**Foundational models for music embeddings in the symbolic domain: from NLP to MIR**

**Project Work in Knowledge Engineering**

**2022-2023**

Artificial Intelligence

Lsei [Li (lesi.li@studio.unibo.it)](mailto:Li(lesi.li@studio.unibo.it))

**Contents**

1. **[Introduction](#_30j0zll)** **3**
2. **CHOCO:Chord corpus** **3**
   1. What’s Choco? 3
   2. How it works? 3
   3. How to use ChoCo?.... 4
3. **JAMS** **4**
   1. What’s JAMS? 4
   2. Why? 4
   3. How to process the jams 4
4. **Create Datasets** **5**
   1. Exporting ChoCo as a JAMS dataset..............................................................5
   2. Extract metadata information 5
   3. Data preprocessing 6
5. **Extract chord and mode annotations** **7**
   1. Extract chord information 7
   2. Extract kymode information 8
   3. Merge chord and mode notes into meta 8
6. **Data analysis** **9**
   1. A glimpse of ChoCo 9
   2. Meta-stats 9
   3. Content-stats 10
7. **MODELS** **11**
   1. Feature selection 11
   2. Music classification 12
   3. music similarity 13
8. **[Conclusion](#_2u6wntf)** **15**

**1 Introduction**

The project will focus on learning representations from specific musical dimensions (melody, harmony, rhythm) by exploring a variety of embedding models (also on KGs); we will use a musicological testbed to verify to which extent the learned representations are musically plausible, and test the embeddings on several downstream tasks, including: music similarity, music generation, and music classification.

Using the ChoCo chord corpus

Developing a music embedding model. This model can transform music data into a low-dimensional vector representation so that it can better capture the characteristics and structure of music.

Once the embedding model is trained, it can be used for various downstream tasks such as music similarity measurement and music classification.

**2 CHOCO:Chord corpus**

ChoCo provides 20K+ timed chord annotations for scores and tracks for a variety of genres and styles, integrated, standardized, and semantically enriched from many repositories and databases

**2.1 What’s Choco?**

As a JAMS dataset, audio and score annotations are distinguished by the type attribute in their sandbox; time/metric information is expressed in seconds (audio) and metric: beats (score);

As a knowledge graph, based on our JAMS ontology to model musical annotations, and the Chord and Roma ontology to semantically describe chords; this link provides the SPARQL endpoint.

**2.2 How it works?**

To achieve consistency across annotations, chords are converted into the following 2 notation families: (i) Harte, which popularizes Leadsheet-based notation and is widely used in music information retrieval systems; (ii) Roman numerals, a well-known notation standard , where chords are named according to their degrees. Additionally, Roman numeral chords are syntactically converted to Hart symbols for interoperability. This means that all tracks/productions in ChoCo are always provided with corresponding Harte annotations.

The resulting annotations are rich in provenance data, including metadata for the annotated work or track, the annotated author, identifiers and links, and more. We emphasize that the current version of ChoCo only includes high-quality timed chord annotations produced by human annotators (e.g., music experts, students) or crowdsourced and verified prior to release.

ChoCo also comes with a range of tools for chord parsing and manipulation , as well as a data transformation pipeline (a Smashub instance) to include new chord data sets in ChoCo.

**2.3 How to use ChoCo?**

1 Use JAMS files:After downloading ChoCo, we can read, manipulate and edit harmonic annotations through the jams library

2 Use RDF files:Another option is to work with ChoCo's knowledge graph and use the RDF files in the publish folder; or simply query our SPARQL endpoint. For example, the output of the above Python snippet can be obtained via a SPARQL query to the endpoint (see query below), which returns this output (Michelle's top 10 chords, sorted by start).

**3 JAMS**

JAMS is a standard format for describing music and audio data, used to store and share music metadata and annotations.A JSON annotated music specification that replicates MIR research.

**3.1 What’s JAMS?**

JAMS is a JSON-based music annotation format.

Formal JSON schema for generic annotations

Ability to store multiple annotations per file

Pattern definitions for various annotation types (beat, chord, section, label, etc.)

Error detection and annotation validation

A translation layer that interfaces with mir eval for evaluating annotations

**3.2 WHY?**

Music notes are traditionally provided as plain text files, using simple formatting patterns (comma or tab delimited) whenever possible. However, as the MIR field continues to evolve, such annotations have become increasingly complex, and custom conventions are more often used to represent this information. Therefore, these custom conventions can be cumbersome and non-trivial to parse and use.

