🎵 Genre-Based Chord Progression Classifier

Empowering Musicians with AI-Driven Insights

🌐 GitHub Repository Link

Picture Credit: pianistmagazine.com

INTRODUCTION

Our topic is about classifying music genres based on chord progressions. Chord progressions can be a strong indicator of genre [2], and we aim to create a model that classifies a song's genre using its chords. In the realm of music theory, certain chords played in succession within a song, such as one of the most

famous progressions I-V-VI-IV, are highly characteristic of a pop song for example. The harmony of multiple notes create a chord and form the basis of a song's structure, richness, and depth, all of which are important features in shaping a song's genre.

Literature Review

Prior research has used song lyrics for genre classification [3], while others have explored chord progressions but relied on CNNs and Spotify API data. Wundervald and Zeviani explored this using random forest within Brazilian music and placed heavy emphasis on PCA and the addition/engineering of new features as a comment on future improvements [2]. We plan to focus specifically on chord progressions as our classification metric while extracting integral features for our predictions.

We have identified a dataset with 135,783 songs, offering details such as artist name, genre, lyrics, and chords, which will help us retrieve the necessary data for our model [4]. Additionally, a MIDI dataset will provide deeper insights into musical features like loudness, pitch, and energy for further analysis [5].

METHODS

Data Processing & Feature Engineering

Picture Credit: ak9.picdn.net

Our data processing methods included basic data cleaning of the raw chord progressions, and feature engineering methods like bag of words, harmonic intervals, trigrams, and modular frequencies.

For the **basic data cleaning**, we standardized the song and artist names by removing extra spaces, punctuation, and converting text to lowercase, then processed chord progressions by removing non-chord elements like tab numbers, irrelevant words (e.g., "intro," "verse"), and special characters. This left them in a uniform format, to ease the processing done later.

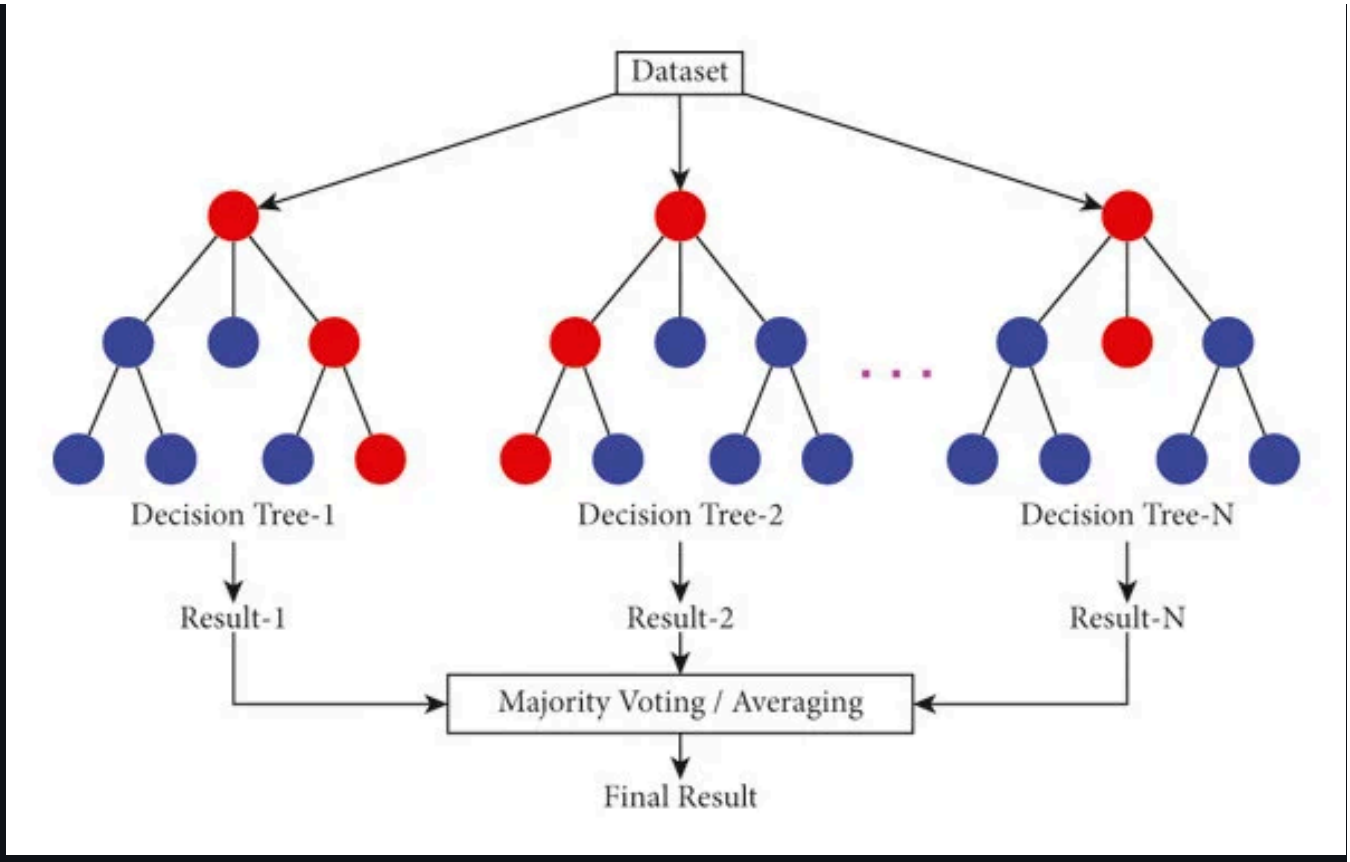
We used **bag of words** to capture the presence of specific chord types across progressions. First, we used a predefined vocabulary of chords, including major, minor, augmented, diminished, and suspended variations, to vectorize each chord progression into a sparse binary matrix (1 if present and 0 if not). This matrix indicates whether each chord in the vocabulary appears in each progression. Then, we grouped equivalent chords (e.g., C# and Db, or C and Cmaj) by summing their binary columns, reducing dimensionality and enhancing interpretability. This added 47 columns of features to the dataset in the end.

To capture the transitions between chords, we encoded sequences of three consecutive chords as **trigrams**. This helped us identify patterns in songs with similar chord progressions. We extracted these sets of three chords from all the chords in the songs and then these trigrams were transformed into feature vectors, resulting in a high-dimensional sparse matrix. To reduce dimensionality, we applied PCA with 50 components to the trigrams matrix in batches to handle memory constraints.

We extracted **harmonic intervals** from the chord progressions by encoding the root note of each chord as a semitone value, which represented its pitch position within a 12-tone scale. We then calculated the intervals (in semitones) between consecutive chords in a progression. The resulting intervals were pooled to obtain summary statistics such as mean, maximum, and minimum interval values, which captured the general "movement" or shifts in pitch in a numeric fashion.

