B. Model Selection and Hyperparameter Settings for Optimal Threshold Selection of Heuristics for Expressive Music Performance Detection

B.1 Machine Learning (ML) Model Selection

Following a series of comparative experiments involving logistic regression, decision trees, and random forests—each implemented using the scikit-learn library—logistic regression was chosen as the most suitable machine learning algorithm for determining optimal thresholds to differentiate between non-expressive and expressive MIDI tracks. This selection was made based on the ground truth data we manually collected, which informed the model's performance evaluation and final decision.

The choice of a machine learning model for identifying optimal thresholds between two classes, such as non-expressive and expressively-performed MIDI tracks, requires careful consideration of the data's specific characteristics and the analysis goals. Logistic regression is often favoured when the relationship between the input features and the target class is approximately linear. This model provides a clear, interpretable framework for classification by modelling the probability that a given input belongs to one of the two classes. The output of logistic regression is a continuous probability score between 0 and 1, which allows for straightforward determination and adjustment of the decision threshold. This simplicity and directness make logistic regression particularly appealing when the primary objective is to identify a reliable and easily interpretable threshold.

However, logistic regression has limitations, particularly when the true relationship between the features and the outcome is non-linear or complex. In such cases, decision trees and random forests offer more flexibility. Decision trees can capture non-linear interactions between features by partitioning the feature space into distinct regions associated with a specific class. Random forests, as ensembles of decision trees, enhance this flexibility by averaging the predictions of multiple trees, thereby reducing variance and improving generalization. These models can model complex relationships that logistic regression might miss, making them more suitable for datasets where the linear assumption of logistic regression does not hold.

Regarding threshold determination, logistic regression has a distinct advantage due to its probabilistic output. The model naturally provides a probability estimate for each instance, and a threshold can be easily applied to classify instances into one of the two classes. This straightforward approach to threshold selection is one of the key reasons logistic regression is often chosen for tasks requiring clear and interpretable decision boundaries. In contrast, decision trees and random forests do not inherently produce probability scores similarly. While they can be adapted to generate probabilities by considering the distribution of classes within the leaf nodes for decision trees or across the trees in the forest for random forests, this process is more complex and can make threshold selection less intuitive.

In our computational experiment, the logistic regression machine learning model, combined with manual threshold inspection for validation, was found to be sufficient for identifying the optimal threshold for each heuristic. This approach was particularly effective given the simplicity of the task, which involved a single feature for each of the three key metrics—Distinctive Note Velocity Ratio (DNVR), Distinctive Note Onset Deviation Ratio (DNODR), and Note Onset Median Metric Level (NOMML)—and the classification of data into two categories: non-expressive and expressive tracks. The problem at hand, being a straightforward binary classification task using a supervised learning algorithm, aligned well with the capabilities of logistic regression, thereby rendering it an appropriate choice for our optimal threshold selection.

B.2 Hyperparameter Settings and Training Details

The process of training a logistic regression model using the leave-one-out cross-validation (LOOCV) method requires a methodical approach to ensure robust model performance. Leave-one-out cross-validation is a special case of k-fold cross-validation where the number of folds equals the number of instances in the dataset. In this method, the model is trained on all data points except one, which is used as the validation set, and this process is repeated for each data point. The advantage of LOOCV lies in its ability to maximize the use of available data for training while providing a nearly unbiased estimate of model performance. However, due to its computational intensity, especially with large datasets, careful consideration is given to the selection and tuning of hyperparameters to optimize the model's performance. In our case, we trained our models with 722 instances using LOOCV, a relatively small amount of data available with the ground truth of non-expressive and expressive tracks due to the scarcity of such ground truth available for expressive music performance detection.

The training environment for our experiments was configured on a MacBook Pro, equipped with an Apple M2 CPU and 16GB of RAM, without the use of external GPUs. Our analysis, which included evaluation using the P4 metric alongside basic metrics such as classification accuracy, precision, and recall, did not indicate any significant impact on performance attributable to the computational setup. Furthermore, we share three logistic regression models in .pkl format, each trained on a specific heuristic, accessible via GitHub. These models correspond to the following heuristics: baseline heuristics, Distinctive Note Velocity Ratio (DNVR), trained in less than 10 minutes;

Distinctive Note Onset Deviation Ratio (DNODR), trained within 10 minutes; and Note Onset Median Metric Level (NOMML), trained in 3 minutes with our MacBook Pro.

For hyperparameter tuning, we employed the scikit-learn library for logistic regression, a widely recognized tool in the machine learning community for its efficiency and versatility. We utilized the GridSearchCV function within this framework, which facilitates an exhaustive search over a specified parameter grid. This approach identifies the most effective hyperparameters for the logistic regression model. GridSearchCV systematically explores combinations of specified hyperparameter values and evaluates model performance based on cross-validation scores, in this case, derived from the LOOCV process.

