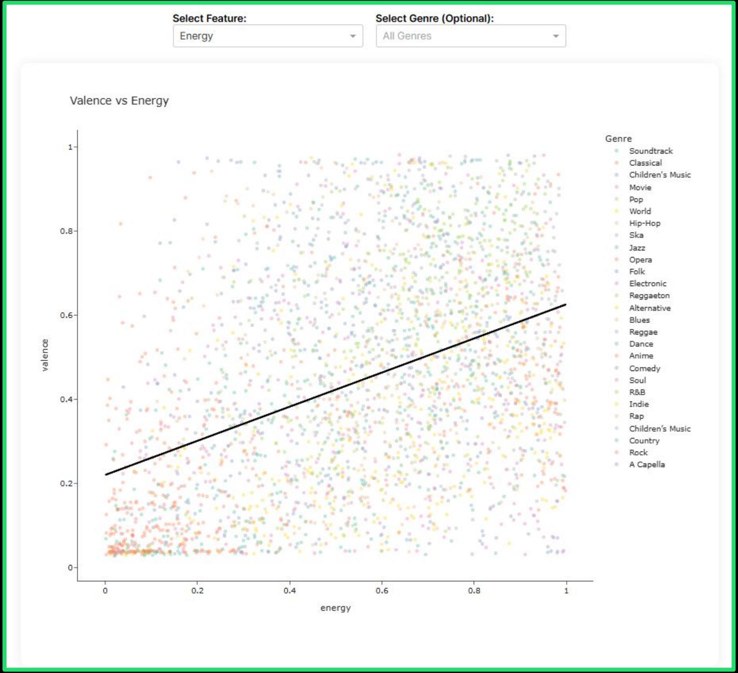





Dashboard Visuals
Visual 1: Valence vs Feature Scatter
Visual 2: Feature Distribution Histogram
Visual 3: Genre Comparison Bar Chart
Visual 4: Genre Radar Profile
Visual 5: Correlation Heatmap
Visual 6: Track Search and Feature Profile

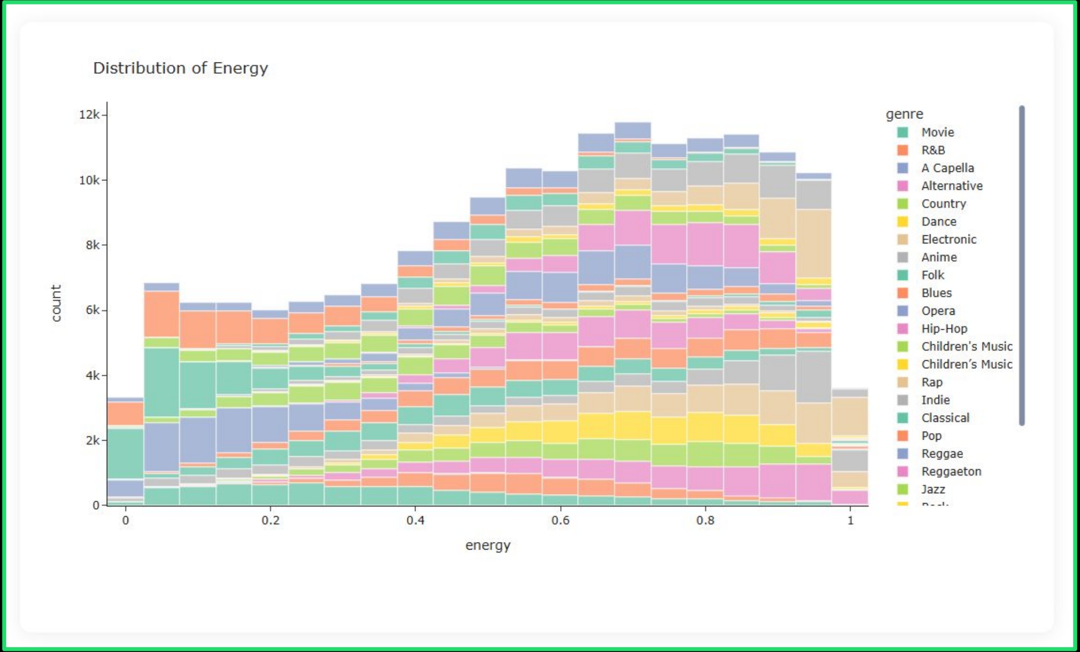
Visual #1: Valence vs Feature Scatter

- *What it shows:*
  - **Scatter:** x = selected feature vs y = valence
  - **Optional:** color by genre (when genre filter is unset)
  - Trendline (OLS) drawn across the sample
- *Controls:*
  - Feature-dropdown (danceability, energy, etc.)
  - Genre-dropdown (optional)
- *Important code notes:*
  - **Sample used:** sample = d.sample(min(3000, len(d)), random_state=42) to keep interactive rendering fast
  - **Trendline:** OLS trend added and styled as a separate trace (black, width 3)



