1 - The Classification Problem

For this project, I attempt to train a model which can identify chord progressions which contain chords entirely within one major key vs. chord progressions which also contain secondary dominants.

What is the problem?

First, I will define a few terms which are important to understanding the problem

Note:

In classical western music theory, there exist twelve distinct notes. We use these twelve notes as building blocks to form complex harmonies and melodies. Note names consist of two parts: a letter and an accidental. The letter is an alphabetical letter in the range A-G. The accidental can be natural (no accidental), sharp (denoted by #), and flat (denoted by b). Examples of note names are: C, Eb, F#

Chord:

A chord is a set of three or more notes played simultaneously in order to create harmony. Two important features of chords are the root note (the note which the chord is named after) and the quality of the chord (e.g. major, minor, and dominant). For example, one name for a chord could be C major.

Scale/Key

Think of a scale/key as a note bank. A scale is a selection of notes from which chords and melodies can be formed. Here I will use the term scale and key interchangeably.

Chord Progression

A chord progression is a series of chords played back to back.

Chord Function

Within any scale, certain chords have certain functions and tendencies depending on what root note they are built from. The two important functions for this project are Tonic and Dominant function. Tonic function indicates that a chord feels at rest. This means that there is no tension in the music and the chord progression could end with this chord. Dominant function indicates high tension in the chord progression. Dominant chords have a high desire to resolve to Tonic chords.

Diatonic

Diatonic means a note within the key. Diatonic chord progressions consists of chords which contain notes entirely in one specific key.

Secondary Dominant

To keep things simple, within any given key, there is one main tonic chord and one main dominant chord. This means that within any key, there is only one possible dominant to the tonic transition. Secondary dominants are an advanced music theory concept where the composer borrows notes from outside the current key. They use these notes to form a dominant chord not found in the current key in order to resolve to a different tonic chord.

The Problem

In this project, I attempt to train a model to classify chord progressions which stay entirely in one key vs chord progressions which contain secondary dominants

Data Collection

In this project, I generate the data points programmatically with python. Every data point is in the same format: eight chords back to back, each played for exactly two seconds. An equal amount of data points is generated for both categories.

First I render the chord progression to a midi file using the midiutil library. I pass in a soundfont and my midi file into a cli tool called Fluidsynth to render the midi file to a wave file. A soundfont can be thought of as a virtual instrument. After this step I have a variety of chord progressions, each played with various instruments.

2 - Feature Set

Here is a general list of features typically used in audio processing for machine learning that I considered using: MFCCs (Mel-Frequency Cepstral Coefficients), Tonnetz coefficients, harmonic pitch class profiles (HPCP).

How Digital Audio is Stored

Before discussing each feature individually, it is important to discuss the format of an audio signal. Digital audio is represented as a waveform, which can be represented as a weighted sum of sine waves at different frequencies, amplitudes, and phases. When audio is recorded and stored digitally, the waveform is sampled at a fixed rate. A typical sample rate is 44100 Hz, meaning that for every second of audio, there are 44100 discrete samples, each storing the amplitude of the waveform at a specific point in time.

A key challenge in audio processing is that audio signals evolve over time. To analyze temporal data, most audio features operate on short frames of the signal rather than the entire waveform at once. These frames typically contain 512 samples, providing a reasonable tradeoff between time resolution and frequency resolution. For each frame, the corresponding feature is computed. For example, Tonnetz features produce six coefficients per frame, resulting in a sequence of feature vectors over time.

This creates a particular issue. All the models we learned in class (Logistic Regression, Neural Network, and SVM) directly operate on fixed length feature vectors and cannot operate on variable length sequential data. To resolve this problem, I perform two techniques to transform the variable length sequential data into fixed length representations: global averaging and per-chord averaging, which I will discuss next.

Acquiring Fixed Length Vectors

Global averaging:

At first, I attempted to use a global averaging method. For each individual coefficient, I analyze its values across all frames in the audio signal, computing the mean and standard deviation. This produces a fixed-length representation that summarizes the entire song.

For example, Tonnetz features consist of six coefficients computed per frame. Applying global averaging, results in 12 total features: the mean and standard deviation for each of the six Tonnetz features.

While this approach is simple, it removes all temporal structure from data making it less optimal for analyzing chord progressions, which requires the model to have an understanding of how each chord affects the neighboring chords in the progression. While I attempted global averaging at first, I eventually settled with per-chord averaging to keep the temporal structure intact.

Per-chord averaging:

Every chord progression in the generated dataset follows a consistent structure. Each data point Each data point consists of eight chords, with each chord lasting for exactly two seconds at a sample rate of 44100 hz. Because the chord boundaries are known, I can determine precisely which samples and frames correspond to each chord.

