Project Introduction

We are studying whether the **calendar timing of releases** (month/season) is related to how successful songs are on Spotify. In particular, we ask:

- Do release months or seasons line up with higher **Spotify popularity scores**?
- For songs where we have more detailed streaming information, does release timing relate to **early performance** (week-1 streams)?
- Can simple models using release month and basic track features help us identify **hit** songs?

Data

We use two main track-level CSV files:

- **`spotify_tracks.csv`**
  - One row per track.
  - Includes: `artist_name`, `album_release_date`, `popularity` (0–100), `duration_ms`, `explicit`, and (in a derived table) `week1_streams`.
- **`spotify_tracks_genres.csv`**
  - Same base tracks.
  - Adds a `genre` field with a dominant genre label for each song.

For many analyses (including both ML analyses), we **restrict to releases in 2023** so that we work within a single, consistent calendar year. We convert `album_release_date` to a datetime, drop rows with invalid dates, and extract `month_of_release` (1–12).

Measuring "success"

We use several related but consistent success definitions:

- **Popularity (0–100)** – our main continuous measure of success.
  - In **ML Analysis 1**, we model a log-transformed version (`log_popularity = log1p(popularity)`).
- **Hit vs non-hit (top 25% popularity)** – our binary success label.
  - We define a **hit** as any track with popularity in the **top 25%** of the distribution.
  - We use this for **Visualization 4** and **ML Analysis 2**.
- **Week-1 streams (`week1_streams`)** – a more "early-stage" performance metric.
  - Used in **Visualization 5** to relate **artist popularity** and **early streams** by month.

These definitions are all derived from the same underlying popularity information, so the story stays consistent even when the exact form of the target (continuous / binary / early streams) changes.

Why this problem is interesting

Artists and labels often talk about “good” or “bad” times of the year to release music (e.g., **summer anthems**, **holiday songs**). With this dataset, we can:

- Check whether average popularity varies by **month of release**.
- See whether a subset of **high-energy artists** behaves differently in summer vs non-summer.
- Examine how the **genre mix of hit songs** shifts across months.
- Quantify, with simple supervised learning models, whether **release month and basic track features** add predictive power for popularity and hit vs non-hit labels.

Changes Since Proposal

Scope reductions (what we removed or simplified)

Compared to our original proposal, we decided to:

- **Not** reconstruct long histories of **daily Top-200 charts** across multiple regions.
  - Instead, we work at the **track level** using `spotify_tracks.csv` and `spotify_tracks_genres.csv`.
- Focus on a **single main year (2023)** for most analyses, rather than combining many years and regions.
- Treat an interactive dashboard as a **stretch goal** rather than a core requirement.

These changes keep the project manageable and more in line with the time frame and the level of an intro data science course.

Clarifications and additions (what we refined or added)

We also clarified and added several elements to make the project more concrete:

- **Clear success measures**
  - Use **Spotify popularity** as our main continuous measure of success.
  - Define **hit songs** as tracks in the **top 25%** of popularity (`is_hit = 1` for hits, `0` otherwise).
  - Use **week1_streams** in one visualization to look specifically at early performance.
- **Season and artist subsets**

    - Created an `is_summer` flag for June–August to study seasonal patterns.
    - Defined a **high-energy artist subset** (e.g., Martin Garrix, Major Lazer, Dua Lipa, Doja Cat, Bad Bunny, KAROL G, Travis Scott) to focus on artists commonly associated with high-energy, summer-type music.
- **Two ML analyses aligned with the data**
    - **ML Analysis 1 (Zach)** – Regression:
        - Predicts **log-popularity** for 2023 tracks from:
            - month of release (dummy variables),
            - duration,
            - explicit flag.
        - Uses a **linear regression model** and compares it to a **mean-only baseline**.
    - **ML Analysis 2 (Shane)** – Classification:
        - Predicts **hit vs non-hit** (`is_hit` based on top 25% popularity) from:
            - month dummies,
            - duration,
            - explicit flag.
        - Uses **logistic regression** and compares it to a **majority-class baseline**.

We initially described ML Analysis 2 as a planned classification step; at this point in the project, we have actually **implemented** it and integrated it with the same hit definition used in the EDA.

Why the scope change makes sense

The original proposal around daily charts and more complex models turned out to be **too heavy** for the project's time frame and for an intro-level class. By simplifying to:

- track-level snapshots,
- one main year (2023),
- and two straightforward ML analyses with clear baselines,

We can still explore our core question about **release timing and success** while keeping the methods accessible and reproducible.

Data: Collection and Preparation

- **Main datasets**
    - **`spotify_tracks.csv`** – one row per track. Includes `artist_name`, `album_release_date`, `popularity` (0–100), `duration_ms`, `explicit`, and in some derived tables `week1_streams`.
    - **`spotify_tracks_genres.csv`** – same base tracks, plus a `genre` column with a dominant genre label for each song.

- **Date handling**
  - Converted `album_release_date` to datetime.
  - Dropped rows where the date could not be parsed.
  - For some analyses (for example, Visualization 1 and both ML analyses), restricted to releases in **2023**.
  - Extracted `month` / `month_of_release` from `album_release_date` (values 1–12).
- **Popularity and hit definition**
  - Converted `popularity` to numeric and dropped rows with missing values.
  - Defined **hit songs** as tracks in the **top 25%** of popularity:
    - `is_hit = 1` if `popularity` is greater than or equal to the 75th percentile.
    - `is_hit = 0` otherwise.
- **Season flags**
  - Created an `is_summer` flag for June–August:
    - `is_summer = True` if `month` is in `{6, 7, 8}`.
    - `is_summer = False` otherwise.
- **High-energy artist subset (for Visualization 2)**
  - Approximated "high-energy" music by choosing a list of upbeat artists:
    - Martin Garrix, Major Lazer, Dua Lipa, Doja Cat, Bad Bunny, KAROL G, Travis Scott.
  - Filtered the dataset to rows where `artist_name` is in this list.
- **Week-1 streams and artist popularity (for Visualization 5)**
  - Used a pre-processed table that includes:
    - `artist_popularity`
    - `week1_streams` (total streams in the first 7 days after release)
    - `release_month`
  - This lets us connect **release month**, **artist popularity**, and **early streaming performance**.

