Lei Zhang

Prof. David Bamman

INFO 202: Information Organization and Retrieval

December 12, 2023

— Final Implementation Project —

Sensemaking of "Morning" and "Night" Music

Background

Listening to music is a personal and emotional experience. Different types of music can trigger different types of physiological response. At different times of the day, our body may prefer different styles of music. For example, in the morning, I would prefer delightful and positive music that can provide me with the energy to start the day; in the evening, I would prefer soft and sometimes gloomy music to receive relief and relaxation before going to bed.

My curiosity about how the underlying attributes of music could potentially determine whether a track is a "morning" song or a "night" song brought me to conduct this study. In this project, I built a supervised machine learning model to make binary classification (either "morning" or "night") using audio features data from online together with binary labels annotated by myself.

It is worth noting that there is *bias* in my dataset and analysis, because the selected tracks came from a subset of the music that I personally enjoy listening to and the labels derived from my own understand of "morning" vs. "night" music.

In the sections below, I will provide the process, technical specifications, and a reflection of this project.

Process & Technical Specifications

Following the process of sensemaking, the documentation of my process is broken down into two parts: Search & Retrieval, and Analysis & Synthesis of Search Results.

Sensemaking Part I: Search & Retrieval

1. Set up API access

I use Spotify as the main platform to listen to music, and I can get access to the metadata and audio features of each track using the Spotify API. In order to do that, I need to first set up my API access. I created an app in their online platform, and got my "client_id" and "client_secret." To make sure that I could obtain the dataset that I wanted, I tested the API through some random tasks.

2. Build playlists

I decided to use 300 tracks in total for this project, and the tracks were separated into 3 playlists. The separation was necessary because the API only allows the retrieval of a maximum of 100 tracks each time.

I used the Spotify app to create 3 playlists with 100 songs in each playlist. In summary, there are around 150 songs with lyrics and 150 instrumental music without lyrics. There is no single artist that is overrepresented. The songs with lyrics included both English and Chinese songs.

3. Retrieve metadata & audio features data

After creating the playlists, I switched into Jupyter notebook. I first imported all the necessary libraries. Then, I initiated access to the API using my credentials. Finally, I called the Spotify API to send back the metadata and audio features of each track in each playlist. This was accomplished by a function provided by Samantha Jones. The call_playlist() function uses the name of the creator and playlist_id as inputs, and generates the following data frame with 12 columns as output. Each row represents one track.

Track Metadata

artist	album	track_name	track_id

"Track_id" is the unique identifier for each track and I will use it later to merge data tables together.

Track Audio Features

acousticness	danceability	energy	instrumentalness	liveness	loudness	tempo	valence

For the audio features, I was able to specify the columns that I wanted. I intentionally chose the variables that are numbers (floats), and excluded other variables, such as integer variables key, mode, and time_signature. I also excluded the "speechiness" float variable because it does not add any value my dataset — it tells you whether the track is music, podcast, or audio book, etc. Since my dataset contains music only, I did not retrieve the "speechiness" variable.

At the end of this step, I got 3 csv files including the "track_id", metadata, and audio features for each track.

Sensemaking Part II: Analysis & Synthesis of Search Results

1. Annotate each track

This is the label annotation part of the project. For this step, I opened the Spotify app and the csv files on my computer. I added a new column in the csv files called "timing", which will indicate either **1 for morning** or **0 for night**. After listening to each track, I entered either 1 or 0 under the "timing" column for the track. This step was repeated 300 times for the 300 annotations.

2. Prepare final dataset

Now, I have the metadata, the audio features, and the classification labels ready. I then merged the data tables using "track_id" as the unique identifier. The final dataset contains 300 rows and 13 columns.