JAMS therefore provides a simple, structured and sustainable way to represent rich information in a human-readable, language-agnostic format. Importantly, JAMS supports the following use cases:

Many types of comments

Multiple annotations for a given task

Rich file-level and annotation-level metadata

**3.3 How to process the jams**

Handles music notation, time signatures and JAMS annotations

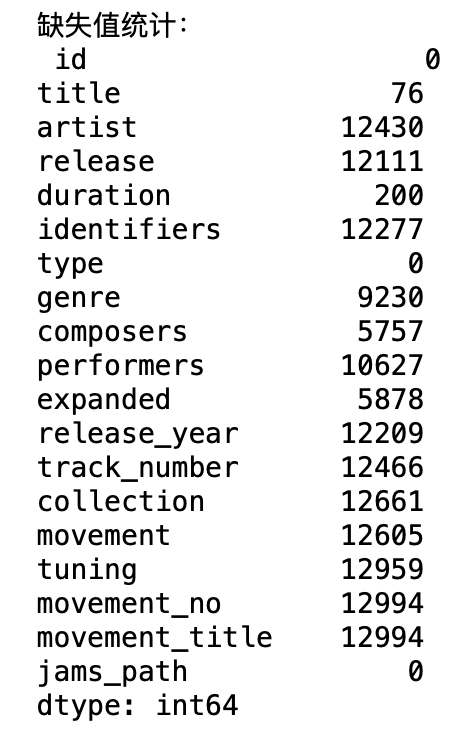
**4 Create datasets**

# **4.1 Exporting ChoCo as a JAMS dataset**

Since there is a large amount of jams data in choco, creating a customized subset from the existing ChoCo music data set allows us to get the data we need faster and analyze the data more accurately. We can choose to include specific partitions while excluding those that are not of interest to meet the needs of the project.

**4.2 Extract metadata information**

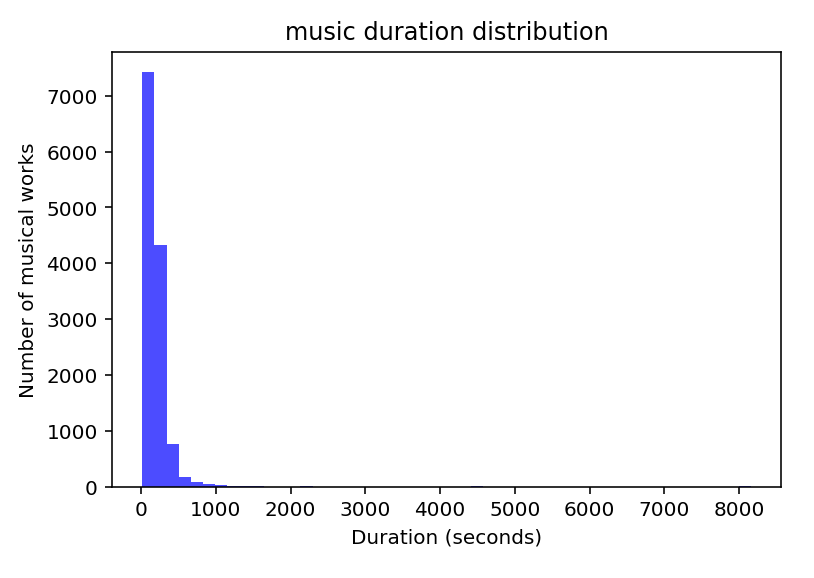
Check for missing values



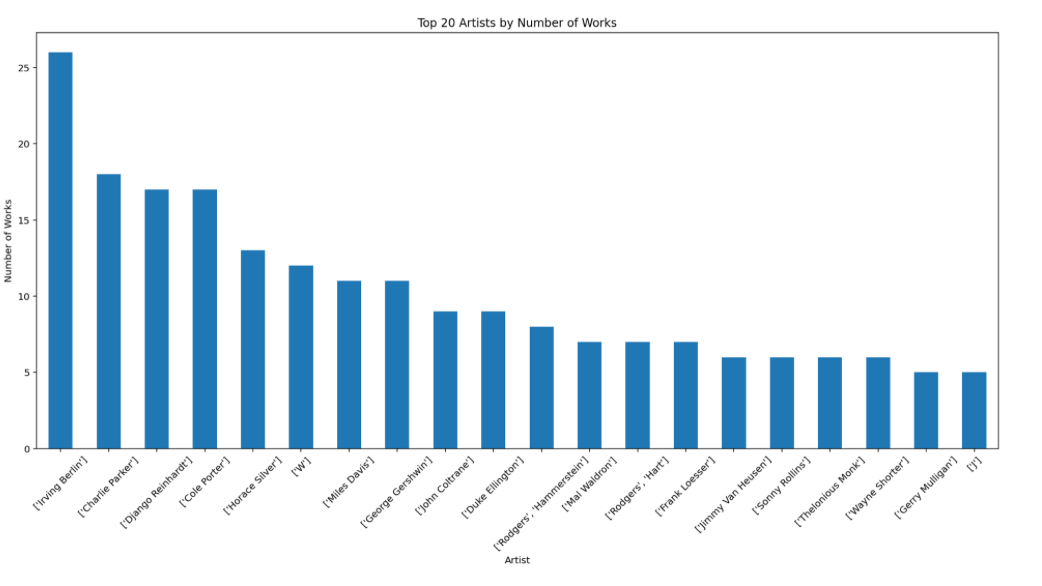
I will use these missing value statistics to decide how to deal with missing values, such as filling missing values, deleting records containing missing values, or performing other data cleaning and processing operations on a case-by-case basis.

