2023 Spring Machine Learning Final Project Report: Group ThreeChens

B09501043 CHEN, LING-WEI, Department of Computer Science and Information Engineering, NTU
B10705005 CHEN, SZU-JU, Department of Computer Science and Information Engineering, NTU
B10902136 CHEN, YEN-SHAN, Department of Computer Science and Information Engineering, NTU

1 INTRODUCTION

In this project, we are required to train a model to predict the danceability of music. We use 3 different ways to generate models, including regression, decision tree, and boosting algorithms. In addition, prior to the model itself, another major focus of this report involves feature and parameter selection. We discuss the intuition behind each experiment / design and provide detailed explanation for the results.

2 DATA PREPROCESSING

The data provided in the training / testing data can be roughly categorized into 3 types: numerical data, categorical data, and text. While we assumed that simply analyzing numerical data, which were the easiest to process, may give fairly good results, preliminary testing proved the opposite: our first submission on Kaggle resulted in a public loss as high as 2.53325, which was about as poor as guessing randomly. Hence, we started investigating how we could possibly filter out the dominating features to increase the effectiveness of the model.

2.1 Numerical Data

To rank the importance of each feature, the most natural and obvious idea that came into our minds was to analyze the correlation coefficient for the numerical data. Surprisingly, most of the numerical features had quite low correlations with the to-be-predicted column. As shown in Table 1, even the most highly related feature "Valence" only had a correlation value of 0.424. Followed by "Valence", we have "Acousticness", "Loudness", "Speechiness", and "Instrumentalness", whose correlation values are -0.31, 0.28, 0.23, and 0.23 respectively. This suggests that solely using numerical data may not be enough - we might need to consider some other categories, or perform some feature transformation to find stronger relationships between our features and labels. Indeed, after extracting the Top-5 slightly more related features, our model performed even worse (2nd submission on Kaggle: public loss = 2.15206; private loss = 2.58026).

Table 1 Correlation coefficient of numerical data with danceability

Energy	Key	Loudness	Speechiness
0.047	0.026	0.27	0.23
Acoustiness	Instrumentalness	Liveness	Valence
-0.31	-0.23	-0.087	0.42
Tempo	Duration ms	Views	Likes
-0.098	-0.098	0.095	0.1
Stream	id	Comments	
0.076	0.19	0.044	

2.2 Categorical Data

Our next step was to analyze categorical data, since this was our second most intuitive idea. For "Licensed" and "official_video", we convert TRUE, FALSE, and N/A to 1, -1, and 0 respectively. Similarly, "single", "album", "compilation", and "N/A" were converted to 0, 1, 2, -1 respectively. While there is no specific rule for mapping these features to integers, we roughly order the categories based on their scale: A compilation usually involves more performers, and an album is often more costly than are singles. However, after the mapping, the data still seemed to exhibit low relevance with "danceability", as shown in Table 3.

Table 2 Correlation coefficient of categorical data with danceability

Album type	Licensed	official video
-0.1	0.017	0.039

2.3 Text Columns

We had no other choice but to start delving into the text columns. We gave up the url columns since there was no simple way to convert them into meaningful data. Initially, we thought that it may be reasonable to count the word frequencies, since we observed that some words had particularly high numbers of occurrences. However, after some discussion, we came up with a more effective and reasonable idea: Instead of just counting the frequencies, we also take information about danceability into consideration. The algorithm we use to convert text into values is as follows:

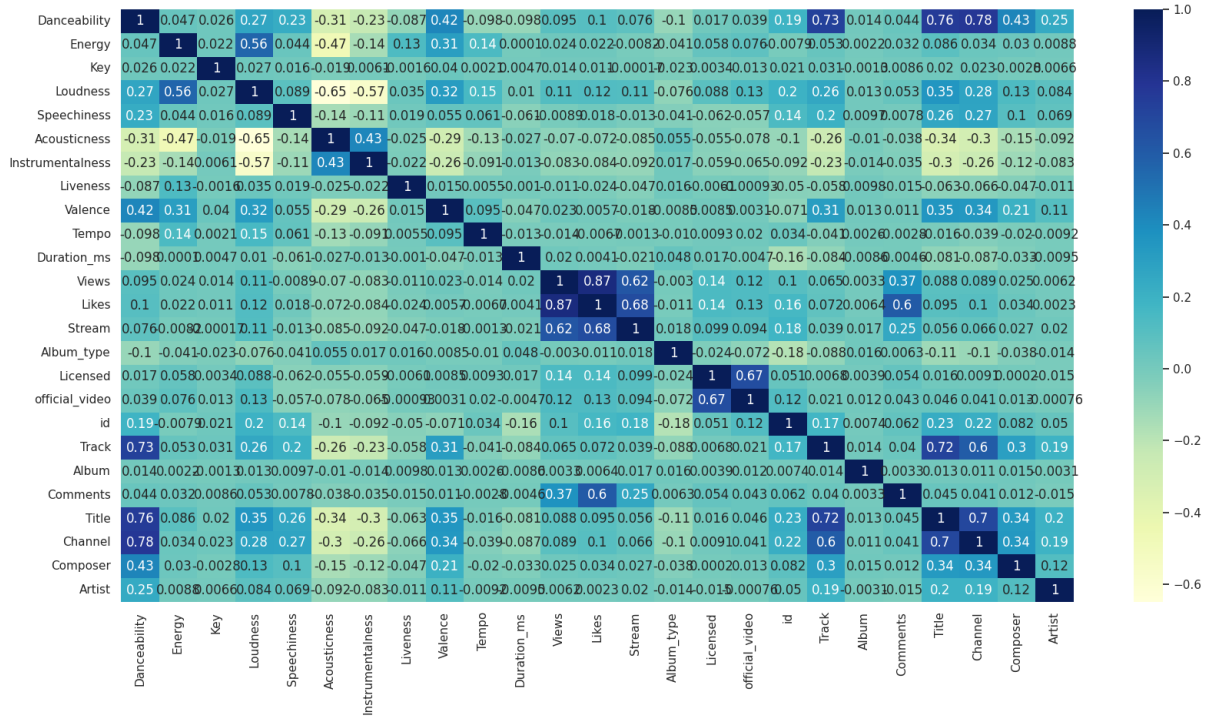
- Create a dictionary whose keys include each word / each person's name.
- For each key, its corresponding value is the average danceability of each occurrence.
  As an example, if the word "dance" appears in Rows 11, 2, and 6, and each of the rows have danceability 6, 2, and 1, then the score for "dance" is (6 + 2 + 1) / 3 = 3.
- Finally, we map each sentence to the average score of all words appearing in the sentence.

Fig. 1 Shows the heatmap of the correlation matrix between each pair of columns. We see that the correlation between danceability and the mapped scores are surprisingly high (0.78 for Cahnnel, 0.76 for Title, 0.73 for Track, 0.43 for Composer, and 0.25 for Artist).

Table 3 Correlation coefficient of text data with danceability

Track	Album	Comments	Title
0.73	0.014	0.044	0.76
Channel	Composer	Artist	
0.78	0.43	0.25	

Fig. 1. Heatmap of the correlation matrix



2.4 Other Notes

- For empty blanks, we simply fill them in with the average values of each column. Testing data and training data are filled with their respective averages to handle the shifting and scaling in data distribution.
- In this stage, we also tried applying 2nd / 3rd / 4th order polynomial transforms to raw data. This, especially 2nd order transformation, should make sense because most of the features, according to our domain knowledge, should showcase a maximum somewhere in between the two extremes. After all, neither “too fast” nor “too slow” tempos are suitable for dancing, right? :-) Despite a reasonable assumption, testing shows that 2 order transform might already be over complicating the model: While testing, we find a significant difference between training and validation data, and thus gave up on this approach.

3 FEATURE SELECTION AND ABLATION STUDIES

After mapping all columns into numbers in the previous stage, we then move on to decide which specific columns to select. While it is apparent that high correlation columns may be more helpful, we have still made several useful observations.

