Classification of Danceability Report- Kavyasri Jadala

Contents

Data Collection and Upload to BigQuery

The following steps outline the process of collecting a dataset containing track features and uploading it to BigQuery:

Project Creation:

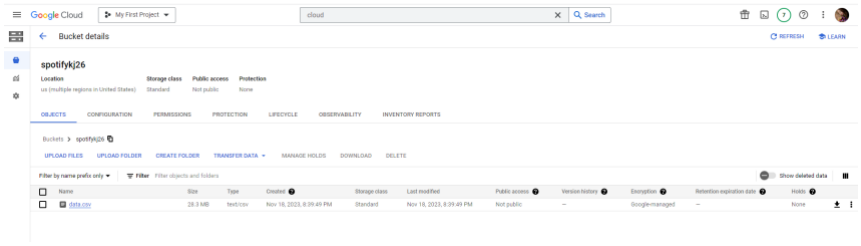
To facilitate the analysis, I initiated the project with the ID **"root-matrix-400415"** in Google Cloud Platform. This project served as the primary environment for all subsequent data handling, storage, and analysis tasks in BigQuery and Google Cloud Storage.

Dataset Acquisition:

Sourced a dataset comprising features essential for the classification task: 'tempo', 'energy', 'valence', 'loudness', 'speechiness', and 'acousticness' and other features. The dataset was in CSV format with 18000 songs, suitable for processing and analysis.

Cloud Storage Upload:

Uploaded the CSV file to a cloud storage solution. For this task, **Google Cloud Storage** was used as it integrates seamlessly with BigQuery. Ensured data integrity during the upload by verifying the file size and content post-upload against the original file. Created **a new bucket** in the Google Cloud Console, selecting the appropriate storage class and location to optimize for access speed and cost. The bucket's settings were configured to ensure data privacy and security.



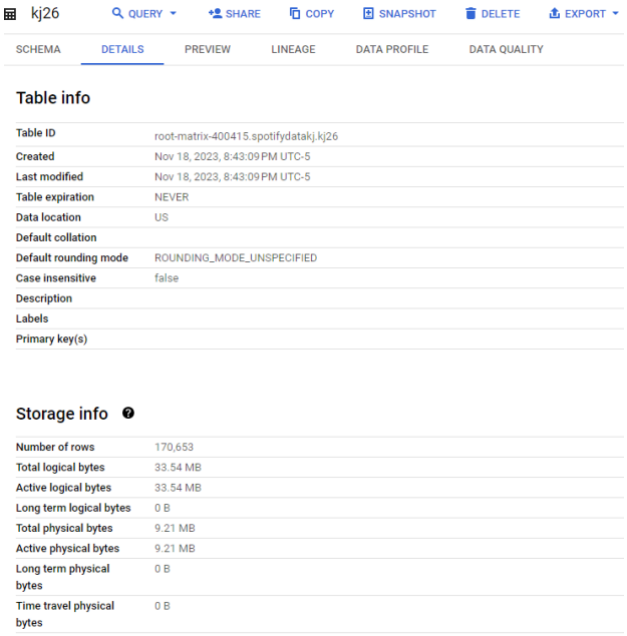
Uploading Data to the Bucket:

Uploaded the **CSV file** containing the track features to the designated Google Cloud Storage bucket.

Linking the Bucket with BigQuery:

Imported the data from the CSV file in Google Cloud Storage into the **BigQuery dataset and table.** This process involved specifying the data source (the CSV file in Cloud Storage), the destination table (the newly created table in BigQuery), and the schema.Ensured that the import process correctly mapped CSV columns to the BigQuery table columns and checked for any errors or warnings during the import**. Schema** is set to detect automatically and ensure jagged rows option is selected to

arrange the data.
Within the project "root-matrix-400415", the dataset "spotifydatakj" was created, and within this dataset, the specific table "kj26" was used to store and manage the track features data essential for our analysis.



Data Verification:

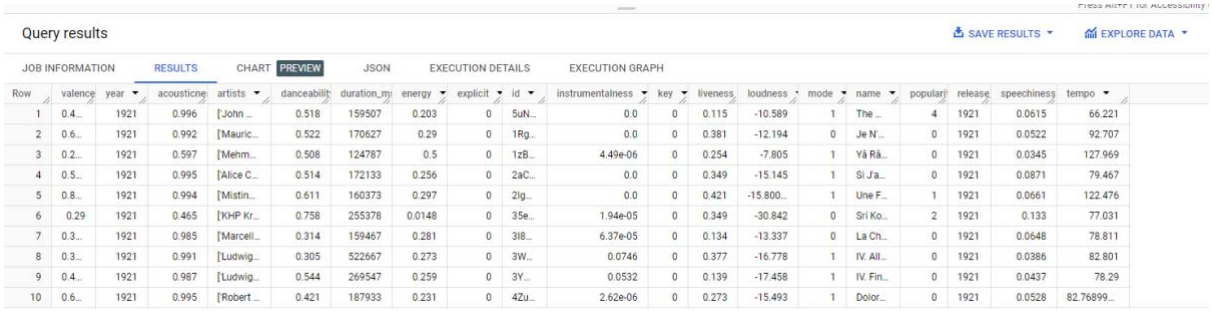
Check Data Integrity:

After the data was loaded into BigQuery, ran a few basic queries to verify the integrity and correctness of the data. This included checking row counts, viewing sample records, and ensuring that no data corruption occurred during the transfer.

```
-- Count the total number of rows in the table

SELECT COUNT(*) AS total_row_count
FROM `root-matrix-400415.spotifydatakj.kj26`;

-- Preview the first 10 records from the table
SELECT *
FROM `root-matrix-400415.spotifydatakj.kj26`
LIMIT 10;
SELECT COUNT(*) AS total_rows

FROM `root-matrix-400415.spotifydatakj.kj26`;
SELECT COUNT(*) AS total_columns
FROM `root-matrix-400415`.spotifydatakj.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'kj26';
```

Row	total_rows
1	170653

Row	total_columns
1	19



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	valence	year	acousticne	artists	danceabilit	duration_m	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popular	release	speechiness	tempo
1	0.4...	1921	0.996	['John ...	0.518	159507	0.203	0	5uN...	0.0	0	0.115	-10.589	1	The ...	4	1921	0.0615	66.221
2	0.6...	1921	0.992	['Mauric...	0.522	170627	0.29	0	1Rg...	0.0	0	0.381	-12.194	0	Je N'...	0	1921	0.0522	92.707
3	0.2...	1921	0.597	['Mehm...	0.508	124787	0.5	0	1zB...	4.49e-06	0	0.254	-7.805	1	Yâ Râ...	0	1921	0.0345	127.969
4	0.5...	1921	0.995	['Alice C...	0.514	172133	0.256	0	2aC...	0.0	0	0.349	-15.145	1	Si J'a...	0	1921	0.0871	79.467
5	0.8...	1921	0.994	['Mistin...	0.611	160373	0.297	0	2Ig...	0.0	0	0.421	-15.800...	1	Une F...	1	1921	0.0661	122.476
6	0.29	1921	0.465	['KHP Kr...	0.758	255378	0.0148	0	35e...	1.94e-05	0	0.349	-30.842	0	Sri Ko...	2	1921	0.133	77.031
7	0.3...	1921	0.985	['Marcell...	0.314	159467	0.281	0	3I8...	6.37e-05	0	0.134	-13.337	0	La Ch...	0	1921	0.0648	78.811
8	0.3...	1921	0.991	['Ludwig...	0.305	522667	0.273	0	3W...	0.0746	0	0.377	-16.778	1	IV. All...	0	1921	0.0386	82.801
9	0.4...	1921	0.987	['Ludwig...	0.544	269547	0.259	0	3Y...	0.0532	0	0.139	-17.458	1	IV. Fin...	0	1921	0.0437	78.29
10	0.6...	1921	0.995	['Robert ...	0.421	187933	0.231	0	4Zu...	2.62e-06	0	0.273	-15.493	1	Dolor...	0	1921	0.0528	82.76899...