**4.3 Data preprocessing**

Visualization can help us better understand and explore music metadata and discover trends, relationships and patterns in the data.



For example, this graph shows the distribution of duration of musical works. It can be seen from the figure that the duration of most musical works is in a short range, and as the duration increases, the number of works decreases sharply. Specifically, the number of musical works with a duration of less than 1,000 seconds (about 16 minutes) is the largest, while the number of works with a duration exceeding this range decreases rapidly. This shows that most musical works are relatively short.



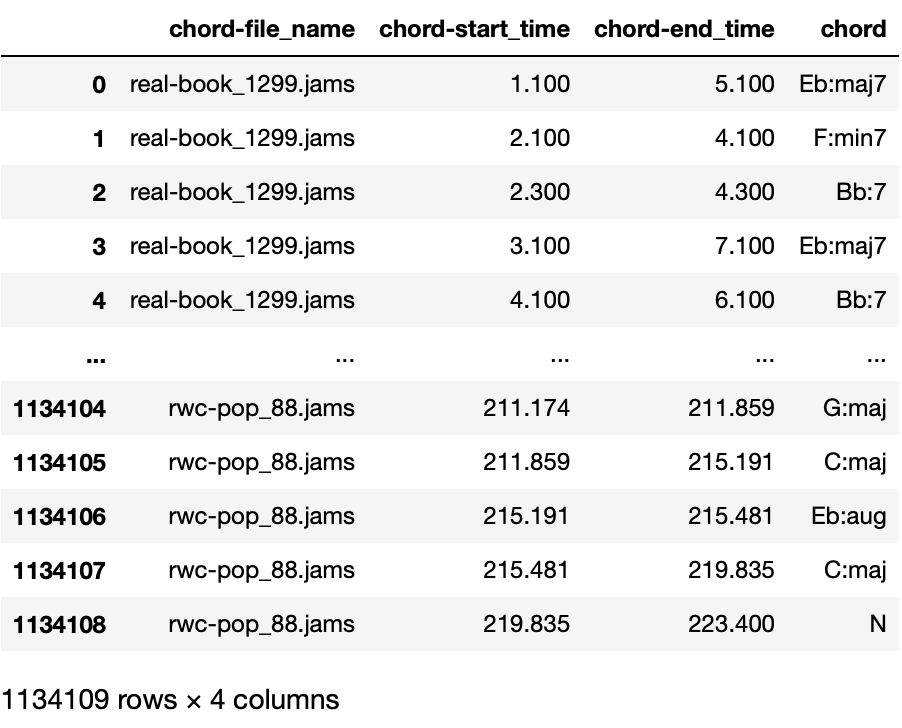
Additionally, this chart shows a bar chart of the top 20 artists ranked by number of works. The height of the bar chart represents the number of works by each artist. It can be seen that the first artist has the largest number of works, close to 25, while the 20th artist has about 5 works. From this distribution, it can be observed that the number of works decreases as the artist ranks lower, which may reflect the productivity of the artist or the popularity of their works. However, since the artists' names are coded, it is not possible to directly identify who they are.

**5 Extract chord and mode annotations**

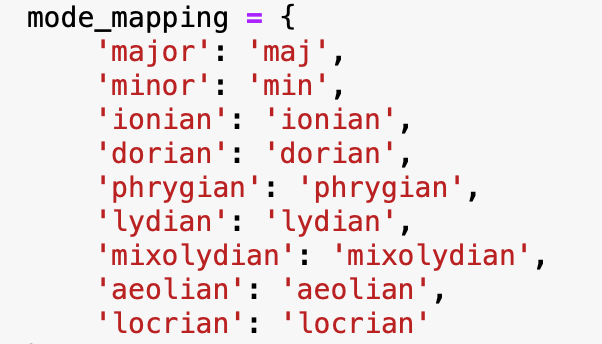
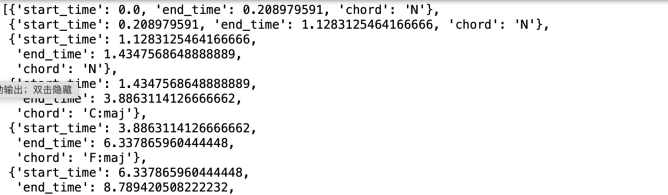
Extracting chord and mode annotations provides insight into the structure and elements of music, providing important information and tools for music research and musical applications. These annotations can be used to analyze, generate, and retrieve music, expanding the scope of applications in the music field.