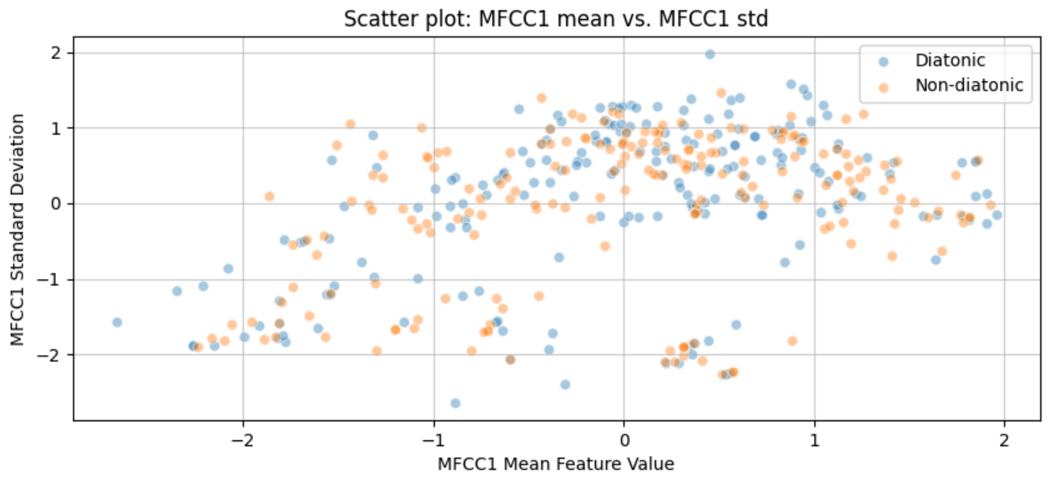
Using this information, I group the frame-level features by chord and compute the mean and for each coefficient within each chord's time window. For a single coefficient, this produces 16 features (two statistics across eight chords). For all six Tonnetz coefficients, this results in a total of 96 features.
Although this method increases the dimensionality of the feature vector, it preserves harmonic and temporal structure that is essential for capturing chord relationships, such as the presence of secondary dominants.

MFCCs

At first, I considered using only MFCCs. MFCCs are a set of coefficients which describe the spectral envelope of an audio signal and are commonly used to capture timbral characteristics. Spectral envelope is the shape/contour of how a sound's energy is distributed throughout its frequency spectrum. Timbre is often described as the color or quality of a sound that distinguishes between different instruments playing the same note. As a result, MFCCs are widely used in speech recognition and genre classification.

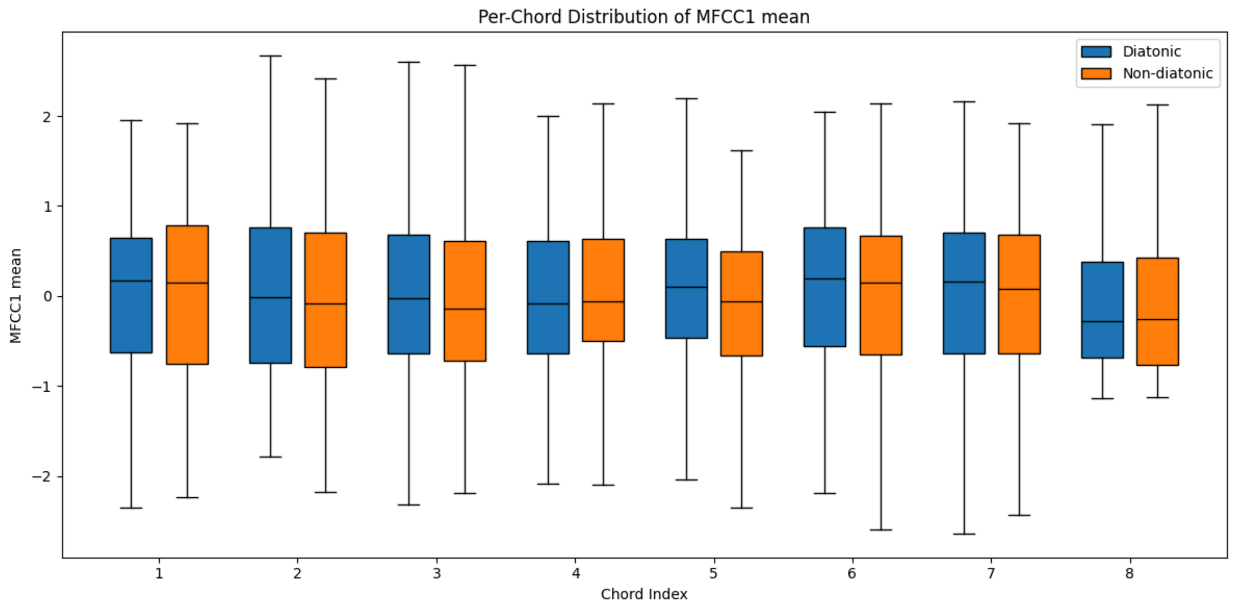
The number of MFCC coefficients can range depending on the desired level of detail. I chose to use 13 MFCC coefficients, which is a common choice for machine learning.

Using global averaging, I was able to extract 26 features from each chord progression. The figure below shows an example scatter plot for the first MFCC coefficient where the X-axis represents the mean value and the Y-axis represents the standard deviation across all frames. The plot contains 200 data points, with diatonic progressions labeled in blue and non-diatonic progressions labeled in orange.



As shown in the figure, the two classes demonstrate significant overlap in both the mean and standard deviation for the first MFCC coefficient. This indicates that MFCC1 provides little

