
Rating the Difficulty of Piano Music

Jacob McKibben
jmckib2@uic.edu

Anshuman Misra
amisra7@uic.edu

Crista Mondragon
cmondr3@uic.edu

Marlon Mueller-Soppart
mlrspp2@uic.edu

Edris Qarghah
mqargh2@uic.edu

Harsh Yadav
hyadav5@uic.edu

Abstract

In this paper, we detail methods for determining the difficulty of piano music. We used existing difficulty rankings and features we extracted from midi files of ranked music to train the following machine learning models: decision trees, neural networks, k-means, k-nearest neighbors and support vector machines. Decision trees and k-nearest neighbors yielded the highest accuracy predictions (79 and 77 percent respectively), with comparatively high F-scores (.77 and .72). We remain optimistic that with better features and a more robust sample, it should be possible to make more accurate predictions of the difficulty of piano sheet music.

1 Introduction

Beginner piano students require professional instruction to learn the basics of posture, fingering and interpretation. As students advance, however, instruction can grow prohibitively expensive. A significant barrier to students progressing independently is the expertise needed to set achievable goals. Our goal in this paper is to lay out a method for making predictions of the difficulty of piano music so that students can determine whether a piece of music is a good next step in their journey toward piano proficiency.

We have treated the difficulty of piano music as a classification problem. In this paper we will describe the rankings used, discuss how and why we collected MIDI files as our digital representation of piano music, outline the features we extracted to describe that music, detail the tools we used to extract those features and explain the models we trained to predict the music’s difficulty. We broadly succeeded in predicting the difficulty of piano music, though we will discuss the limitations of our findings and what we believe could be done to create better predictions.

2 Data

As novices ourselves, we were not equipped to make definitive determinations as to whether a particular piano piece is difficult or not. To this effect, we needed an expert ranking of a large amount of piano music to serve as our ground truth and we also needed some digital representation of that music to use for training our models.

2.1 Ground truth: difficulty levels

Pianostreet [1] is a general resource for classical piano music and one of the features on the site is a list of 2867 pieces of classical piano music and their corresponding difficulty level from 1-8+. While other rankings exist of piano music [2][3], we used pianostreet because of the size of the sample, the cost (free) and the ease of getting all ranked pieces with some other details (they were available in a tabular format).

Unfortunately, as there are many more difficult pieces of classical piano music than there are easier ones, the sample was almost exponentially distributed (as shown in figure 1). This posed a challenge to our models, which we discuss further in the Results section.

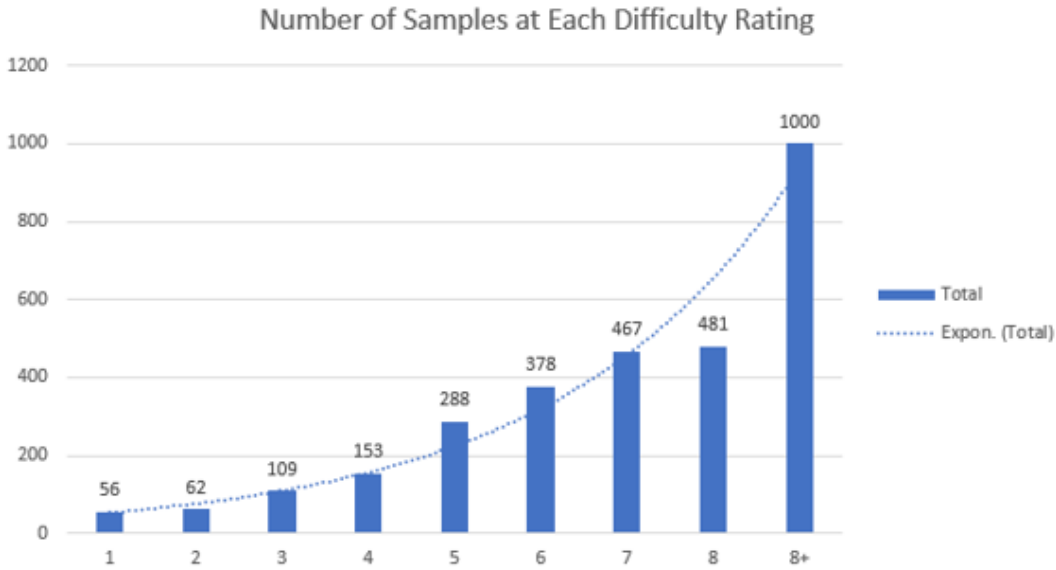


Figure 1: The number of samples increased almost exponentially with difficulty.