3.1 Removal of “Id”

While the feature “Id” has a relatively high correlation, it is artificially labeled and has nothing to do with the song itself. We removed "ID" directly considering the feature would just create a bias when training: If the training data were sorted according to their danceability, we would get a misleading relationship between the 2 features.

3.2 Analyzing Text Columns

3.2.1 Observations. Ever since we noticed the beautifully high correlation between certain text columns (such as "Title”, “Channel”, and “Track”), we had used them for every experiment. Not until a seemingly random submission one of our teammates made did we notice picking those columns might not always be the best choice. After multiple tries, we found that the performance was best when we only considered “artist” and “composer” from the text columns.

3.2.2 Hypothesis 1. This might happen because there is no guarantee whether the word set used in the training data and testing data would be similar enough, so any slight difference in the distribution may result in poor generalization. This phenomena may be especially severe when a word appears very few times in the training data but many times in the testing data. Say, for example, a word occurs only once in the training data, and we use the danceability for that specific instance to represent the score for that word. Then, we use the score, which was calculated from only one instance, to represent every instance with that word in the testing data. Certainly, this would lead to poor reliability. What’s worse, if the training data isn’t representative enough, neither validity is satisfied. In this case, the higher the occurrence of the word in the testing data, the more the error is amplified. This is the main reason why the performance wasn’t as good as expected.

3.2.3 Hypothesis 2. Besides the above reason, another risk for this algorithm is over-weighting unimportant words. Pronouns, articles, and prepositions appear rather frequently in natural language, but they may have less to do with the content of the words. When we calculate the score of each sentence, we do not weigh each word according to their importance. However, we believe this doesn't affect performance as much because titles and descriptions on YouTube are often concise and usually do not contain redundant words. We have listed the most common words in the training data to support our thought (Fig. 2).

Fig. 2. Most Common words in the training data

```
[119]  1 Counter(" ".join(dt).split()).most_common(100)

      [('music', 63),
       ('video', 36),
       ('christmas', 34),
       ('follow', 33),
       ('instagram', 32),
       ('solo', 29),
       ('facebook', 28),
       ('youtube', 27),
       ('twitter', 26),
       ('official', 24),
       ('love', 22),
       ('wan', 21),
       ('gon', 19),
       ('subscribe', 19),
       ('whatd', 19),
       ('spotify', 18),
       ('album', 16),
       ('song', 16),
       ('means', 16),
       ('videos', 15),
       ('things', 15),
       ('production', 14),
       ('performing', 14),
       ('listen', 13),
       ('producer', 13),
```

3.3 Why "Composer" and "Artists" Still Yield Good Results

Continuing from the above discussion, it is interesting to examine why "composer" and "artist" still give superior results. After some testing, we find that the two features have only 10 and 97 types respectively, which means that they are rather similar to "categorical" data instead of "text features". **However, by our algorithm, we not only treat them as discrete and independent categories; instead, the number each category maps to captures some hidden ordinal information about the danceability.** In our opinion, this might be the key reason our model worked quite well.

3.4 Remarks

While testing different combinations for feature selection, we have found that "good" and "bad" features have significant influence on the final performance. In our case, the loss varies from around 1.8 to 2.5 in extreme cases. By merely changing feature selection, a sequence of Kaggle submissions resulted in public losses: 2.00572, 2.05849, 2.43142, and 1.9509.

4 MODEL COMPARISON

After gaining sufficient knowledge about the features, we began to try out and compare several types of models. In the project, we tried three types of models, namely regression-based models, decision-tree-based models, and AdaBoost-based Models. We did random shuffling for all experiments. In most trials, we saved 20% of the training dataset for validation, and for some cases, we also used other methods introduced in class, such as k-fold validation or leave-one-out cross validation error. In the following sections, we elaborate on the details of these models and compare their performance.

4.1 Regression Based Models

In this model, we simply use linear regression to make predictions. Since the concept and implementation of regression is relatively straightforward, this is also our baseline model for preliminary testing. As discussed previously, since regression mainly works with numeric fields, we started by selecting only numeric columns. As we progressed in data preprocessing, we began to integrate the "text features" into the regression model and tested different feature combinations for better performance. Fig.3 shows the result of running this method. A total of 14 columns were selected, and the generated $\tilde{\mathbf{W}}$ is also shown in the figure. As for the result, the error of the local test is around 1.83, and the one of the public test is around 1.95444.

