Machine Learning Final Report

b08902005 資工四 黃品淳
b09502158 機械三 詹宜昇
b09502046 機械三 胡睿宸

I. Abstract

In this survey paper, we try to predict a song's danceability based on its 28 features with machine learning. First of all, we analyze the meaning of these features and visualize them to assist in the subsequent selection of data preprocessing methods and result analysis. We try many different machine learning models, and mainly study at four types of them - SGD linear regression, random forest, SVM, and CNN models. In Section IV, we define the model evaluation criteria and analyze the model accordingly. Then compare models by those analysis. Our most recommended model is SGD linear regression, which has advantages such as efficiency and interpretability, but still has room for improvement in preprocessing and choosing of the features.

II. Data Analysis

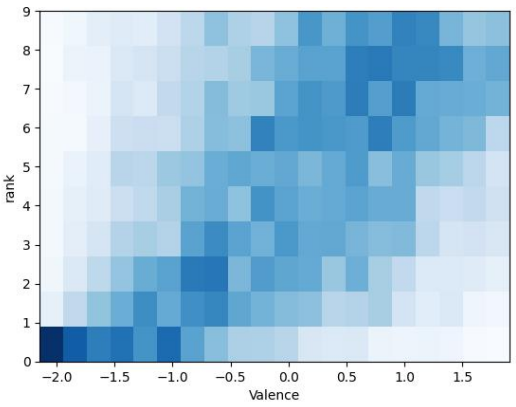
Before designing models, we read the training dataset and dataset description on Kaggle. Then we use this background knowledge and the distribution of each feature on the training set for data analysis. And speculate on its impact on "danceability" after visualizing each feature. We have divided these 28 features into five categories, which will be introduced one by one below.

1. Numeric Features

Energy, Key, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Duration_ms, Views, Likes and Stream are numeric features. We observe the average of features in the data of different danceability and make 2D histogram. The average results are shown in the table below.

We marked feature rows with more obvious numerical trends and speculate that they have a greater impact on danceability. Taking "Valence" as an example, which describes the musical positiveness conveyed by a track, we guess that a song with higher valence tends to have higher danceability. This trend can also be observed in the 2D histogram of Valence as shown in the figure below.

feature \ rank	0	1	2	3	4	5	6	7	8	9
Energy	0.242	0.337	0.357	0.346	0.349	0.356	0.355	0.358	0.349	0.301
Key	4.955	5.127	5.115	5.256	5.346	5.208	5.113	5.284	5.487	5.398
Loudness	-12.5	-8.24	-7.52	-7.31	-7.12	-7.08	-6.71	-6.53	-6.46	-6.55
Speechiness	0.058	0.075	0.076	0.087	0.087	0.09	0.096	0.104	0.116	0.16
Acousticness	0.33	0.176	0.127	0.115	0.102	0.086	0.075	0.064	0.051	0.039
Instrumentalness	0.248	0.075	0.047	0.033	0.033	0.034	0.023	0.023	0.023	0.02
Liveness	0.048	0.042	0.042	0.045	0.036	0.036	0.028	0.025	0.019	0.019
Valence	0.266	0.411	0.456	0.511	0.542	0.549	0.598	0.625	0.639	0.63
Tempo	121.3	127.5	125.2	123.1	121.7	120.8	118.9	120.1	116.6	115.1
Duration_ms	2E+05	2E+05	2E+05	2E+05	2E+05	2E+05	2E+05	2E+05	2E+05	2E+05
Views	5E+07	7E+07	6E+07	8E+07	8E+07	8E+07	1E+08	1E+08	1E+08	1E+08
Likes	4E+05	5E+05	4E+05	6E+05	6E+05	6E+05	7E+05	9E+05	8E+05	9E+05
Stream	9E+07	1E+08	1E+08	1E+08	1E+08	1E+08	1E+08	2E+08	1E+08	1E+08



2. Categorical Features

Licensed, official_video and Album_type are categorical features. We observed the line chart of the true/false ratio of the first two features under different danceability ranks and found that it fluctuated up and down in a small range, indicating low correlation.

The part of Album_type has four categories: album, single, compilation and N/A, which are displayed in a bar graph as follows, and the above four categories are blue, orange, green and red in order. It can be seen from the figure that the single category is generally positively correlated with danceability, and the other two categories are negatively correlated, but neither is obvious, which means that the reference value of this data may be small.

