Predicting song popularity using track features

Miguel Cozar Tramblin A20522001
Carlos Munoz Losa A20521562

16th November 2022

Abstract

Regression problems are common in Deep Learning. They are use to predict house prices, stock markets, and mostly everything that comes into our minds. One field that we consider of interest is music. It is crucial for the industry to estimate in advance how popular songs are going to be. For this purpose, we have found several papers. Most of them employ machine learning techniques, but we will try to approach it with deep neural networks starting from different sets of data, studying the viability of all of them and giving details about our experience.

1 Problem Statement

The main problem is to estimate how popular a song is going to be according to its features. These features to be included are distributed in three main groups: general information about the song, music features analysis, and lyrics analysis.

2 Previous papers

The paper that triggered us to start was [1]. This paper tries to solve the problem using four machine learning algorithms (MLAs): Linear Regression, Random Forest Classifier, and K-means Clustering. We considered their approach as admissible, but improvable.

After a bit more of research we discovered [2]. This paper is much more in line with what we wanted to do. They approach the problem by using a big neural network which has as inputs: raw audio, lyrics, high-level audio features and additional metadata.

Apart from the previously work motivator papers, we will be mentioning more papers among the report.

3 Data

This sections presents the data sources, how were they obtained and treated.

3.1 Million Song Dataset

The Million Song Dataset is an open source collection of songs audio features for a million contemporary songs [3]. They have a total amount of 280 GB of data, and they provide a subset of 10,000 songs (1.8 GB). We started working with this subset to get familiar with the platform, to finally reach a final number of 5,648 songs. The reduction of the number of songs is caused by the absence of the target value in some of the cases.

To retrieve and organize the data, we created a class "DataGatherer", based on a script provided in their documentation, that retrieved all the possible data of these songs, 55 features, of types array, string, integer and float. From this source we built two of the datasets previously explained: the General Information Dataset and the Sections/Tatums dataset. An explanation of each feature is provided in Annex A. The gatherer is designed to have scalability, so more data could be introduced in a future from the big song dataset.

3.2 Lyrics Dataset

The lyrics dataset was a bit more difficult to obtain. The Million Song Dataset is partered with Musixmatch [4], but they provide limited access to songs lyrics. Therefore, we searched for an alternative and found genius.com [5]. This web provides an API to retrieve songs' lyrics. We used it to retrieve the raw text of each song, while the target feature (songs' popularity) was still extracted from [3].

After that, we could face the problem as a Natural Language Processing one. We faced to main challenges.

- Lyrics Language: songs can be in multiple languages, but for treating them we need as much uniformity as possible. We leveraged a pre-trained model for language detection [6]. If language detected was not English, we proceed to automatically call Google Translator and retrieved the raw text in English
- From raw text to features: this was another challenge. Deep learning networks work with numbers, so we had to be as meticulous as possible to get a good training dataset. For this purpose, we leveraged another pretrained model from HuggingFace called "all-mpnet-base-v2", whose information is further explained in [7]. We encoded our raw text with this model, generating 768 dimensional vectors, ready to be trained.

4 Approach

Our approach was to test the dataset separately. The complexity of the problem joining every feature was considered to be too complex. We had 768 features from lyrics, 2,118 from the musical analysis of the songs, and 8,074 from the general data (previously explained). This makes the problem too complex for our resources.

Moreover, the number of features was still too big to deal with it, and as seen in lectures, we have to start from the simplest solution to the most complex. For approaching the problem, we have leveraged PCA dimensional reduction. PCA is a widely used dimensional reduction method that uses eigenvalues, eigenvectors and co-variances of a big dataset to build principal components (new dimensions). It is fast and more reliable than others like TSNE. It has previously used for musical problems such as song lyrics

treatment [8]. We also considered kernel PCA to fight against non linearity, but the optimization of parameter γ made the the problem more complex.

Our analysis consists of a dissection of each of the problems. We will make a trial with all the datasets joined. We will analyze the usability of the three datasets and the possibility of designing an "ensemble regressor" weighting the outputs of the model.