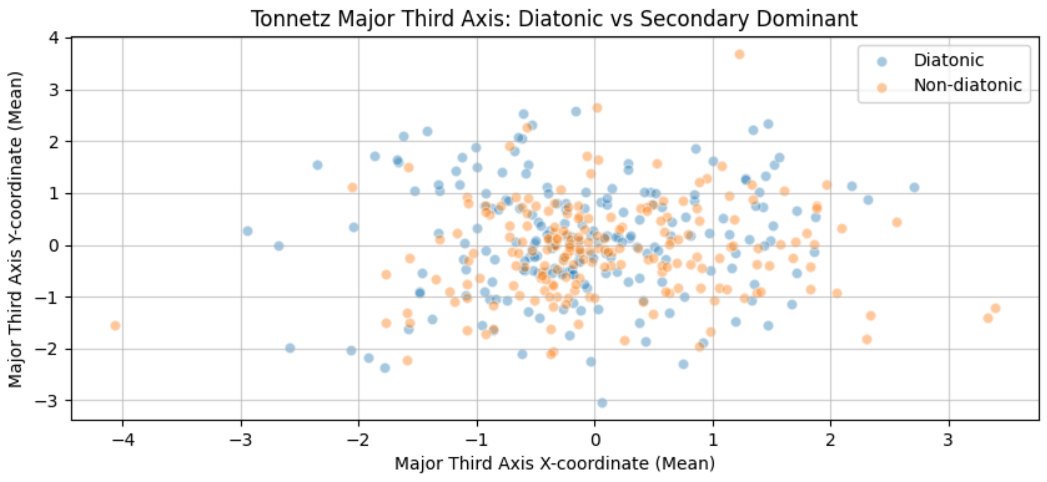
information that helps with categorizing the two classes. Similar patterns were observed for the remaining MFCC coefficients (see graphs labeled “global mfcc mean vs std” in the graphs/mfcc directory). This result is expected, as MFCCs primarily capture timbral information, while secondary dominants are distinguished by harmonic information.
Using per-chord averaging, I reached a similar conclusion. Below is a box plot of the distribution for the mean MFCC1 for each chord across all the data points. The X-axis represents the chord index (1-8) and the Y-axis represents the mean feature value. For every chord, there is significant overlap between diatonic progressions and progressions containing secondary dominants. Similar patterns were also observed for the remaining features (see graphs labeled “per-chord-distribution-mfcc” within the graphs/mfcc directory). These results indicate that MFCCs provide limited useful information for this classification task.

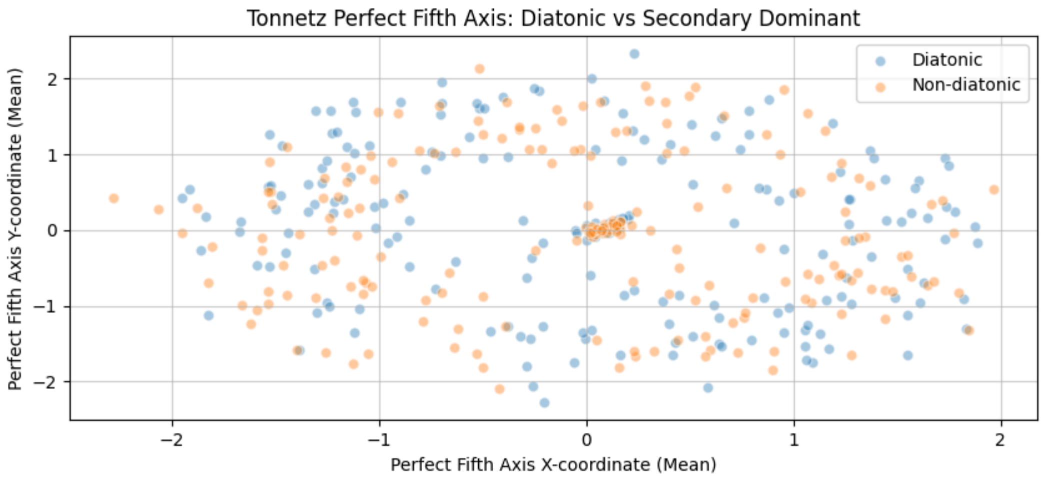


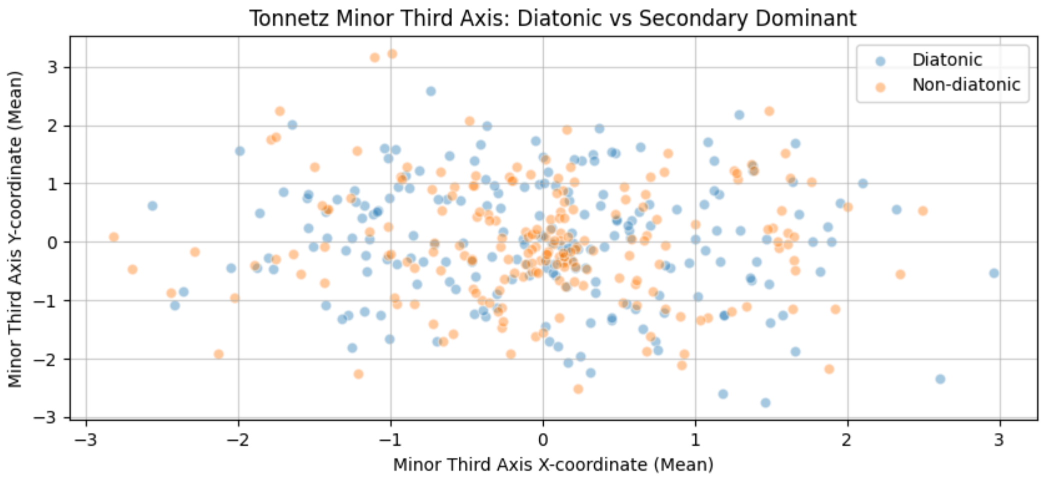
Tonnetz

After MFCCs, the next feature I attempted to use was Tonnetz. Tonnetz feature extraction returns six coefficients. These coefficients are meant to be interpreted in pairs of 2. The first and second coefficients represent the strength of the perfect fifth interval. The third and fourth coefficients represent the strength of the minor third interval. Lastly, the fifth and sixth coefficients represent the strength of the major third interval. These intervals represent intervals between two individual musical notes and they represent the building blocks of chords.