The ranked pieces included some that were for two pianos, which we omitted. We also replaced 8+ with 9, for numerical consistency.

2.2 Digital piano music

Finding digital representations of piano music posed a significant challenge, as we needed them in large quantities (as many of the almost 3000 ranked songs as we could manage) and they needed to be specific matches to the pieces for which we had rankings. We decided not to use common file formats that explicitly represent sheet music (e.g., lilypond, musicxml, humdrum, etc.) because the amount of free sheet music in any of these formats was incredibly limited.

Instead, we opted to use MIDI, which is fairly ubiquitous and can be found for free at a variety of websites [4][5][6]. Unfortunately, most of these websites had caps on the number of files that could be downloaded in a given time frame or presented other barriers to downloading the files (e.g., limited search functionality, many steps to download a single file, file names that don't properly match the title of the piece of music, etc.).

This meant we were only slowly accruing MIDI files and working with what we had at any given time (as seen in figure 4.4). We ended with 240 samples (less than a tenth of the pieces for which we had rankings), which had consequences we discuss further in the Results.

MIDI files contain every pitch in a piece of music along with the time when they are played, the velocity at which they are played (i.e., their volume) and the duration for which they are sustained. This is not directly equivalent to what a student would encounter written on sheet music, but many of those features (i.e., dynamics, tempo, accidentals, etc.) can be inferred using the tools described in 3.1.

3 Approach to predicting difficulty

The primary consideration in pre-processing our data for use in training our predictive models is the extraction of descriptive features. As discussed above, the difficulty of piano music is likely dependent on features that are not directly expressed in a MIDI file, so we defined those features and

extracted them. For each sample, these features were represented as a vector which was then used for training the models outlined below.

Our final dataset consisted of 238 music samples with eight extracted features (using `music21` [7] and `pretty_midi` [8]), which were continuous numerical values, and two categorical features, Composer Name and Key. The result was a multi-class classification problem with 9 classes.

We observed that the size of our dataset was too small to satisfactorily capture the variance in the features across all nine classes. As there were very few samples in the lower difficulty classes and there were a lot of samples at higher difficulties, we decided to aggregate difficulties across three classes to ensure there were enough samples in each class to reflect variations in features:

- **Easy:** Difficulty levels 1-3
- **Intermediate:** Difficulty levels 4-6
- **Hard:** Difficulty levels 7-9

This made it easier to rank features because we could now observe the variation across the 3 classes in many of our features. This also made the process of determining difficulty much easier because the ratio of samples across the three classes was not as skewed as it was across the nine classes.

3.1 Tools used for pre-processing

As MIDI files are not encoded in a manner conducive to our analysis, we used two python libraries, `pretty_midi` [8] and `music21` [7], which provided tools for extracting the features we needed.

`pretty_midi` was used to extract certain rudimentary information which act as variables in formulas comprising complex features, for example, tempo. Tempo is a part of the formula to compute playing speed. `pretty_midi` was also used to directly extract time signature changes. `music21` was used to extract most of the other information required to compute features, such as beats per note for the playing speed.

3.2 Feature extraction

Though some features could be extracted directly using functions in the python libraries we used (e.g., the aforementioned time signature changes), others had to be defined separately. See Table 1 for a detailed breakdown of each feature and how we calculated it (several descriptions are from a research paper on this topic [9]).