5 Model Implementations and Results

This section discusses the implementation details and results obtained for each problem that we have explained. [1]

5.1 Lyrics Dataset

The lyrics dataset resulted to be the "a priori" most promising dataset. We wanted to illustrate a map of the songs according to their popularity, to observe if the regression problem was promising or not. For that purpose, we used PCA to reduce the initial 768 dimensions to 2 and 3D. Fig. 1. shows that there are certain regions were there is a density of popular songs. This demonstrates that the semantics of the lyrics may have influence on the popularity of a song. The train-test-validation partition is of 4046-1125-450 samples.

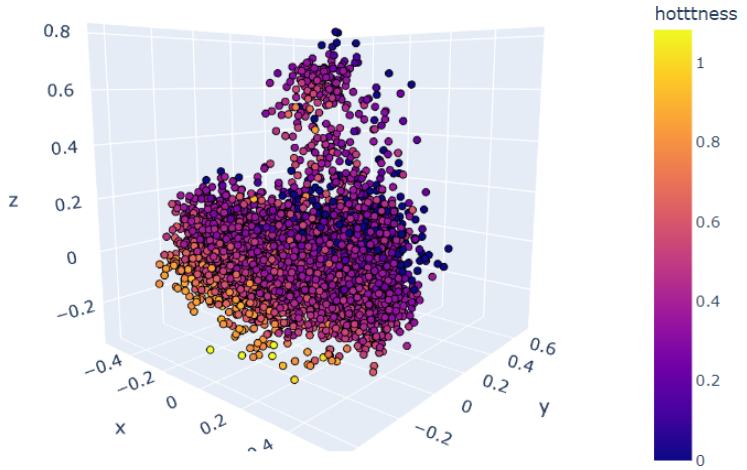


Figure 1: Lyrics Dataset in 3 dimensions

5.1.1 Proposed solution and Implementation

After try and failure tests, we achieved a configuration that we consider optimum. It consists of a first PCA dimensional reduction of the initial 768 features to 100 dimensions. After that, the deep learning part starts with a convolution layer of 1 dimension, with 4 filters, a kernel size of 6 and the inclusion of l2 regularization. Afterwards a Max Pooling is applied with a pool size of 3 and 2 strides. The output is flattened, with 188 dimension vectors. A first dense layer with 128 relu activated neurons is applied to the 1d vector, followed by a layer of 100 sigmoid neurons. The output layer consists of a single sigmoid activated neuron giving the estimated song hotttness. We minimize the MSE and use MAE as performance metric. The chosen optimizer is Adam.

[1] All the model architectures are found in Annex B. Test results graphics can be found in Annex C

We tried simpler models with no convolutions, and also more complex convolutional networks. We also considered to implement transfer learning (discarded because it is more focused on image treatment), but this network designed resulted to be better than the previous.

5.1.2 Results and discussion

After a previous chapter of epoch optimization and batch optimization, we reached an optimum configuration with a batch size of 64 and 40 epochs of training. Fig. 2. illustrates how the optimum point of training is reached at epoch 40 (we are minimizing MSE). The results are very good, since it is obtained a loss of 0.03 in the validation step. The same happens in the test step, where MSE is 0.0031, although it is true that the MAE is 0.04. This indicates the robustness of our model against outliers.

Analyzing a bit more the test results. The mean of the difference between prediction and true hotttness is of 0.01, demonstrating the model does not overestimate in excess the popularity of songs. Moreover, the gap is in between 0.2 and -0.2 points of difference, with no dramatic differences. The error density curve (provided in Annex B), shows a perfect gaussian curve centered in 0 and with a big slope. From the 1125 test cases, only 5 predictions have an error over 0.15 and 4 are under -0.15, reaffirming the feeling that the model works really well. Previous work has been done using only lyrics such as in [9], but they got stuck into only a classification problem, while we go further and achieve great results giving a prediction.