**5.1 Extract chord information**

Extract the chord sequence and find the chord annotation



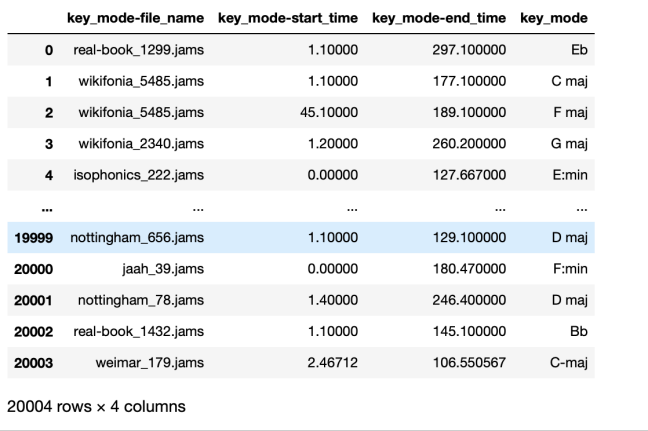
This image shows a data frame containing four columns of information: chord-file\_name, chord-start\_time, chord-end\_time, and chord. The data comes from music files, and each line records the start time, end time and chord type of a chord. For example, the first line shows a chord in the file "real-book\_1299.jams", starting at time 1.100 seconds and ending at 5.100 seconds, and the chord is Eb:maj7. The last row shows that there are 1134109 rows and 4 columns in total.



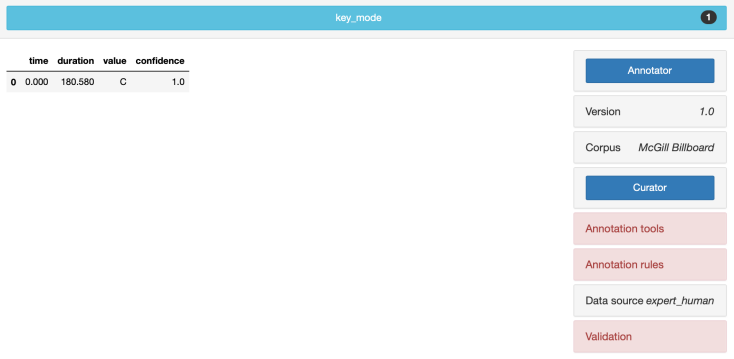
Additionally, it is very important to reformat the mode annotations. Define a map that maps common mode names to the expected abbreviations, splitting the mode string into notes and modes. If the mode part is in the map, convert it. If the mode Not in expected format, returns original string.

**5.2 Extract kymode information**

Similarly, by extracting the mode information, find the mode annotation



This picture shows a data table that contains four columns of data: key\_mode-file\_name, key\_mode-start\_time, key\_mode-end\_time and key\_mode. The data in the table are related to their key and timestamp. For example, the file name real-book\_1299.jams starts at 1.10000 seconds and ends at 297.10000 seconds, and the key is Eb (E flat major). This table has 20004 rows.



This image shows a simple data table and part of a software interface. The data table contains four columns: time, duration, value and confidence. In this example, we can see an entry that starts at time 0.000, has a duration of 180.580 seconds, a value of C (probably referring to the key of C major), and a confidence level of 1.0, which means that the data is accurate very high.

**5.3 Merge chord notes and mode notes into meta**

Add all the columns of the chord annotation and mode annotation files to the end of all the columns of the target file, and save the divided data set to the file for subsequent use. This is for subsequent feature selection and model building.

**6 Data analysis**

Visualize and process statistics of music annotations.This step helps us understand and analyze data to conduct model building and project development more efficiently.

# **6.1 A glimpse of ChoCo**

**Notes on the format of statistics :**

Duration, per score and audio, distplot for each (+ mu and std in text)

Type: proportion of score/audio can be mentioned in text, or table.

Identifiers, mentioned in text -- number, proportion and counts.

Composers and performers -- number, proportion, and counts.

No. of annotations (in total), just mention in text.

**Per annotation namespace (e.g. key, chord in Harte)**

Annotation type: count, not interesting if separated because many keys-chords would follow the same (can avoid).

Annotators, same as before, mention for how many this is available, possibly mention the top 3.

No. of observations (and unique): distplot + reference to mean and std

For keys, we can filter out the global, and report for when more are given (to avoid inflating 1s).