Fig. 3. Result of the linear regression model

```
parsing training data...
done
training...
DIM = 14
{{-1.40678}, {0.0337742}, {0.0364694}, {4.61957}, {-2.0351}, {-0.320214}, {
-1.64117}, {4.10578}, {-0.00801518}, {-8.85106e-07}, {3.36536e-10}, {6.0623
8e-08}, {2.67007e-10}, {0.847149}}
error: 1.83065
done
```

4.1.1 Optimization. One of our teammates proposed to round all floats to integers. This idea comes from the knowledge that the ground truth for danceability are all integers, which implies that as long as our models are in the correct direction of prediction, rounding may eliminate the slight deviation due to regression. Of course, if our model deviates away from the ground truth for more than 0.5 (more precisely, given an error x, if $x - \lfloor x \rfloor$ is within the range $(0.5, 1)$ or $[0.5, 1)$, depends on whether the error is larger or smaller than the ground truth), we become negatively impacted by rounding. Thankfully, our model was on the right track and improved a little (from public loss 1.95444 to 1.9509) after rounding. In fact, one of our most successful predictions came from regression + rounding.

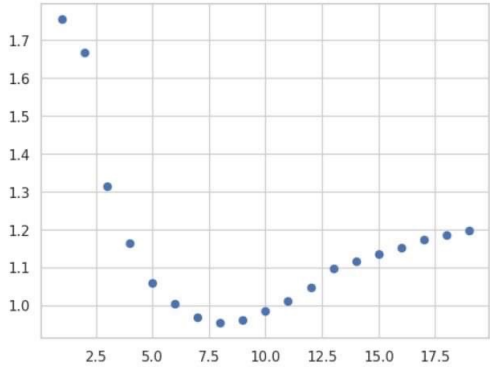
4.2 Decision Tree Based Models

Secondly, we try decision tree based models. We use gini-index, which is one of the most common criteria to evaluate purity, to decide how to partition the samples for each branch. Two experiments were done for this part.

4.2.1 Max-Depth Tuning. It is well known that the restriction of the hyperparameter "depth" plays an important role in training a decision tree model. A shallow model may result in insufficient complexity of the model, while an unnecessarily deep model leads to overfitting and a waste of computation resources. To estimate the scale of the "optimal depth", we tested the validation loss when

limiting the recursion depth from 1 to 20. As shown in Fig. 4, it is clear that the decision tree model converges quite quickly and exhibits a notable rise in error after the critical point 8. Therefore we focus on the range 8 to 13 for our parameter "maximum depth" in the following experiments.

Fig. 4. Relation between decision tree depth and MAE error.
X axis: depth of tree; Y axis: MAE error



4.3 Random Forest and Voting

As introduced in class, “ensembling” and “voting” are methods to increase the robustness of our model. Random forest is just like a strengthened version of decision tree that considers more possibilities of feature selection and sampling, and this, from our perspective, can in no way be worse than using a single tree. Oddly enough, Kaggle submission results showed a slight increase in MAE for random forest. Our guess is that the existing difference between the distribution of training and testing data may result in random forest “getting the wrong answer with a higher confidence”. With this distribution shift, a more widely varying sampling method may by some chance perform better than the more aggregated / comprehensive random forest.

4.4 Boosting Based Models

Finally, we have boosting based algorithms. Starting from the classic AdaBoost algorithm, we also went on to discover some cool online packages, such as XGDBoost, CatBoost, Gradient Boost, Light Boost and so on. Here we discussed CatBoost, which is the model that enabled us to reach our highest public and private score. Catboost, short for Categorical Boost, is similar to AdaBoost, except that it is designed for categorical data. The way it handles categorical data is not just one-hot encoding (e.g., mapping TRUE to 1 and FALSE to 0). Instead, it uses order encoding and incorporates categorical information through the training process. Extremely interestingly, this is actually highly similar to what we did for feature selection in Section 3.3. None of our teammates had background knowledge about CatBoost beforehand, so it was truly amazing that our original idea coincided with one of the most exceptional categorical machine learning algorithms. :-) As a final remark, we also tried gradient boost which didn’t perform so well and XGBoost which results in overfitting because it is a kind of tree boost .