First, I attempted to use global averages for the Tonnetz coefficients. Below are three scatter plots, each one representing a different interval that Tonnetz encodes. A similar observation can be made with the scatter plots for the MFCC coefficients above. The globally averaged Tonntez provides little distinction between the two classes.


Tonnetz Major Third Axis: Diatonic vs Secondary Dominant
Diatonic
Non-diatonic
Major Third Axis X-coordinate (Mean)
Major Third Axis Y-coordinate (Mean)


Tonnetz Perfect Fifth Axis: Diatonic vs Secondary Dominant
Diatonic
Non-diatonic
Perfect Fifth Axis X-coordinate (Mean)
Perfect Fifth Axis Y-coordinate (Mean)


Tonnetz Minor Third Axis: Diatonic vs Secondary Dominant
Diatonic
Non-diatonic
Minor Third Axis X-coordinate (Mean)
Minor Third Axis Y-coordinate (Mean)

The globally averaged Tonnetz coefficients didn't provide any useful information, however what about the per-chord averaged coefficients? After applying per-chord averaging Tonnetz and passing the results into a logistic regression model, I received interesting accuracy results. The training accuracy was around 80% while the test accuracy was around 50%. 80% was the highest accuracy I had seen so far, with all the numbers being around 50%. This indicated that although my model was currently overfit, Tonnetz provided at least a few useful features that would allow for my models to recognize the songs distinctly.

Global Average Deprecation

After attempting the global average technique with both MFCCs and Tonnetz, I came to the realization that global averaging was not a good technique. Chord progressions inherently rely on temporal context of the next and previous chords. Global averaging removes all temporal context, flattening the entire song into one small vector. From hereon, I would not use global averaging anymore

Harmonic Pitch Class Profiles (HPCP)

The final feature representation explored in this project was the Harmonic Pitch Class Profile (HPCP). HPCP is a pitch based feature designed to capture the harmonic content of an audio signal. HPCP focuses directly on the pitch class energy, making it well suited for harmonic analysis.

For each frame of the audio signal, HPCP produces a 12-dimensional vector where each element corresponds to one of the twelve pitch classes in western music theory (e.g. C, Db, D…). Each value represents the relative magnitude of harmonic energy associated with that pitch class. HPCP discards octave information while preserving the structure of the signal.

Final Feature Set

Ultimately, the feature set I ended up going with consisted of per-chord HPCP concatented with per-chord Tonnetz. The goal was to combine the harmonic information that HPCP provided with the chord relationship information that Tonnetz provided.

3 - Logistic Regression

The first model that I attempted to train was a logistic regression using a feature vector formed by concatenating per-chord Harmonic Pitch Class Profiles (HPCP) and Tonnetz features.

Dataset Splitting

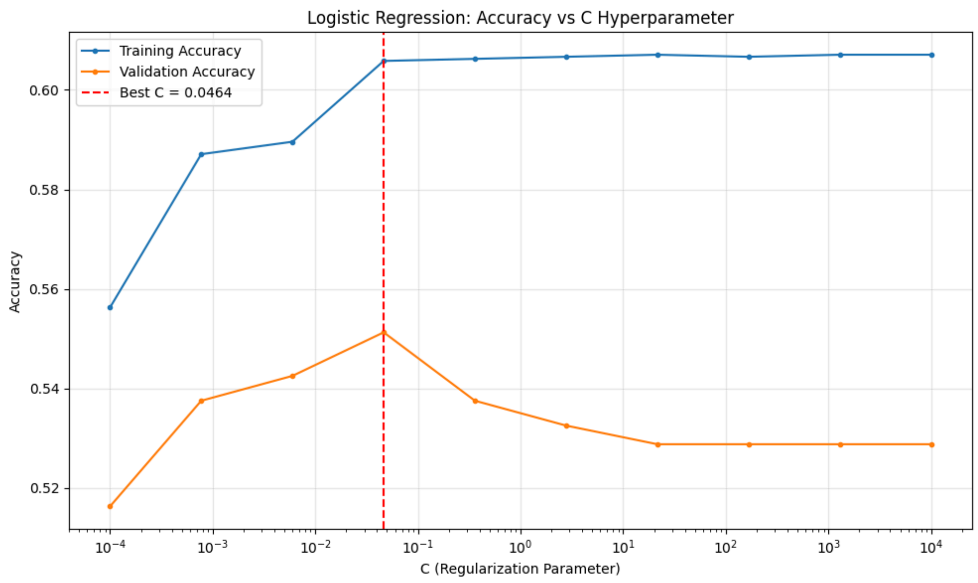
I partitioned my dataset randomly into three disjoint subsets:

- Training set (60%): used to fit the model parameters
- Validation set (20%): used for hyperparameter tuning
- Test set (20%): used only for the final validation

Hyperparameter Tuning

Logistic regression has a single primary hyperparameter: the regularization strength C, which controls the trade-off between fitting the training data and preventing overfitting. Smaller values of C correspond to stronger regularization while larger values of C allow the model to fit the training data more closely.

To tune this parameter, a logarithmically spaced range of c values between 10^{-4} and 10^{4} was evaluated. For each value of C, the model was trained on the training set and its accuracy was measured on both the training and validation sets.



The figure above shows the training and validation accuracy as a function of C. as C increases, the training accuracy gradually improves. However the validation accuracy peaks at an intermediate value and then decreases for larger C values, suggesting overfitting. The optimal value of C was selected as 0.0464 because it achieved the highest validation accuracy.