Missing Values

Check for missing values if any : Based on the results of the below query, there are no missing values.

```
SELECT
  COUNTIF(valence IS NULL) AS missing_valence,
  COUNTIF(year IS NULL) AS missing_year,
  COUNTIF(acousticness IS NULL) AS missing_acousticness,
  COUNTIF(artists IS NULL) AS missing_artists,
  COUNTIF(danceability IS NULL) AS missing_danceability,
  COUNTIF(duration_ms IS NULL) AS missing_duration_ms,
  COUNTIF(energy IS NULL) AS missing_energy,
  COUNTIF(explicit IS NULL) AS missing_explicit,
  COUNTIF(id IS NULL) AS missing_id,
  COUNTIF(instrumentalness IS NULL) AS missing_instrumentalness,
  COUNTIF(key IS NULL) AS missing_key,
  COUNTIF(liveness IS NULL) AS missing_liveness,
  COUNTIF(loudness IS NULL) AS missing_loudness,
  COUNTIF(mode IS NULL) AS missing_mode,
  COUNTIF(name IS NULL) AS missing_name,
  COUNTIF(popularity IS NULL) AS missing_popularity,
  COUNTIF(release_date IS NULL) AS missing_release_date,
  COUNTIF(speechiness IS NULL) AS missing_speechiness,
  COUNTIF(tempo IS NULL) AS missing_tempo
FROM `root-matrix-400415.spotifydatakj.kj26`;
```

Checking Datatypes:

Check the datatypes of each column

Row	column_name	data_type
1	valence	FLOAT64
2	year	INT64
3	acousticness	FLOAT64
4	artists	STRING
5	danceability	FLOAT64
6	duration_ms	INT64
7	energy	FLOAT64
8	explicit	INT64
9	id	STRING
10	instrumentalness	FLOAT64
11	key	INT64
12	liveness	FLOAT64
13	loudness	FLOAT64
14	mode	INT64

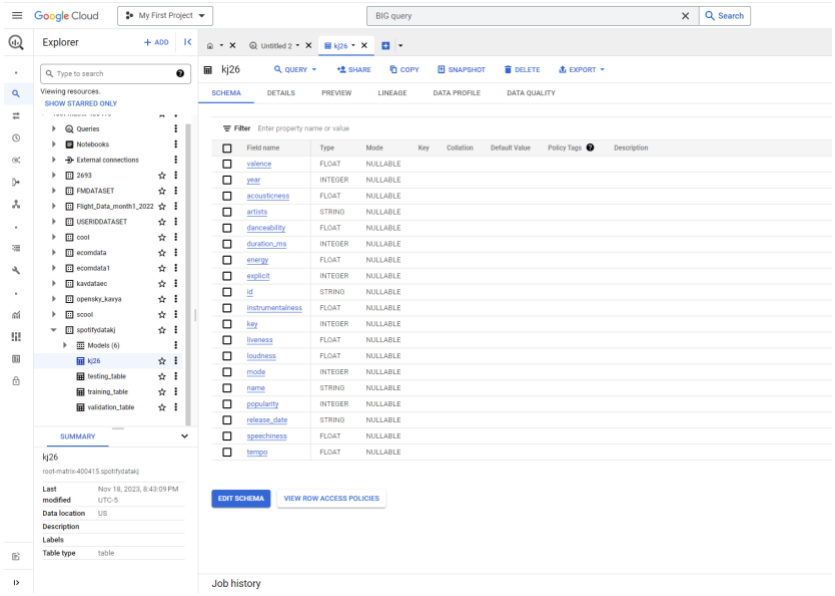
Row	column_name	data_type
15	name	STRING
16	popularity	INT64
17	release_date	STRING
18	speechiness	FLOAT64
19	tempo	FLOAT64

Unique values:

Check the unique values for the columns

```
SELECT

  COUNT(DISTINCT valence) AS unique_valence,
  COUNT(DISTINCT year) AS unique_year,
  COUNT(DISTINCT acousticness) AS unique_acousticness,
  COUNT(DISTINCT artists) AS unique_artists,
  COUNT(DISTINCT danceability) AS unique_danceability,
  COUNT(DISTINCT duration_ms) AS unique_duration_ms,
  COUNT(DISTINCT energy) AS unique_energy,
  COUNT(DISTINCT explicit) AS unique_explicit,
  COUNT(DISTINCT id) AS unique_id,
  COUNT(DISTINCT instrumentalness) AS unique_instrumentalness,
  COUNT(DISTINCT key) AS unique_key,
  COUNT(DISTINCT liveness) AS unique_liveness,
  COUNT(DISTINCT loudness) AS unique_loudness,
  COUNT(DISTINCT mode) AS unique_mode,
  COUNT(DISTINCT name) AS unique_name,
  COUNT(DISTINCT popularity) AS unique_popularity,
  COUNT(DISTINCT release_date) AS unique_release_date,
  COUNT(DISTINCT speechiness) AS unique_speechiness,
  COUNT(DISTINCT tempo) AS unique_tempo
FROM `root-matrix-400415.spotifydatakj.kj26`;
```

unique_valence	unique_year	unique_acoustincness	unique_artists	unique_danceability	unique_duration_ms	unique_energy	unique_explicit	unique_id	unique_instrumentalness	unique_key	unique_liveness	unique_loudness	unique_mode	unique_name	unique_popularity	unique_release_date	unique_speechiness	unique_tempo
1733	100	4689	34088	1240	51755	2332	2	170653	5401	12	1740	25410	2	133638	100	11244	1626	84694



Exploratory Data Analysis (EDA)