Lastly, we extracted the **modulation frequency** of chord progressions, which is a measure of how often the "key" or dominant root note changes in a song. First, we extracted root notes from each chord in a progression, isolating the primary pitch for each chord symbol. Then, we determined the most frequently occurring root note, or "dominant root," for each section of the progression. As the dominant root changed between sections, we counted each change as a modulation, representing a shift in musical key. The final modulation frequency for each progression was stored as a feature for each song.

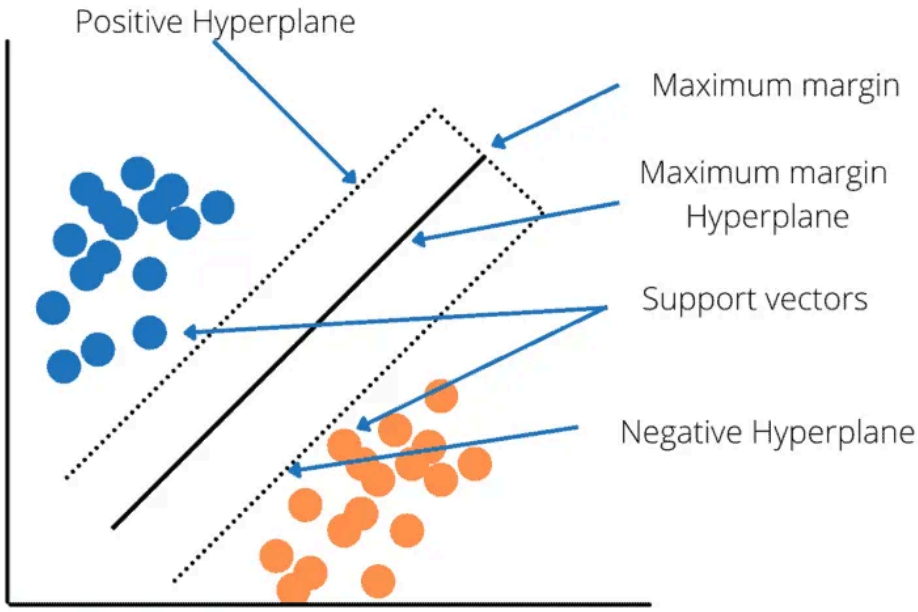
Random Forest



Picture Credit: mlarchive.com

Using our final feature set, the first machine learning algorithm we implemented was a random forest. Musical features have many complex relationships that would go into predicting a specific genre. The ensemble of decision trees provided by random forest allows it to handle that complexity and form non-linear relationships between features. We decided to utilize this model also because of its ability to prevent overfitting. Overfitting is a large concern especially when dealing with a vast amount of intricate features and data points. Random forest averages out individual tree predictions which can significantly reduce overfitting, thus helping with the efficacy of our model. Another important reason for using Random forest is its ability to provide the strength of each feature when making a decision, allowing us to easily identify patterns and features to look out for [6].

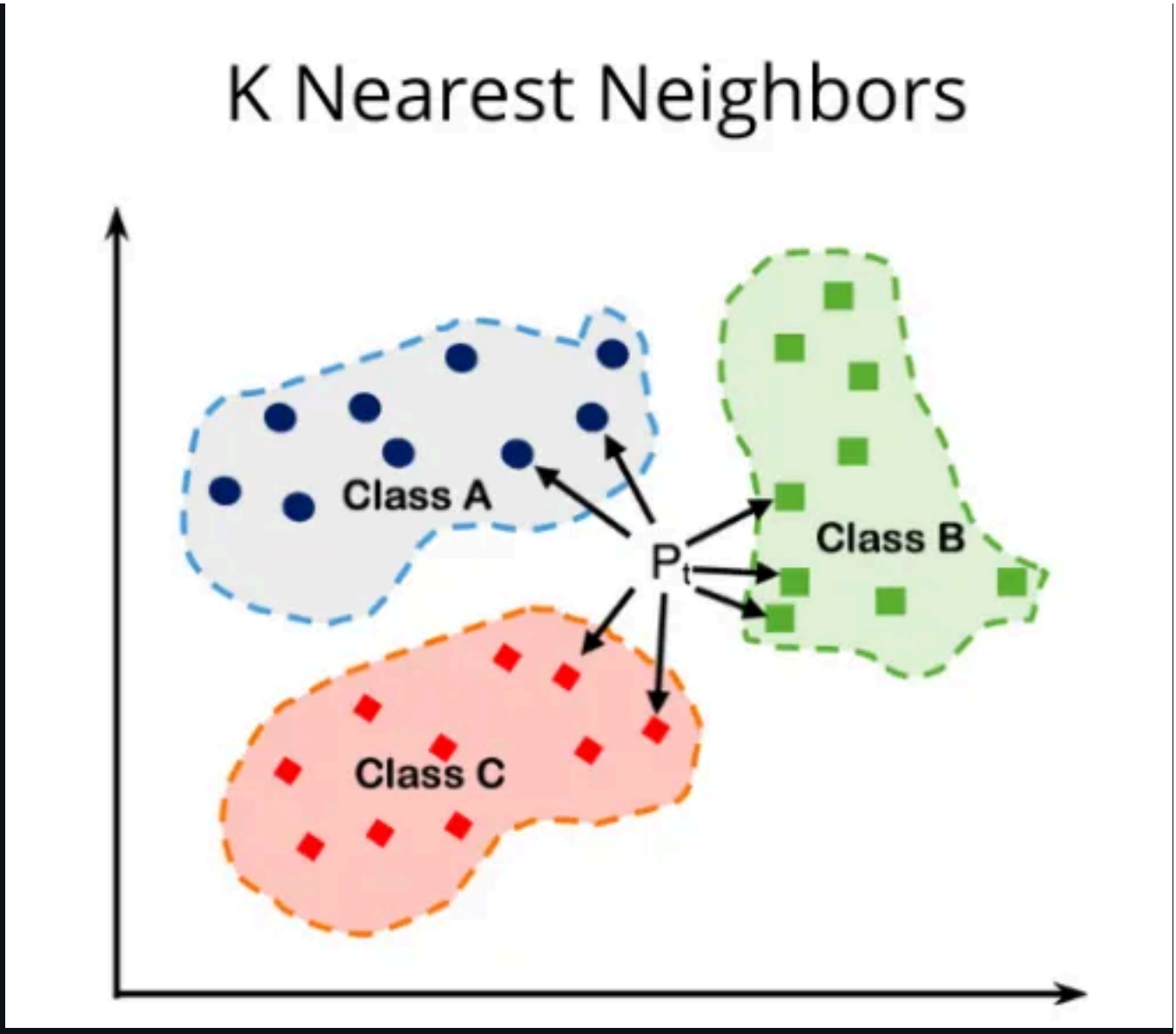
Support Vector Machine



Picture Credit: hands-on.cloud

The second machine learning algorithm we decided to implement is a Support Vector Machine (SVM), once again using our final feature set. Some of the processing we did to take the chord progressions and turn it into features resulted in a rather large feature matrix (currently at about 101 features) and this was after using the PCA algorithm on a couple of the heftier features. Because of this larger feature set, we needed an algorithm that would be able to handle this and SVM is thus a great choice. SVM is able to handle all these features (and large data sets in general) on lower computational power due to its memory efficiency [7]. This memory efficiency comes from the model using support vectors that are formed from a subset of the input data and used in the decision function. This algorithm is also useful due to the kernel functions we can modify: similar to the random forest, if the relationship between our features and the genre classifications is non-linear, the kernel can transform the data into a hyperplane, allowing for more accurate classification and maintaining a linear divide in the data. We chose to use the RBF kernel for our training due to the non-linearity nature of our data.