3. String Features

Track, Album, Title, Channel, Composer and Artist are string features. These features consist of a small number of words and are nouns rather than descriptive sentences. Judging from the physical meaning of the features, when the value of the string feature is the same, there may be a closer danceability rank. Therefore, we print out the number of repeated values of each string feature. Composer and Artist have the largest number of repetitions. To be precise, the former has only 11 different values, and the latter has 97. Relatively speaking, Track and Title have 12,977 and 13,281 values respectively, which are likely to be less helpful to model prediction.

4. Dropped Feature

The description feature consists of a large block of text, which is difficult to process. From the meaning of the data, we reasonably guess that link-type data such as Uri, Url_spotify, and Url_youtube are not helpful for judging danceability. Therefore, the data in these four fields are usually dropped during pre-processing.

III. Model Design

1. CNN model

a. Reason

(1) Popularity:

CNN model is one of the most popular machine learning models. In recent years, it is also often used when dealing with big data. As a result, packages such as pytorch, tensorflow and sklearn provide a large number of related tools to assist the training and testing processes.

(2) Performance:

The CNN model is mostly used for image input problems, and usually has a high accuracy rate. We're curious about how it performs for pure numerical data. We also observe changes in MAE by adjusting parameters such as the number of layers of the CNN model and the learning rate.

b. Pre-processing

For the CNN model, we pre-process the data in two ways. They differ only in the processing of the Description feature. For numeric features, we fill missing values with the mean of the feature. For categorical features, we map true/false to 1/0, and map Album_type feature to 0~3. Missing values are filled with intermediate values or in a class of their own. For the six string features, we process them together and map them to a ten-dimensional vector. As for the Description feature, we drop it directly in method 1, and map it to a 50-dimensional vector in method 2, similar to string features.

c. Experiment Settings

The CNN model we use is constructed by ourselves with torch.nn package. For method 1 and 2, hyperparameters are: batch_size=256, optimizer=SGD, epoch=600, learning rate= 0.002*max(0.005, (1 - (epoch / epochs))). The model architectures of method 1 and 2 are shown in the left and right figures respectively. To shorten the training time, we use the GPU acceleration of google colab.

```
MyModel(
  (numeric_layer): Linear(in_features=13, out_features=64, bias=True)
  (boolean_layer): Linear(in_features=2, out_features=32, bias=True)
  (categorical_layer): Embedding(4, 10)
  (string_layer): Embedding(500, 10)
  (text_layer): Embedding(500, 10)
  (conv1d): Conv1d(10, 32, kernel_size=(3,), stride=(1,))
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (cnn): Sequential(
    (0): Conv1d(1, 32, kernel_size=(3,), stride=(1,))
    (1): ReLU()
    (2): Flatten(start_dim=1, end_dim=-1)
    (3): Linear(in_features=4352, out_features=512, bias=True)
    (4): ReLU()
    (5): Linear(in_features=512, out_features=64, bias=True)
    (6): ReLU()
  )
  (output_layer): Linear(in_features=64, out_features=1, bias=True)
)
```

```
MyModel(
  (numeric_layer): Linear(in_features=13, out_features=64, bias=True)
  (boolean_layer): Linear(in_features=2, out_features=32, bias=True)
  (categorical_layer): Embedding(4, 10)
  (string_layer): Embedding(500, 10)
  (text_layer): Embedding(500, 10)
  (conv1d): Conv1d(10, 32, kernel_size=(3,), stride=(1,))
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (cnn): Sequential(
    (0): Conv1d(1, 32, kernel_size=(3,), stride=(1,))
    (1): ReLU()
    (2): Flatten(start_dim=1, end_dim=-1)
    (3): Linear(in_features=5376, out_features=512, bias=True)
    (4): ReLU()
    (5): Linear(in_features=512, out_features=64, bias=True)
    (6): ReLU()
  )
  (output_layer): Linear(in_features=64, out_features=1, bias=True)
)
```

d. Post-processing

Set the prediction results less than 0 to 0, and those greater than 9 to 9. No rounding is done.

2. SGD Linear Regression

a. Reason

(1) Efficiency:

The major advantage of using SGD (Stochastic Gradient Descent) is its efficiency, which is basically linear to the number of training examples. Also linear regression has a closed-form solution, known as the norm equation, which allows for fast training and prediction. Here the model we choose is a variant of simple linear regression where it can handle large datasets efficiently by performing updates on subsets of the data instead of the entire dataset at once.