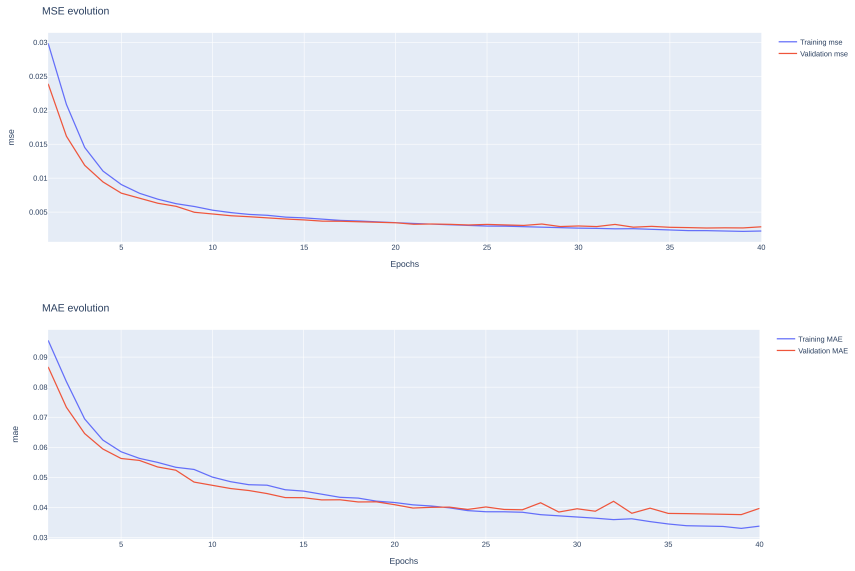


Figure 2: Final Model for lyrics dataset training and validation loss curves

5.2 General Dataset

The general dataset is mainly composed by the tags that an artist is assigned. The tags are a mix between mbtags (musicbrainz tags assigned by human to a particular artist [10]), and terms (tags provided by The Echo Nest, coming mostly from blogs [11]). The Million Song Dataset gives more reliability to the mbtags, but we are going to consider both the same reliable. The total amount of features is 8073. The terms are categorically

encoded and multiplied by the confidence. Fig. 3 represent the number of tags assigned to each row of the dataset. The average number of labels assigned to each row is 27.5, with a maximum number of 78. This results in vectors with a high percentage of zeros. Moreover, we have discovered that our employed dataset has 4350 empty labels. This means that no song/artist from our dataset is labelled with 4350 labels. This becomes a problem and we can approach it from two sides. The first one is to take into account this full-zero rows, the other is to delete them. The first approach would need to increase the amount of data, making the model training more complex. The second approach might improve the results of the model, but it would make it less scalable to other music data. After analyzing the problem,we moved for the first choice.

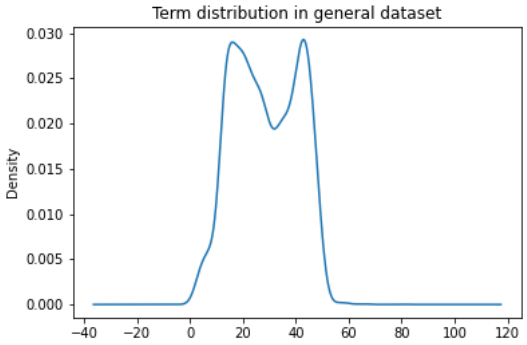


Figure 3: General dataset number of terms distribution

5.2.1 Proposed solution and implementation

We have tried several model designs, and several data treatments. We made an intent with a couple fully convolutional networks, we also tried with simpler dense models but the results were not as expected. Next section deepens into the numerical details behind our reasoning. We have opted for the same model design as in the lyrics dataset. The only change is that we decided not to reduce the dimensions of the features, so we had an input of 8073 features. We trained the model above 50 epochs with a batch size of 128 and a data distribution of 4046-1125-450 samples of train-test-validation.

