

HANDCRAFTED MIDI FEATURES AND GRADIENT BOOSTING FOR PIANO DIFFICULTY ESTIMATION FROM AUDIO

Vighnesh Ganti

Archbishop Mitty High School, San Jose, CA, USA

vvsganti@gmail.com

ABSTRACT

This extended abstract describes submission VG1 to the MIREX 2026 Music Performance Difficulty Prediction task. The system transcribes a solo-piano recording to MIDI with the high-resolution piano transcription model of Kong et al. [1], the same transcriber used to create the MIDI files of the PSyllabus dataset [2]. It then computes 48 handcrafted, interpretable statistics of the transcription (piece length, pitch range and variety, note density, polyphony, hand span and hand independence, articulation and ornament proxies) and maps them to a real-valued difficulty score with a gradient-boosted regressor trained on all 7,900 usable PSyllabus pieces. The features leave out note velocity and the sustain pedal, so that a score does not depend on recording gain or on whether a rendering uses the pedal. On the five official PSyllabus splits the model reaches a Kendall's τ_c of 0.785 with the default labels. In a held-out audio round trip (66 unseen pieces rendered to audio both as performances and as pedal-free, flat-velocity renderings), the full audio pipeline reaches a τ_c of 0.873, against 0.878 when the same pieces are scored from their original MIDI (a small, favorable sample; see Section 4). The system runs on CPU or GPU and needs about 2 GB of memory.

1. SYSTEM OVERVIEW

Each recording is scored independently in three steps.

1. Transcription. The audio is converted to mono, resampled to 16 kHz (Kaiser-windowed resampling), and transcribed with the note-and-pedal CRNN of Kong et al. [1] using the released checkpoint (note F1 0.9677, pedal F1 0.9186) and its default thresholds. PSyllabus distributes MIDI files that were produced with this transcriber from YouTube performances [2], so the features the model sees at test time come from the same kind of transcription it was trained on.

2. Features. From the transcribed notes the system computes 48 handcrafted features with the same code that built the training table. They summarize each piece globally: length (note count, duration); pitch range, spread, number of distinct pitches, and pitch and pitch-class entropy; note density and inter-onset-interval statistics, including a log-bucketed inter-onset-interval entropy; polyphony and chord density; pitch span within 0.5 s onset windows; left/right balance (split at C4); per-hand leaps and hand-synchrony proxies (split at the piece's median

pitch); legato and staccato fractions, trill and scale-run rates; out-of-key fraction from a Krumhansl key estimate; and an onset-based tempo estimate.

3. Regression. A scikit-learn HistGradientBoostingRegressor [3] maps the 48 features to a score. The raw regression output is returned without rounding or clipping, which keeps ties rare under a rank metric. Higher scores mean harder pieces; the values lie roughly on the 0–10 PSyllabus scale but are meant only as an ordering. If a recording cannot be scored (for example, fewer than four notes are transcribed or the file cannot be read), the system returns the training median (6.0) and logs a warning instead of failing.

2. TRAINING DATA AND LABEL SPACE

The regressor is trained only on PSyllabus [2] (Zenodo record 14794592): 7,901 classical piano pieces with 11-level grades (0–10) from the Piano Syllabus community, of which one MIDI file contains no notes, leaving 7,900 pieces. I used the dataset's own transcribed MIDI files and its default `ps_rating` labels, rounded to integers; the other ranking systems in PSyllabus were not used. No other datasets, augmentation, or label remapping were used to train the regressor, and no composer information enters the model. The transcription checkpoint [1] is used as released, without fine-tuning; its authors trained it on the MAESTRO v2.0.0 dataset [4] (about 200 hours of piano competition performances). As of October 1, 2026, no list of forbidden test pieces or composers had been published on the task page (last edited August 20, 2026), so no training items were excluded; the model can be retrained without any piece or composer on such a list in minutes.

3. FEATURES AND MODEL

The feature code computes 54 features, and the submitted model uses 48 of them. It leaves out three velocity features and the sustain-pedal fraction, because transcribed velocities scale with recording level and the synthesized renderings in the test set may not use the pedal, and two tempo-map features that are constant on transcribed MIDI. The final model is trained on all 7,900 pieces and outputs the continuous prediction. Hyperparameters: 600 maximum iterations, learning rate 0.05, 31 leaves, L2 regularization 1.0, at least 30 samples per leaf, and early stopping on a random 12% of the training data (173 iterations in the final fit).