K-Nearest Neighbors



Picture Credit: researchgate.net

The third machine learning algorithm we decided to implement is a K-Nearest-Neighbors algorithm, using the final feature set like our other algorithms. KNN fits our problem because it doesn't assume any type of linear relationship between the data and the output, allowing for flexibility within the processing, be it linear or complex patterns. KNN also inherently deals with multiclass classification, something we clearly need to distinguish between multiple different music genres. Our dataset also implies some similarity between the pop and rock chord progressions, with fuzzy differences to the naked eye. However, KNN can adapt to these and deal with complicated decision boundaries. It does this by using proximity metrics. It takes the closest neighbors in the feature space and gives more weight to those data points to accurately classify and cluster the data even if there's overlap. All these reasons make KNN an effective choice for our specific problem.

RESULTS AND DISCUSSION

Random Forest Results

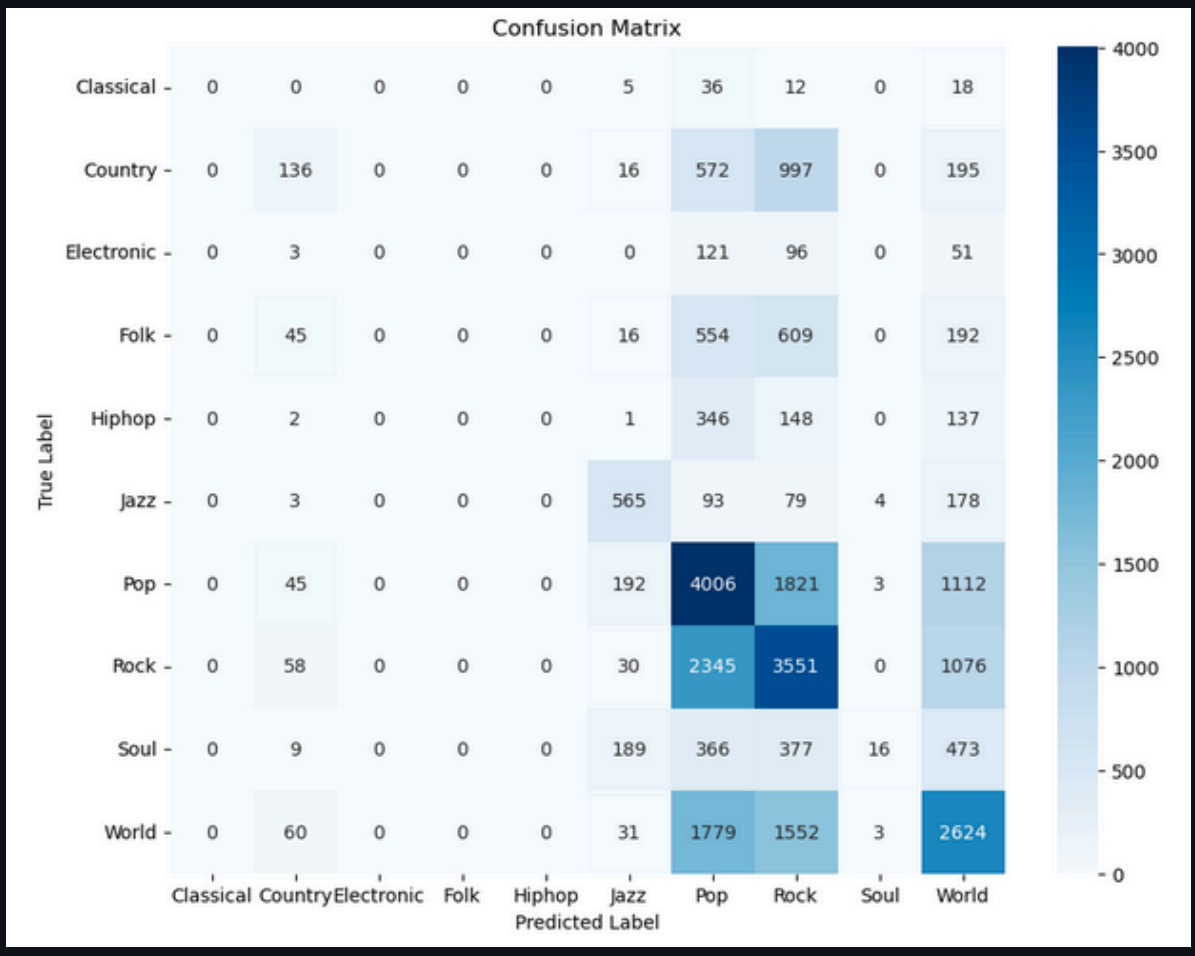


Figure 1: Confusion Matrix for Random Forest

Strong prediction for overrepresented classes:

The model performed the best in recognizing Pop, Rock, and World which dominate the dataset. These genres have the highest number of correct predictions on the diagonal of the confusion matrix, where the predicted class matches the true class:

- Pop: 3,823 samples were correctly classified.
- Rock: 3,410 samples were correctly classified.
- World: 2,977 samples were correctly classified.

This would indicate that the model has learned to recognize features associated with these majority genres. However the matrix also shows strong genre confusion.

Genre Confusion for overrepresented classes:

Despite their high number of correct predictions, Pop, Rock, and World also had high misclassification rates, particularly as each other. In the confusion matrix, the shading highlights where Rock was confused with Pop and World with both Pop and Rock. This suggests that, even with more data, the model struggles to differentiate these genres, likely due to overlapping features. The lack of distinction between these

genres, despite their high representation, indicates similar or insufficiently unique features that lead to frequent misclassifications.

	Genre	Precision	Recall	F1-Score	Support
0	Classical	1.0000	0.0100	0.0300	67
1	Country	0.3800	0.1800	0.2500	2035
2	Electronic	1.0000	0.0000	0.0100	255
3	Folk	0.2900	0.0000	0.0100	1512
4	Hiphop	0.7000	0.0200	0.0400	672
5	Jazz	0.5400	0.5300	0.5400	896
6	Pop	0.4000	0.5300	0.4500	7223
7	Rock	0.4000	0.4800	0.4400	7035
8	Soul	0.3400	0.0300	0.0600	1408
9	World	0.4300	0.4900	0.4600	6040

Figure 2: Qualitative Metrics for Random Forest Model

This is further represented by the qualitative metrics (Shown in Figure 2)

- Precision
- Recall
- F1-Score
- Support

The F1-score, which balances precision and recall, is essential for evaluating this imbalanced dataset, as it highlights the model's effectiveness across both frequent and rare genres. Higher F1-scores for Pop (0.45), Rock (0.44), and World (0.46) indicate that the model is more effective at predicting these classes, likely due to their overrepresentation. This allows the model to learn nuanced patterns, leading to higher confidence in predictions.