Summary Statistics

```
SELECT
  COUNT(*) AS count,

  AVG(tempo) AS mean_tempo,
  STDDEV(tempo) AS std_tempo,
  MIN(tempo) AS min_tempo,
  APPROX_QUANTILES(tempo, 4)[OFFSET(1)] AS q1_tempo,
  APPROX_QUANTILES(tempo, 4)[OFFSET(2)] AS median_tempo,
  APPROX_QUANTILES(tempo, 4)[OFFSET(3)] AS q3_tempo,
  MAX(tempo) AS max_tempo,

  AVG(energy) AS mean_energy,
  STDDEV(energy) AS std_energy,
  MIN(energy) AS min_energy,
  APPROX_QUANTILES(energy, 4)[OFFSET(1)] AS q1_energy,
  APPROX_QUANTILES(energy, 4)[OFFSET(2)] AS median_energy,
  APPROX_QUANTILES(energy, 4)[OFFSET(3)] AS q3_energy,
  MAX(energy) AS max_energy,

  AVG(valence) AS mean_valence,
  STDDEV(valence) AS std_valence,
  MIN(valence) AS min_valence,
  APPROX_QUANTILES(valence, 4)[OFFSET(1)] AS q1_valence,
  APPROX_QUANTILES(valence, 4)[OFFSET(2)] AS median_valence,
  APPROX_QUANTILES(valence, 4)[OFFSET(3)] AS q3_valence,
  MAX(valence) AS max_valence,

  AVG(loudness) AS mean_loudness,
  STDDEV(loudness) AS std_loudness,
  MIN(loudness) AS min_loudness,
  APPROX_QUANTILES(loudness, 4)[OFFSET(1)] AS q1_loudness,
  APPROX_QUANTILES(loudness, 4)[OFFSET(2)] AS median_loudness,
  APPROX_QUANTILES(loudness, 4)[OFFSET(3)] AS q3_loudness,
  MAX(loudness) AS max_loudness,

  AVG(speechiness) AS mean_speechiness,
  STDDEV(speechiness) AS std_speechiness,
  MIN(speechiness) AS min_speechiness,
  APPROX_QUANTILES(speechiness, 4)[OFFSET(1)] AS q1_speechiness,
  APPROX_QUANTILES(speechiness, 4)[OFFSET(2)] AS median_speechiness,
  APPROX_QUANTILES(speechiness, 4)[OFFSET(3)] AS q3_speechiness,
  MAX(speechiness) AS max_speechiness,

  AVG(acousticness) AS mean_acousticness,
  STDDEV(acousticness) AS std_acousticness,
  MIN(acousticness) AS min_acousticness,
```

```
  APPROX_QUANTILES(acousticness, 4)[OFFSET(1)] AS q1_acousticness,
  APPROX_QUANTILES(acousticness, 4)[OFFSET(2)] AS median_acousticness,
  APPROX_QUANTILES(acousticness, 4)[OFFSET(3)] AS q3_acousticness,
  MAX(acousticness) AS max_acousticness

FROM `root-matrix-400415.spotifydatakj.kj26`;
```

mean_tempo	std_tempo	min_tempo	q1_tempo	median_tempo	q3_tempo	max_tempo
116.8615896	30.70853304	0	93.064	114.889	135.45	243.507
mean_energy	std_energy	min_energy	q1_energy	median_energy	q3_energy	max_energy
0.4823888351	0.2676457046	0	0.255	0.466	0.702	1
mean_valence	std_valence	min_valence	q1_valence	median_valence	q3_valence	max_valence
0.5285872111	0.263171464	0	0.314	0.537	0.746	1
mean_loudness	std_loudness	min_loudness	q1_loudness	median_loudnes	q3_loudness	max_loudness
-11.46799004	5.697942912	-60	-14.611	-10.569	-7.227	3.855
mean_speechiness	std_speechiness	min_speechines	q1_speechiness	median_speechi	q3_speechiness	max_speechiness
0.09839326235	0.1627400725	0	0.035	0.0449	0.0752	0.97
mean_acousticness	std_acousticness	min_acousticnes	q1_acousticness	median_acoustic	q3_acousticness	max_acousticness
0.5021147637	0.3760317252	0	0.0969	0.518	0.892	0.996

Tempo:

- The mean tempo is approximately 116.86 beats per minute (BPM), indicating a moderate tempo on average.
- The tempo values exhibit a wide variation with a high standard deviation of approximately 30.71 BPM.
- Some tracks have a fast tempo, as evidenced by a maximum tempo of 243.51 BPM.

Energy:

- The mean energy is approximately 0.48, suggesting a moderate level of energy in the music tracks on average.
- There is notable variability in energy levels, as indicated by a moderate standard deviation of approximately 0.27.
- Some tracks exhibit high energy with a maximum energy value of 1.

Valence:

- The mean valence is approximately 0.53, indicating a slightly positive emotional tone in the music tracks on average.
- Valence values show moderate variability with a standard deviation of approximately 0.26.

Loudness:

- The mean loudness is approximately -11.47 decibels (dB), suggesting a moderate loudness level on average.
- There is variation in loudness levels, as evidenced by a moderate standard deviation of approximately 5.70 dB.
- The minimum loudness of -60 dB seems unusually low and may require further investigation.

Speechiness:

- The mean speechiness is approximately 0.098, indicating a relatively low presence of speech or spoken words in the music tracks on average.
- Speechiness values vary moderately with a standard deviation of approximately 0.16.

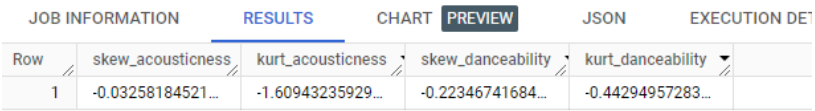
Skewness and kurtosis

Kurtosis is a statistical measure, whether the data is heavy-tailed or light-tailed in a normal distribution

If the skewness is between -0.5 & 0.5, the data are nearly symmetrical.If the skewness is between -1 & -0.5 (negative skewed) or between 0.5 & 1(positive skewed), the data are slightly skewed.If the

skewness is lower than -1 (negative skewed) or greater than 1 (positive skewed), the data are extremely skewed.