5.2.2 Results and discussion

For determining how many dimensions we should keep, we trained the model employing several dimensional reductions with PCA to see if results varied as in the previous case. Table 1 shows how the difference is not big. This led us to decide not to reduce dimensions for this specific problem

Number of dimensions kept	**MSE in early epochs**
8073 (All)	0.037
2058	0.04
1008	0.037
108	0.034
18	0.036

Table 1: Scores obtained with different dataset dimensions

We trained the model for 50 epochs as per Fig. 4. It is shown that there is overfitting over epoch 5-6, so we trained our final model with less epochs. The results are clearly worse than in the previous dataset. Our best validation score is 0.037, which clearly is under the previous ones. Focusing on the test results, we obtained an MSE of 0.04, and a MAE of 0.155. It is not satisfactory, but despite our efforts to make it work better, it was not possible

In this case, the mean of the errors is 0.008, which tells that the model is not underestimating or overestimating the hotttness, but in this case we obtain errors over 0.5, which is too much. Almost the 50% of our test samples have errors over 0.15 or under -0.15, so it is not a good model. Some possible reasons might lead to a lack of data. The presence of zero values and non represented features may lead to a malfunctioning of the model. Also, the possibility of an unreliable set of artist terms could decrease the results. Lately, we have checked that there is less correlation between this set of features and the hotttness target variable than it was with the lyrics (it is provided in the code).

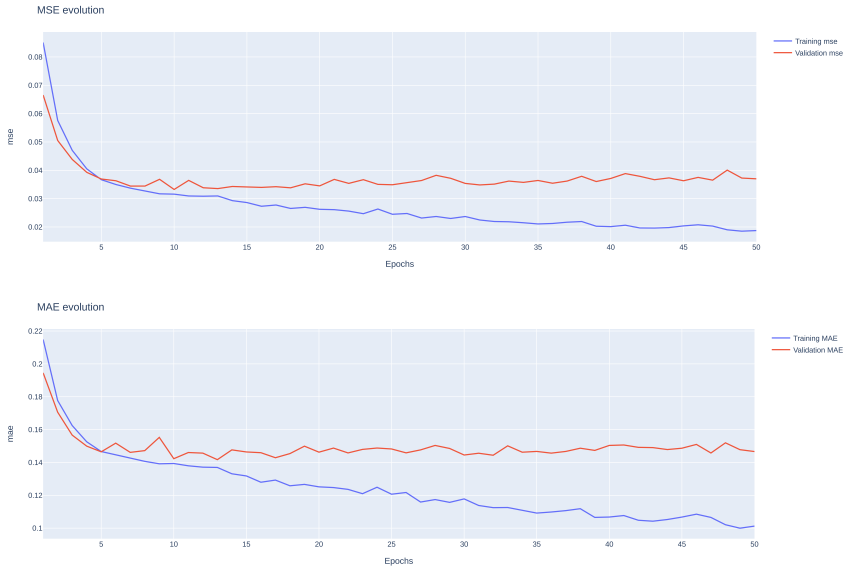


Figure 4: Final Model for general dataset training and validation loss curves

5.3 Musical Dataset

This dataset has two different type of features, the first 8 columns of the dataset represent numerical values, but the rest of the columns contain values of a sequence of different type of measures (such as segments, tatums, sections, beats or bars). Al the measures are sequential. After a several attempts we noticed that it was illogical to treat them in the same way, so we decided to apply a transformation for dimensional reduction based on the PCA method as we have explained earlier. In this case we decided to reduce each sequence to 3 dimensions. After doing so we end up with 29 features.

5.3.1 Proposed solution and implementation

For the model we have splitted the dataset into training, testing and validation subsets with the 4046-1125-450 samples. The architecture used is represented in the following Fig B. We included two convolutional layers with a pooling layer between them, followed

by a flatten layer, a dense layer with batch normalization and another dense layer with dropout before reaching the output layer.