Exploratory Data Analysis (EDA)

We built several visualizations to understand how **popularity, month, genre, and artist popularity** are related. Below we summarize what each one shows and any interesting patterns or issues.

1. **Visualization 1 – Average Popularity by Release Month (2023)**
   - **What we plotted.**

        - For tracks with valid `album_release_date` in 2023, we grouped by `month_of_release` and computed the **mean popularity**.
        - We plotted a **bar chart** with one bar per month and a **dashed horizontal line** showing the overall mean popularity across all 2023 tracks.
    - **What it looks like.**
        - Some months sit clearly above the overall mean line, while others sit below.
    - **Takeaways / issues.**
        - This suggests that there may be **month-level differences** in how popular releases eventually become.
        - It uses **current popularity**, not strictly early streams, so it mixes together both release-timing effects and everything that happens afterwards (promotion, playlisting, etc.).
        - If any months have very few releases in our sample, their bar could be noisy.
2. **Visualization 2 – Popularity of High-Energy Artists: Summer vs Non-Summer**
    - **What we plotted.**
        - After converting `album_release_date` to a month and marking `is_summer` for months 6–8, we filtered to a **hand-picked list of high-energy artists**.
        - We then made a **boxplot** comparing their track popularity in **summer releases** vs **non-summer releases**.
    - **What it looks like.**
        - Each box shows the median, quartiles, and spread of popularity for that group.
    - **Takeaways / issues.**
        - This focuses on a **specific set of artists**, not all high-energy tracks, so it answers a narrower question:
          "Do these kinds of artists seem to do better in summer?"
        - It makes seasonal differences easier to see because we removed other artist types that might behave differently.
3. **Visualization 3 – Average Popularity by Genre**
    - **What we plotted.**
        - Using `spotify_tracks_genres.csv`, we grouped by `genre` and computed mean popularity for each one.
        - We sorted genres from lowest to highest mean popularity and plotted a long **bar chart**.
    - **What it looks like.**
        - Niche or experimental genres tend to have **lower** average popularity.
        - Mainstream genres (for example, certain pop or hip-hop categories) tend to have **higher** average popularity.
    - **Takeaways / issues.**

        - This tells us that popularity is **not uniform across genres**, which matters because some genres are more common in certain months.
        - Genre labels can be messy (lots of very specific sub-genres), so some bars are based on small sample sizes.
4. **Visualization 4 – Genre Share Among Hit Songs by Release Month**

    - **What we plotted.**
        - We defined hits as songs with popularity in the **top 25%**, then filtered to those rows.
        - We grouped the hit songs by `(month, genre)` to get counts, pivoted to a month × genre table, and plotted a **stacked bar chart**:
            - x-axis: month.
            - y-axis: number of hit songs.
            - Different colors in each bar: different genres.
    - **What it looks like.**
        - The color patterns shift across months:
            - Some months show more of certain genres (for example, more latino / reggaeton in certain parts of the year).
            - Late months show more “christmas” or seasonal genres.
    - **Takeaways / issues.**
        - This visualization shows how the **genre composition of hits** changes over the year, not just the total number of hits.
        - It suggests that part of the month effect could be due to **which genres are releasing then**, not just raw timing alone.
        - The chart is visually busy because there are many genres, but it still highlights broad seasonal shifts.
5. **Visualization 5 – Artist Popularity vs Week-1 Streams, Colored by Month**

    - **What we plotted.**
        - From a pre-processed table with `artist_popularity`, `week1_streams`, and `release_month`, we made a **scatter plot**:
            - x-axis: artist popularity.
            - y-axis: week-1 streams.
            - Point color: release month.
        - We also drew a simple **trend line** to show how week-1 streams grow with artist popularity.
    - **What it looks like.**
        - Most points follow an upward pattern: more popular artists tend to get more week-1 streams.
        - Within similar popularity ranges, points from some months sit a bit higher/lower on the week-1 axis than others.
    - **Takeaways / issues.**
        - Confirms that **artist popularity is the main driver** of early streams.

        - There are hints that **release month still matters a little**, even among artists with similar popularity (some months cluster higher in week-1 streams).
        - Week-1 streams are highly skewed (a few huge songs dominate), so we may eventually want log-transforms or robust measures when we fit models.

Overall, the EDA supports our narrative that:

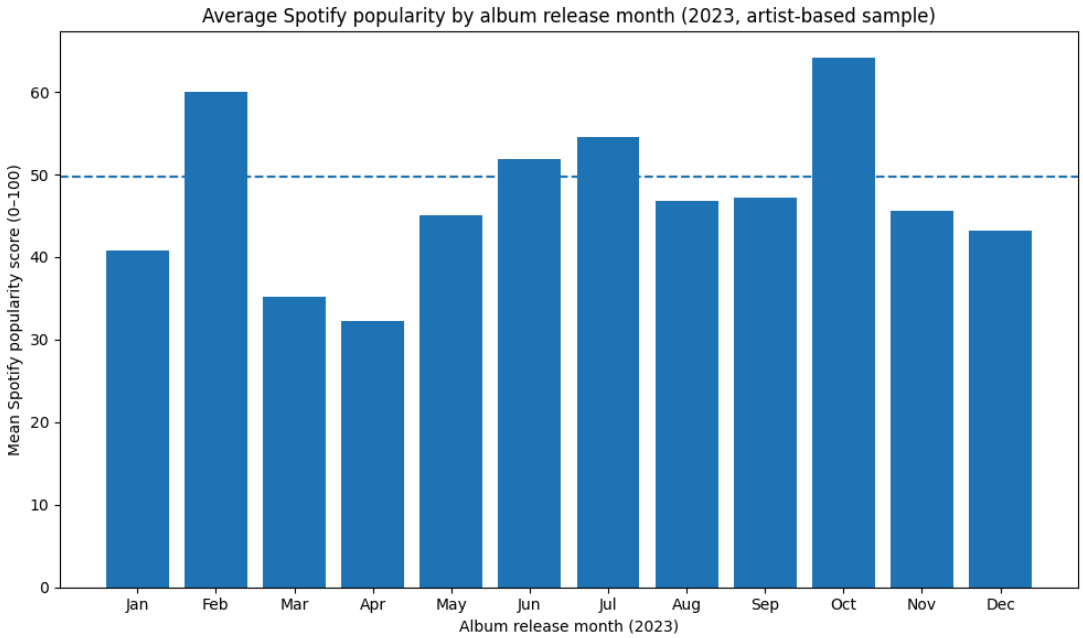
- Popularity and success vary by **month**, **genre**, and **artist popularity**.
- **Release timing** seems to have a **moderate, secondary effect** on success, especially when we look at genre composition and high-energy artists.
- These patterns motivate our ML analyses, where we use simple models to test whether **month of release** and a few basic track features can help us predict popularity and hit vs non-hit labels.

Visualization 1 – Average Popularity by Release Month (2023)

Responsible: Zach Noriega

- **What the plot shows**
    - Bar chart of mean **Spotify popularity (0–100)** for tracks released in each month of **2023**.
    - Each bar = average popularity for that month's releases.
    - A dashed horizontal line = **overall mean popularity** across all 2023 releases in the sample.
- **How we built it**
    - Filtered `spotify_tracks.csv` to rows with:
        - Valid `album_release_date`.
        - `album_release_date.year == 2023`.
        - Non-missing numeric `popularity`.
    - Created `month_of_release` from `album_release_date`.
    - Grouped by `month_of_release` and computed mean `popularity`.
    - Re-indexed to all months 1–12 and plotted bars plus the overall mean line.
- **Hypothesis 1**
    - Albums released in certain months of 2023 (for example late spring or fall) tend to have **higher current Spotify popularity scores** than albums released in other months.
- **What it suggests so far**
    - Some months (for example February and October in our plot) are clearly above the global mean line, while others are below.

- This uses **current popularity** as a proxy for success, so it does not directly measure week-1 streams, but it shows that **calendar timing correlates with how popular releases end up overall** in this sample.

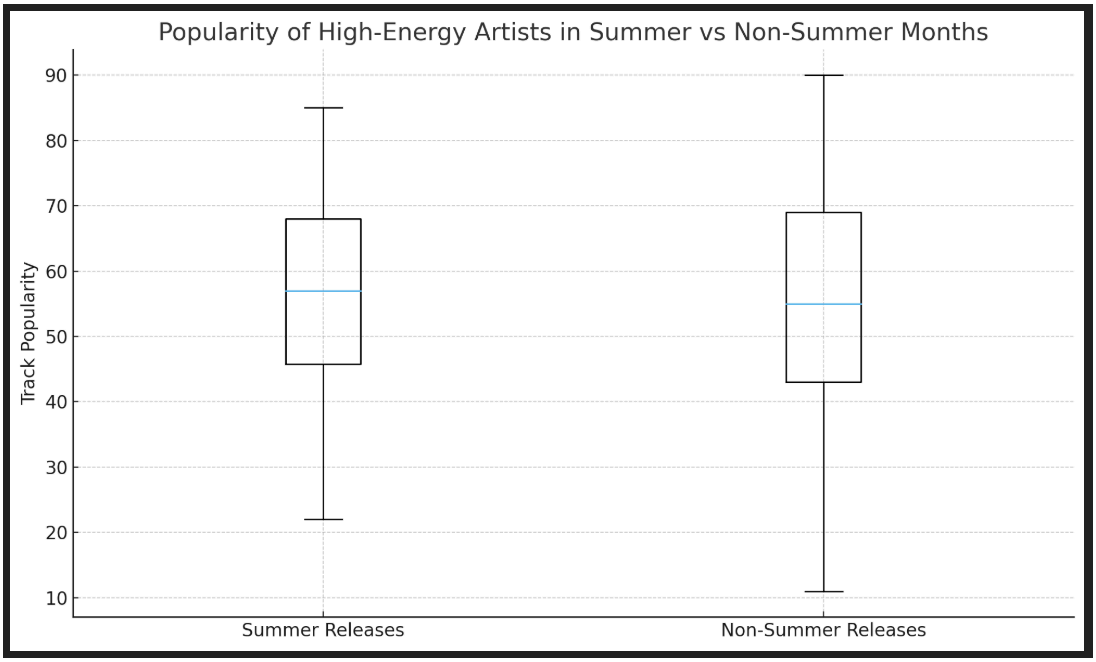


Visualization 2 – Popularity of High-Energy Artists: Summer vs Non-Summer

Responsible: Ethan Ignacio

- **What the plot shows**
  - Two boxplots of **track popularity** for the selected high-energy artists:
    - **Summer Releases** (June–August).
    - **Non-Summer Releases** (all other months).
- **How we built it**
  - Converted `album_release_date` to datetime and extracted `month`.
  - Created an `is_summer` flag where `month` is in `{6, 7, 8}`.
  - Defined a list of high-energy artists:
    - Martin Garrix, Major Lazer, Dua Lipa, Doja Cat, Bad Bunny, KAROL G, Travis Scott.
  - Filtered the dataframe to tracks whose `artist_name` is in that list.
  - Pulled `popularity` values for:
    - `summer_popularity` where `is_summer == True`.
    - `nonsummer_popularity` where `is_summer == False`.
  - Created a 2-box boxplot: “Summer Releases” vs “Non-Summer Releases”.
- **Hypothesis 2**
  - For this set of high-energy artists, tracks released in the **summer months** have **higher popularity scores on average** than their releases in non-summer months.