Final Results

At the optimal C value, the overall accuracy of the logistic regression model is shown in the table below:

Datapoints	Accuracy
Training	60.58%
Validation	55.12%
Testing	49.75%

Overall, the logistic regression model achieved performance close to 50% accuracy on the test set, indicating that it performs no better than random guessing. This suggests that a linear decision boundary is insufficient to separate the two categories when using the concatenated per-chord HPCP and Tonnetz feature presentation.

4 - SVM

Due to the limited performance of logistic regression, I next attempted to use the SVM model from sklearn in order to separate non-linearly classifyable data . I create the same feature vector as before by concatenating per-chord Harmonic Pitch Class Profiles (HPCP) and Tonnetz features. Dataset partitioning is also done in the same manner as specified in the logistic regression session, maintaining a 60/20/20 training, validation, and testing split.

Hyperparameter tuning

When creating the SVM, three parameters are provided to the model: the kernel, a C value, and a gamma value.

Kernel

The kernel function defines how input features are transformed, allowing the classifier to separate data that is not linearly separable.

A linear kernel can be specified which applies no nonlinear transformation. The results from the logistic regression prove that the data is non-linear, so this kernel was not chosen.

Polynomial kernels introduce interactions between features by mapping the data into higher order polynomial relationships. Polynomial kernels require many additional hyperparameters and features and can be prone to overfitting so it was not chosen.

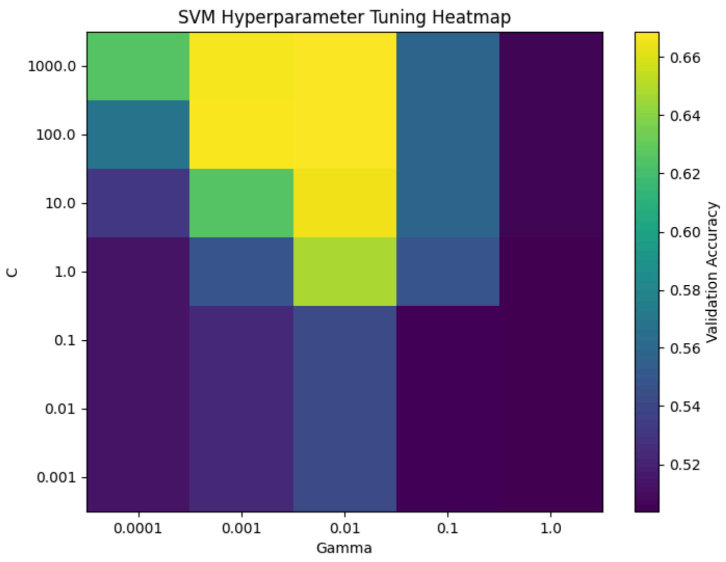
The radial basis function (RBF) kernel measures similarity based on the distance between features, enabling the model to learn nonlinear decision boundaries. Because the extracted audio features exhibit a complex nonlinear relationship where distances between chord level features are important, the RBF kernel was chosen.

C and Gamma Values

The parameter C in an SVM controls the strength of regularization and acts similarly to the C value in the logistic regression model. A smaller C allows for more regularization while a larger C value encourages the model to fit the training data more closely.

The gamma value is specific to the RBF kernel. Gamma determines the influence of a single training example. A smaller gamma means that points influence a larger region of the feature space whereas a larger gamma means that points only affect points closer to them in the feature space.

Both C and gamma values were tuned simultaneously to maximize classification accuracy on the validation dataset. The figure to the right is a heatmap that shows the validation accuracy across different combinations of the C and gamma hyperparameters. The X-axis represents the gamma value, the Y-axis represents the C value, and the color intensity represents the validation accuracy. Values of gamma were logarithmically spaced in the range 0.0001 to 1.0 and values of C were logarithmically spaced between in the range 0.001 to 1000. After testing the SVM with every pair of C and gamma values, the best values are C=100 and gamma=0.01.



Final Results

Using the selected hyperparameters of C=100 and gamma=0.01, the accuracy of the SVM on each dataset is shown in the table below.

Datapoints	Accuracy
Training	100.00%
Validation	66.87%
Testing	64.87%

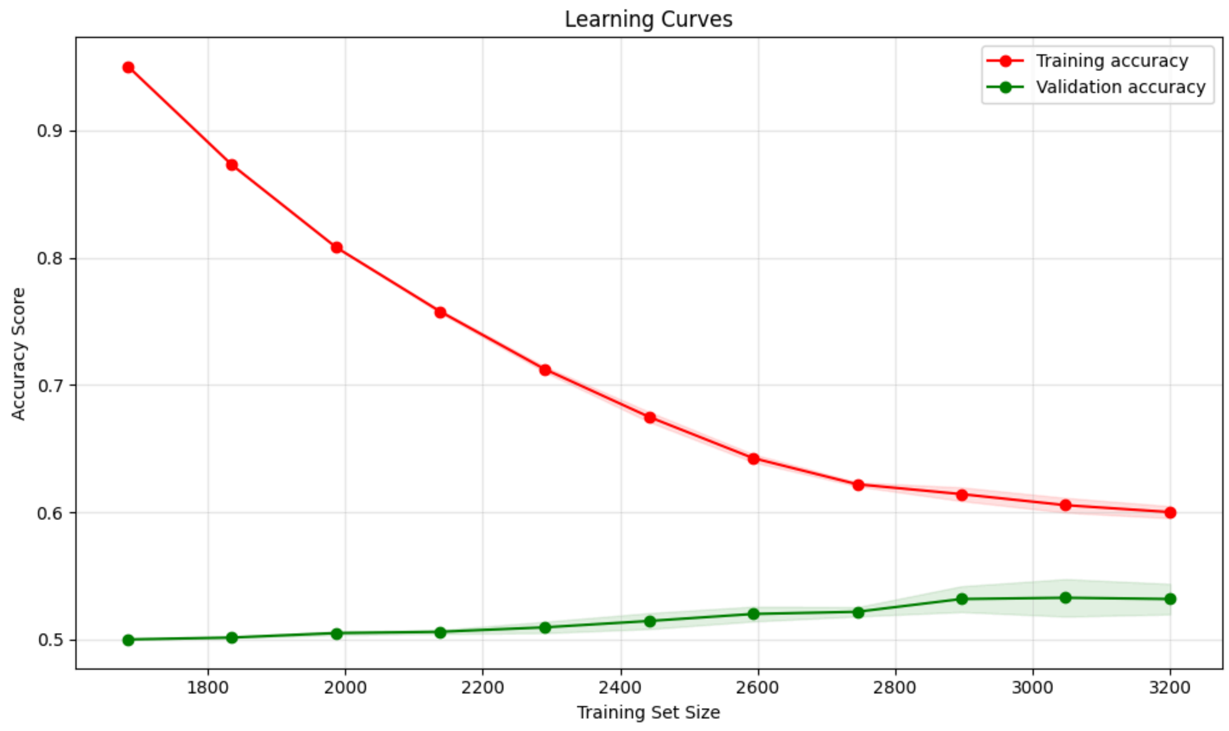
The model achieved 100% accuracy on the training data, which is expected when using an RBF kernel with a relatively large C value.