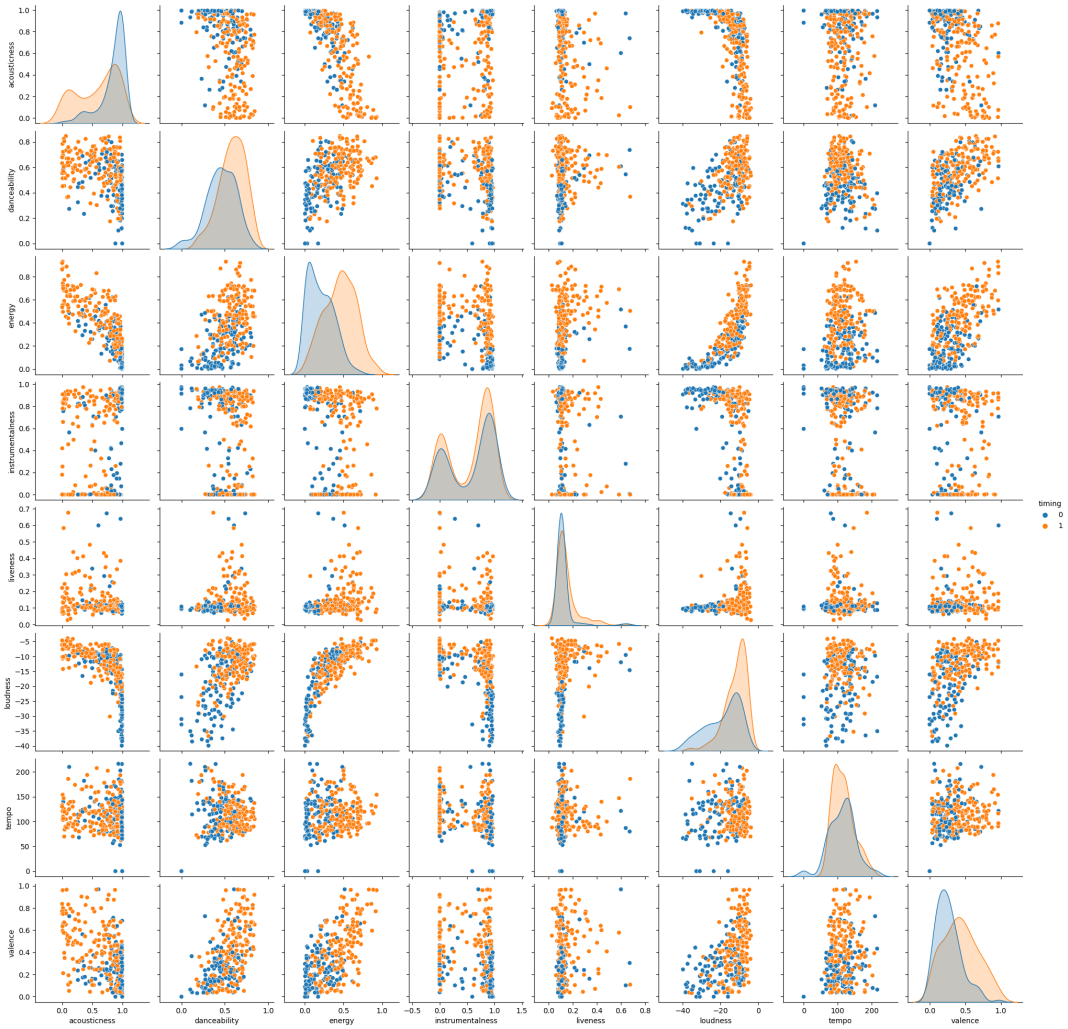
The important columns for classification are shown below. The first 8 columns will be used as predictors, and the "timing" column is the binary label.

acousticness	danceability	energy	instrumentalness	liveness	loudness	tempo	valence	*timing*

3. Data Visualizations

Following instructions by Carla Martins, before building the model, I first took a look at how my data is distributed.

