ML final project

b10902025, b10902027, b10902044

June 2023

1 Introduction

The report wants us to study at least 3 ML approaches. However, since we performed our project through several steps and there are different choices in each step, every approach may be a combination of these steps. Hardly could we have enough space to write all these approaches in 6 pages. Hence, we would explain different choices in different steps and discuss the efficiency and other perspectives against different models in the fifth section.

2 Data preprocessing

2.1 Dealing with missing values

Both train and test data contain some missing data, which is similar to any real-world data. We came up with several methods to deal with this problem:

1. Delete the data including missing values.
   Deleting these data can easily filter the data with missing values. However, we observe that almost all rows and columns have some missing values in the dataset, so this approach may result in loss of valuable information.

2. Fill in the mean.
   Since the missing value in the data set almost happen in different features. By filling missing values with the mean, we can retain as much data as possible and avoid potential biases that may arise from deleting rows or columns with missing values. However, mean values can be sensitive to outliers, as it takes into account the magnitude of all values. Outliers can significantly influence the mean, pulling it towards extreme values.

3. Fill in the median.
   To avoid the situation that mean values affect by extreme values, we can use the median. Median value of the feature is less influenced by extreme values, maintaining the same central value without affecting the spread or variability.

To decide which method to use for missing values, we can take the following example into consideration.

$$y = |x - 1| + |x - 2| + |x - 6|$$

When x=2, the median of 1, 2, 6, y has a minimum value.

$$y = (x - 1)^2 + (x - 2)^2 + (x - 6)^2$$

When x=3, the mean of 1, 2, 6, y has a minimum value.
In conclusion, to minimize mean squared error, we would fill in the mean. To minimize mean absolute error, we would fill in the median. Hence, we mostly fill in median values in this project.

2.2 Dealing with text values

1. artist, composer
   Our assumption is that musics created by the same artist or composer tend to have similar danceability. To have numeric value to represent the data, we calculate the mean danceability of the same artists'(composers') songs and put the values in the entries. However, when some artists or composers have too few musics compared to our dataset size, they may become noise data. To address this issue, we consider these instances as missing values and fill in null. They would be seen as missing values to be processed.

2. key, channel
   We tried to deal with these 2 features using the method mentioned above. However, since channel isn't very related to danceability, it may result in overfitting. There is also limited improvement using this strategy to deal with key.

3. description
   There are some techniques to transform text data into numerical representations that can be processed by our models. We tried:

   (a) Vectorization
   Vectorization converts text data into numerical vectors, where each dimension represents a unique term and the value represents the measure of its presence or importance, each text is represented as a vector in a high-dimensional space.

   (b) Word Embedding
   Word embeddings are dense vectors that represent words in a continuous vector space with lower dimensions and capture semantic relationships, word similarities, and contextual understanding, enabling models to better comprehend and generalize from text data.

   However, these method have a limited improvement. To be honest, we didn't figure out a suitable way that can significantly decrease E_{out}.

2.3 Standardization

For those features with numerical and continuous data, we can standardize them to improve performance. We think we should do this because:

1. Balance feature influence and handling different units:
   Features with larger scales, ranges, or different units would potentially bias the results. We think standardization eliminates such biases and provides a more balanced representation of the data, making it easier to compare and interpret their relative importance or weights in the model.

2. Enhancing Model Performance:
   Standardization can improve the performance of certain machine learning models, such as those based on distance calculations or regularization. Models like support vector machines using distance metrics to make predictions can benefit from this. Besides, since we have tried models such as neural networks, which rely on techniques like gradient descent to find optimal model parameters. Standardizing the data can make the optimization process more efficient, as the algorithm can take larger steps and converge faster.

3 Feature selection

3.1 Correlation coefficient

We calculate the correlation coefficient between danceability and other features. By doing this, we could gain insight into the dataset. The correlation coefficient is as follows:

```
Correlation Coefficient between Danceability and Energy : 0.04449445221363738
Correlation Coefficient between Danceability and Key : 0.10296993187094751
Correlation Coefficient between Danceability and Loudness : 0.25265159896975975
Correlation Coefficient between Danceability and Speechiness : 0.20944431391913546
Correlation Coefficient between Danceability and Acousticness : -0.2816535032304039
Correlation Coefficient between Danceability and Instrumentalness : -0.21272583377160034
Correlation Coefficient between Danceability and Liveness : -0.07715274963108701
Correlation Coefficient between Danceability and Valence : 0.3913676767827113
Correlation Coefficient between Danceability and Tempo : -0.09205655788660638
Correlation Coefficient between Danceability and Duration_ms : -0.08487327592453672
Correlation Coefficient between Danceability and Views : 0.08783172778771559
Correlation Coefficient between Danceability and Likes : 0.0947353078851534
Correlation Coefficient between Danceability and Stream : 0.06916354826167787
Correlation Coefficient between Danceability and Channel : 0.5718710274038936
Correlation Coefficient between Danceability and Composer : 0.42150461423306756
```

Some results went beyond our imagination. For instance, we thought that the higher energy is, the higher danceability would be. However, the correlation coefficient between them is only 0.04. Another example is liveness. We thought that liveness and danceability have positive relationship, but their correlation coefficient is negative. The reason is that the correlation coefficient is limited to linear relationships. If the relationship is polynomial or kernel, the correlation coefficient may be low. However, for these values, if we choose an appropriate model, these features may still be useful.