Features	τ_c	ρ	MAE	MSE _m	BAcc	Acc $\pm$ 1
54 (all)	.787	.898	0.959	1.74	34.3	79.6
48 (submitted)	.785	.896	0.964	1.79	34.1	79.0

Table 1. Mean over the five official PSyllabus splits (features from the dataset’s MIDI). τ_c : Kendall’s tau-c; ρ : Spearman; MAE on the continuous prediction; MSE_m: class-macro MSE and BAcc: balanced accuracy (%), both on rounded predictions; Acc $\pm$ 1 in %. Split-to-split standard deviation of τ_c is about 0.005.

Input to the system	τ_c	ρ	MAE
Original MIDI (no audio)	.878	.953	0.75
Audio, performance rendering	.867	.945	0.81
Audio, score-like rendering	.877	.952	0.77
Audio, mean of both renderings	.873	.950	0.79

Table 2. Held-out audio validation (66 split-0 test pieces, model trained on split-0 training data, 48 features). The last row averages the two renderings’ scores per piece, as the task aggregates a piece’s recordings.

4. DEVELOPMENT RESULTS

Table 1 reports cross-validation on the five official PSyllabus splits, using features computed from the dataset’s MIDI files. Removing the six features costs almost nothing: τ_c falls from 0.787 to 0.785.

To test the whole audio pipeline on unseen pieces, I retrained the submitted model on the training portion of split 0 only and took 66 pieces from that split’s test portion, six per grade. Each piece was rendered to 44.1 kHz audio with the Salamander Grand Piano soundfont in two ways: from the PSyllabus MIDI as is (performed timing, dynamics, and pedal), and as a score-like rendering with the pedal removed and every velocity set to 80 (performed timing kept), a partial stand-in for the task’s synthesized renderings. Table 2 compares predictions made from the original MIDI with predictions after the audio round trip (render, then transcribe).

The audio round trip costs little: averaging each piece’s two renderings gives a τ_c of 0.873, against 0.878 from the original MIDI, and the two renderings’ scores agree closely ($\rho = 0.99$). Without the velocity and pedal features, the score-like rendering does not shift the scores (mean difference 0.00, against -0.08 with all 54 features), which is the reason for removing them; the τ_c difference between the two feature sets (48 minus 54) is within noise: $+0.005$ on the mean-of-both row (paired bootstrap over pieces, 95% CI $[-0.01, 0.02]$). This 66-piece sample is easier to rank than the split as a whole: with the same model and MIDI input, random six-per-grade samples of the split-0 test set average $\tau_c = 0.81$ (s.d. 0.03), against 0.785 on the full test set. The rows of Table 2 should therefore be compared with each other, not with Table 1.

5. COMPUTE DECLARATION

Transcription dominates the cost; feature extraction and the regressor take well under a second per piece. On an 8-core Apple M1 Pro CPU with no other load, a 95-second recording took 34 s end to end (0.37 s per second of audio), with a peak memory of 1.9 GB. At the validation recordings’ mean length (162 s) that is about 1 min per recording, or 2 min per piece with its two recordings. At that CPU rate the 24-hour budget covers about 65 hours of audio. Inside the container the transcriber (43.0 M parameters) runs on a CUDA GPU when one is visible and otherwise on the CPU; I had no GPU to measure on. GPU memory should stay under about 2 GB regardless of recording length, since the transcriber processes 10-second segments one at a time (172 MB of weights plus one segment’s activations); this is an estimate, not a measurement.

6. LIMITATIONS

The model only sees what the transcriber recovers, so transcription errors (missed notes in dense textures, spurious notes from reverberation) propagate into every feature. It was trained on a single dataset whose MIDI comes from YouTube performances, while the test set uses repertoire outside public corpora and includes soundfont renderings. Its labels follow one community grading scheme, and the official splits are not composer-disjoint, so the cross-validation numbers likely overstate performance on unseen composers. Both conditions of the audio validation were rendered from the same soundfont, so it does not measure robustness to real recording conditions such as room acoustics or microphone placement, and the score-like condition keeps performed timing, so quantized score timing is untested.

7. AI USAGE STATEMENT

I used Claude (Anthropic) to write and run the packaging, inference, and validation code and to draft this abstract. I checked the results and take responsibility for the content.

8. REFERENCES

- [1] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, “High-resolution piano transcription with pedals by regressing onset and offset times,” *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 29, pp. 3707–3717, 2021.
- [2] P. Ramoneda, M. Lee, D. Jeong, J. J. Valero-Mas, and X. Serra, “Can audio reveal music performance difficulty? Insights from the Piano Syllabus dataset,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 33, pp. 1129–1141, 2025.
- [3] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

- 172 [4] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-
173 Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and
174 D. Eck, “Enabling factorized piano music modeling
175 and generation with the MAESTRO dataset,” in *Proc.*
176 *Int. Conf. Learning Representations (ICLR)*, 2019.