The hyperparameters tuned during this process include the regularization strength (denoted as C), which controls the trade-off between achieving a low training error and a low testing error, as well as the choice of regularization method (L1 or L2). By conducting an exhaustive search over these parameters, we aimed to identify the configuration that minimizes the validation error across all iterations of the LOOCV. This rigorous tuning process is crucial, as these hyperparameters can significantly affect logistic regression's performance, particularly in the presence of imbalanced data or feature correlations. The result is a logistic regression model that is finely tuned to perform optimally under the specific conditions of our dataset and evaluation framework.

The following parameters and model configuration were determined through hyperparameter tuning using leave-one-out cross-validation and GridSearchCV using the scikit-learn library for the logistic regression model. Notably, these optimal hyperparameters were consistently identified across all three models corresponding to each heuristic.

- Hyperparameter for the logistic regression models: C=0.046415888336127774
- Logistic regression setting details using the scikit-learn Python ML library:
  LogisticRegression(random_state=0, C=0.046415888336127774, max_iter=10000, tol=0.1)

This configuration represents the optimal hyperparameters identified through comprehensive parameter exploration using GridSearchCV and LOOCV, thereby ensuring the logistic regression model's robust performance.

B.3 Procedure of Optimal Threshold Selection

Our curated evaluation set comprises 361 non-expressive (NE) tracks labelled 0 and 361 expressively-performed (EP) tracks labelled 1. We have five features for training each: baseline heuristics (the number of distinct velocity levels and onset time deviations), DNVR, DNODR, and NOMML (more sophisticated heuristic) feature values. To train the logistic regression models for selecting optimal thresholds for our heuristics, 80% of this curated evaluation set was allocated as the training set. The remaining 20% was reserved as the testing set, which was subsequently used to validate the model's performance during the evaluation phase, so the testing set is not involved with the optimal threshold selection process to prevent potential data leakage.

To determine the optimal threshold for expressive music performance detection using logistic regression with a focus on the P4 metric, the following steps were undertaken:

- Step (1): Prepare the logistic regression algorithm using GridSearchCV to identify optimal hyperparameter settings, followed by leave-one-out cross-validation to maximize the P4 metric. This ensures that the model is fine-tuned for the specific task of classifying non-expressive and expressively-performed MIDI tracks.
- Step (2): Train the logistic regression model on the training set, incorporating the relevant features and ground truth labels, using the pre-determined optimal hyperparameters.
- Step (3): Apply leave-one-out cross-validation on the validation set (within the training set) to obtain predicted probabilities for the positive class, i.e., expressively-performed MIDI tracks.
- Step (4): Validate the performance of the classifier at various threshold values, focusing on optimizing the P4 metric, which is particularly suited for imbalanced and small sample size datasets.
- Step (5): Identify the index of the optimal threshold value within the threshold array that maximizes the P4 metric, ensuring that the model effectively distinguishes between the two classes.
- Step (6): Use this index to extract the corresponding optimal value from the feature array, translating the identified threshold into actionable feature values.
- Step (7): Lastly, we conduct a manual inspection to ensure that the selected thresholds are consistent with the distribution of feature values within the dataset. We then determine the optimal percentiles for these thresholds based on the feature value distribution.

Details of Steps (4), (5), and (6): Initially, predicted probabilities for the positive class are obtained using the `predict_proba` method of the logistic regression model. Next, the precision-recall curve is computed using the `precision_recall_curve` function, and this curve is plotted as a function of different threshold values. The P4 metric is then maximized to identify the optimal threshold, given its effectiveness in handling imbalanced and small sample size datasets by prioritizing the accurate classification of the minority class. By adjusting the threshold value, the trade-off between precision and recall can be controlled—higher thresholds increase

precision but reduce recall, whereas lower thresholds have the opposite effect.

The precision and recall analysis are related to the P4 metric in that both are used to evaluate model performance, especially in imbalanced and small sample size datasets. Precision and recall measure the accuracy of positive predictions and the model's ability to identify all positive cases, respectively. The P4 metric builds on this by optimizing for the correct classification of the minority class, making it particularly useful when the dataset is imbalanced and handing small sample size data. While precision and recall help select optimal thresholds, the P4 metric provides a more tailored validation for scenarios where the minority class is of primary concern.

Following the precision and recall analysis, we convert the identified threshold value into the corresponding feature value. For instance, to translate a P4 metric threshold value (0.9952) into the corresponding Note Onset Median Metric Level (NOMML), the index of the threshold value is determined within the threshold array derived from the precision-recall curve analysis, ensuring that the P4 metric is maximized. This index is then used to extract the corresponding feature value from the NOMML list. As a result, the threshold is set at the corresponding percentile within our curated set used during the optimal threshold selection, establishing the boundary between non-expressive and expressively-performed ground truth data. Finally, we perform a manual review to verify that the selected thresholds align with the distribution of feature values within the dataset. Following this, we identify the optimal percentiles for these thresholds by analyzing the distribution of the feature values.