3.2 Experiments

Since the correlation coefficient may not represent non-linear data, we have to conduct experiments to select our feature. We have tried:

1. only numeric values: This is the most basic feature selection. However, these features aren't enough to interpret danceability, resulting in $E_{out} > 2$.

2. numeric values + 'Composer', 'Artist': After dealing with these text values, we had a huge improvement, resulting in $E_{out} \approx 1.9$.

3. numeric values + 'Composer', 'Artist', 'Channel': Adding channel results in overfitting. The E_{in} and E_{val} may be lower than 1.5 but E_{out} would be worse than the previous case.

4. 'Energy', 'Loudness', 'Speechiness', 'Acousticness', 'Instrumentalness', 'Liveness', 'Valence', 'Tempo', 'Durationms', 'Likes', 'Composer', 'Artist', 'Key': After several experiments, we reached to a conclusion that this is the best combinations of features we should select.

4 Splitting data

We split the training data randomly into 2 parts, 80 percent is for training while the other 20 percent is for validation. We only train the data using the training set and evaluate the performance of our model using the validation set. We claim that there are the following advantages:

1. Model performance evaluation: Having a tool to directly assess the performance of our model, we could test different parameters and compare different models more conveniently.

2. Preventing data leakage: If we use the whole dataset to train, the validation data may inadvertently influence our model. By splitting the model, we ensure that the model's performance is assessed on genuinely unseen data.

5 Six approaches

5.1 Linear Regression

Linear Regression is a simple method and comes to our mind first. Though it is simple, we figured out that as long as we performed good data preprocessing, the results could still be not bad. If we only select numeric features, $E_{out} \approx 2.08$, it is already enough to pass the TA baseline. After we add artist and composer into our features, $E_{out} \approx 1.9$.

1. **Efficiency and scalability:** The total runtime of this algorithm is less than 1 second. Hence, we claim that linear regression is a suitable model to handle large datasets and multiple features.

2. **Interpretability:** It has easily understandable coefficient values, where each coefficient represents the impact of a feature on the target variable. Also, the magnitude and sign of the coefficients provide insight into the direction and strength of the relationships. This transparency allows for a clear interpretation of the model's predictions and identification of important features.

5.2 Ridge Regression

Since a member of our group took a course about data analysis, we also tried Ridge and Lasso regression, expecting to deal with possible overfitting. However, we found out that when using linear models, there is rarely no overfitting. Although the regularizers can guarantee the model wouldn't become worse, the improvement isn't obvious. We tried different alphas 0.01, 0.1, 1, 10, 100, 1000 and find out that E_{val} only differs in a 0.0001 scale, which indicates the regularizers don't play an important role in ridge regression. With only selecting numeric features, both linear and ridge regression have $E_{in} \approx 1.98$ and $E_{val} \approx 2.03$.

5.3 Stepwise Regression

Since we didn't figure out a good pattern to select features, we performed stepwise regression using R language. It automatically selects the most relevant subset of features from a larger set of potential predictors and iteratively adds or removes features from the model based on certain criteria until an optimal subset of features is determined. However, the result isn't significant since it always selects every feature.

5.4 Support Vector Regression

To deal with possibly overfitting, we think the SVR with the concept of support vector machine can be a choice. In the beginning, we use the model with only numeric features, the rbf kernel and regularization C = 0.15, $E_{in} = 1.69$, $E_{val} = 2.06$ $E_{out} = 2.06$, after using transformed text features $E_{in} = 1.59$, $E_{val} = 1.65$, $E_{out} = 1.9$ which is a great improvement. Besides, we also try different regularization constants and kernels to see their effect. We figure out that using other kernels leads to a worse result and when using rbf kernel, $C = 0.15$, $epsilon = 0.5$, we obtain the best result with $E_{in} = 1.66$, $E_{val} = 1.68$ $E_{out} = 1.86$

1. **Efficiency and scalability:** In our implementation we use SVR in sklearn library with rbf kernel, which time complexity is approximately $O(n^2 \times m)$, where n is the number of training instances and m is the number of support vectors. The kernel trick allows SVR to work in the high-dimensional feature space without explicitly computing and the regularization come to deal with overfitting, both two techniques make it practical for training the model.