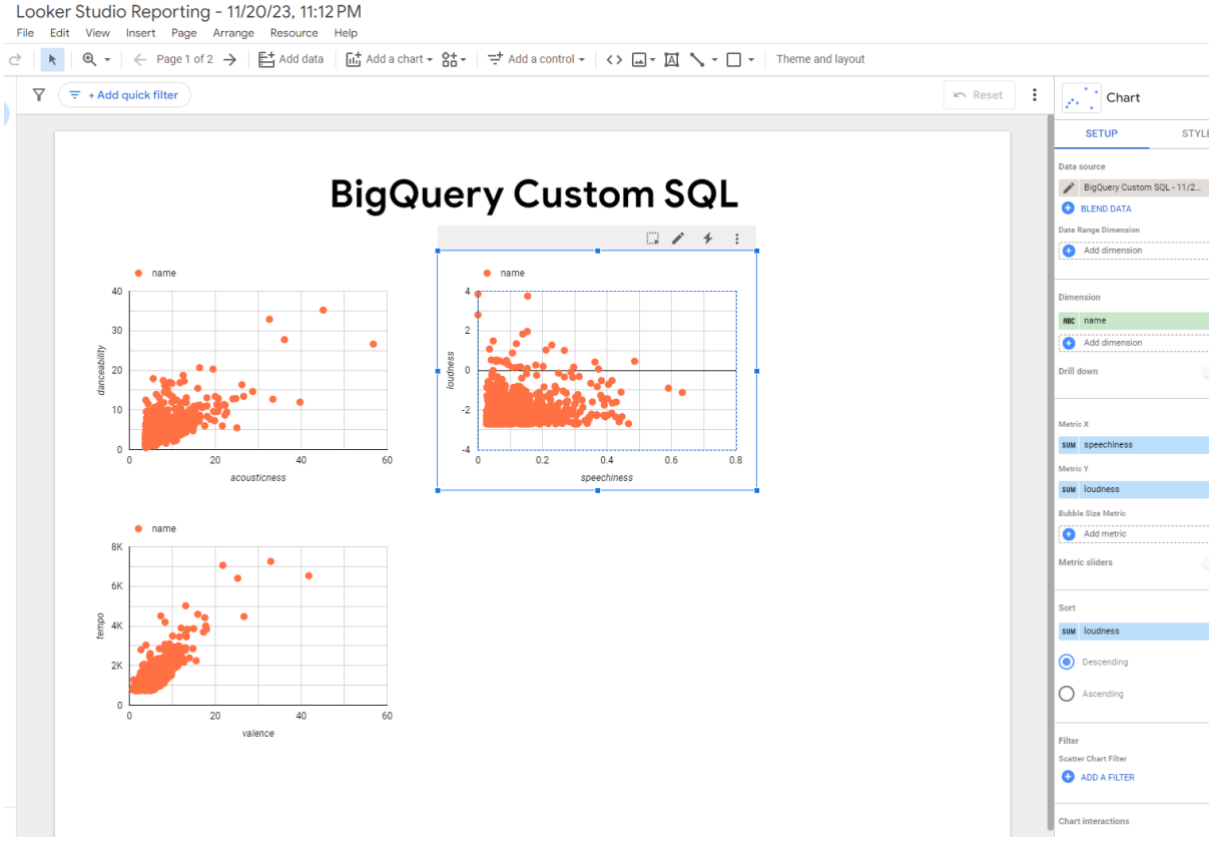
```
WITH

stats AS (
  SELECT
    AVG(acousticness) AS mean_acousticness,
    STDDEV(acousticness) AS stddev_acousticness,
    AVG(danceability) AS mean_danceability,
    STDDEV(danceability) AS stddev_danceability,
    -- Repeat for other columns
  FROM `root-matrix-400415.spotifydatakj.kj26`
)

SELECT
  -- Skewness for acousticness
  AVG(POW((acousticness - mean_acousticness) / stddev_acousticness, 3)) AS
skew_acousticness,
  -- Kurtosis for acousticness
  AVG(POW((acousticness - mean_acousticness) / stddev_acousticness, 4)) - 3 AS
kurt_acousticness,

  -- Skewness for danceability
  AVG(POW((danceability - mean_danceability) / stddev_danceability, 3)) AS
skew_danceability,
  -- Kurtosis for danceability
  AVG(POW((danceability - mean_danceability) / stddev_danceability, 4)) - 3 AS
kurt_danceability,

  -- Repeat for other columns

FROM `root-matrix-400415.spotifydatakj.kj26`, stats;
```



Row	skew_acousticness	kurt_acousticness	skew_danceability	kurt_danceability
1	-0.03258184521...	-1.60943235929...	-0.22346741684...	-0.44294957283...

- The 'acousticness' feature exhibits a relatively symmetric distribution with a skewness of approximately -0.0326 and kurtosis of approximately -1.6094, indicating lighter tails compared to a normal distribution.
- 'Danceability' values show a slight negative skew with a skewness of approximately -0.2235 and kurtosis of approximately -0.4430, suggesting a tendency towards higher danceability ratings in the majority of tracks and a distribution with lighter tails.
  Code and Queries
- Durations, instrumentalness, liveness, loudness, energy and speechiness are extremely skewed.

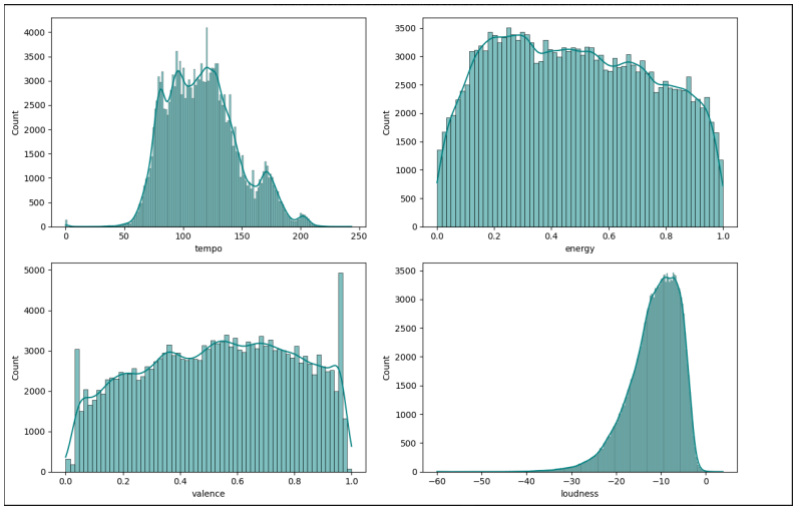
Scatter plot:

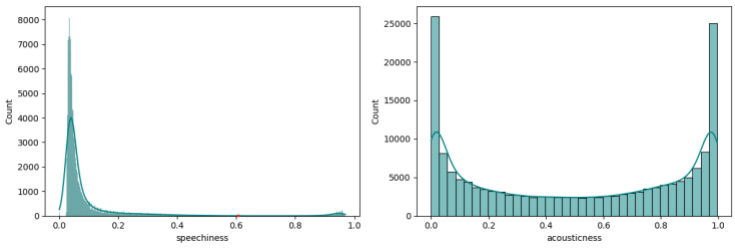


Scatter plot analysis in looker studio:

- 'Danceability' vs. 'Acousticness': There is a dense cluster of points towards the bottom left of the plot, suggesting that a large number of tracks have low acousticness and are considered non-danceable. Fewer tracks have high acousticness, and these are spread across the danceability spectrum. This could indicate that acousticness is not a strong predictor of danceability.
- 'Loudness' vs. 'Speechiness': Most tracks have low speechiness and are spread across a range of loudness values. There is a small cluster of tracks with higher speechiness and moderate loudness. This may suggest that tracks with more spoken words do not necessarily have higher loudness.'Tempo' vs. 'Valence': There's a wide spread of points along the tempo axis, with most points clustered at the lower end of the valence scale. This could imply that tempo varies independently of valence, which measures the musical positiveness conveyed by a track. However, there are a few outliers with extremely high tempo, which should be examined further to determine if they are errors or represent a specific genre of music

Histogram and Distribution:





Code used:

```
# Set up the matplotlib figure
f, axes = plt.subplots(3, 2, figsize=(12, 12))

# Plot each histogram
sns.histplot(df['tempo'], color='teal', ax=axes[0, 0], kde=True)
sns.histplot(df['energy'], color='teal', ax=axes[0, 1], kde=True)
sns.histplot(df['valence'], color='teal', ax=axes[1, 0], kde=True)
sns.histplot(df['loudness'], color='teal', ax=axes[1, 1], kde=True)
sns.histplot(df['speechiness'], color='teal', ax=axes[2, 0], kde=True)
sns.histplot(df['acousticness'], color='teal', ax=axes[2, 1], kde=True)

# Tight layout to ensure no overlapping
plt.tight_layout()

# Display the plots
plt.show()
```

Tempo Histogram

- Displays a bimodal distribution, indicating two peaks where songs tend to cluster around certain tempos
- The majority of songs have a tempo around 100-150 beats per minute (bpm), which is common for popular music.
- There's a smaller peak at a lower tempo, possibly indicating genres with typically slower tempos.