In contrast, underrepresented genres like Classical, Electronic, and Folk have extremely low F1-scores (0.03, 0.01, and 0.01, respectively), showing the model's struggle with these classes. Low recall for these genres, such as Classical and Electronic (0.01 and 0.00), suggests the model rarely identifies these classes, reflecting an inability to generalize to minority classes. High precision but near-zero recall for these genres implies the model predicts them correctly when it does, but rarely makes those predictions, impacting the F1-scores.

SVM Results

	Genre	Precision	Recall	F1-Score	Support
0	Classical	0.0000	0.0000	0.0000	71
1	Country	0.3800	0.0700	0.1200	1916
2	Electronic	0.0000	0.0000	0.0000	271
3	Folk	0.0000	0.0000	0.0000	1416
4	Hiphop	0.0000	0.0000	0.0000	634
5	Jazz	0.5400	0.6100	0.5700	922
6	Pop	0.3900	0.5600	0.4600	7179
7	Rock	0.3800	0.5000	0.4400	7060
8	Soul	0.6200	0.0100	0.0200	1430
9	World	0.4300	0.4300	0.4300	6049

Figure 3: Qualitative Metrics for SVM Model

The SVM model demonstrates inconsistent performance across genres, with Jazz being the strongest performer (f1-score: 0.57) and complete failure in classifying Classical, Electronic, Folk, and Hiphop (all metrics at 0.00). Pop and Rock show moderate performance (f1-scores of 0.46 and 0.44 respectively), benefiting from large sample sizes (7179 and 7060 samples).

Despite high precision in some cases, like Soul (0.62), extremely low recall values lead to poor overall performance, suggesting the model struggles with consistent genre identification. The significant class imbalance in the dataset, ranging from 71 samples (Classical) to 7179 samples (Pop), likely contributes to these varying results.

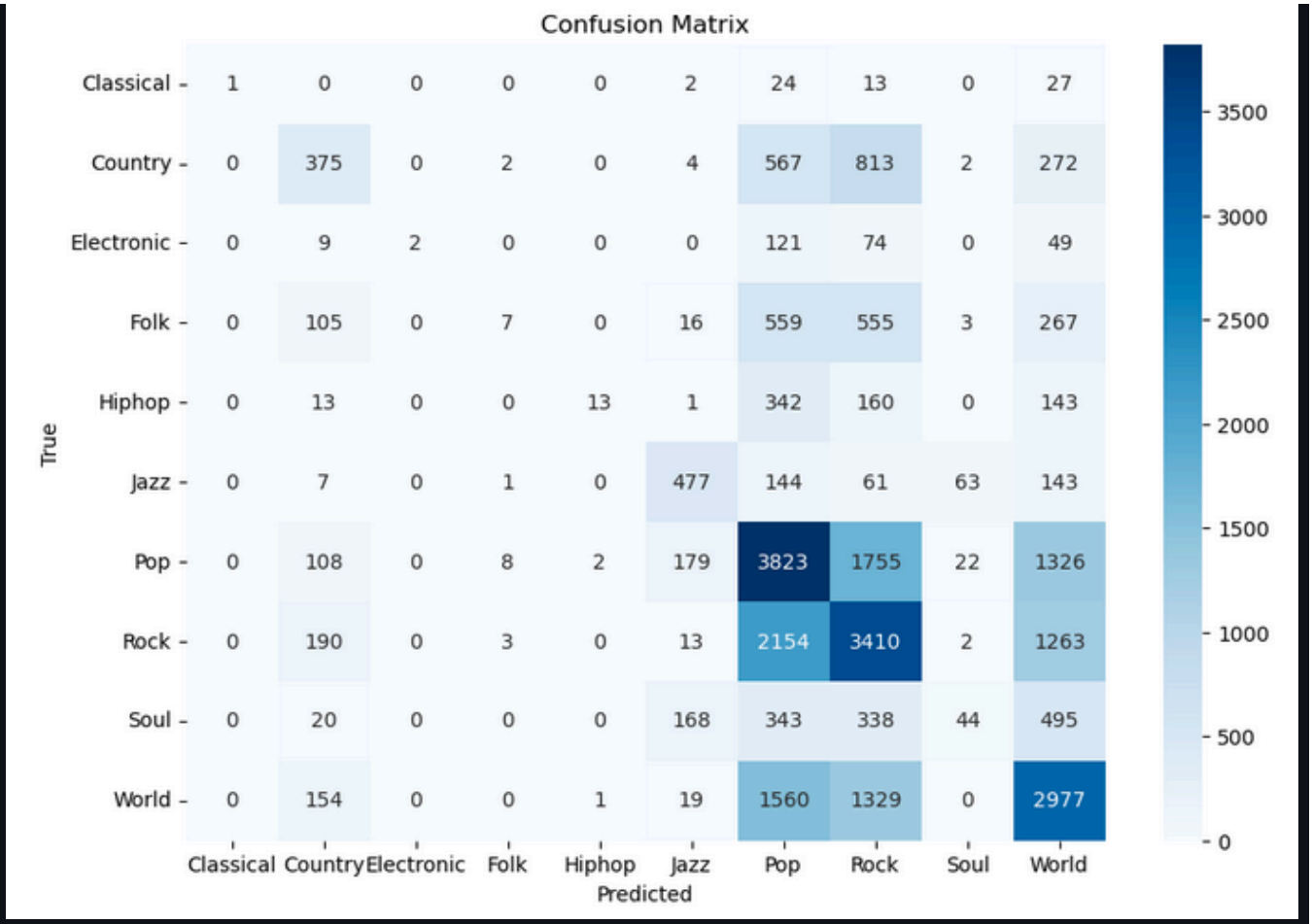


Figure 4: Confusion Matrix for SVM

The confusion matrix reveals significant classification challenges in the model's performance. Pop and Rock show the highest number of correct predictions (4006 and 3551 respectively), but there's substantial confusion between these two genres, with 1821 Pop songs misclassified as Rock and 2345 Rock songs misclassified as Pop. World music also shows a decent number of correct predictions (2624), but experiences considerable confusion with Pop and Rock genres. Most notably, Classical, Electronic, Folk, and Hiphop genres show extremely poor performance with almost no correct predictions, with their samples being predominantly misclassified as either Pop or Rock. This pattern suggests the model has a strong bias toward the two dominant classes (Pop and Rock), likely due to their larger representation in the training data.