2. **Interpretability:** SVR's interpretability is generally lower than linear regression, as the prediction is based on the relationships between support vectors and their associated coefficients, which may be more complex to interpret and explain.

5.5 Random Forest

We learned decision tree in class and thought it was a cool algorithm. Therefore, after searching for information, we tried random forest, which combines several decision trees, to predict our data. We found out that it is a time-consuming model and it may lead to overfitting very easily. E_{in} could be extremely small while E_{val} would be quite larger. Therefore, we have to set the parameters "max depth" and "min sample leaf" carefully. After several experiments, there is the best $E_{out} = 1.89$ when "max depth" $= 10$ and "min sample leaf" $= 4$ where $E_{in} = 1.54$ and $E_{val} = 1.58$. The results are slightly better than linear regression, but worse than support vector regression.

1. **Efficiency and scalability:** Random Forest can be computationally efficient since it can handle large-scale datasets effectively, as it can parallelize the training process across multiple processors or machines. However, prediction time can be slower for large forests or complex trees due to the need to traverse multiple decision tree.

2. **Interpretability:** The interpretability of Random Forest is generally lower than linear regression, as it lacks the simplicity and direct interpretability of coefficients.

5.6 Neural Network Model

We have tried several neural network models such as NNR and CNN. We found out that NNR has the best performance. When epoch $= 5$, batch size $= 32$ and using "adam" optimizer, we get $E_{in} = 1.64$, $E_{val} = 1.66$ and $E_{out} = 1.84$.

1. **Efficiency and scalability:** Neural Network models can handle large datasets and scale well with increasing data size. However, it would be computationally intensive for large and complex architectures if training too large or too deep, requiring significant resources and training time.

2. **Interpretability:** Neural Networks are often considered less interpretable than linear regression due to their complex architecture and multitude of interconnected nodes.

5.7 Best Model

After several tests of these models, Neural Network Model performs the best. We believe that this is because not all the features are linear to danceability, some may have complex, nonlinear relationships. The advantage of Neural Network Model is that it is capable of learning intricate patterns and leveraging their ability to model nonlinearities to achieve better performance which cannot be done well by other models. However, it is very hard to interpret and understand the learned relationships, the inner workings of neural networks, especially in deep architectures, may lack transparency.

6 Dealing with overfitting

After some tries, we found that there is always a huge gap between E_{in} and E_{out}. Though we make E_{in} smaller, expecting E_{out} to have same tendency, the gap still exists or becomes even bigger. Hence, dealing with overfitting carefully is second to none:

6.1 Cross validation & Adaboost

We use cross validation with 5 fold in different models. Besides, we also tried using 50 weak estimators and 0.1 learning rate to train the model with adaboost techniques. However, when it comes to the same model with same parameters, both method could only make E_{out} decline at most 0.01.

7 Other adjustments

7.1 Rounding

Suppose the result of a danceability is 4.7. Since danceability is an integer between 0 and 9, if $P(y \leq 4) < P(y \geq 5)$, rounding would make our results better and vice versa. Since our prediction is closer to 5, we could reasonably infer that $P(y \leq 4) < P(y \geq 5)$. Therefore, we should round our results.

7.2 Blending

Since we tried several models, some models are so close but still can't lead to an optimal result and we want to make good use of them. One opinion that comes into mind is to calculate the average of these models and result in our new prediction. We found out that when blending, there is limited improvement if we use the same model with different parameters. What really improves our result is that we blend 3 different regressions with $E_{out} \approx 1.85$. We reached our optimal $E_{out} = 1.83$. Note that when we blend our results, we would use the original numbers to calculate the average first, then round the numbers.

8 Conclusion

We think that using E_{in} and E_{val} to estimate E_{out} plays a crucial role in the project. Generally, when E_{in} and E_{out} are close enough, this means that overfitting isn't obvious. However, in order to make them close, the trade off is that the values would get larger, also resulting in a larger E_{out}.
We also realize the importance of processing the data. Whatever models we tried and whatever parameters we set, the improvement of E_{out} is within 0.03. What really makes a difference is to deal with text values such as "artist" and "composer". It made our E_{out} improve from 2.06 to 1.88.

9 Work load

Generally, 044 preprocesses the data such as selecting features and doing standardization. 025 tries different models and figures out the best parameters. 027 tries other adjustments and tries to deal with overfitting. However, since we usually work in person together, we would discuss the E_{in} and E_{val} together, figure out what adjustments we can make, and determine whether we should submit the predictions.

10 Reference

1. sklearn: We use this package in python to calculate MAE, perform standardization and conduct all the models except neural network.
2. keras: We use this package in python to conduct all neural network models.
3. pandas: We use this package in python to read and write csv file, especially when blending.

x