4.5 Overall Comparison

Of the three types of algorithms, for both public and private data, boosting based models, CatBoost in particular, had the best performance. Followed by boosting based models were regression-based models and decision-tree based models. Table 4 shows a comparison for $E_{in}, E_{val}, E_{public}$, and $E_{private}$.

Table 4 Comparison of error for different model

model	E_{in}	E_{val}	E_{public}	$E_{private}$
regression	1.8587	1.8955	1.9509	2.18461
catboost	0.8036	0.8053	1.9509	2.15516
random forest	0.7918	0.8485	2.3177	2.74509

4.5.1 Local Testing vs. Kaggle Public. From Table 4., it is clear that random forest suffer a serious overfit: Their validation loss can be as low as 0.85 when tested locally, but the results on Kaggle were never lower than 2.2. On the other hand, Regression Algorithms overfit the least. They usually seem to perform poorly locally, but their performances do not decline too much after submitting to Kaggle.

4.5.2 Kaggle Public vs. Kaggle Private. While both Regression and CatBoost performed well on the public testing dataset on Kaggle, the latter performed significantly better than the former (and of course, rest of the models too) for the private testing dataset. This sheds light on the robustness of boosting algorithms, and how “weighing” important features may play an important role in Machine Learning. In feature selection, we see how weighing key features influences the error rate, and from boosting algorithms, we once again see how emphasizing on mistakes and adjusting accordingly may be a simple yet effective way of improving the overall performance.

4.5.3 Classification vs. Regression. Apart from the differences between models, our team also had a debate on whether a “classification model” or a “regression model” would stand out in this task. At first 2 of us firmly believed that the classification model would outperform the other because it seemed to us that “danceability” was a categorical feature with 10 possible values, instead of a to-be-predicted value that can possibly be any real number. However, after noticing how well the regression model works, we start to understand why regarding this as a regression problem also makes sense: the 10 categories for danceability actually have a clear ordinal relationship, and there *is* actually a physical meaning for a floating number. A float, say 4.7, would mean that the danceability is somewhere in between 4 and 5, and it might more likely be 5. Having this interpretation in mind, we thought of rounding the predicted regression values in Section 4.1, which more or less improved the performance of our model.

5 CONCLUSIONS AND FUTURE PROSPECTS

In this project, we study how regression-based models, decision-tree-based models, and boosting-algorithm-based models perform on the Spotify and YouTube Dataset, which involves numerical values, categorical data, and text features. After trying many models and obtaining various results, we find that outstanding approaches usually have a straightforward intuition / idea behind, often related to filtering out relevant information / key factors, or considering

domain knowledge. In addition, we have observed how diverse thoughts and capabilities from different team members may be helpful. For example, since we had different opinions on whether the problem was a regression or classification one, we were able to go on and explore different potential models individually.

In addition, we also observe that the most simple model can be very powerful: regression is one of the first models we learn in class, and it is incredible that it can perform so well.

For the final part of the report, though we unexpectedly won fifth place and the least overfitting award for the first stage of competition, we still have a lot of room for improvement.

For instance, instead of using an average-danceability based metric to convert sentences into numbers, we can further learn how to encode sentences to latent vector embeddings using Word2Vec or other common language models to preserve contextual information about the sentences.

Also, the mathematical theory behind (1) why a decision tree may be more effective than a random forest, (2) how the shift in training / testing data distribution would affect the prediction results, (3) why 2nd order linear transformation doesn't work as expected, and many more, may also be interesting topics for further study.

6 RESOURCES

The following references are provided with hyperlinks, since we handed in the report online.

- Decision tree ref1
- Decision tree ref2
- Random forest ref1
- Random forest ref2
- xgboost ref1
- xgboost ref1
- catboost ref1
- catboost ref2
- catboost ref3
- catboost ref4