KNN Results

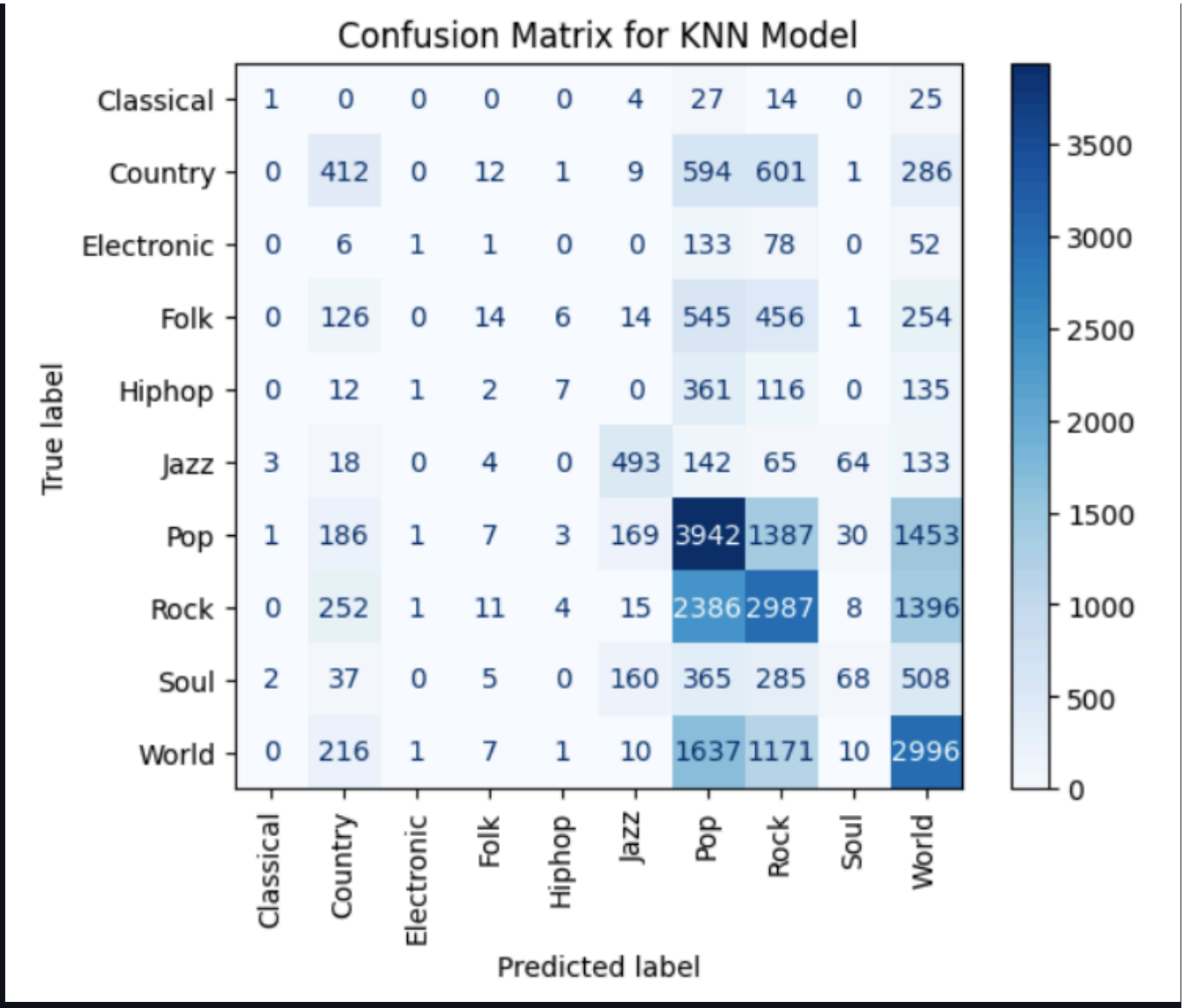


Figure 5: Confusion Matrix for KNN

The KNN model performed the best in recognizing Jazz, Pop, Rock, and World, which were the most occurring in the dataset. These genres have the highest number of correct predictions as shown in the confusion matrix, where the predicted class matches the true class. In terms of correct classifications, Pop had 3,942, Rock had 2,987, and World had 2,996. This shows that the model learned to recognize features associated these genres the best. Unfortunately, the matrix also shows a lot of confusion between the genres, as the shading around misclassifications surrounding those genres is darker.

Pop, Rock, and World genres were often misclassified for one another. In the confusion matrix, the shading highlights where Rock was confused with Pop and vice versa and World was confused with both Pop and Rock. This implies that the model struggles to differentiate these genres, most likely due to overlapping features, since the genres use similar chord progressions and styles. The lack of distinction between these genres, despite their high representation, likely lead to features that were also non-distinct, making it harder for the model to differentiate them.

	Genre	Precision	Recall	F1-Score	Support
0	Classical	0.1400	0.0100	0.0300	71
1	Country	0.3300	0.2200	0.2600	1916

	Genre	Precision	Recall	F1-Score	Support
2	Electronic	0.2000	0.0000	0.0100	271
3	Folk	0.2200	0.0100	0.0200	1416
4	Hiphop	0.3200	0.0100	0.0200	634
5	Jazz	0.5600	0.5300	0.5500	922
6	Pop	0.3900	0.5500	0.4600	7179
7	Rock	0.4200	0.4200	0.4200	7060
8	Soul	0.3700	0.0500	0.0800	1430
9	World	0.4100	0.5000	0.4500	6049

Figure 6: Qualitative Metrics for KNN Model

The F1-score is a good representation of both the precision and recall the model has for each genre. Jazz(0.55), Pop (0.46), World (0.45), and Rock (0.42) all had higher F1-scores which suggests that the model is more effective at predicting these classes, likely due to their overrepresentation and also Jazz's distinct nature musically. This allowed the model to learn more subtle pattern differences, leading to higher confidence in predictions.

When looking at genres like Soul, Classical, Folk, and Electronic we see extremely low F1-scores (0.08, 0.03, 0.02, and 0.01, respectively), showing the model's struggle with these classes likely due to how underrepresented they are in the dataset. Low recall for these genres, such as Classical, Folk, Hip-Hop, and Electronic (0.01, 0.01, 0.01, and 0.00), also highlights that the model rarely identifies these classes, exposing the model's inability to notice clear patterns for minority genres. There was also relatively high precision but near-zero recall for these genres which reveals that the model predicts them fairly correctly when it does, but rarely makes those predictions due to their sparseness in the dataset, impacting the F1-scores. It's clear the KNN model does better with the genres we had more to train on, and still requires some more advanced feature engineering to distinguish between them.