The first convolutional one dimensional layer of the model is implemented with 6 filters and 6 kernels with L2 regularization. The following layer is a Max Pooling layer with a pool size of 6 and 3 strides.The other convolutional one dimensional layer with 6 filters and 6 kernels and again L2 regularization again. The output is flattened and connected to the next layer which is formed by 128 nodes with rectilinear activation function. In this layer we applied batch normalization with a 0.99 momentum and a value of 0.001 for ϵ. Then we included another dense layer with 100 nodes but with sigmoid activation function this time we applied a 0.5 dropout for this last layer. Finally we have the output layer with a simple node with a sigmoid activation function to predict the final predicted value of the hotttness of each song. We compute the MSE and MAE as metrics during the training stage with Adam optimizer. After several attempts we decided to compute 30 epochs to visualize if the model overfits and when.

Before this architecture we have tried several different models with no convolution layers, changing the number of layers and the number of nodes in each layer, including and erasing dropouts and normalization and we have tried this models with different number of dimensional reduction values.

5.3.2 Results and discussion

Before the training the model we plotted the 3 dimension representation of the sequential features, it is very interesting to explore how the songs are distributed. However we can not find any pattern.7

The validation and training MSE and MAE evolution over epochs overfits around the 10th epoch. We can appreciate that the validation MAE value does not perform as well as the MSE that makes us thing that we have a lot of errors in our predictions but very close to the real value, with no large errors that will penalize more in the MSE metric.

With this model we made an evaluation stage with the purpose of obtaining performance metrics for the test dataset. We obtained for 0.06277 MSE and 0.2089 for MAE.

We have also performed some predictions to calculate the error that each prediction make in detail. Finally we obtained similar results as in the evaluation stage. A value of 0.06128 for MSE and 0.2089 for the MAE.

5.4 Comparison of Models Performance

In this stage of the project we will compare the performance on the test datasets of each model against the others.

	MSE	**MAE**
Lyrics	0.0031	0.04
General	0.04	0.155
Musical	0.0627	0.2089

Table 2: Test results comparison

As we can see the Lyrics model performance is by far the model with best performance.

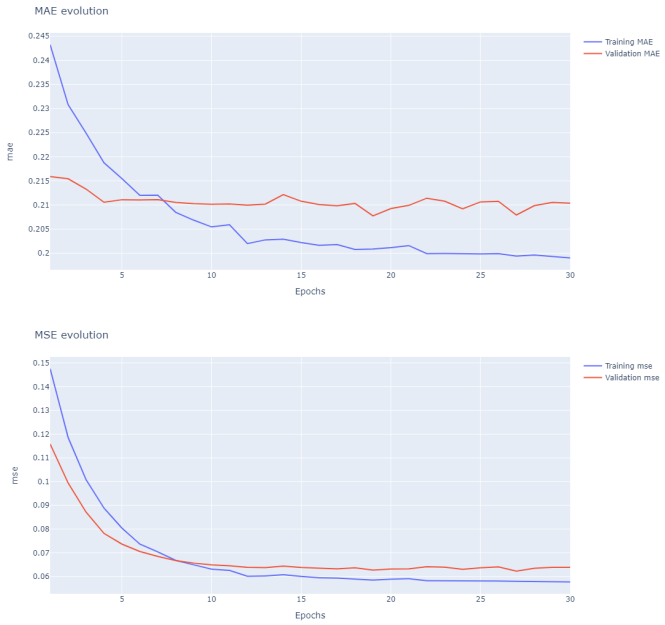


Figure 5: Training evolution metrics

6 Conclusions

After the development of the project, we have experienced difficulties gathering the data and processing it, due to our lack of equipment. We tried to manage it with AWS instances. This supposed an added challenge to the main one. We have found in PCA a useful tool for complexity reduction, since some of the datasets were too complex. It has been demonstrated that with the current amount of data we that we have, the general and the musical model are not capable of doing the task correctly. We believe that adding more data could lead to better results, especially in the general dataset.

Aditionally for the musical model, we should make a deeper approach in the preprocessing stage for improving its performance. With a better understanding on how each feature affects to the song, we could make better treatment of the information and not just reduce its dimensions. For doing that, we would need the help of an expert in the musical theory field.

On the other hand, we have demonstrated that the semantic of the lyrics can determine the success of a song. Our developed model works extremely well, thanks to the leverage of pre-trained deep learning models [7]. We think this way of research is the most promising.