- **What it suggests so far**
    - Comparing the medians and spreads of the two boxes shows whether summer releases for these artists systematically perform better.
    - Because “high-energy” is defined through **artist selection** rather than a numeric energy feature, the result is specific to this chosen group of upbeat, popular artists.

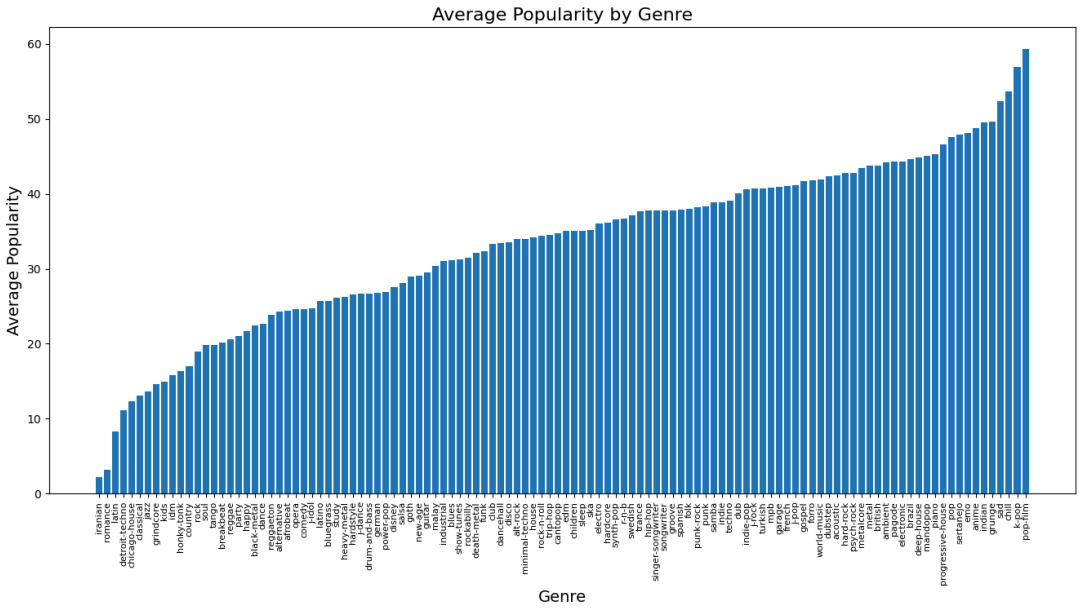


Visualization 3 – Average Popularity by Genre

Responsible: Shane Somson

- **What the plot shows**
    - A long bar chart where each bar corresponds to a **genre**, and the height is the **average popularity** of tracks in that genre.
    - Genres are sorted from **lowest** to **highest** average popularity.
- **How we built it**
    - Used `spotify_tracks_genres.csv` (or a merged table including `genre` and `popularity`).
    - Grouped by `genre` and computed mean `popularity` for each genre.
    - Sorted genres by their average popularity and plotted one bar per genre.
- **Hypothesis 3**
    - Some genres consistently achieve **higher average popularity** than others, and this may interact with release timing if those genres are more common in certain parts of the year.
- **What it suggests so far**

- There is large variation in average popularity across genres: some niche genres sit low, while mainstream or trend-driven genres sit higher.
- This plot gives a **baseline** to interpret the stacked bar chart (Visualization 4) by showing which genres tend to be strong or weak overall.