3.3 Overview of models used

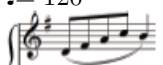




1. KNN with $K = 3$ (determined with guess and check)
2. K-Means $K = 3$ (Using top 5 features determined from ANOVA)
3. Decision Trees - Based on the estimate that top features will be utilized automatically via recursive greedy approach
4. Neural Networks

4 Evaluation of methods

4.1 Feature Ranking using ANOVA

After aggregating samples to three classes, we checked if these features were effective in differentiating between the three classes and then ranked them based on their effectiveness. The categorical features were limited in the sense that for certain composers, we did not find MIDI files corresponding to all difficulty levels, therefore using composer name lead to overfitting. Similarly, using the key also lead to overfitting as the dataset did not have all difficulty levels for all keys (there were also many samples with key not listed). Therefore, we chose to leave out the categorical features and focus entirely on the features we extracted.

Table 1: Features

Feature	Description	Function definition	Example
Playing speed (PS) [9]	Tempo: speed or pace of a musical piece. It is indicated by a word (e.g. Andante grazioso) or a value in BPM (Beats Per Minute) Pulsation: reference value indicated in the tempo: $\circ = 4 \text{ } \flat = 2 \text{ } \natural = 1 \text{ } \sharp = 0.5$	$PS = \frac{60}{tempo} \times \frac{1}{N} \sum_{i=1}^N b(n_i)$ where n_i is i -th note; N is the total number of notes. Unit: second	$\natural = 120$  $PS = \frac{60}{120} \times \frac{1}{5} \times (4 \times 0.5 + 1 \times 1) = 0.3$
Pitch entropy (PE) [9]	There are 128 distinct pitch values in MIDI. Generally speaking, the pitch range of piano is from A0 (27.5Hz) to C8 (4186Hz). The pitch entropy measure the information content of pitch value.	$PE = -\sum_{i=1}^N p(x_i) \log_2 p(x_i)$ where $p(x_i)$ is the probability of pitch value x_i ; N is the total number of distinct pitch values.	There are 12 distinct pitch values in total, including 2 B4, 6 D5, 2 G3, 1 F#3 and 1 B2.  $PE = -(\frac{1}{12} \log \frac{2}{12} + \frac{6}{12} \log \frac{6}{12} + \frac{2}{12} \log \frac{2}{12} + \frac{1}{12} \log \frac{1}{12} + \frac{1}{12} \log \frac{1}{12})$
Polyphony Rate (PPR) [9]	Notes sounded simultaneously (i.e., a chord).	Proportion of chords and chords sequences in a given piece P . $PPR = \text{chord strokes} / \text{distinct strokes}$	$PPR = 3/6 = 50\%$ 
Altered note rate (ANR) [9]	Different tonalities impose different sharps and flats as key. C major and A minor are the most basic ones, without alteration.	Proportion of the altered notes. $ANR = \text{number of altered notes} / \text{total number of notes}$	$ANR = 8/18$ 
Change of Meter	The meter of a piece is the number of beats per measure.	A boolean that returns 1 if meter changes at any point in the piece.	$CM = 1$ 
Average Pitch (AP)	Pitch is determined by the frequency of a note. Average pitch is a measure of central tendency across all the different note frequencies in the composition.	$AP = \frac{1}{N} \sum_{i=1}^N \text{freq}(n_i)$	

We used Analysis of variance (ANOVA) to determine which extracted features correlate strongly to the difficulty levels. ANOVA is basically a test of hypothesis to compare means of a continuous variable in two or more independent comparison groups and determine if the difference in means are statistically significant. If the means are considered different across the groups, it means that the variable in question can be used to model the groups as a function of the variable.

The prerequisites to apply ANOVA are:

1. The number of samples should be equal for each class of the dependent variable.
2. If condition 1 is not satisfied, the following two conditions should be satisfied:

- (a) All the features should be normally distributed.
- (b) All the features should have the same standard deviation.