Looking at All Models by F1 Scores

	Genre	KNN	Random Forest	SVM
0	Classical	0.0300	0.0300	0.0000
1	Country	0.2600	0.2600	0.1200
2	Electronic	0.0100	0.0100	0.0000
3	Folk	0.0200	0.0200	0.0000
4	Hip-Hop	0.0200	0.0200	0.0000
5	Jazz	0.5500	0.5500	0.5700

	Genre	KNN	Random Forest	SVM
6	Pop	0.4600	0.4600	0.4600
7	Rock	0.4200	0.4200	0.4400
8	Soul	0.0800	0.0800	0.0200
9	World	0.4500	0.4500	0.4300

Figure 7: F1 Scores for Three Models Trained

This project evaluated three machine learning models—K-Nearest Neighbors (KNN), Random Forest, and Support Vector Machine (SVM)—for classifying musical genres based on chord progressions. KNN performed best, achieving higher F1-scores for genres like Jazz, Pop, Rock, and World due to its adaptability to complex, non-linear feature distributions. However, it struggled with underrepresented genres like Classical and Electronic, reflecting its sensitivity to class imbalances. Random Forest, while effective for majority genres and strong in feature interpretation, suffered from significant genre confusion, particularly between overlapping genres such as Pop and Rock. SVM, despite its computational efficiency with an RBF kernel, performed poorly overall, especially for minority genres, due to its inability to handle class imbalances and subtle feature variations.

Confusion matrices revealed consistent difficulties in distinguishing between Pop, Rock, and World genres across all models. Genre confusion was most pronounced in Random Forest, while KNN showed better adaptability but remained constrained by dataset imbalances. SVM's reliance on sparse data representations resulted in near-zero recall for underrepresented genres. Overall, KNN proved the strongest for majority genres, though its limitations, along with those of the other models, underscore the need for enhanced data balancing and feature engineering. Future improvements could include synthetic data generation, exploring additional harmonic and rhythmic features, and integrating deep learning models like LSTMs to better capture sequential patterns in chord progressions.

Further Analysis

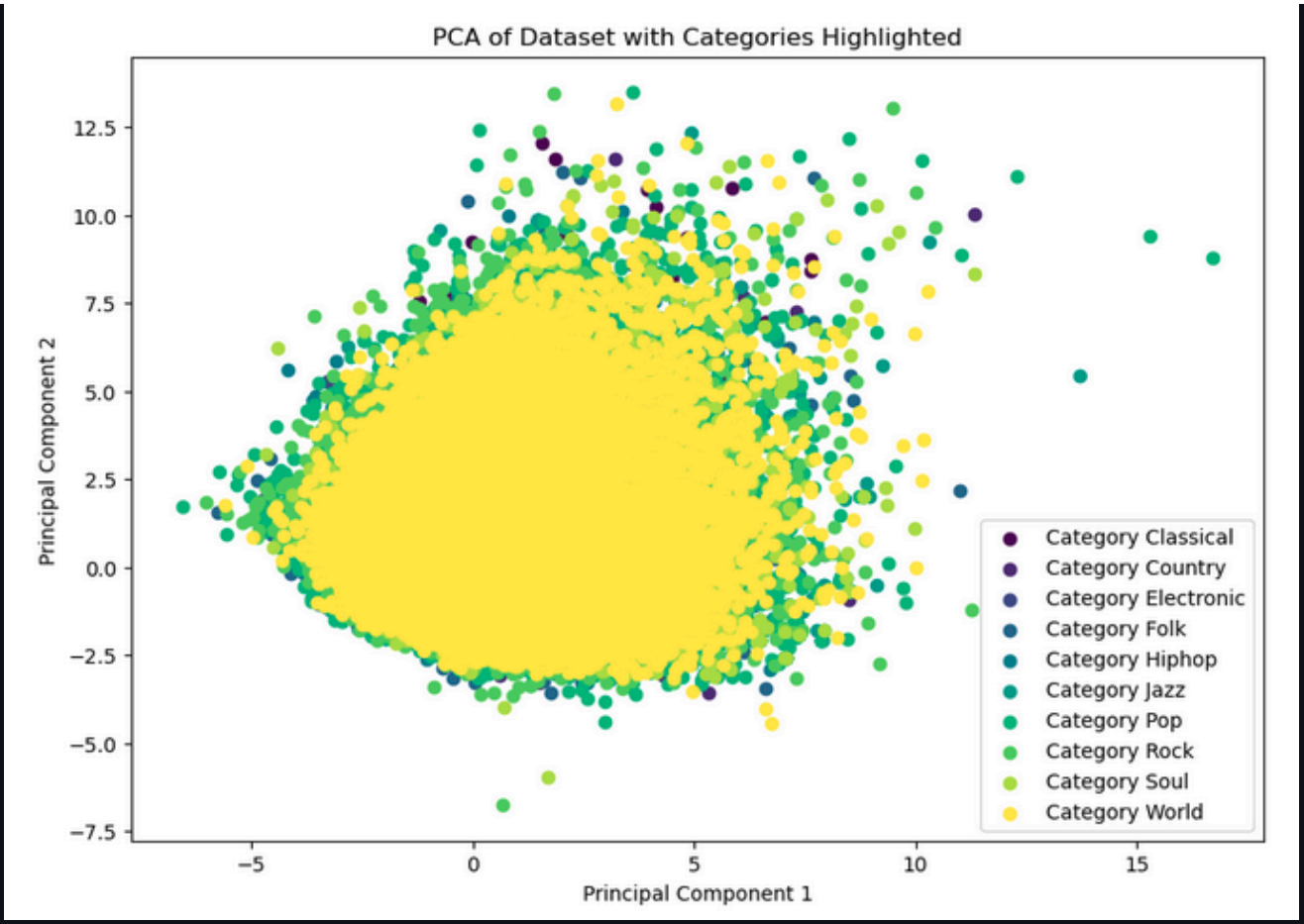


Figure 8: 2D PCA Plot of Our Features

We performed Principal Component Analysis (PCA) on the dataset to examine the variability and potential genre clusters based on our features. The results indicate a lack of variance in the principal components, suggesting that the features do not effectively distinguish genres.

We experimented with several techniques to improve model performance:

- Class Weights: Adjusted to give more importance to underrepresented genres
- Subset Sampling: Created balanced subsets by sampling equal numbers for each genre
- SMOTE: Applied to synthetically balance the dataset

Despite these efforts, each approach yielded an accuracy around 40%, with SMOTE actually lowering accuracy further.

FUTURE WORK

Although we were able to do some work on engineering the features of the engineering model, for the timing of this project we were not able to fully implement the chord embedding model. In the future, we would like to delve into how this different representation of chords will impact the accuracy of the 3 models for classifying genres.