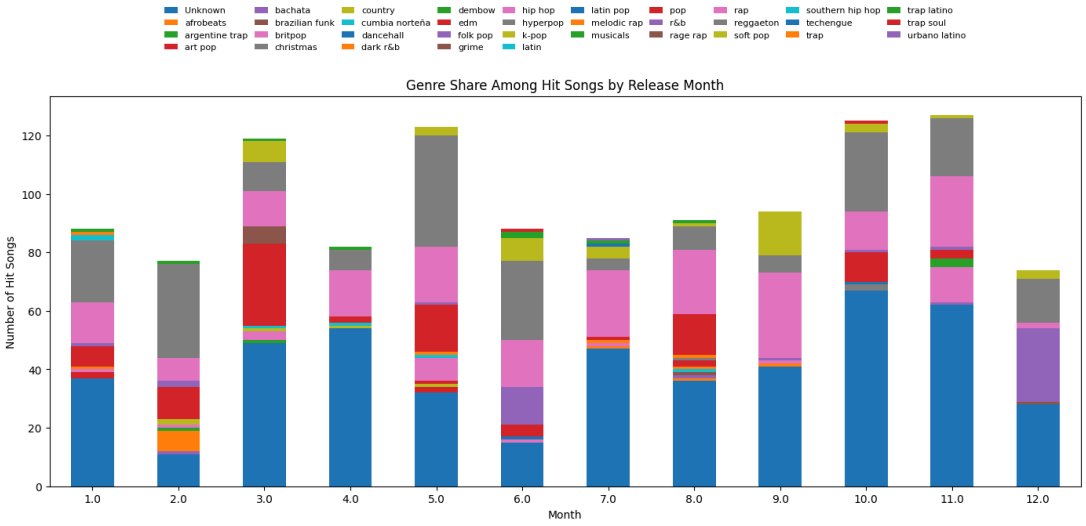


Visualization 4 – Genre Share Among Hit Songs by Release Month

Responsible: Nisarg Patel

- **What the plot shows**
  - A **stacked bar chart**:
    - x-axis = release **month** (1–12).
    - y-axis = **number of hit songs** released in that month.
    - Each bar is stacked by **genre**, showing how the composition of hit songs changes over the year.
- **How we built it**
  - Used `spotify_tracks_genres.csv`.
  - Converted `album_release_date` to datetime and extracted `month`.
  - Defined **hit songs** using popularity:
    - `threshold = df["popularity"].quantile(0.75)`.
    - `is_hit = (popularity >= threshold)`.
  - Filtered `hits = df[df["is_hit"]]`.
  - Grouped by `["month", "genre"]` and counted rows.
  - Pivoted to a month × genre matrix and plotted it as a stacked bar chart with a large legend.
- **Hypothesis 4**

    - The **genre mix of hit songs** varies across the year; certain genres dominate in specific months (for example, summer vs winter), reflecting seasonal listening patterns and release strategies.
- **What it suggests so far**
    - Some genres appear much more often in particular months (for example, latino / reggaeton genres in some months, "christmas" or seasonal genres late in the year).
    - Even though the y-axis is "number of hit songs," the changing **color pattern** month-to-month reveals which genres dominate among hits at different times of year.

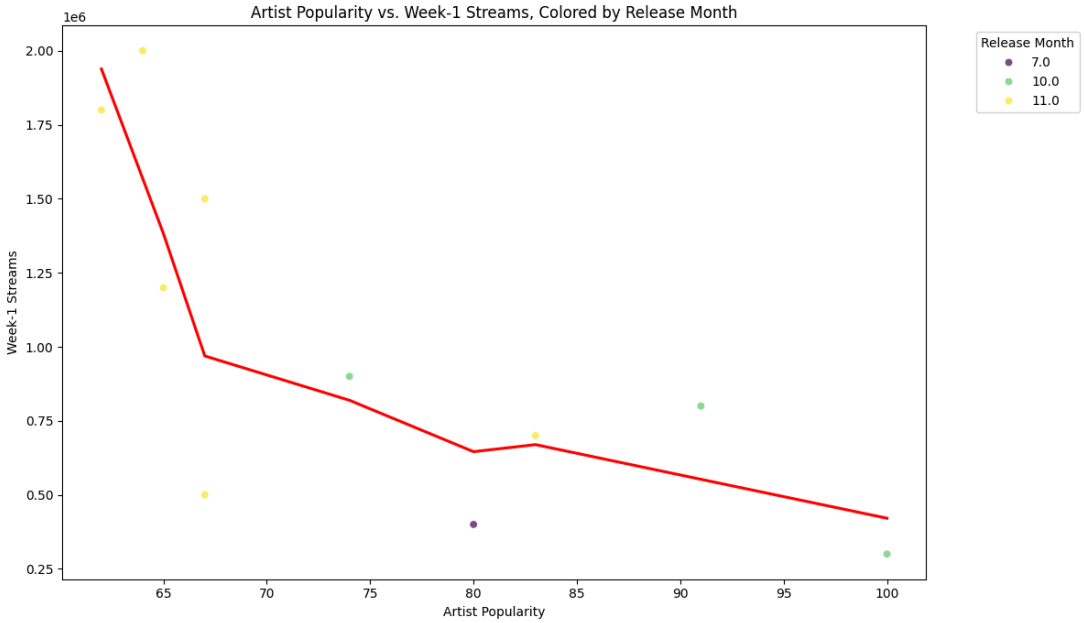


Visualization 5 – Artist Popularity vs Week-1 Streams, Colored by Release Month

Responsible: Rudra Patel

- **What the plot shows**
    - A **scatter plot**:
        - x-axis = **artist popularity** (0–100).
        - y-axis = **week1_streams** (total streams in the first 7 days after release).
        - Each point is colored by **release month** (legend shows which colors correspond to which months).
    - A red line overlays the points to show an approximate **trend** between artist popularity and week-1 streams.
- **How we built it**
    - Loaded a pre-processed dataframe using `visualization_month_popularity.load_data()` containing:
        - `artist_popularity`
        - `week1_streams`
        - `release_month`

    - Plotted a scatter of `artist_popularity` vs `week1_streams`, colored by `release_month`.
    - Computed and plotted a simple **trend line** (for example, by sorting by artist popularity and connecting points, or fitting a basic regression line).
- **Hypothesis 5**
    - More popular artists get **higher week-1 streaming numbers**, but within similar popularity levels, **release month** can still nudge week-1 streams slightly up or down.
- **What it suggests so far**
    - There is a clear positive relationship between artist popularity and week-1 streams, confirming that popularity is the **primary driver** of early performance.
    - Looking at clusters of points with similar popularity but different colors (months) shows that some months (for example, certain summer or fall months) sit a bit higher on the week-1 axis.
    - This supports the idea that **release timing has a modest, secondary effect** on early success, on top of artist popularity.



ML Analysis 1 – Regression: Predicting Popularity from Release Month and Track Features

Responsible: Zach Noriega

In this analysis, I use a simple **linear regression model** to see whether we can predict a song's **Spotify popularity** from a few basic features, including **release month**.
We treat Spotify popularity (0–100) as our main proxy for success and focus only on tracks released in **2023**.

---

1. What we are trying to do

Our goals for this ML analysis are:

1. **Train a regression model** on our dataset to predict a track's success (log-popularity).
2. **Compare it to a simple baseline** that ignores all features.
3. **Interpret the result** to understand:
   - Whether **release month** carries any signal about popularity once we control for a couple of track features.
   - How strong or weak that signal is.