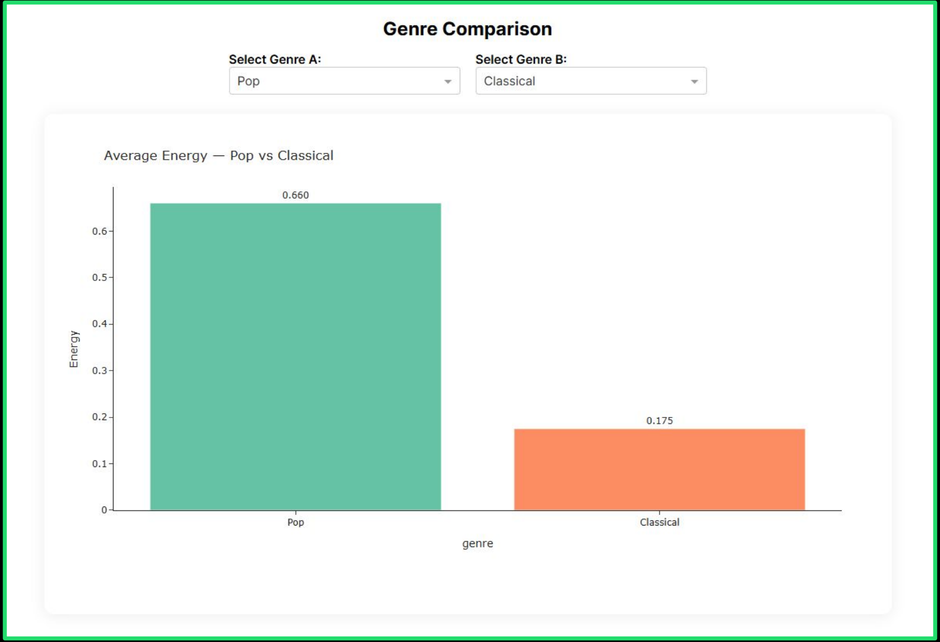
Visual #2: Feature Distribution Histogram

- *What it shows:*
  - Distribution of chosen feature across dataset or a filtered genre
- *Controls:*
  - Feature-dropdown (danceability, energy, etc.)
  - Genre-dropdown (optional)
- *Important code notes:*
  - Uses px.histogram(..., nbins=40).
    - If genre is set, histogram uses that single genre; otherwise colored by genre.
  - Helps identify skew, multi-modality, and outliers.



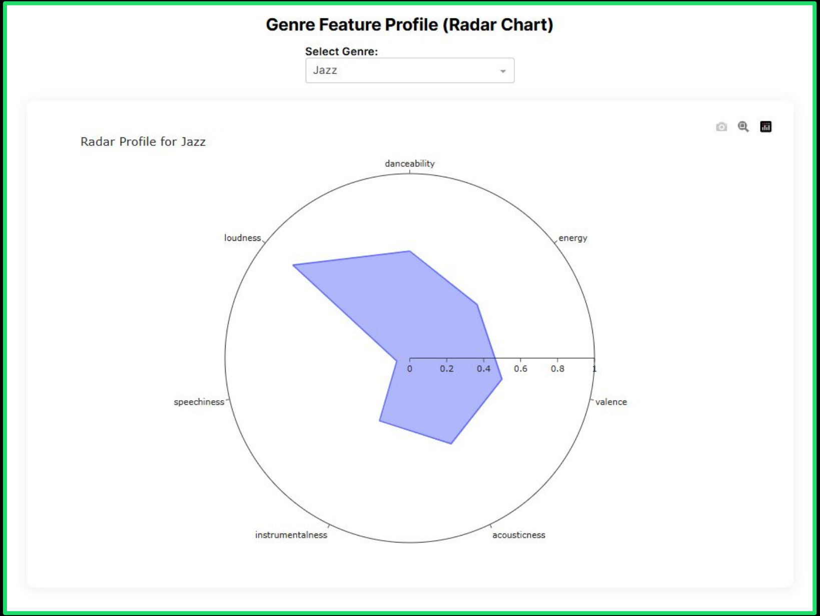
Visual #3: Genre Comparison Bar Chart

- *What it shows:*
  - Average value of selected feature for two chosen genres
- *Controls:*
  - GenreA-dropdown
  - GenreB-dropdown
- *Important code notes:*
  - Compares mean of selected feature for genreA vs genreB.
  - **Example:** "Pop mean energy = 0.660 vs Classical = 0.175 → Pop songs are on average more energetic."