Chord Embeddings as Language

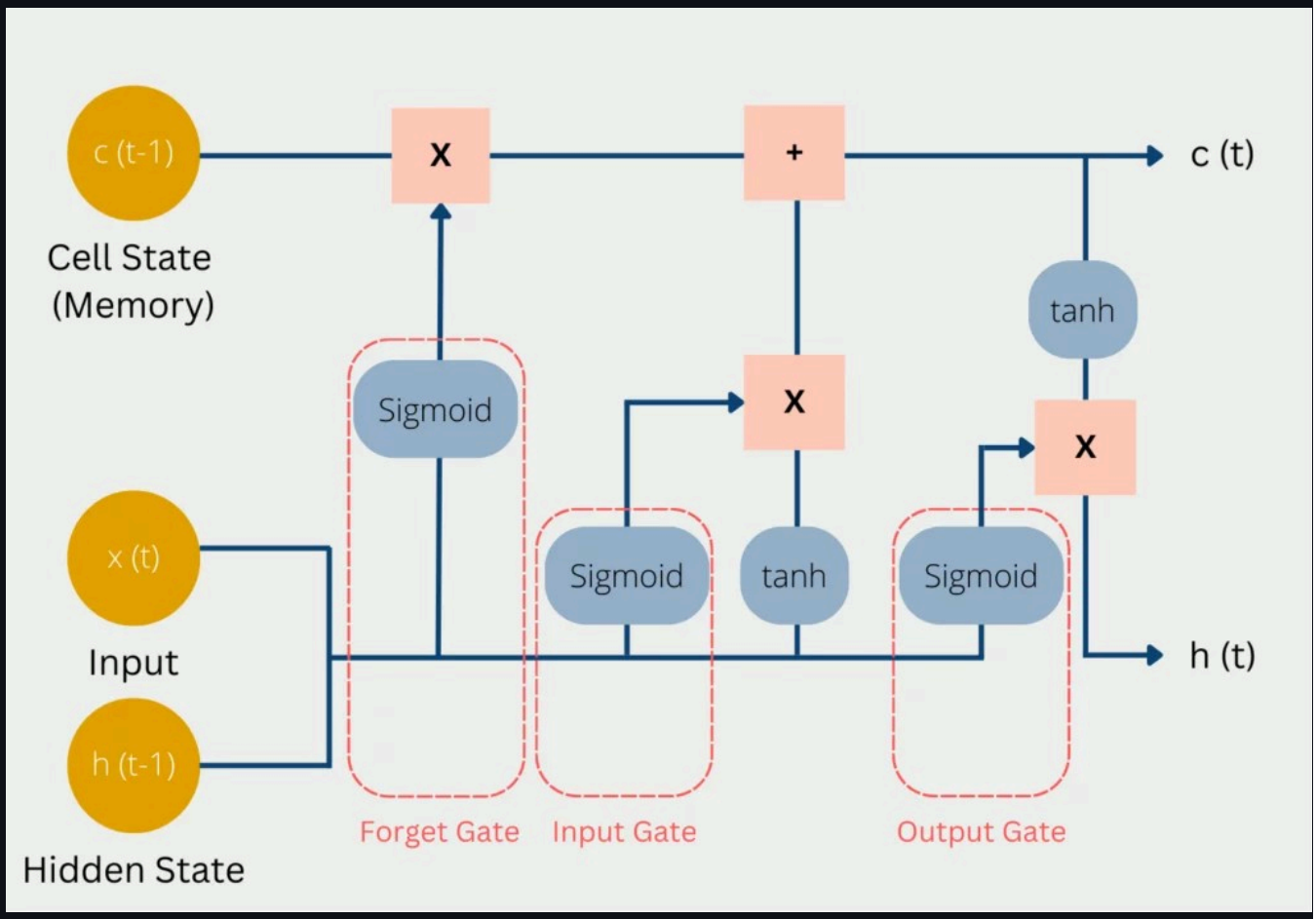


Figure 9: LSTM Architecture that could be used to train for genre classification

Chord embeddings can be treated like a language due to their sequential structure, similar to how words form sentences. Transformer models, such as those used in symbolic music generation, can be used for processing sequential data like chord progressions. By leveraging self-attention mechanisms, these models can capture long-term dependencies and context within sequences. For example, pretrained transformer models like BERT or GPT can be fine-tuned on chord sequences to learn relationships specific to musical genres, building on techniques shown to enhance symbolic music processing [8][9].

For the sake of simplicity, models like Word2Vec or FastText could create embeddings based on harmonic relationships, although these may lack the contextual depth of transformers. Alternatively, recurrent neural networks (RNNs) like LSTMs or GRUs can account for temporal dependencies in chord sequences, capturing the influence of earlier chords on later ones [9].