The validation and test accuracies are within approximately 2% of each other, indicating that the model generalizes reasonably well and that the validation set provided a reliable estimate of the test performance.

Compared to the logistic regression which averaged around 50% accuracy on the unseen data, the SVM improved performance by roughly 15 percentage points, reaching an average accuracy of around 65%. This supports my hypothesis that the underlying feature set is nonlinear in nature and cannot be captured by a linear classifier such as logistic regression.

5 - Learning Curve

Logistic Regression



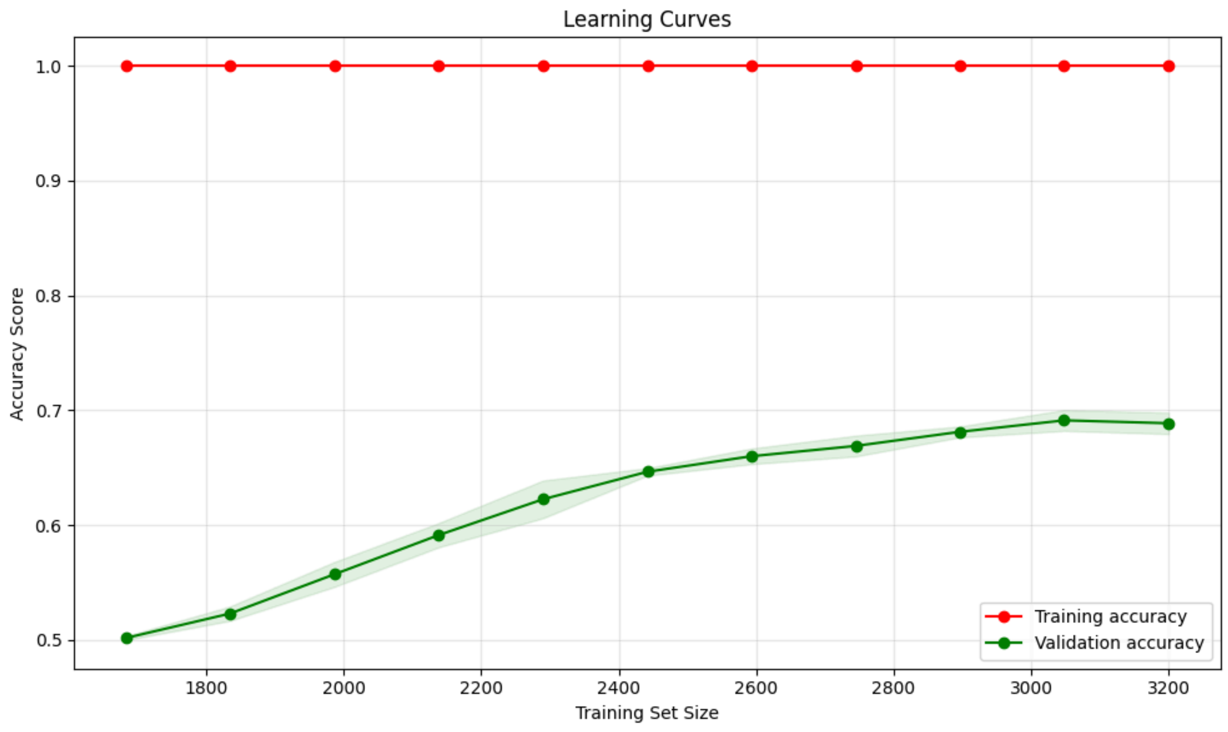
Displayed above is the learning curve graph for the logistic regression model. The X-axis represents the size of the training set and the Y-axis represents the classification accuracy. The red curve corresponds to the training set accuracy and the green curve corresponds to the validation set accuracy.

When trained on small subsets of data, the model achieves very high training accuracy, indicating that it can closely fit a limited number of samples. However, as the training set size increases, the training accuracy steadily decreases and converges to approximately 60%. This behavior suggests that the model is unable to maintain high accuracy as more data is introduced.

The validation accuracy remains consistently low across all training set sizes, increasing only marginally from approximately 50% to 54%. The validation curve plateaus early and does not show significant improvement as more training data is added. This indicates that the size of the training set does not meaningfully improve performance.