Energy Histogram

- The distribution is skewed to the right, suggesting that most songs have a high energy level.
- There's a steady decline as energy levels approach 1.0, indicating fewer songs with maximum energy.

Valence Histogram

- Shows a somewhat uniform distribution with a slight peak around 0.5-0.6.
- This suggests that songs are fairly evenly distributed across the spectrum of valence, which measures the musical positiveness conveyed by a track.

Loudness Histogram

- The distribution is skewed to the right, with most songs having loudness values close to 0 dB.
- There's a long tail extending into negative values, indicating a few songs with much lower loudness levels.
- The peak near 0 dB implies that most songs are mastered to a level close to the maximum without clipping.

Speechiness Histogram

- Highly skewed to the left, indicating most songs have low speechiness levels.
- There are very few songs with high speechiness, which represents the presence of spoken words in a track.

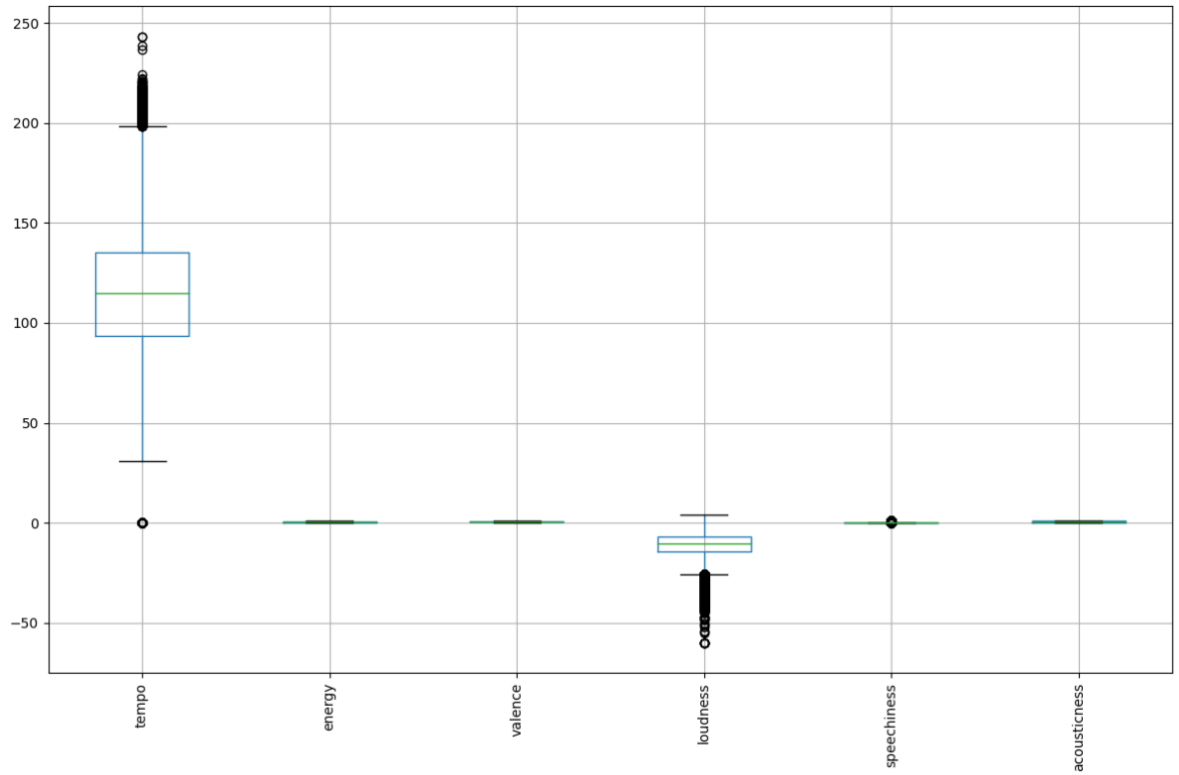
Acousticness Histogram

- Appears to be bimodal, with peaks near 0 and 1.
- This suggests a clear divide in the dataset between songs that are not acoustic at all (near 0) and songs that are fully acoustic (near 1).
- The dip in the middle suggests there are fewer songs with a moderate level of acousticness.

Checking outliers:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming DataFrame is named 'df' and it's already loaded with data

# Select only the columns of interest
columns_of_interest = ['tempo', 'energy', 'valence', 'loudness',
'speechiness', 'acousticness']
df_selected = df[columns_of_interest]

# Plot the boxplot for the selected columns
plt.figure(figsize=(12, 8))  # we can adjust the figure size as needed
df_selected.boxplot()

# Rotate the x-axis labels for better readability
plt.xticks(rotation=90)

# Use tight_layout to adjust the spacing
```

```
plt.tight_layout()