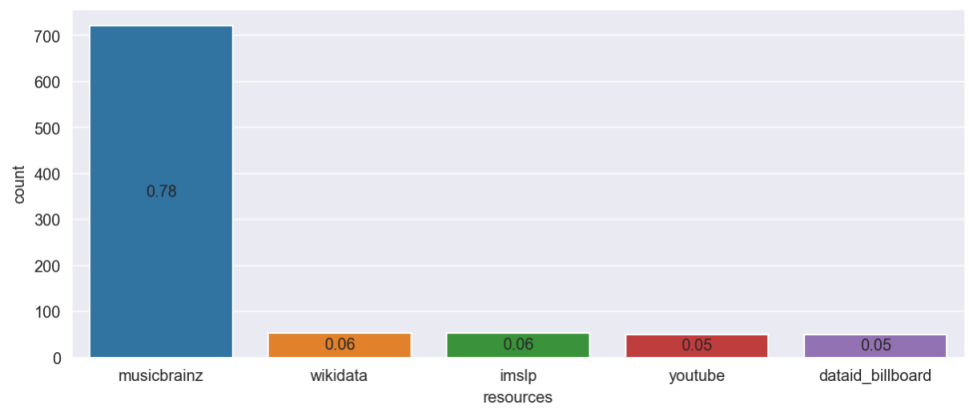
Observation occurrences (and w/o reps), plot histograms of count with relative freqs as plot annotation.

N-grams, same as before. Maybe use table if this is cluttered by labels.

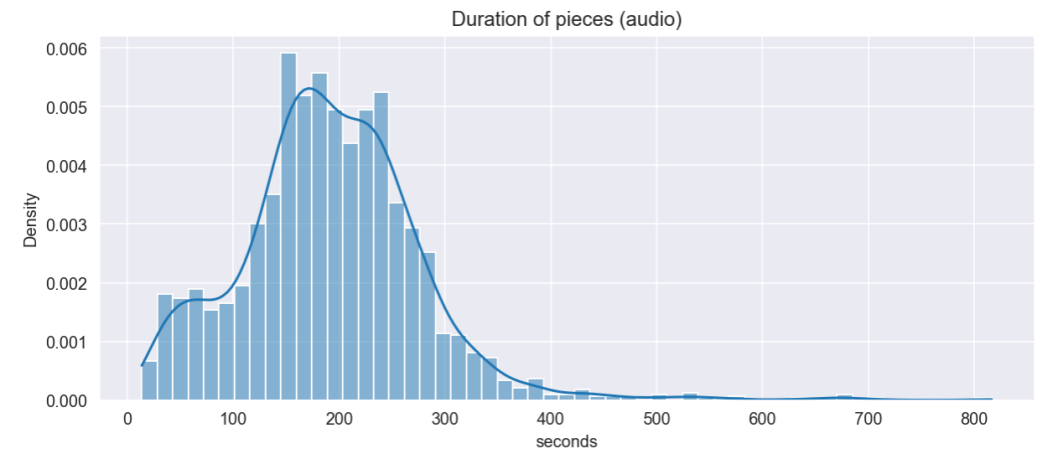
Durations of observations, per audio and score separately, distplot of averages.