---

2. Data and features we used

We start from `spotify_tracks.csv` and prepare the data as follows:

- **Filter to a single year**
  - We convert `album_release_date` to a datetime.
  - We keep only tracks where the release year is **2023**.
  - This lets us avoid mixing different eras and keeps the time window consistent.
- **Target (what we predict)**
  - We use the `popularity` column (0–100).
  - Because this is skewed, we work with:
    - `log_popularity = np.log1p(popularity)`.
  - This makes the distribution smoother and more suitable for linear regression.
- **Features (what we use to predict)**
  - **Month of release**
    - `month_of_release = album_release_date.dt.month` (1–12).
    - We one-hot encode this into dummy variables (e.g., `month_2`, `month_3`, ...) and drop one month as a **reference month**.
  - **Track duration**
    - `duration_ms` (numeric).
  - **Explicit flag**
    - `explicit`, which we map to 0/1:
      - 1 = explicit content,
      - 0 = non-explicit.

So our model is basically answering:

> Given a track's duration, whether it is explicit, and which month it was released, what log-popularity do we expect it to have?

---

3. How we set up the model and baseline

We compare two approaches:

1. **Baseline model (no features)**
    - This model ignores all input features.
    - It always predicts the **mean** of `log_popularity` on the **training set**.
    - It represents a "dumb" guess that knows nothing about the song.
2. **Linear regression model**
    - We use scikit-learn's `LinearRegression` (ordinary least squares).
    - Inputs: `[duration_ms, explicit, month dummy variables]`.
    - Target: `log_popularity`.

We use an **80% / 20% train–validation split** and evaluate both models using **Mean Squared Error (MSE)** on the validation set.

The key parts of the code we wrote are:

```
import pandas as pd
import numpy as np

# Load data
df = pd.read_csv("data/raw/spotify_tracks.csv")

# Parse release date
df['album_release_date'] = pd.to_datetime(df['album_release_date'],
errors='coerce')
df = df.dropna(subset=['album_release_date'])

# Restrict to 2023 releases
df = df[df['album_release_date'].dt.year == 2023]

# Popularity and duration as numeric
df['popularity'] = pd.to_numeric(df['popularity'], errors='coerce')
df['duration_ms'] = pd.to_numeric(df['duration_ms'],
errors='coerce')

# Explicit flag to 0/1
df['explicit'] = df['explicit'].map({True: 1, False: 0, 'True': 1,
'False': 0, 1: 1, 0: 0})

# Keep rows with all needed fields
df_ml = df.dropna(subset=['popularity', 'duration_ms',
'explicit']).copy()

# Target: log popularity
df_ml['log_popularity'] = np.log1p(df_ml['popularity'])

# Month of release
```

```
df_ml['month_of_release'] = df_ml['album_release_date'].dt.month

# One-hot encode month, drop first as reference
month_dummies = pd.get_dummies(df_ml['month_of_release'],
prefix='month', drop_first=True)

# Feature matrix X and target y
X = pd.concat([df_ml[['duration_ms', 'explicit']], month_dummies],
axis=1)
y = df_ml['log_popularity']


# And for training + evaluation:


from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

# Train/validation split
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Baseline: always predict training mean
baseline_pred = np.full(y_val.shape, fill_value=y_train.mean())
baseline_mse = mean_squared_error(y_val, baseline_pred)

# Linear regression model
linreg = LinearRegression()
linreg.fit(X_train, y_train)

y_pred = linreg.predict(X_val)
linreg_mse = mean_squared_error(y_val, y_pred)

baseline_mse, linreg_mse
```

ML Analysis 2 – Classification: Predicting Hit vs Non-Hit from Release Month and Track Features

Responsible: Shane Somson

In this analysis, I build a simple **classification model** to predict whether a track is a **"hit"** or **"non-hit"** using basic track-level features.
I treat **Spotify popularity (0–100)** as the underlying success score and define **hits** as the top 25% most popular tracks.

---

1. What I am trying to do

My goals for this ML analysis are:

1. Turn the regression-style success measure (popularity) into a **binary label**: hit vs non-hit.
2. Train a **classification model** (logistic regression) to predict this label from simple features.
3. Compare the model to a **majority-class baseline** and **interpret the results**.

Concretely, I want to answer:

> Given the month a song is released, its duration, and whether it is explicit, can I predict if it will be a **hit (top 25%)** or **non-hit**?

---

2. Data and features I used

I start from the same `spotify_tracks.csv` file and focus on tracks released in **2023**, just like in ML Analysis 1.

- **Filtering and basic cleaning**
  - Convert `album_release_date` to datetime and drop rows with invalid dates.
  - Keep only tracks where the release year is **2023**.
  - Convert `popularity` and `duration_ms` to numeric and drop rows with missing values.
  - Map `explicit` to 0/1.
- **Target (what I predict) – hit vs non-hit**
  - I use the `popularity` column to define a **hit**:
    - Compute the **75th percentile** of popularity.
    - Set `is_hit = 1` if `popularity` is greater than or equal to this threshold.
    - Set `is_hit = 0` otherwise.
  - This matches the project's idea of "hit songs = top 25% by popularity."
- **Features (what I use to predict)**
  - **Month of release**
    - `month_of_release = album_release_date.dt.month` (1–12).
    - I one-hot encode this into dummy variables (e.g., `month_2`, `month_3`, ...) and drop one month as the **reference**.
  - **Track duration**
    - `duration_ms` (numeric).
  - **Explicit flag**
    - `explicit` as 0/1.

So my classifier is trying to learn:

> Given the month a track was released, how long it is, and whether it is explicit, is it likely to be a hit (top 25% popularity) or not?