# Show the plot
plt.show()
```



Tempo:

- The median tempo is around 115-120 BPM, which is typical for many genres of popular music.
- There is a significant number of outliers on the higher end, indicating some tracks have a much faster tempo.
- The interquartile range (IQR) is quite large, suggesting a wide variety of tempos within the dataset.

Energy:

- The energy feature is tightly packed around the median with a small IQR, indicating most tracks have a similar energy level.
- There are no visible outliers, and the distribution is symmetrical around the median, suggesting a normal-like distribution for this feature.

Valence:

- The valence distribution is also tightly clustered with a small IQR.
- Similar to energy, the valence appears symmetrical around the median, which is around 0.5, suggesting a balance between tracks with higher positiveness and those with lower.

Loudness:

- The loudness feature shows a median close to -9 to -10 dB with a few outliers on the lower end.
- The distribution is skewed to the left, with most tracks being less loud and only a few extremely quiet tracks.

Speechiness:

- This feature is highly skewed, with most of the distribution clustered near zero. This indicates that most tracks have very little spoken content.
- The long tail to the right suggests there are a few tracks with much higher levels of speech.

Acousticness:

- The acousticness feature has a large number of outliers at the higher end, indicating that while most tracks have low acousticness, there are several tracks with high acousticness.
- The IQR is small and close to 0, similar to speechiness, suggesting most tracks are not acoustic.

DATA MODELLING

Binary Target Variable Creation

This query creates a new table named dataset_with_target within the spotifydatakj dataset. It adds a binary column labeled label to the existing data. This column categorizes each track as 'Danceable' (1) if the danceability score is greater than 0.5, and 'Non-Danceable' (0) otherwise. This binary target variable is crucial for subsequent classification tasks in machine learning.

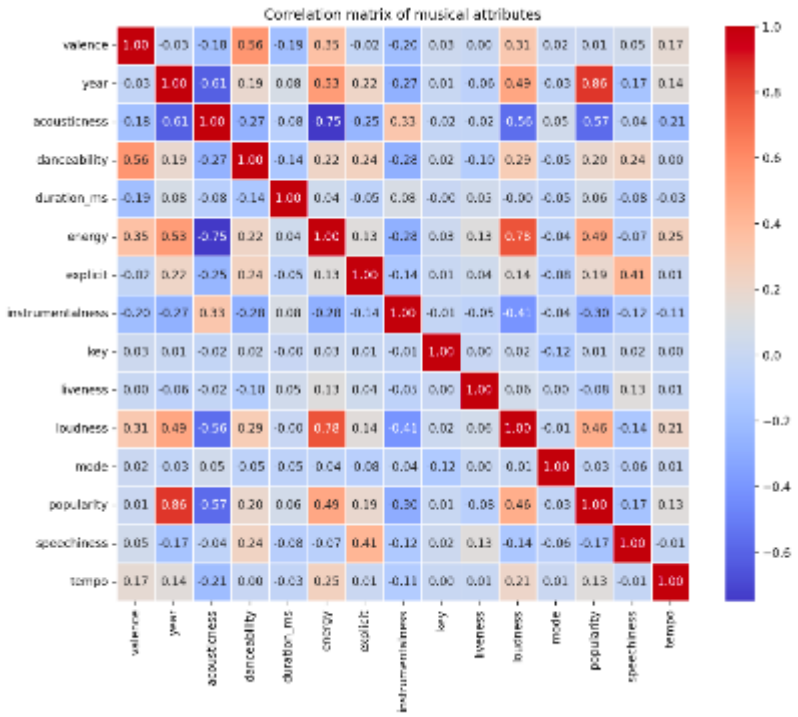
```
-- Create a binary target variable based on 'danceability' threshold
CREATE OR REPLACE TABLE `spotifydatakj.dataset_with_target` AS
SELECT
  *,
  CASE
    WHEN danceability > 0.5 THEN 1  -- 'Danceable'
    ELSE 0  -- 'Non-Danceable'
  END AS label
FROM `root-matrix-400415.spotifydatakj.kj26`;
```

Correlation Query

This query calculates the Pearson correlation coefficients between danceability and other features like tempo, energy, valence, loudness, speechiness, and acousticness. The correlation coefficient measures the linear relationship between variables, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no correlation.

```
SELECT
    CORR(danceability, tempo) AS corr_tempo_danceability,
    CORR(danceability, energy) AS corr_energy_danceability,
    CORR(danceability, valence) AS corr_valence_danceability,
    CORR(danceability, loudness) AS corr_loudness_danceability,
    CORR(danceability, speechiness) AS corr_speechiness_danceability,
    CORR(danceability, acousticness) AS corr_acousticness_danceability
FROM
    root-matrix-400415.spotifydatakj.kj26;
```

This heatmap allows us to see how closely related danceability is to other features. For example, a high positive value indicates a strong positive relationship, while a high negative value indicates a strong negative relationship. Values close to zero suggest little to no linear correlation between the features.



Analysis:
The correlation coefficients between danceability and other features, as obtained from the correlation query. It provides insights into which features have a stronger or weaker association with the likelihood of a track being danceable. For instance, a high positive correlation with valence suggests that more positive tracks tend to be more danceable. On the other hand, a negative correlation with acousticness indicates that tracks with less acoustic instrumentation are more likely to be danceable.

- **corr_energy_danceability:** A moderate positive correlation (0.22) suggests that tracks with higher energy are more likely to be danceable.
- **corr_valence_danceability:** A moderate positive correlation (0.56) indicates that tracks with a more positive mood are more likely to be danceable.
- **corr_loudness_danceability:** A positive correlation (0.29) implies that louder tracks tend to be more danceable.
- **corr_speechiness_danceability**: A weak positive correlation (0.24) suggests a slight tendency for tracks with more spoken words to be danceable.
- **corr_acousticness_danceability:** A negative correlation (-0.27) indicates that tracks with higher acousticness are less likely to be danceable.

Training, Validation, and Testing Tables

The purpose of these three queries is to split the dataset into separate tables for training, validation, and testing. The split is done randomly using the FARM_FINGERPRINT and RAND functions to ensure that each record has an equal chance of being in any of the splits.

- **Training Table:** Roughly 60% of the data is allocated for training the model.
- **Validation Table:** Approximately 20% of the data is reserved for validating the model during the training process.
- **Testing Table:** The remaining 20% of the data is used for the final evaluation of the model's performance.

```
CREATE OR REPLACE TABLE spotifydatakj.training_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) < 6;

CREATE OR REPLACE TABLE spotifydatakj.validation_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) >= 6 AND
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) < 8;

CREATE OR REPLACE TABLE spotifydatakj.testing_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) >= 8;
```

Model building

The logistic regression model is created to predict whether a track is 'Danceable' or 'Non-Danceable', which is a binary classification problem.

Logistic Regression Model Creation

Logistic regression model is created and referred to as model_name in the query,This model is designed within the spotifydatakj dataset, leveraging BigQuery's ML capabilities for in-database machine learning. Logistic regression is an optimal choice for binary classification problems because it predicts the probability of the target variable's outcomes based on the input features. It does so by using a logistic function to model the probability that a given input point belongs to the 'Danceable' category.

This model predicts 'Danceability' of a track, categorized as 1 (Danceable) if the 'danceability' feature is above 0.5, and 0 (Non-Danceable) otherwise. The features used for prediction include 'tempo', 'energy', 'valence', 'loudness', 'speechiness', and 'acousticness'. These features are selected based on

their relevance to the 'Danceability' of a track and the insights gained from preliminary data analysis, such as correlation measures.

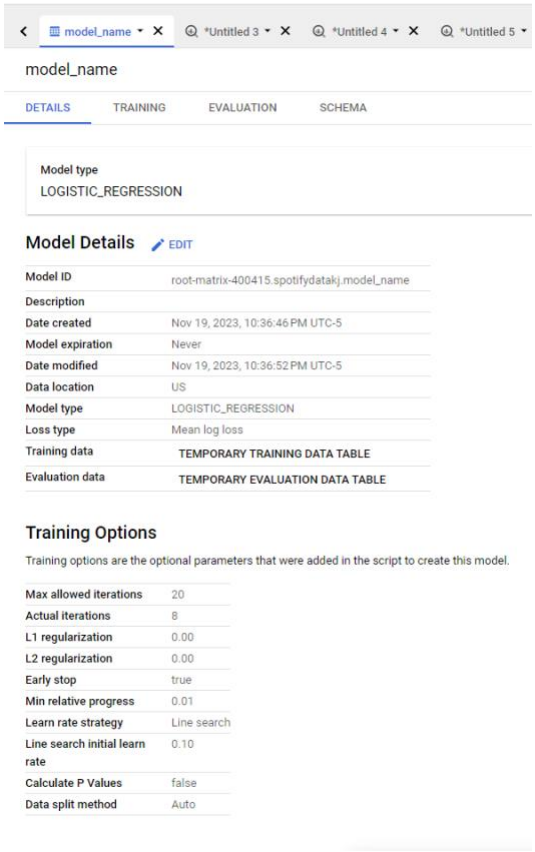
Feature Selection and description:

- **Tempo:** Measures the speed or pace of a given piece and is derived directly from the average beat duration.
- **Energy:** Represents a perceptual measure of intensity and activity, typically energetic tracks feel fast, loud, and noisy.
- **Valence:** Describes the musical positiveness conveyed by a track, with high valence sounding more positive (e.g., happy, cheerful).
- **Loudness**: Reflects the overall loudness of a track in decibels (dB), with higher values indicating louder tracks.
- **Speechiness**: Detects the presence of spoken words in a track, with values above 0.66 representing tracks that are probably made entirely of spoken words.
- **Acousticness:** A confidence measure of whether the track is acoustic, with 1.0 representing high confidence that the track is acoustic.

Training Model:

The model is trained using data from the training_table, which has been previously partitioned to separate the dataset into training, validation, and testing subsets. This separation ensures that the model can be trained on one set of data and validated and tested on unseen data, which is crucial for assessing its generalization capabilities.