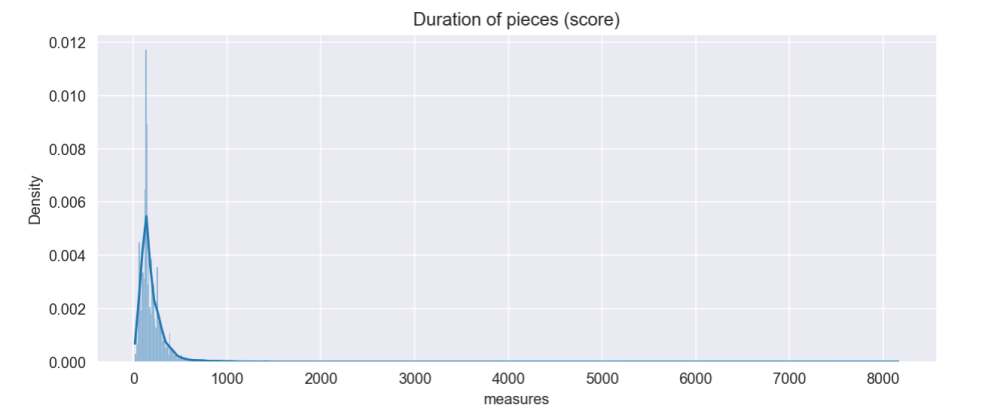
**6.2 Meta-stats**

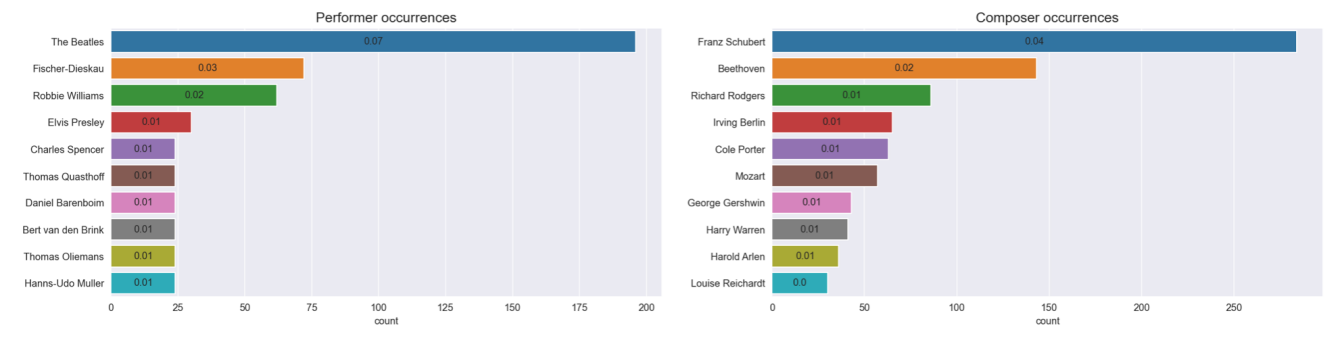
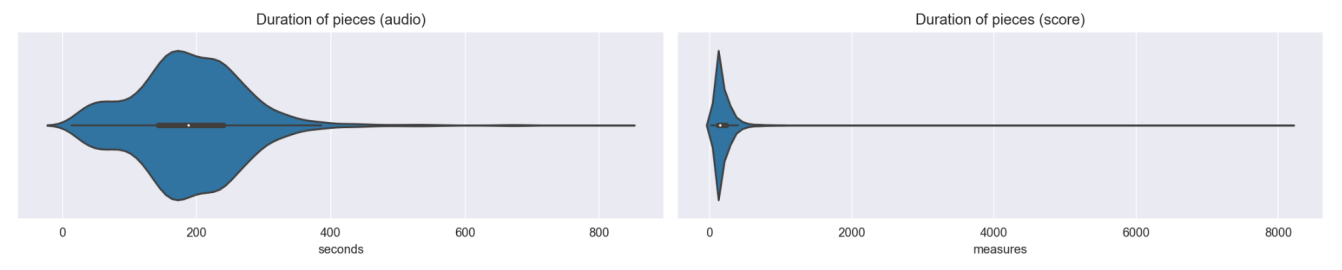
Some key information is visualized, please refer to the code for specific content.



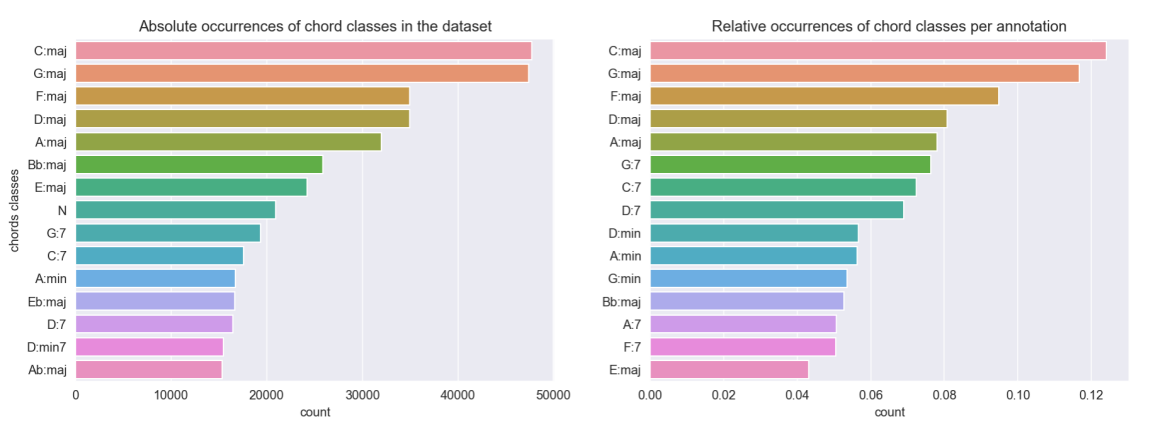
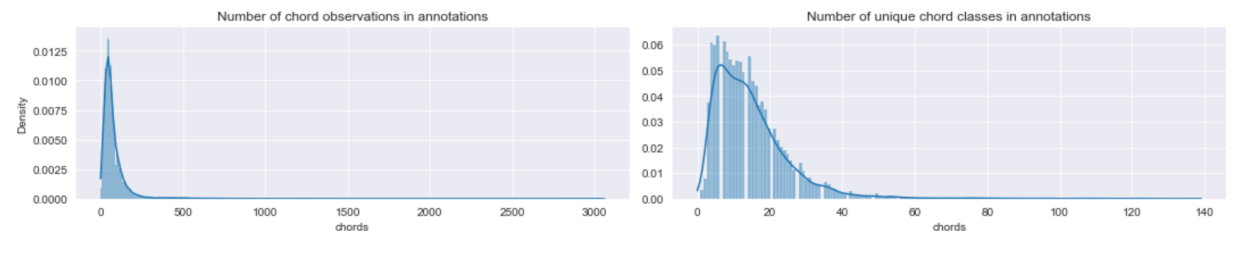
identifiers['sum']







## **6.3 Content-stats**



For other analyses, please see the code.