Chord Embeddings as Graphs

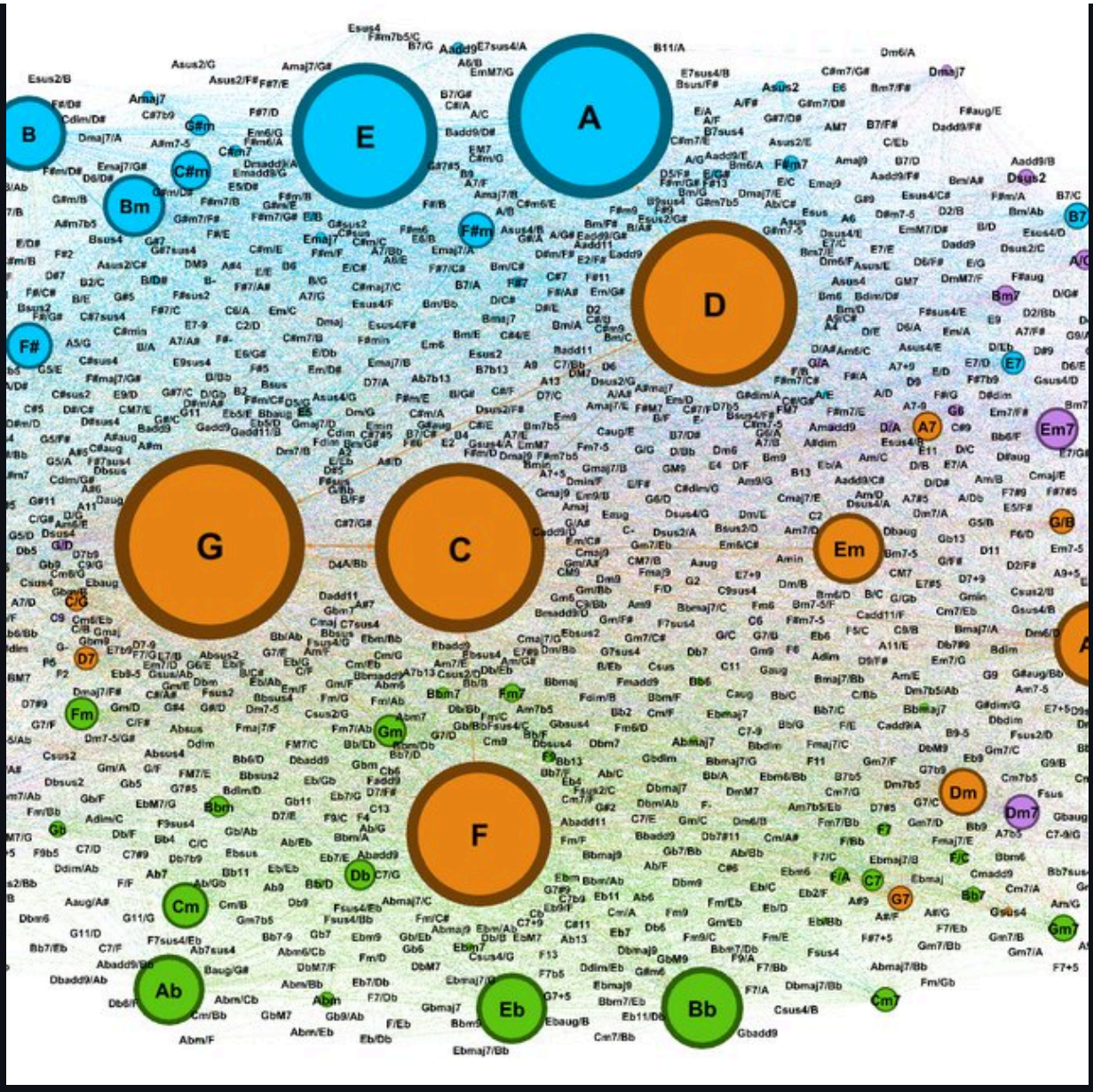


Figure 10: A possible diagram of treating chord as graphs

Another approach would involve representing chord embeddings as graphs, where nodes represent unique chords and edges represent transitions between chords, weighted by their frequency or context. Graph-based approaches have been successfully used in tasks like melody-to-chord harmonization, where structural patterns in music are important. For example, integrating Graph Neural Networks (GNNs) to learn chord relationships could help model genre-specific motifs such as secondary dominants in jazz or cyclic structures in pop[9]. GNNs can be enhanced by incorporating additional features, such as edge Directionality (capturing the difference between A → D major and D → A major transitions) and embedding nodes with harmonic intervals and frequency distributions.

Graph-based embeddings, combined with sequence models like transformers, could offer a hybrid solution to address both structural and temporal aspects of chord progressions, making this a promising area for further research [8][9].

REFERENCES

[1] A. Lahnala, et al., "Chord embeddings: Analyzing what they capture and their role for next chord prediction and artist attribute prediction," in Artificial Intelligence in Music, Sound, Art and Design: 10th International Conference, EvoMUSART 2021.

[2] B. Wundervald, W. Zeviani, "Machine learning and chord based feature engineering for genre prediction in popular Brazilian music," arXiv preprint arXiv:1902.03283, 2019.

[3] M. Leszczynski, A. Boonyanit, and A. Dahl, "Music Genre Classification using Song Lyrics Stanford CS224N Custom Project."

[4] Chords and Lyrics Dataset, Kaggle.

[5] Lakh MIDI Dataset, Kaggle.

[6] AIML.com. (2024, May 22). What are the advantages and disadvantages of Random Forest?

[7] Scikit-Learn Developers. (n.d.). Support vector machines. Scikit-Learn.

[8] N. Zhang, "Learning Adversarial Transformer for Symbolic Music Generation," in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 4, pp. 1754-1763, April 2023, doi: 10.1109/TNNLS.2020.2990746.

[9] S. Rhyu, H. Choi, S. Kim and K. Lee, "Translating Melody to Chord: Structured and Flexible Harmonization of Melody With Transformer," in IEEE Access, vol. 10, pp. 28261-28273, 2022, doi: 10.1109/ACCESS.2022.3155467

Picture Credit: images.coolwallpapers.me

PROJECT MANAGEMENT

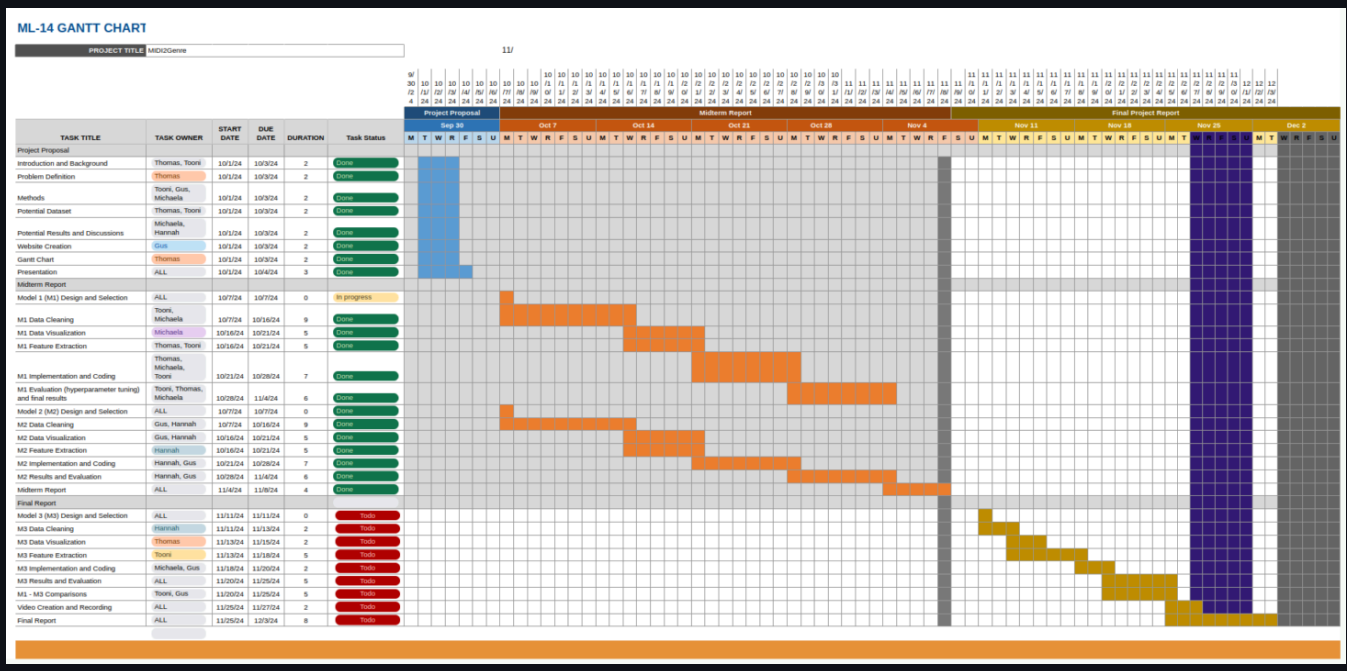


Figure 11: Gantt Chart used for this project

	Name	Contribution
0	Thomas	Gantt Chart, Data preprocessing/feature engineering, Github file directory, Video
1	Hannah	Methods - KNN, Video
2	Kasikrit	Website Creation using stremlit, Comparison between all the models
3	Michaelah	Future work
4	Tooni	Data Processing Methods KNN, Video

Figure 12: Contribution Table of the Team