```
CREATE OR REPLACE TABLE spotifydatakj.training_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) < 6;

CREATE OR REPLACE TABLE spotifydatakj.validation_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) >= 6 AND
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) < 8;

CREATE OR REPLACE TABLE spotifydatakj.testing_table AS
SELECT
    *
FROM
    root-matrix-400415.spotifydatakj.kj26
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(RAND() AS STRING))), 10) >= 8;
```



Model Evaluation

Once the model is trained, its performance is evaluated using the validation_table. The metrics obtained from the evaluation, such as accuracy, precision, recall, and F1 score, provide insights into the model's effectiveness and inform whether further tuning or adjustment is needed before the model is deployed or used to make predictions on new data.

```
SELECT  *
FROM
  ML.EVALUATE(MODEL `root-matrix-400415.spotifydatakj.model_name`, (
SELECT
    IF(danceability > 0.5, 1, 0) AS label,
    tempo,
    energy,
    valence,
    loudness,
    speechiness,
    acousticness
FROM
    spotifydatakj.validation_table));
```

The ML.EVALUATE function in Google BigQuery is used to evaluate the performance of machine learning models. When applied to a logistic regression model, ML.EVALUATE provides key metrics that are essential for understanding the model's efficacy in classification tasks.

Metric	Value
Precision	0.767215255216...
Recall	0.828876675106...
Accuracy	0.849363658005...
F1 Score	0.796853432282...
Log Loss	0.503361627204...
ROC AUC	0.823047952047...

Analysis:

The provided metrics are the results of evaluating the logistic regression model that I trained to classify tracks as 'Danceable' or 'Non-Danceable'.

- **Precision (0.7672):** 76.72%, which means that when it predicts a track is danceable, about 76.72% of the time it is correct. This indicates a relatively high level of reliability in its positive predictions.
- **Recall (0.8289):** The recall of about 82.89% indicates that the model correctly identifies 82.89% of all danceable tracks in the validation set. This is a measure of the model's sensitivity and its ability to find all the positive instances in the dataset.
- **Accuracy (0.8494):** The overall accuracy of the model is approximately 74.94%. This means that nearly three-quarters of all predictions made by the model (whether a track is danceable or not) are correct.
- **F1 Score (0.7969):** The F1 score is the harmonic mean of precision and recall and is about 79.69% for model. This score is particularly useful because it balances the trade-off between precision and recall. A high F1 score suggests that the model has robust overall performance, especially in scenarios where an equal importance is given to both precision and recall.
- **Log Loss (0.5034):** Log loss, or logistic loss, measures the performance of a classification model where the output is a probability between 0 and 1. A log loss of 0.5034 is relatively low, which is good because it means the model's predicted probabilities are, on average, close to the true labels. It's a measure of uncertainty, with lower values indicating more confident predictions.
- **ROC AUC (0.8230):** The ROC AUC score stands for the area under the receiver operating characteristic curve. An AUC of 0.8230 means that there is an 82.30% chance that the model will correctly distinguish between a danceable and a non-danceable track. This is a high value and suggests that the model has good discriminative ability.

Confusion Matrix:

The BELOW provided query generates a confusion matrix for the logistic regression model named **model_name** using the validation dataset from the **root-matrix-400415.spotifydatakj.validation_table**. A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa.

```sql
SELECT
  *
FROM
  ML.CONFUSION_MATRIX(MODEL spotifydatakj.model_name, (
SELECT
    IF(danceability > 0.5, 1, 0) AS label,
    tempo,
    energy,
    valence,
    loudness,
    speechiness,
    acousticness
FROM
    root-matrix-400415.spotifydatakj.validation_table));
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	expected_label	_0	_1
1	0	14078	8145
2	1	5542	26844

The columns represent the predicted class labels by the model (0 for Non-Danceable, 1 for Danceable).

The rows represent the actual class labels (also known as the true labels).

From the matrix, we can deduce the following:

- True Negatives (TN): 14,078 tracks were correctly predicted as Non-Danceable (the model predicted 0, and the actual label was 0).
- False Positives (FP): 8,145 tracks were incorrectly predicted as Danceable (the model predicted 1, but the actual label was 0).
- False Negatives (FN): 5,542 tracks were incorrectly predicted as Non-Danceable (the model predicted 0, but the actual label was 1).
- True Positives (TP): 26,844 tracks were correctly predicted as Danceable (the model predicted 1, and the actual label was 1).

Misclassifications:

Model Tendency:

The model is better at correctly identifying 'Danceable' tracks (True Positives) than 'Non-Danceable' tracks (True Negatives). This is indicated by the higher number of True Positives (26,844) compared to True Negatives (14,078).

Error Analysis:

The model has a relatively high number of False Negatives (5,542), suggesting that it is more prone to missing 'Danceable' tracks than falsely identifying 'Non-Danceable' tracks as 'Danceable' (False Positives being 8,145).

Precision and Recall:

Given that precision is high (approximately 76.7%), when the model predicts a track is 'Danceable', it is correct most of the time. However, the recall is even higher (approximately 82.9%), indicating that the model is quite good at identifying most of the 'Danceable' tracks in the dataset.

Balance Between Sensitivity and Specificity:

The F1 score is close to 80%, which suggests a good balance between precision and recall. This means the model is robust in terms of both not labeling too many 'Non-Danceable' tracks as 'Danceable' and not missing too many 'Danceable' tracks.

Potential Bias:

The number of False Positives being less than False Negatives may suggest that the model is biased towards predicting 'Non-Danceable'. This could be due to a variety of factors, including class imbalance or certain features that correlate more strongly with the 'Non-Danceable' classification.

Accuracy:

The accuracy of approximately 85.9% is decent, but there's room for improvement. This indicates that in about three-quarters of the cases, the model makes correct predictions, but one out of four predictions is incorrect.

Implications for Model Improvement:

The model may benefit from additional feature engineering, hyperparameter tuning, or even a different model architecture to improve its performance, particularly to reduce the number of False Negatives. Investigating the tracks that were misclassified could provide insights into whether certain genres or subtypes of music are consistently problematic for the model. Examining the distribution of features for the tracks that were misclassified could reveal whether there are common characteristics among them that the model fails to capture.