**7 MODELS**

Now we start modeling!

**7.1 Feature selection**

For tasks such as music similarity, music generation, and music classification, we can select and process features based on the specific needs of the task.

For example: In a **music classification** task, the task involves classifying music into different categories or genres. Key features may include:

1. Metadata: `artist`, `genre`, `composers`, `release\_year`, etc.

2. Music characteristics: such as `duration`, `chord`, `key\_mode`.

For music classification tasks, the goal is to classify music into predefined categories based on its characteristics, such as different genres, emotional states, or artist styles. In this case, your target variable would be the category you wish to predict, such as the genre of music. Features should be those musical attributes that help distinguish different categories.

In the **music similarity** task, the goal is to determine whether two songs are similar. Key features may include:

1. Chord (`chord`) and mode (`key\_mode`): Chord progression and mode are important aspects of musical similarity.

2. Duration (`duration`): Tracks with similar duration may be closer in style.

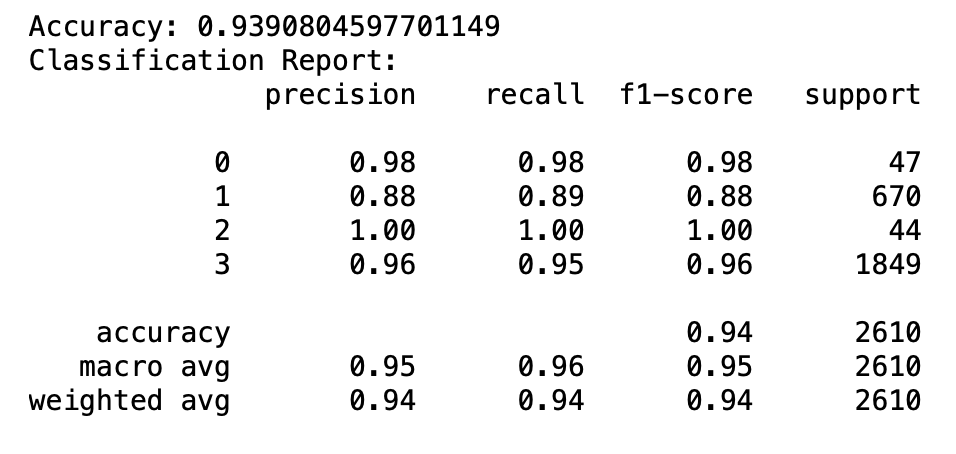
3. Music metadata: such as `genre`, `artist`, `composers`, etc., which can help identify similar styles or genres.

**7.2 Music classification**

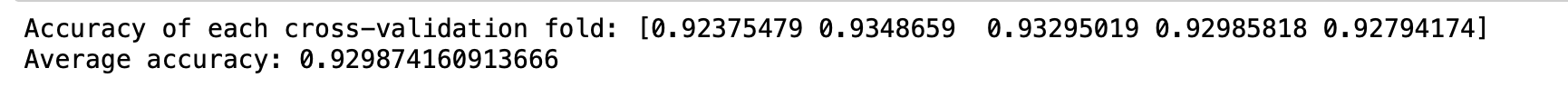
In my classification model, select

'chord', 'key\_mode', 'duration','release\_year', 'composers', 'artist'

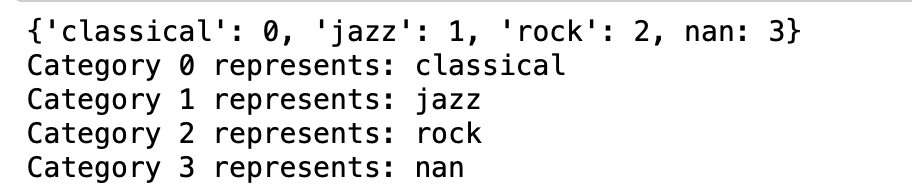
These serve as feature columns, while the target variable is 'genre'. By training a random forest classifier, predictions are made on the dataset, and finally the model performance is evaluated.