Together, these characteristics of the training and validation learning curve indicate that the model is underfit. Logistic regression lacks the ability to capture nonlinear interactions between features in the dataset, resulting in consistently low validation scores.

SVM



Displayed above is the learning curve graph for my SVM model. Once again, the red curve represents training accuracy and the green curve represents cross-validation accuracy.

Across all training sizes, the model achieves near perfect training accuracy, indicating that the SVM has sufficient capability to closely fit the training data. This behavior is expected for a model with a large C value, which allows the model to better fit the training data.

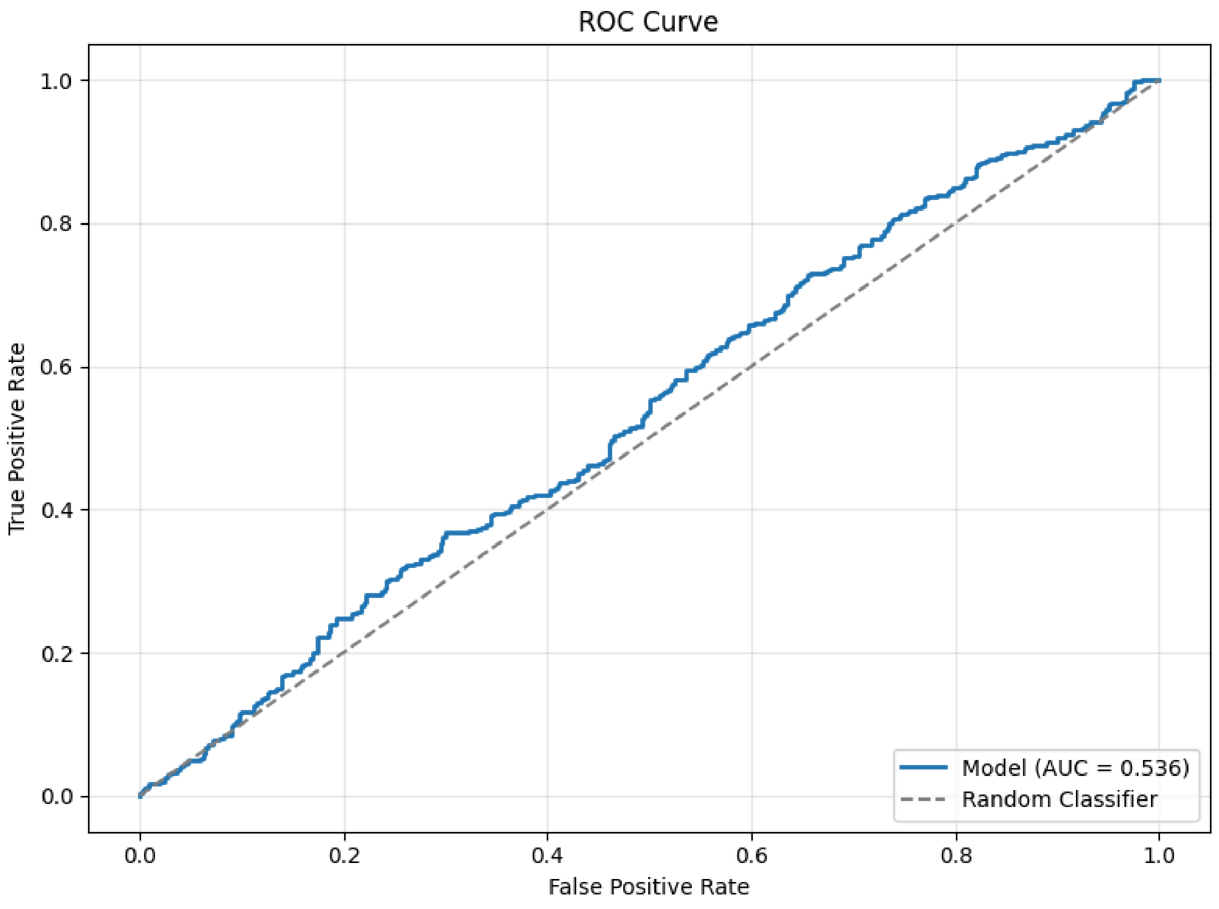
In contrast, the validation accuracy starts at around 50%, steadily increasing until it reaches approximately 65%. Unlike the logistic regression model, the validation curve shows consistent improvement with additional training data, demonstrating that the SVM is able to learn increasingly meaningful patterns rather than plateauing early.

The gap between the training and validation curves indicates overfitting, as the model fits the training data perfectly while the generalized performance for unseen data remains lower. However, the upward trend of the validation accuracy suggests that this overfitting is mitigated as more data is added. This implies that the model benefits from increasing the sample size. More gains may be possible if the sample size is further increased.

6 - Analyzing Success

Logistic Regression

ROC Curve



The figure above shows the ROC and AUC for the logistic regression model trained on the concatenated per-chord HPCP and Tonnetz features. The blue curve depicts the ROC curve while the dashed gray line is the ROC for a random classifier which randomly guesses the classes for each data point.

The ROC curve for the logistic regression closely resembles the curve for the random classifier. The AUC value of 0.536, is only marginally above 0.5, further confirming that the model performs only slightly better than random guessing. The concatenated per-chord HPCP and Tonnetz features do not provide strong linearly separability between diatonic chord progressions and chord progressions containing secondary dominants.