Since our dataset did not have equal samples across the three classes, condition 1 was violated. We then checked if condition 2 was applicable to our dataset. We used the Shapiro-Wilk test to check if the features are normally distributed. We found that five out of eight features were normally distributed. Next, we checked if the features had similar variances, which was an issue since they had completely different standard deviations because their ranges were completely different. For example average pitch had a standard deviation of 103.4 and play speed had a standard deviation of 0.2. This is because play speed ranges between 0.4 and 1.85, whereas average pitch ranges from 205.4 to 822.4. Therefore, we then scaled all features into a normal distribution with standard deviation of 1 and mean of zero using the following standardization procedure:

1. Calculate mean and variance for each feature:

$$\mu = \frac{1}{N} \sum_{i=1}^n x_i$$

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$$

2. Within each feature, scale each datapoint z , using the formula below:

$$x_i = \frac{x_i - \mu}{\sigma}$$

This allowed us to fulfill the requirements of condition 2. Although, in practice ANOVA is robust to mild violations of its prerequisite conditions. In applying ANOVA to our features, we wanted to find out if each feature had a statistically significant difference in its mean across the three classes. We set the following hypotheses:

- $H_0 : \mu_1 = \mu_2 = \mu_3$
- $H_1 : \text{The means are not all equal}$

We set confidence interval at 95% and $\alpha = 0.05$. We used a one way ANOVA test for all features and calculated F statistics and p values to determine whether the features are statistically significant and to find a ranking among the features.

The results from running ANOVA on the features are:

Rank	Features	F Statistic	P Value
1	Pitch Entropy	61.40829855	3.52E-21
2	Altered Rate	24.50784888	3.62E-10
3	Pitch Range	11.7395609	1.59E-05
4	Time Signature Change	9.664135456	1.02E-04
5	Average Pitch	3.168630985	4.44E-02
6	Average Note Duration	0.608645063	5.45E-01
7	Polyphony Rate	0.232406371	7.93E-01
8	Play Speed	0.120214265	8.87E-01

Figure 2: ANOVA F Statistic Table

The top five features have $p < 0.05$, therefore the null hypotheses was rejected for them which means that these features can be used to classify musical pieces into difficulty levels. The bottom 3 features have $p > 0.05$, therefore, the null hypotheses is accepted for them, which means that these features may lead to overfitting if used to classify musical pieces. The ranking of the features has been done on the basis of F statistic, the higher the better. The results from ANOVA were encouraging because five out of eight extracted features have statistical significance.

4.2 KNN

KNN was evaluated with two metrics. The first metric was accuracy. The overall accuracy was .77 on the test set of 44 observations. This is a favorable result. The second metric, F -scores tell a different story.

Figure 2 shows that most of the error took place in the upper left quadrant of the confusion matrix. This concentration of error clearly shows the model's inability to separate easy pieces from medium pieces and the converse.

The class imbalance is likely the culprit here. Even within the subset of easy and medium songs, there was an inner imbalance. There was a concentration of values at 4, which is the border of the two class and probably are the least reliable of rated songs since they do represent the core identity of the bin.

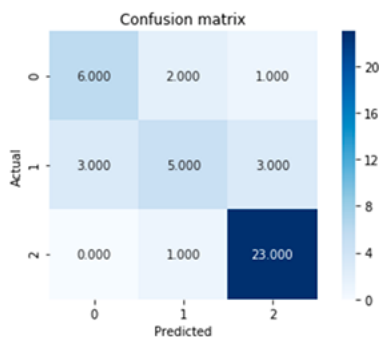


Figure 3: A confusion matrix for KNN.

4.3 K-Means

This model was used to see if the three categories could be roughly discovered by how the observations organize themselves in space. K-means was not able to distinguish the groups well. The contingency matrix in figure 3 shows this.

Medium songs confounded this model. The first group was mainly hard songs with four easy songs and three medium songs. The second centroid grouped the zeros almost perfectly but unfortunately added along too many medium difficulty songs in that cluster and oddly fifteen hard songs. The final cluster took the other half of the medium songs and grouped them with hard songs with, again, oddly a few easy songs.

The model split everything into roughly three equally sized groups unable to discern the existence of a class imbalance. The silhouette score of .23, suggests the separation and cohesion of each cluster are lacking and did not discover a new and better ratings for the songs.

It's fair to say that the observations we collected formed a gradient that is difficult to separate into circular clusters. It must be noted that reducing the number of feature to the top 5 helped the accuracy greatly, ≈ 5 percent.