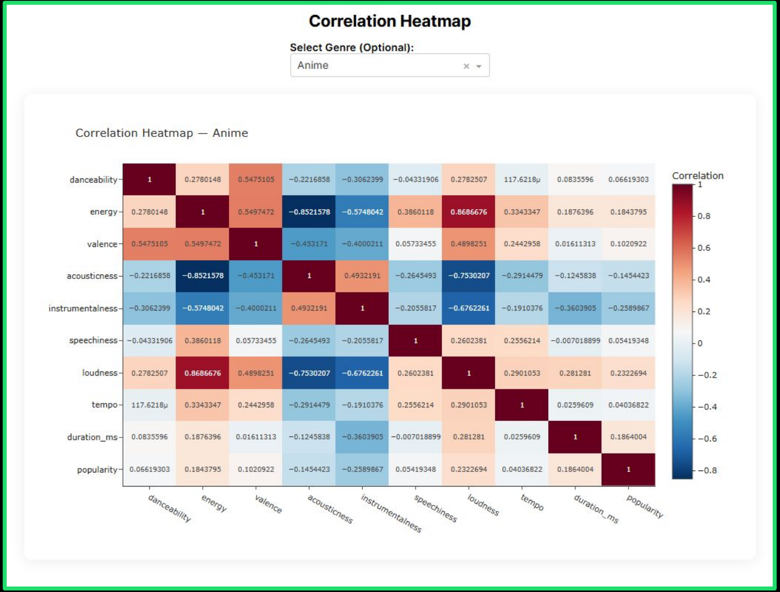
Visual #4: Genre Radar Profile

- *What it shows:*
  - Mean profile across multiple features for selected genre
- *Radar features shown:*
  - Danceability, energy, valence, acousticness, instrumentalness, speechiness, loudness
- *Important code notes:*
  - **Loudness scaled:** (mean_loudness + 60) / 60 → maps [-60, 0] to [0, 1] so radar maintains consistent scaling
  - Creates a high-level genre fingerprint



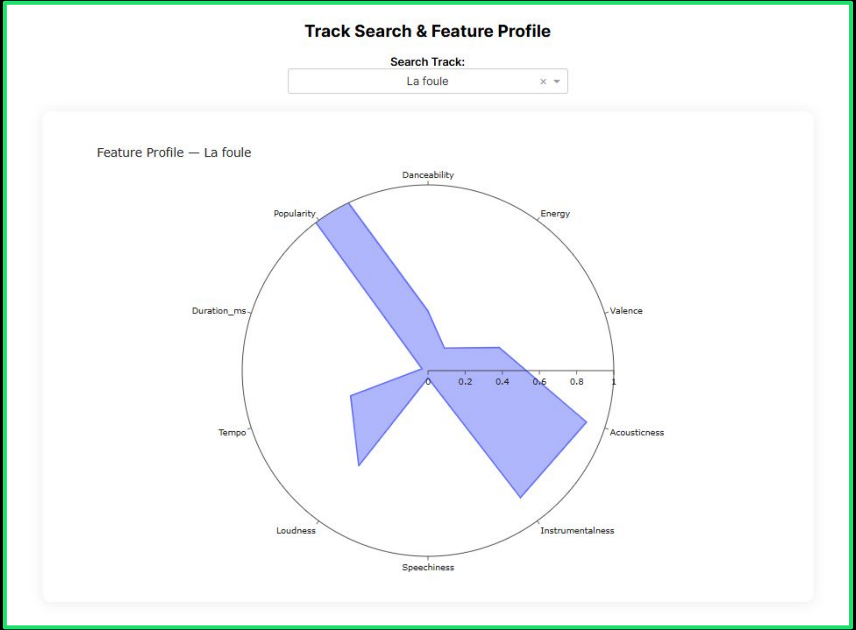
Visual #5: Correlation Heatmap

- *What it shows:*
  - Pearson correlations between key features (danceability, energy, valence, loudness, tempo, duration_ms, popularity)
- *Features:*
  - Danceability, energy, valence, acousticness, instrumentalness, speechiness, loudness, tempo, duration_ms, popularity
- *Controls:*
  - Heatmap-genre-dropdown (choose a single genre or all)
- *Important code notes:*
  - Corr = d[features].corr() and px.imshow(...) with RdBu_r scale
  - Identify which features co-vary with valence and and whether relationships are genre-specific



Visual #6: Track Search and Feature Profile

- *What it shows:*
  - Normalized audio features for a single track (radar)
- *Controls:*
  - Track-dropdown — select a track name
- *Normalization:*
  - Loudness_norm = (loudness + 60) / 60
  - Tempo_norm = tempo / 220 (approx max tempo)
  - Duration_norm = duration_ms / df["duration_ms"].max() (normalized by dataset max
- *Purpose:*
  - Compare track-level profile to genre mean radar



Demonstration!