(2) Baseline model:

Linear regression can be viewed as baseline model which can be used to compare the performance with other sophisticated or complex model such as SVM or neural network. It provides the baseline for evaluating the model power of additional features or more complicated algorithm.

b. Pre-processing

For SGD Linear Regression we use all numeric features and categorical features for training and prediction. For numeric features, we fill the missing value with the corresponding mean, as for categorical features we select one hot encoding to transform the original data into boolean type.

c. Experiment Settings

The model we use is SGDRegressor from sklearn package, the model implements a plain stochastic gradient descent learning routine which support different loss function and penalties to fit linear regression models. The final hyperparameters we select after several tuning by minimizing the cross validation error are “loss='huber', penalty='l2', alpha=0.001, max_iter=10000, tol=0.00001, random_state=721, learning_rate='constant', eta0=0.00001”.

d. Post-processing

Set the prediction results less than 0 to 0, and those greater than 9 to 9. Round the result value.

3. Random Forest

a. Reason

First, Random Forest is a non-parametric model, meaning it does not assume a specific underlying distribution of the data. This flexibility makes it suitable for handling complex relationships and interactions between features. Second, techniques like bagging and random feature selection, which reduce the risk of overfitting.

b. Pre-processing

For missing values of numeric features, we fill them with mean of the feature. For categorical feature, we fill the missing value with false boolean and turn them into 1 and 0. As for Album_type, we find out that the ratio of each type are almost the same, so we encode it with frequency value.

c. Experiment Settings

We use random forest regressor model from sklearn package. We didn't use classifier because after several experiment we observe the model of "regressor" outperform "classifier" in this project, and we split the training data into three parts, first is the training data, second is validation data, and the last one is the test data that haven't been seen or touch by the machine. We want to see if the in sample MAE is close to Public MAE.

The final hyperparameters we select after several tuning by minimizing the cross validation error are:'max_dapth ': 100, 'n_estimator':900, 'random_state':42.

d. Post-processing

Set the prediction results less than 0 to 0, and those greater than 9 to 9. Round the result value.

4. SVM

a. Reason

(1) Robust to outliers:

SVM is less sensitive to outliers compared to other algorithms such as linear regression or neural networks. It achieves this by maximizing the margin between the decision boundary and the support vectors, making it more robust to noisy data.

(2) Regularization:

SVM possesses a regularization parameter (C) that helps control the trade-off between maximizing the margin and minimizing the classification error. This regularization parameter allows SVM to combat overfitting.

b. Pre-processing

Same as the pre-processing of Random Forest.

c. Experiment Settings (Model Selection & Parameter Chosen)

First we use the SVM classifier and regressor model from sklearn package. The kernel we use are linear and rbf, and we split the trainnig data into three parts, first is the training data, second is validation data, and the last one is the test data that haven't been seen or touch by the machine. We want to see if the in sample MAE is similar to Public MAE.

The final hyperparameters we select after several tuning by minimizing the cross validation error are: 'kernel':rbf,'C':1.4, 'gamma':0.1

d. Post-processing

Same as the pre-processing of Random Forest.

IV. Result Analysis

1. Evaluation Metrics

To measure the quality of our machine learning models, we design metrics to evaluate important model properties. Additionally, we visualize part of them in different ways. First, we measure model performance by public score and mean absolute error on train and validation set. The train and validation sets are cut from the training data through train_test_split in the sklearn package, and the parameters are set to: test_size=0.1, random_state=0. After that, we visualize the classification result of validation set by confusion matrix.

Second, we evaluate model efficiency by training time and testing time. Third, we try to explain the meaning of the models' weights after training. Last but not least, we briefly introduce the general characteristics of this type of model, including generalization ability and scalability.

2. CNN model
   a. Scores (method 1 / method 2)

Public MAE	validation MAE	in sample MAE	Private MAE
2.07002 / 2.06762	1.6454 / 1.6428	1.5223 / 1.4763	2.39761 / 2.41569

