# HTML2023 Spring Final Project

b10902003, b10902067

June 14, 2023

## 1 Initial Settings

### 1.1 Data Inspection

The features in the dataset can be classified into three types:

- Ordinal: Energy, Key, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Duration_ms, Views, Likes, Comments
- Categorical: Album_type, Licensed, official_video, Album, Channel, Composer, Artist
- Text: Track, Uri, Uri_spotify, Url_youtube, Description, Title

Notice that we drop ID since it is a manually added feature in the dataset, it should provide no insight about the danceability of each track. We also drop text data, due to the fact that it is provided by the uploader and has no standardized format. We believe that in such cases, it contains more noise than the insight about danceability.

Diving into the dataset, we found out that there are only 11 composers and 97 artists. It seemed unnatural to us that there are such less unique composers and artists given that we have more than 17,000 tracks. We dug into several tracks and realized that both features are totally incorrect, and provide us nothing but misleading information. As a result, we also dropped those two features.

We also observed that there are some uncanny information in the testing dataset. For example, negative values in `Duration_ms` and decimals in `Views`! Those information appear after `id=19170`, which is the 2001-th test data. Moreover, aside from unreasonable values, some features (e.g. Loudness, Speechiness, etc.) have over 10 decimal places after the 2001-th test data and only 3 decimal places before. In light of those weirdness, we made a assumption that **Nijika** had fiddled those test data. To counter such mischief, we had different approaches toward the two sections of data. For the first 2000 test data, we believe they are more related to the training data and thus apply models with stronger power and less regulizations; as for the remaining data, we believe there are huge amounts of noise and thus apply stronger regulization on the models.

In conclusion, we mainly focus on the ordinal features. Inside each model, the feature selection is performed manually by permuation test or automatically by the model (e.g. ShapValues in CatBoost). The details are specified in each model subsection.

### 1.2 Combat Missing Values

We have observed that 15% of the data is missing in each feature. In order to facilitate the use of commonly employed machine learning models, it is necessary to fill in these missing portions with appropriate data. Firstly, we need to determine whether the missing data is randomly selected. In some features, such as Channel, where it is relatively easier to obtain the original data, we have found that the missing data is likely to be randomly selected. Based on our belief that the missing data is randomly selected, we have chosen to use mean and median imputation to fill in the missing values, without further exploring whether alternative approaches would yield better performance.

### 1.3 Validation Set

In the first stage, we use k-fold cross validation to tune the parameters and evaluate each model. Nonetheless, the public score is quite far away from the validation score. When the public score is around 1.9, selecting a model with lower validation score does not necessarily give a lower public score. It felt like the public score was randomly hovering between 1.8 to 2.0. We believe that this is due to the manual distortion in test data and thus the validation score can only provide a considerably vague approximation on the public / private score.

Blindly follow the validation score seemed to lead to an overfitting on the training dataset, and did not help us much on selecting the best model. Therefore, we utilized the 5 submissions per day to try out every possible model we had, and managed to reach a relatively lower public score by the end of the first competition. However, on the award

ceremony, professor Lin revealed that there is a distribution shift between public and private dataset. Thus what we had done was overfitting on the public dataset, resulting in an appalling private score.

In the second stage, we are given a small subset of the private dataset. Considering the problematic validation score above, we decided to apply it on validating and give up on cross validation. We believed that we had had enough training data but lack in validating, thus apply it on validation may make the greatest use on it.

# 2 Individual Models

## 2.1 SVM

1. Package: LIBSVM
2. Selected Features: Instrumentalness, Speechiness, Energy, Valence, Acousticness, Liveness, Tempo, Key, Composer
3. Parameters: `-s 3 -t 2 -c 0.01 -g 0.5 -e 0.00001 -h 0`

In our initial attempts to surpass the baseline, we opted to utilize a model with which we were familiar and had good control over the issue of overfitting. While tuning this model, we observed that the numerical ranges of each feature significantly influenced the model trained using Support Vector Machine (SVM). Therefore, we selected features with similar numerical ranges in both the train and test data sets and normalized their values to fit within the range of $[0, 1]$. For data points that exhibited excessive concentration, we applied mathematical functions (such as square root) to disperse their distributions. As for categorical features, we attempted one-hot encoding for the Composer and Artist, which were deemed more likely to be related to Danceability in their physical meaning. However, we found that while the inclusion of Artist reduced Ein (training error), it marginally increased Ecv (cross-validation error) and the public score. Consequently, we dropped the Artist feature.

Ultimately, this model achieved a score of 2.084 on the released data, representing the second-best performing standalone model prior to our submission deadline.

## 2.2 SGD Regressor

## 2.3 Cat Boost

1. Package: CatBoost
2. Selected Features: Instrumentalness, Speechiness, Energy, Valence, Acousticness, Liveness, Tempo, Key, Composer, Artist
3. Parameters: loss function = MAE, uses eval set in feature select, drop 2 features, train final model = True

CatBoost is an optimized version of Gradient Boosting Trees. It natively supports handling categorical features and allows the inclusion of a validation dataset to retain the best iteration, thereby reducing the need for parameter tuning while mitigating overfitting. We utilized the similar input as SVM and employed two methods to enhance our model. The first involved constraining the tree depth and applying l2 regularization at the leaf nodes, while the second utilized the feature selection tool built into CatBoost. We found that the approach utilizing feature selection exhibited better performance on the released dataset, leading us to adopt the feature selection version as our final choice.

Ultimately, this model achieved a score of 2.111 on the released data, representing the best-performing tree model prior to our submission deadline.

## 2.4 Other Models

In addition to the aforementioned three models, we also experimented with various models, including neural networks and random forests. However, their performance on the public score and validation dataset fell short of the aforementioned models, leading us to exclude these two models from our final selection.

Furthermore, we discovered that several linear models, such as BayesianRidge, ARDRegression, ElasticNetCV, LassoLarsCV, as well as neural networks utilizing only linear activation, achieved scores around 2.13 on the released dataset. These models demonstrated favorable performance after fine-tuning, but due to limitations in space, we refrain from elaborating on them further in this paper.

# 3 Blending

In the final stage, we merged the aforementioned models that yielded satisfactory results into a single model. As we had exhausted all available datasets during the training of the preceding models, to prevent overfitting, we simply assigned different weights to each model based on our understanding of their performance. SVM and SGDRegressor exhibited the best performance, thus receiving higher weights, while the other models were assigned lower weights. We believe

that such an allocation contributes to controlling the upper bound of the private score while providing opportunities for effective error correction. Through simple mathematical reasoning, it can be deduced that the worst-case scenario for the private score is the sum of each model's private score multiplied by their respective weights. Due to the higher weights assigned to SVM and SGDRegressor, the worst-case scenario for Blending does not deviate significantly from the performance of these two models. Simultaneously, when different models make predictions on the same value, if their predictions fall on opposite sides of the correct answer, we can anticipate improved results through the blending of predicted values.

Ultimately, this model achieved a score of 2.039 on the released data, outperforming all individual models.

# 4 Final Result