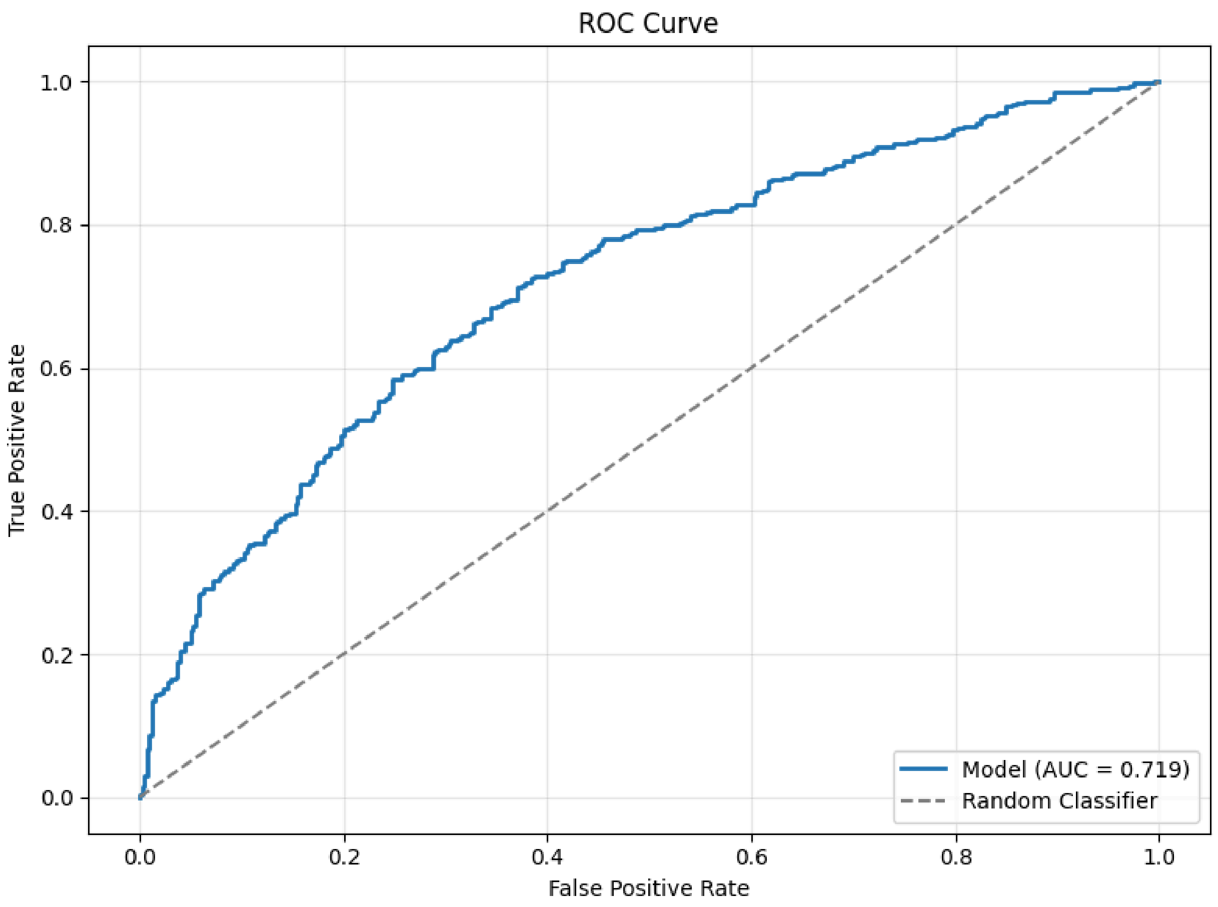
Precision and Recall

Precision and recall for my logistic regression model is displayed in the table below:

Precision	0.513
Recall	0.485

This aligns with the ROC curve from above, suggesting that the logistic regression model is no better than randomly guessing the class.

SVM



Depicted above is the ROC curve for my SVM model using the concatenated per-chord HPCP and Tonnetz feature set. Once again, the blue curve shows the ROC curve for the model while the dashed gray is an ROC curve for a random classifier.

The ROC curve for the SVM model indicates a clear separation from the random classifier. The resulting AUC value of 0.719 indicates that the SVM substantially outperforms logistic regression. The shape of the curve suggests that the SVM is able to capture non-linear relationships within the feature set.

Precision and Recall

Precision and recall for my SVM model is displayed in the table below:

Precision	0.656
Recall	0.625

While the precision and recall for the SVM model are not too high, it aligns with the comparative increase compared with the logistic regression model that we saw earlier. Both the precision and recall for SVM are around 15 percentage points higher than the precision and recall for the logistic model.

7 - Conclusion

In this project, I investigated whether a logistic regression or support vector machine is capable of distinguishing chord progressions which are purely diatonic vs. chord progressions which contain the use of secondary dominants. The dataset consisted of programmatically generated chord progressions which consisted of eight-chords each played for two seconds and rendered using a variety of different instruments.

Although many audio features are available for music analysis, the final feature set consisted of concatenated per-chord averaged Harmonic Pitch Class Profiles and Tonnetz coefficients. These features were selected to capture the harmonic energy of the different chords and the harmonic motion across chord boundaries.

Both models were evaluated using the same 60/20/20 split of training, validation, and testing sets to ensure a fair comparison. Logistic regression, which is only able to model a linear decision boundary, performed only marginally better than randomly guessing. This was reflected in its ROC curve, low AUC, and precision and recall values near 0.5. In contrast, the SVM demonstrated substantially improved performance, achieving a higher AUC and stronger precision and recall.

These results indicate that HPCP and Tonnetz features alone are insufficient for linear classification, but can become more effective when used in conjunction with a nonlinear model.