   b. Analysis

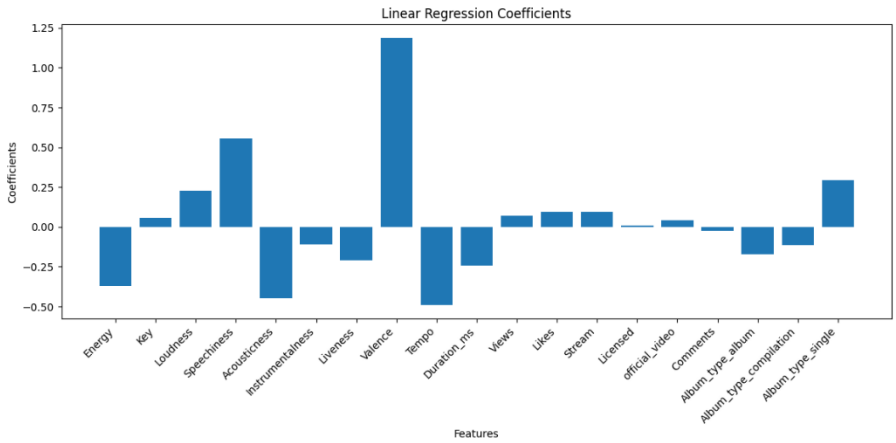
First, after using GPU acceleration, the training time of methods 1 and 2 are 5 minutes 16 seconds and 6 minutes 38 seconds, respectively. However, it takes 1 to 2 hours to train with only CPU. Obviously, the resource consumption is higher than our other models. In terms of scores, Description feature didn't make the model better, which may be related to the data processing method or the low correlation between itself and danceability. From the MAE of validation and in sample, the overfitting of method 2 is more obvious, which is also reflected in the gap between public and private score. From the perspective of interpretability, both models have more than two million parameters, and it is difficult to interpret their respective meanings.

3. SGD Linear Regression
   a. Scores

Public MAE	validation MAE	in sample MAE	Private MAE
2.08837	2.00155	1.97483	2.4531

   b. Analysis

- After comparing the MAE between using squared_error and huber as loss function we select huber as our final decision since it focus less on getting outliers correct by switching from squared to linear loss past to epsilon, where epsilon is the parameter to determine the threshold at which it becomes less important to get the prediction exactly right.
- First, SGD Linear regression is a very efficient model, the total training time only took 1.19s for this dataset. It has the advantage of computationally efficiency which allows this model to be used for larger dataset. Secondly, the interpretability is much better compared with our other model, since we can easily tell which feature has stronger impact on the prediction outcome by analyzing the model's coefficient. The model's coefficients are shown in below bar chart which reveal Energy, Loudiness, Speechiness, Acousticness, Valence and Tempo have stronger impact on danceability which is approximately same as our conclusion after data analysis in section II.



4. SVM Regressor & Random Forest
   a. Scores (SVM/RF)

Public MAE	validation MAE	in sample MAE	Private MAE
2.10389/2.17453	1.7885/1.81362	1.7864/1.71166	2.48555/2.43609

   b. Analysis

- From the result, we can infer that the regressor perform better than classifier in this project, since the goal of this project is to predict the danceability rank from 0 to 9, not to classify the type. Regressor will output continuous number. On the other hand, we find that the SVM training time is longer since we use both linear and rbf kernel to see which one is better and their parameters, as for random forest we only choose the number of decision tree and depth, then we can surely compare with our result. It has shown that the training time for SVM and RF are 922.72s, 1216.83s, respectively.
- For interpretability, the decision boundary in SVM is represented by a hyperplane, making it easier to understand the influence of different features on the classification/regression outcome.
- For scalability, Random Forest is known for its scalability. It can handle large datasets efficiently due to its parallelization capabilities.

V. Conclusion

After all the experiment and analysis we have done in this project, we can conclude that CNN model has the best performance on prediction but it falls short on efficiency and interpretability. As for SGD linear regression, although it doesn't perform as well as CNN in terms of prediction, it is much efficient and performs well. We also reasonably believe that it has much room to improve, since we didn't use complete features on model training. Therefore, among the models discussed in this survey paper, we recommend the SGD Linear Regression model the most.

To improve the prediction using SGD Linear regression, we can use the features we dropped previously such as string features and description features. Although as we mentioned in section IV, these features didn't help too much on prediction when we used CNN. However, this might result from our encoding technique. By selecting other encoding techniques, we believe these features can improve our model.

VI. Work Loads

黃品淳：Abstract, Data Analysis, CNN model
詹宜昇：SGD Linear Regression, Conclusion
胡睿宸：Abstract, SVM, Random Forest

VII. Reference

1. Sklearn.Linear_model.SGDRegressor. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
2. (2020, February 5). [機器學習]sklearn-SVM 調參. https://blog.csdn.net/ztf312/article/details/98594359
3. Weiaweiww (Ed.). (2022, August 8). Sklearn 的系統學習-隨機森林調參. https://huaweicloud.csdn.net/63807e62dacf622b8df88cfc.html
4. Polanitzer, R. (2022, March 23). The Minimum Mean Absolute Error (MAE) Challenge. Medium. https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031
5. 李弘毅教授的機器學習講義