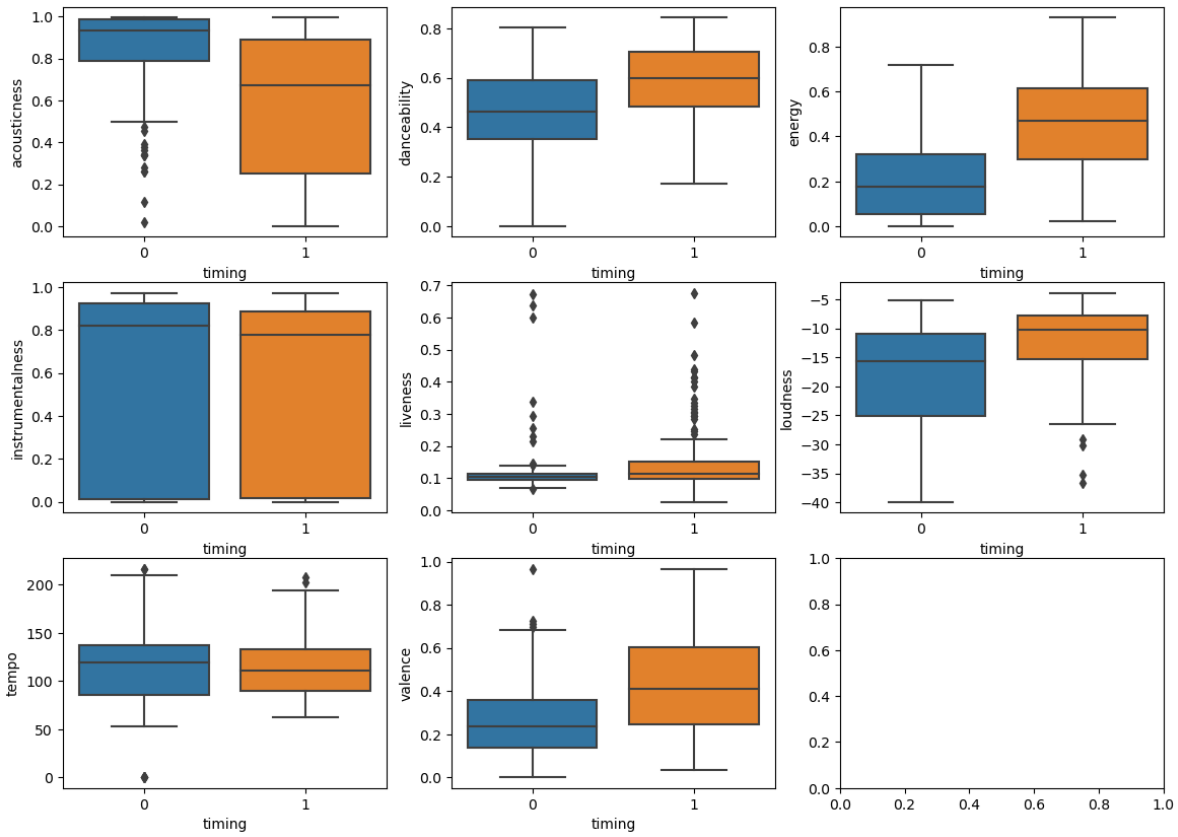
Pairplot from the seaborn package generated the visualization below. Orange represents "morning" tracks, and blue represents "night" tracks.



Findings:

- It seems that there is a strong correlation between “energy” and “loudness” because the higher the “loudness”, the higher the “energy”.
- We also see “morning” tracks generally have a higher “danceability”, “energy”, “loudness”, and “valence”.
- “Acousticness”, “instrumentalness”, “liveness”, and “tempo” do not seem to differentiate between “morning” and “night” tracks.

The box plots below further prove the conclusion above. “Danceability”, “energy”, “loudness”, and “valence” appear to be better predictors.



4. Build the model

Now it's time to build the model. I created the X variable after dropping the "timing" label and the unnecessary metadata columns. The y variable includes data from the "timing" label column.

Then, I split the dataset into training and testing sets. Afterwards, as suggested by Raunaq Aman Jaswal, I built a Linear SVM for model because we are conducting a supervised binary classification task using a small dataset.