Results & Insights

Global Audio Relationships

What drives "happy-sounding" music (valence)?

- Danceability → strongest positive effect:
  (slope = 0.83, r = 0.59)
- Energy → moderate effect:
  (slope = 0.44, r = 0.45)

Genre Fingerprints

Each genre has a unique feature profile

- Pop: high danceability & energy
- Jazz: highest acousticness & instrumentalness
- Hip-Hop: highest speechiness, very high danceability

Popularity Patterns

Popularity is not driven by audio features

- Pop: tempo (r = –0.04), loudness (r = –0.01)
- Classical: tempo (r = –0.00), loudness (r = 0.06)

Overall: Clear emotional drivers, strong genre identities, and weak feature-based popularity predictions

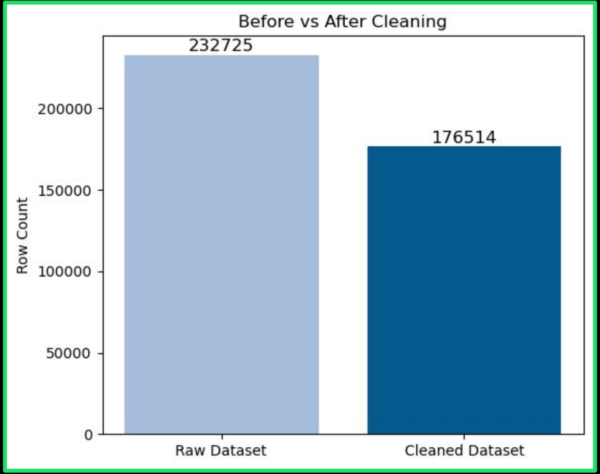
Challenges & Solutions

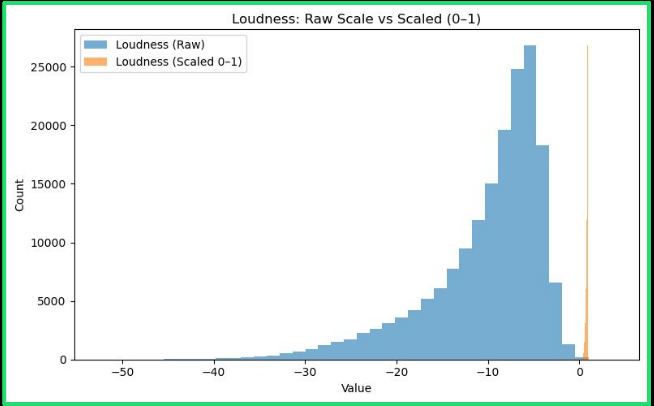
- *Challenge #1: Large dataset caused slow plotting*
  - Fix: sample scatter plots (3k rows) and precompute summaries (means) for bar/radar charts

- *Challenge #2: Mixed column formats and missing values*
  - Fix: normalized column names and filled missing numerics with column mean; categories → "Unknown"

- *Challenge #3: Loudness scales differ from other features*
  - Fix: rescale loudness from decibels to to [0,1] for radar plots

- *Challenge #4: Dashboard Integration*
  - Fix: Clean file paths and auto-populated dropdowns





Conclusion & Future Work

- *Conclusion:*
  - We built an interactive dashboard to explore valence across audio features and genres using a cleaned 232k-track dataset
  - **Key result:** valence relates to energy/danceability globally, but genre-specific patterns are strong and informative

- *Future work:*
  - Add PCA / UMAP embedding view to show high-dimensional clusters (project_proposal.md)
  - Add median/IQR metrics to genre comparisons and server-side caching for full-dataset interactive plots
  - Integrate Spotify preview snippets or links for details-on-demand

References

- *Kaggle:*
  - Ultimate Spotify Tracks DB: https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db
- *Libraries:*
  - Pandas, numpy, plotly, dash, scikit-learn
- *Repo:*
  - https://github.com/saanavig/Spotify-Visualization
- *Notebooks:*
  - EDA.ipynb
  - cleanDataEDA.ipynb

Thanks For Listening

Any Questions?!