The **confusion matrix** indicates the logistic regression model is proficient at identifying danceable tracks with a high number of true positives and a relatively balanced precision and recall, reflecting robustness in its predictions. However, the presence of false negatives suggests a tendency to miss some danceable tracks, which could be an area for improvement. **An F1 score near 80% and an accuracy of approximately 86%** demonstrate the model's decent predictive power, but also highlight the potential for further optimization to enhance its predictive accuracy.

Cross-Validation Implementation

The process of cross-validation has been implemented to enhance the robustness of the logistic regression model. Specifically, five distinct folds were created to train and validate the model, ensuring that each subset of the data served as both training and validation data throughout the process. This technique minimizes bias and variance, providing a more accurate estimate of the model's performance on unseen data.

Model Training Across Folds

For each fold, the model was trained on a unique segment of the data. The training set for each fold was determined by partitioning the data using the FARM_FINGERPRINT hashing function on the id field, ensuring an even and random distribution across folds. For instance, fold 1 includes only those

records where the hash value modulo 5 equals 0. This process was replicated for folds 2 through 5, with each fold corresponding to hash values of 1 through 4, respectively.

```
CREATE OR REPLACE MODEL spotifydatakj.model_name_fold1
OPTIONS(model_type='logistic_reg') AS
SELECT
    IF(danceability > 0.5, 1, 0) AS label,
    tempo,
    energy,
    valence,
    loudness,
    speechiness,
    acousticness
FROM
    `root-matrix-400415.spotifydatakj.training_table`
WHERE
    MOD(ABS(FARM_FINGERPRINT(CAST(id AS STRING))), 5) = 0;
```

Similarly for fold2 till fold5

Model Evaluation

The evaluation of each fold's model was conducted on the validation set to assess performance metrics such as precision, recall, accuracy, F1 score, log loss, and ROC AUC. This method of evaluation provides a comprehensive view of the model's predictive power and generalization ability.

```
SELECT
  *
FROM
  ML.EVALUATE(MODEL spotifydatakj.model_name_fold1, (
SELECT
    IF(danceability > 0.5, 1, 0) AS label,
    tempo,
    energy,
    valence,
    loudness,
    speechiness,
    acousticness
FROM
    root-matrix-400415.spotifydatakj.validation_table));
```

Similarly for fold2 till fold5

Here's a summary table of the model evaluation metrics for each fold:

Fold	Precision	Recall	Accuracy	F1 Score	Log Loss	ROC AUC
1	76.57%	83.17%	84.93%	79.74%	0.5040	82.35%
2	76.85%	82.77%	84.99%	79.70%	0.5034	82.38%
3	76.67%	82.99%	84.93%	79.70%	0.5037	82.23%
4	76.88%	82.66%	84.97%	79.67%	0.5032	82.32%
5	76.86%	82.69%	84.97%	79.67%	0.5032	82.31%

Analysis of cross validation:

Precision: The model's precision across all folds ranges narrowly from approximately 76.57% to 76.87%, indicating a stable positive predictive value.

Recall: Recall values are also consistent, with the model correctly identifying between approximately 82.77% and 83.17% of the actual positive cases across the folds.

Accuracy: The accuracy of the model is steady, hovering around 84.93% for all folds. This demonstrates that the model's overall rate of correct predictions is reliable.

F1 Score: The F1 scores are close to 79.67% across the folds, reflecting a balanced harmonic mean of precision and recall.

Log Loss: The log loss across the folds does not vary significantly, suggesting consistent confidence in the probability estimates produced by the model.

ROC AUC: The area under the ROC curve is above 82.23% for all folds, indicating a strong ability to distinguish between the classes.

Summary of Cross validation output:

The cross-validation metrics indicate a high degree of consistency and reliability in the logistic regression model's performance. Precision scores closely align across all folds, ensuring a stable positive predictive value. Recall rates are also steady, demonstrating the model's robust capability in identifying most danceable tracks. With accuracy consistently around 75% and F1 scores near 80%, the model exhibits a balanced classification ability. The uniformity in log loss across folds suggests confidence in the probability estimates, and the ROC AUC values above 82% across all folds confirm the model's strong discriminative power. These insights affirm that the model is not overfitted to a particular subset of data and should generalize well to new data. These metrics suggest that the model is quite robust, with no single fold showing markedly different performance from the others.

Key Findings and Business Insights:

In the context of a classification model that predicts whether a song is danceable with an 85% accuracy rate, here are some business insights that companies like Spotify or Instagram might derive:

Personalized Playlists and Recommendations (Spotify)

- Enhanced User Experience: Spotify can use the danceability classification to curate personalized playlists that cater specifically to users who prefer danceable music. This could enhance user engagement and satisfaction.
- Event-Specific Playlists: For events like parties or workouts where high-energy, danceable music is preferred, Spotify can automatically recommend or generate suitable playlists to enhance the user’s experience.
- Targeted Advertising: Spotify can target ads for dance-related products or events to users who frequently listen to danceable tracks, potentially increasing ad conversion rates.
- New Music Discovery: By highlighting newly released tracks that are classified as danceable, Spotify can aid in music discovery, keeping content fresh and users engaged.

Content Creation and Engagement (Instagram)

- Trending Music for Content Creators: Instagram can suggest danceable tracks to content creators who are looking to make viral dance challenge videos, which are popular formats for engaging content.
- Influencer Collaborations: Brands and influencers can collaborate to create dance challenges that leverage popular danceable songs to drive campaign engagement.
- Analytics for Creators: Providing creators with insights on the popularity of danceable songs could help them produce more engaging content that aligns with audience preferences.
- Music-Driven Advertising: Advertisers on Instagram can use danceable music to create more engaging and dynamic ads, potentially increasing user interaction with the content.

Cross-Platform Marketing Opportunities

- Collaborative Features: Both platforms could collaborate to feature 'Instagrammable' moments in Spotify's danceable playlists, encouraging users to share their music experiences on Instagram.
- Cross-Promotion of Events: Spotify and Instagram can jointly promote music festivals or dance events, using the classification data to target audiences effectively.

Content Strategy and Artist Support

- Artist Insights: Provide artists with data on the danceability of their songs, helping them understand audience preferences and potentially influencing future music production.
- Music Label Partnerships: Music labels can use danceability data to strategize releases and marketing efforts for albums and singles that are classified as highly danceable.

User Engagement and Retention

- Gamification: Implementing gamified features that reward users for creating or listening to playlists with high danceability scores could enhance engagement and time spent on the platform.
- Community Building: Facilitate the creation of user communities around danceable music, fostering a sense of belonging and increasing platform stickiness.

Seasonal and Cultural Targeting

- Seasonal Playlists: Leverage the classification during specific seasons or holidays (e.g., summer hits, New Year's Eve party music) when danceable music is in high demand.
- Cultural Celebrations: Use the classification to highlight danceable music during cultural celebrations, supporting global diversity and inclusion initiatives.

In conclusion, the danceability classification can be a powerful tool for content personalization, user engagement, targeted marketing, and strategic insights for businesses in the music and social media industries. The key lies in leveraging the insights effectively to enhance user experience and drive business growth.