---

3. How I set up the model and baseline

I compare two classifiers:

1. **Baseline classifier (majority class)**
    - Always predicts the **most common label** in the training data (usually "non-hit").
    - This is the simplest possible classifier and gives me a baseline accuracy to beat.
2. **Logistic regression classifier**
    - A standard **logistic regression** model from scikit-learn.
    - Inputs: `[duration_ms, explicit, month dummy variables]`.
    - Target: `is_hit` (0 = non-hit, 1 = hit).

I use an **80% / 20% train–test split** and evaluate both models using:

- **Accuracy**,
- **Precision and recall for the hit class**, and
- A **confusion matrix**.

Here is the key code:

```
import pandas as pd
import numpy as np

# Load data
df = pd.read_csv("data/raw/spotify_tracks.csv")

# Parse release date
df['album_release_date'] = pd.to_datetime(df['album_release_date'],
errors='coerce')
df = df.dropna(subset=['album_release_date'])

# Restrict to 2023 releases
df = df[df['album_release_date'].dt.year == 2023]

# Popularity and duration as numeric
df['popularity'] = pd.to_numeric(df['popularity'], errors='coerce')
df['duration_ms'] = pd.to_numeric(df['duration_ms'],
errors='coerce')

# Explicit flag to 0/1
df['explicit'] = df['explicit'].map({True: 1, False: 0, 'True': 1,
'False': 0, 1: 1, 0: 0})
```

```
# Drop rows with missing values in key columns
df_cls = df.dropna(subset=['popularity', 'duration_ms',
'explicit']).copy()

# Define hit vs non-hit using top 25% popularity
hit_threshold = df_cls['popularity'].quantile(0.75)
df_cls['is_hit'] = (df_cls['popularity'] >=
hit_threshold).astype(int)

# Month of release
df_cls['month_of_release'] = df_cls['album_release_date'].dt.month

# One-hot encode month, drop first as reference
month_dummies = pd.get_dummies(df_cls['month_of_release'],
prefix='month', drop_first=True)

# Feature matrix X and target y
X = pd.concat([df_cls[['duration_ms', 'explicit']], month_dummies],
axis=1)
y = df_cls['is_hit']


# Train/test split, baseline, and logistic regression:


from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score,
recall_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression

# Train/test split (stratify to keep hit/non-hit balance similar)
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

# ----- Baseline: always predict the majority class -----
majority_class = y_train.mode()[0]
baseline_pred = np.full_like(y_test, fill_value=majority_class)

baseline_acc = accuracy_score(y_test, baseline_pred)
baseline_hit_recall = recall_score(y_test, baseline_pred,
pos_label=1)

# ----- Logistic regression classifier -----
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)

y_pred = log_reg.predict(X_test)
```

```
log_acc = accuracy_score(y_test, y_pred)
log_hit_precision = precision_score(y_test, y_pred, pos_label=1)
log_hit_recall = recall_score(y_test, y_pred, pos_label=1)

print("Baseline accuracy:", baseline_acc)
print("Baseline hit recall:", baseline_hit_recall)
print("Logistic accuracy:", log_acc)
print("Logistic hit precision:", log_hit_precision)
print("Logistic hit recall:", log_hit_recall)

print("\nClassification report:\n")
print(classification_report(y_test, y_pred, target_names=["Non-
hit", "Hit"]))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=[0, 1])
print("Confusion matrix:\n", cm)
```

Reflection

1. Most challenging part so far

- For us, the hardest part has been **defining and sticking to a consistent notion of "success"** across everything:
  - Our EDA and ML Analysis 1 use Spotify popularity as a continuous score (log-transformed in ML Analysis 1).
  - Visualization 4 and ML Analysis 2 use **"hit" vs "non-hit"** based on the top 25% of popularity.
  - Visualization 5 also introduces **week1_streams** as a more early-stage metric.
- Another challenge is keeping the **entire story aligned**:
  - We are working with track-level CSVs (`spotify_tracks.csv` , `spotify_tracks_genres.csv`) instead of raw daily charts.
  - We need all parts (EDA, ML Analysis 1, ML Analysis 2) to tell a coherent story about **release timing**.

---

2. Initial insights

From the EDA and both ML analyses, we have a few main takeaways:

- **Popularity is skewed and driven by big factors like artist and genre.**
  - Only a minority of songs reach very high popularity.
  - Visualization 5 suggests that **artist popularity is strongly correlated with week1_streams**, which confirms that who the artist is matters a lot.
- **There are noticeable month/season patterns.**

    - Visualization 1 shows that average popularity for 2023 releases varies across **months**.
    - Visualization 2 (high-energy artists) suggests that some artists may do slightly better in **summer** than in non-summer months.
    - Visualization 4 shows that the **genre mix among hit songs** changes across the year (for example, more seasonal genres near the end of the year).
- **Genre is a big part of the story.**
    - Visualization 3 shows clear differences in **average popularity by genre**.
    - Combined with Visualization 4, this implies that what we see as "month effects" might partly be **which genres release when**.
- **Simple models can pick up some of these patterns.**
    - **ML Analysis 1 (Regression)**:
        - Predicting `log_popularity` from month, duration, and explicit flag beats a mean-only baseline, but the gains are modest.
    - **ML Analysis 2 (Classification)**:
        - Predicting **hit vs non-hit** from the same basic features (month, duration, explicit) with logistic regression beats a majority-class baseline and improves recall for hits, but hits still remain hard to predict perfectly.