To help choose the most appropriate model or algorithm to avoid overfitting or underfitting, I use K-fold cross-validator for cross-validation



Model performance results on the validation set revealed some key observations:



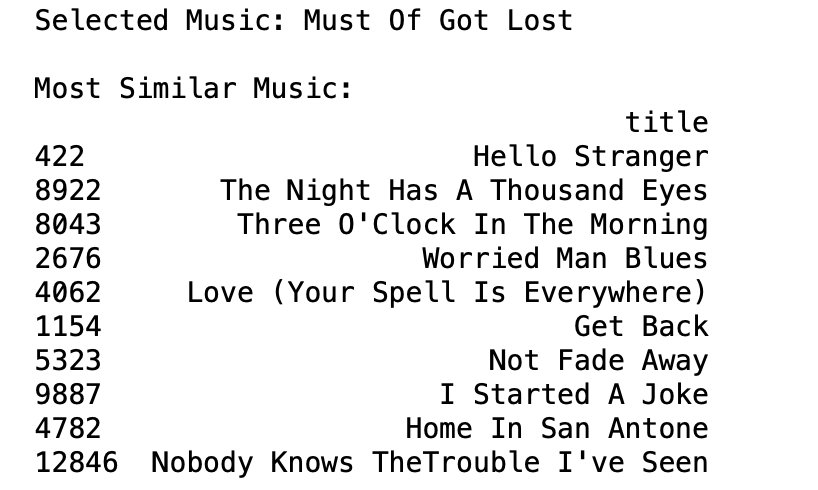
Performance on validation set

**7.3 music similarity**

The goal of music similarity measurement is to determine whether two pieces of music are similar. In this case, there is usually no "target variable" in the traditional sense, since we are not predicting a label or output, but rather comparing the similarity between two instances (in this case, two songs). Therefore, the focus is on selecting features that reflect the characteristics of the music and defining a suitable similarity metric.

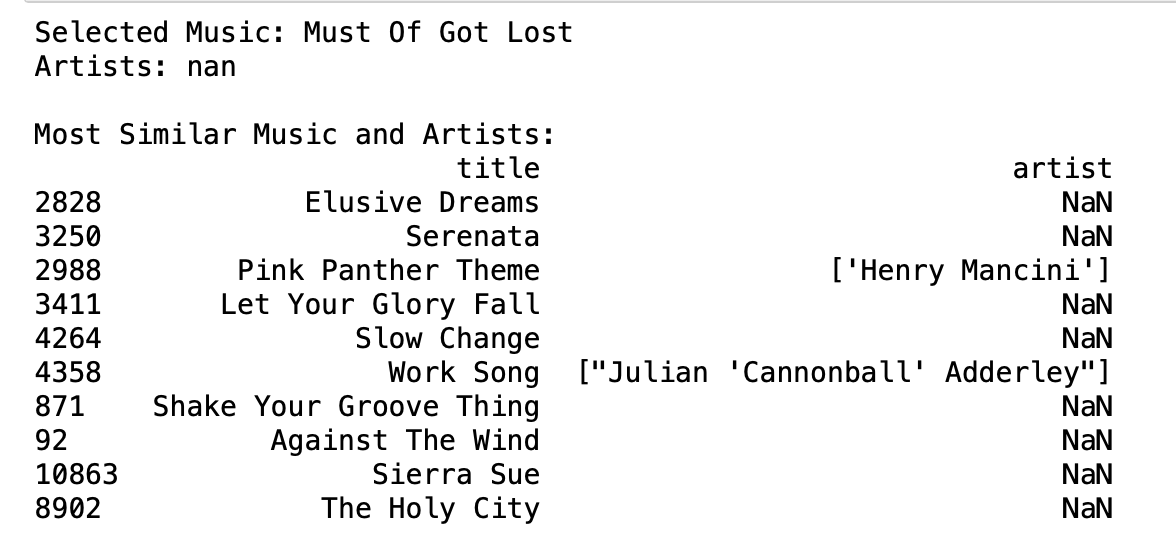
In this task, I first select 'duration', 'chord-start\_time', 'chord-end\_time', 'key\_mode-start\_time', 'key\_mode-end\_time' as features, use cosine similarity to calculate music similarity, and find Music most similar to the selected music

By specifying a piece of music (for example, selecting the first piece of music), find the music most similar to the selected music based on the similarity matrix



For example, I selected the song "Must of Got Lost", calculated the similarity between them through measures , and obtained some of the songs that are most similar to it.

Now, I changed the selection of features: 'duration', 'release', 'genre', 'composers', 'performers' to calculate the most similar 'title' and 'artist'



**8 Conclusions**

This project is about basic research on music embedding models in the semiotic field. It explores how to learn representations from specific dimensions of music (melody, harmony, rhythm) and examines various embedding models (including applications on knowledge graphs). The project uses a musicology testbed to validate the extent to which the learned representations are musically credible and to test embedding models on multiple downstream tasks, including music similarity, music generation, and music classification.

At the same time, it also introduces in detail data acquisition and preparation, data analysis and statistics, model construction and other aspects. For example, it uses the ChoCo and JAMS datasets to analyze and process music data, covering all aspects from extracting and processing music annotations, to feature selection and model evaluation. These contents are of great significance for understanding the field of music information retrieval (MIR) and the application of music embedding technology.