4.4 Decision trees

We tried using Decision trees with entropy as the split attribute to classify musical pieces. The idea was that since we have five features that are relevant (three of the five features have very high F statistic values) to the predictions that need to be made, a greedy recursive approach capitalizing on the relevant features should work well.

The decision tree uses the top three features – Pitch Entropy, altered rate and pitch range as the first three features to split the samples. It uses pitch entropy multiple times at various depths of the tree. The decision tree also utilized some irrelevant features (polyphony rate, play speed) at certain depths, which was leading to over-fitting.

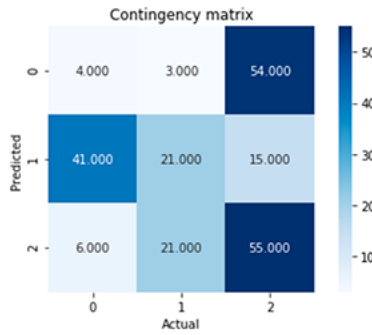


Figure 4: A contingency matrix for K-means.

Removing the irrelevant features did not greatly increase the accuracy and F1 score. This was probably because of the small number of training samples. If there was a larger training set (at least 1000 samples), our estimate is that decision trees would have a much higher accuracy and the recursive greedy approach would utilize only the top 5 features highlighted by ANOVA.

Decision trees scaled very well with the data – the accuracy and F Score increased as the training dataset size increased. We ended up with a test accuracy of .79 and F1 score of .77, which is encouraging because the dataset is so small. The training accuracy and F1 score were 1 were all training set sizes. Figure 4.4 shows the increase in accuracy of the decision tree model versus the number of training samples.

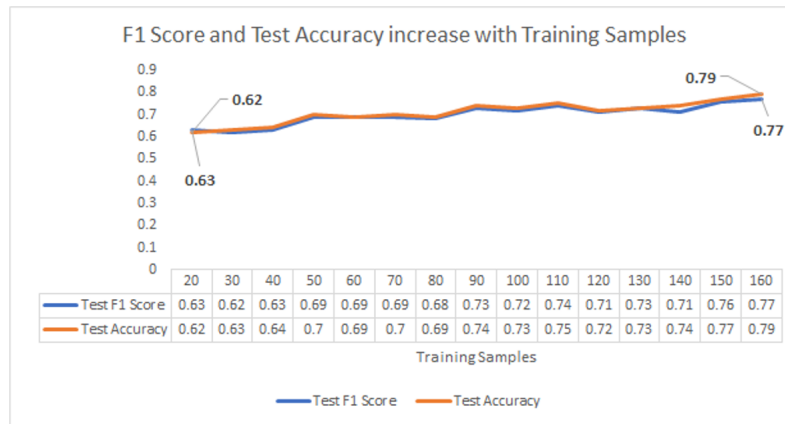


Figure 5: The Performance of the Decision Tree Improves as the training data increases

4.5 Neural Networks

We also tried using a neural network on the features extracted from the dataset. However, this did not prove effective in the classification task. The network did not seem to learn any effective information from the data and was mostly making the naive prediction of the highest frequency class of the dataset.

We implemented a network with 2 hidden layers, sigmoid activations, and squared error loss, with the following results.

Training accuracy: 55.11%

Testing accuracy: 61.36%

The primary reason behind this seems to be the lack of abundant data to train on. Additionally, our dataset was skewed and contained much larger number of difficult samples as compared to easy