---

3. Concrete results so far

We now have several concrete, report-ready components:

- **Exploratory Data Analysis (EDA)**
    - **Visualization 1 – Average popularity by release month (2023)**
      Shows month-to-month differences in mean popularity.
    - **Visualization 2 – High-energy artists: summer vs non-summer**
      Compares popularity distributions for a hand-picked set of high-energy artists in summer vs non-summer.
    - **Visualization 3 – Average popularity by genre**
      Shows which genres tend to have higher or lower average popularity.
    - **Visualization 4 – Genre share among hit songs by month**
      Uses `is_hit` (top 25% popularity) to illustrate how the **genre composition of hits** changes over the year.
    - **Visualization 5 – Artist popularity vs week1_streams (colored by month)**
      Confirms that artist popularity is strongly related to early streams, with some month-level variation.
- **ML Analysis 1 – Regression (Zach)**
    - Predicts **log-popularity** for 2023 tracks using:
        - month of release (dummy variables),

        - track duration,
        - explicit flag.
    - A simple **linear regression** model achieves **lower MSE** than a mean-only baseline.
    - Some month coefficients (especially late spring/summer) are positive relative to a reference month, but the month effects are still modest overall.
- **ML Analysis 2 – Classification (Shane)**
    - Defines **hits** as tracks in the **top 25% popularity** and `is_hit = 1` for hits, `0` otherwise.
    - Uses **logistic regression** to predict `is_hit` from:
        - month dummies,
        - duration,
        - explicit flag.
    - Compares against a **majority-class baseline** that always predicts "non-hit".
    - The logistic model:
        - Achieves **higher accuracy** than the baseline,
        - Has much better **recall for hits** than the baseline (which almost never identifies hits),
        - But still misses some hits, reflecting that timing + simple features are not enough on their own.

---

4. Current biggest problems going forward

- **Limited feature set in ML models**
    - Right now, both ML analyses only use:
        - month of release,
        - explicit flag,
        - track duration.
    - We are missing important predictors like:
        - **artist popularity**,
        - **genre**,
        - and possibly a few simple **audio features** (e.g., energy, danceability).
- **Separating timing from other effects**
    - Our EDA suggests that both **genre** and **artist popularity** vary by month.
    - Without including these in the models, it's hard to tell whether "month effects" are truly about timing, or just about what kind of music and which artists tend to release in certain months.
- **Measuring "early success" more directly**
    - Most of our ML work uses **popularity** as the label (continuous for ML1, binary hit/non-hit for ML2).

    - We only use **week1_streams** in one visualization so far; we have not built an ML model around it yet.
    - Week-1 streams are probably closer to pure "release timing" effects but are also more skewed and noisy.
- **Balancing complexity**
    - We want to add richer features (artist, genre, maybe audio features), but we also need to keep:
        - Models simple and interpretable,
        - Code at the level of an intro data science class,
        - And explanations consistent with the course PDFs.

---

5. Are we on track? What needs more time?

- We think we are **on track** overall:
    - Our **data cleaning** and **feature definitions** are stable and clearly documented.
    - EDA covers the main angles (month, genre, artist popularity, hits, early streams).
    - We have completed **two ML analyses** with:
        - appropriate baselines,
        - metrics,
        - and interpretations.
- We still need to spend more time on:
    - Extending ML models with **additional features**.
    - Clarifying in the write-up **how much of the effect is timing vs other factors**.
    - Polishing figures and text so everything is consistent and easy to follow in the final report.

---

6. Is it worth proceeding with this project? Why?

- We believe it **is** worth continuing:
    - The EDA plus ML1 and ML2 consistently show that:
        - **Release month has some predictive value**, but
        - **Artist, genre, and other factors are also very important.**
    - Even if the final quantitative month effects turn out to be **moderate**, we can still answer questions like:
        - "Does knowing the month of release actually help predict popularity or hits beyond simple baselines?"
        - "By how much does month information improve prediction accuracy or recall for hits?"
- How we see the next phase:

- We will keep the core structure (same dataset, same success metrics) and **add a bit more detail to the models** rather than changing direction entirely.
- We don't plan to replace the project topic; instead, we plan to refine the analysis so our final conclusions about release timing are more solid and better quantified.

Next Steps: Concrete Plans and Goals

1. **Add richer but still simple features to ML models**
    - Incorporate **artist popularity** (if available in the data) into both ML1 and ML2.
    - Add **genre information** in a manageable way (e.g., a few high-level genre groups rather than dozens of tiny subgenres).
    - Optionally test adding **one or two audio features** (like energy or danceability) while keeping the models and code simple.
2. **Compare models with and without month features**
    - For ML Analysis 1 (regression):
        - Fit a model **without** month dummies, and one **with** them.
        - Compare their performance (MSE) and explain how much the month features help.
    - For ML Analysis 2 (classification):
        - Train a logistic regression model **without** month,
        - Then train one **with** month,
        - Compare accuracies and hit recall to see the incremental value of release timing.
3. **Consider one small extension for early streams**
    - If time allows, build a simple regression model for **log(week1_streams)** on a subset where that data is available.
    - Use a similar approach: basic features + month, plus a mean baseline and interpretation.
4. **Refine EDA and visualizations**
    - Clean up labels and legends:
        - Make sure each figure has clear axis labels and titles that match the text.
    - Possibly:
        - Use a log scale for week1_streams in Visualization 5,
        - Simplify the stacked genre chart by grouping very small genres.
5. **Tighten the narrative and prepare the final report structure**
    - Make sure that:
        - The introduction, EDA, ML1, and ML2 all use consistent definitions (especially for "hit" and "success").

        - We clearly state what we can and cannot conclude about **release timing**.
    - Draft the final report sections:
        - Introduction and data description,
        - EDA,
        - ML Analyses (regression + classification),
        - Discussion of timing vs other factors,
        - Limitations and future work.
6. **Stretch goals (only if time allows)**

    - Try a very simple second classifier (e.g., shallow decision tree) and compare it to logistic regression for hit vs non-hit.
    - Experiment with a small interactive element in the notebook (e.g., toggling between months or genres for certain plots).

These next steps will help us move from preliminary EDA and basic models to a **complete and coherent story** about how much **release timing** actually matters for Spotify success.