5. Evaluate results and fine-tuning

In this step, I experimented with many combinations of different predictors, different number of predictors, and C values. The scenarios below include some noteworthy findings.

Scenario 1: Using all 8 variables as predictors

C=1:

training set score: 0.604167

test set score: 0.583333

C=100:

training set score: 0.629167

test set score: 0.666667

C=0.01:

training set score: 0.683333

test set score: 0.700000

Scenario 2: Using “danceability”, “energy”, “loudness”, and “valence” as predictors

C=1:

training set score: 0.758333

test set score: 0.716667

C=100:

training set score: 0.575000

test set score: 0.583333

C=0.01:

training set score: 0.670833

test set score: 0.650000

Scenario 3: Using “energy" and “loudness” as predictors

C=1:

training set score: 0.750000

test set score: 0.816667

```
predictions = LinearSVM.predict(X_test)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.71      0.87      0.78        23
           1       0.91      0.78      0.84        37

    accuracy                           0.82        60
   macro avg       0.81      0.83      0.81        60
weighted avg       0.83      0.82      0.82        60

[[20  3]
 [ 8 29]]
```

C=100:

training set score: 0.550000

test set score: 0.583333

C=0.01:

training set score: 0.654167

test set score: 0.783333

Scenario 4: Using "energy" as predictor

C=1:

training set score: 0.729167

test set score: 0.733333

C=100:

training set score: 0.725000

test set score: 0.733333

C=0.01:

training set score: 0.704167

test set score: 0.600000

Scenario 5: Using "loudness" as predictor

C=1:

training set score: 0.554167

test set score: 0.633333

C=100:

training set score: 0.533333

test set score: 0.616667

C=0.01:

training set score: 0.658333

test set score: 0.583333

In summary, Scenario 3 under C= 1 gave us the best performing model with a test score of 82%.

Scenario 3: Using “energy" and “loudness” as predictors

C=1:

training set score: 0.750000

test set score: 0.816667

```
predictions = LinearSVM.predict(X_test)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.71      0.87      0.78        23
           1       0.91      0.78      0.84        37

    accuracy                           0.82        60
   macro avg       0.81      0.83      0.81        60
weighted avg       0.83      0.82      0.82        60

[[20  3]
 [ 8 29]]
```

Reflection

Making sense of music is a complex process. However, through this project, I was able to find the key attributes of music that can help me find music more suitable for either morning or night. Although artistic expression can vary dramatically, the latent audio features can bring a logical framework for classification and recommendation tasks.

I also experienced how bias can contribute to the result of analysis. It seems that I personally prefer instrumental songs and there is a specific range of tempo that I enjoy the most. The “morning” and “night” songs I selected both fell into a similar range of “instrumentalness” and “tempo”. Therefore, the two variables were not helpful for making predictions in my dataset.

If I have more time, I would annotate more data, which could improve model performance. Ideally, more random participants should annotate the same set of tracks to achieve a consensus of labeling for each track. In this way, the individual bias can be reduced. Through this project, I built my first machine learning model referencing online tutorials. Once I gain more knowledge on machine learning in a formal course, the technical aspects of the project can be further improved.

References

1. Spotify API. https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features
2. Support Vector Machine (SVM) for Binary and Multi-Class Classification: Hands-On with Scikit-Learn by Carla Martins. https://pub.towardsai.net/support-vector-machine-svm-for-binary-and-multiclass-classification-hands-on-with-scikit-learn-29cdbe5cb90e
3. Extracting Your Fav Playlist Info with Spotify's API by Sammie Jones. https://www.linkedin.com/pulse/extracting-your-fav-playlist-info-spotifys-api-samantha-jones/
4. Classification of modern pop songs into ‘Day’ and ‘Night’ songs by Raunaq Aman Jaswal, Department of ECE, University of Rochester. https://hajim.rochester.edu/ece/sites/zduan/teaching/ece477/projects/2019/RaunaqJaswal_SongClassification_FinalReport.pdf
5. Spotipy Documentation. https://spotipy.readthedocs.io/en/2.22.1/