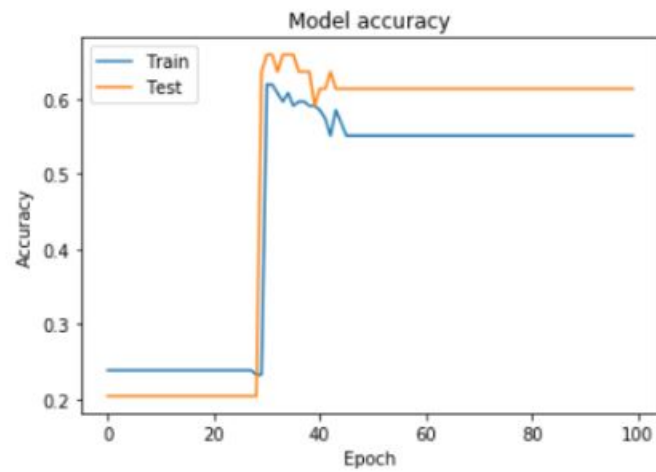


Figure 6: Neural network accuracy

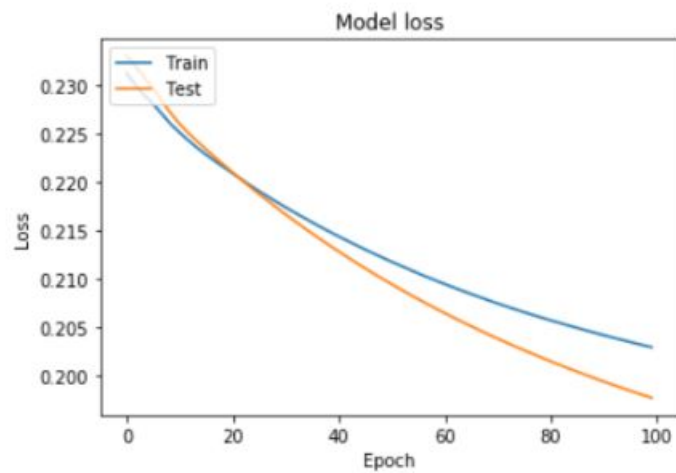


Figure 7: Neural network Loss

ones, which made it more harder to train correctly. Given a larger and well balanced dataset, we expect to get a much better performance with this model.

4.6 Support Vector Machine

Support vectors were also used on the extracted features. This method proved to be ineffective in classifying the dataset. The score was very low implying that many of the data points were misclassified. These results could have been due to the number of features being tested as well as the number of classifications. When the SVM was tested on only the top 5 features, the score was improved some. Using linear vectors proved to have a better score but the results were still far from accurate. The dataset was possibly a poor fit for SVM because of the small sample size as well.

Score with all features: 38% Score with top 5 features: 45%

5 Conclusion

5.1 Overall results

In comparison to the other models, the decision tree and KNN models outperformed the others. The heavily skewed dataset and limited sample size meant that the other methods, particularly neural networks and K-means, predicted that the difficulty was higher for the test set. Although we focused on a variety of models, future implementations would include multiple linear and pace regression.

5.2 Future implementations

Moreover, improvements would most likely be seen if we were to address our issues with the dataset. During our initial phases in this project, data was the most difficult task we dealt with. Although our team was able to collect some MIDI files, it proved too small of a dataset to give an accurate ranking. Because our data was collected from homogeneous sources and were mainly classical pieces, the ratings were skewed. Classical pieces are typically more advanced so our data was skewed towards higher end rankings. Other genres of music would make our rankings more even and our models more robust.

We also had several features we hope to implement that we believe will be predictive. One such feature is the distance between consecutive notes played with a single hand. There are a couple papers on feature extraction using music21 [10][11] that we hope will inform some of our future endeavors.

References

- [1] https://www.pianostreet.com/piano_music/download_1/sheet_1.php
- [2] <http://www.pianosyllabus.com/default.php>
- [3] <http://www.gradedpianorepertoire.com/>
- [4] <https://www.classicalarchives.com/>
- [5] <http://www.kunstderfuge.com/>
- [6] <https://musescore.com/>
- [7] <https://web.mit.edu/music21/>
- [8] <http://craffel.github.io/pretty-midi/>
- [9] S. Chiu and M. Chen, "A Study on Difficulty Level Recognition of Piano Sheet Music," 2012 IEEE International Symposium on Multimedia, Irvine, CA, 2012, pp. 17-23.
- [10] Cuthbert, Michael Scott, Christopher Ariza and Lisa Friedland. "Feature Extraction and Machine Learning on Symbolic Music using the music21 Toolkit." ISMIR (2011).
- [11] Cuthbert, Michael Scott and Chris Ariza. "Hidden Beyond MIDI's Reach: Feature Extraction and Machine Learning with Rich Symbolic Formats in music 21." (2012).