Overall, we are happy with the outcomes and with our work, we have achieved great results and we have identified ways of improving it. We feel motivated about it and we are eager to continue our research in this beautiful topic.

References

[1] Joshua S. Gulmatico et al. “SpotiPred: A Machine Learning Approach Prediction of Spotify Music Popularity by Audio Features”. In: *ICPC2T 2022 - 2nd International*

Conference on Power, Control and Computing Technologies, Proceedings (2022). DOI: 10.1109/ICPC2T53885.2022.9776765.

[2] David Martin-Gutierrez et al. "A Multimodal End-to-End Deep Learning Architecture for Music Popularity Prediction". In: *IEEE Access* 8 (2020), pp. 39361–39374. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2976033. URL: https://ieeexplore.ieee.org/document/9007339.

[3] *Million Song Dataset*. URL: http://millionsongdataset.com/.

[4] *Musixmatch*. URL: https://www.musixmatch.com/.

[5] *Genius*. URL: https://genius.com/.

[6] *langdetect*. URL: https://pypi.org/project/langdetect/.

[7] *all-mpnet-base-v2 · Hugging Face*. URL: https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[8] Vidit Gupta, S. Jeevaraj and Somesh Kumar. "Songs Recommendation using Context-Based Semantic Similarity between Lyrics". In: *2021 IEEE India Council International Subsections Conference, INDISCON 2021* (2021). DOI: 10.1109/INDISCON53343.2021.9582158.

[9] Abhishek Singhi and Daniel G Brown. "Can song lyrics predict hits?" In: *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research*. 2015, pp. 457–471.

[10] *MusicBrainz - The Open Music Encyclopedia*. URL: https://musicbrainz.org/.

[11] *The Echo Nest Blog*. URL: https://blog.echonest.com/.

A Features explanation

Music General information Data

For this data we have selected the parameters with Spotify metadata information about the song. We have selected the most relevant parameters dropping some features that will no have relevant information to our model such us identifiers, location features, songs in the album among others. We have normalize some values with respect to the duration of the song. The features used are:

- Fade-out: it is the time when the song starts to normalized with the duration of the song.
- Fade-in: it is the time that takes the song to reach its regular volume at the beginning of the song normalized with the duration of the song.
- Danceability: subjective parameter that tells how likely the people are willing to dance this song.
- Duration: duration of the song in seconds.
- Energy: subjective parameter.
- Loudness: subjective parameter that tells how loud the music is likely to be perceived.
- Year: year of release.
- Artist Hotness: subjective parameter of the popularity of the artist.

- Artist Terms: it is a catalogue of musical classification terms, each song has different terms and different number of terms so we decided to perform a one hot encoding including a new feature per term and filling it with its "weight" multiplied by its "frequency".
- Artist mbtags: each song has different number of tags, we decided to transform each tag into a feature and fill with a 1 if the song contains the tag.

After this process we obtain a dataset with 10000 songs with 8073 features.

Musical features data

We created an additional sub-dataframe for the specific musical features because we wanted to study which type of data performs better. For some features, the Million song dataset, provided arrays with different length so we have to normalize the length of those arrays. The method used consist in split the arrays with more length and add the resting positions to the previous ones and compute the mean value. With this method we lose the sequence but we use, somehow, all the data.

- Key:the key of a song is the note or chord the music is centered around, the tonic.
- Mode: type of scale with distinct melodic characteristics.
- Fade-out: it is the time when the song starts to normalized with the duration of the song.
- Fade-in: it is the time that takes the song to reach its regular volume at the beginning of the song normalized with the duration of the song.
- Duration: duration of the song in seconds.
- Tempo (bpm): is the speed at which a piece of music is played.
- Time signature: indication of rhythm following a clef, generally expressed as a fraction with the denominator defining the beat as a division of a whole note and the numerator giving the number of beats in each bar.
- Analysis sample rate.
- Segments (timbre,start, loudness): for this feature we have split the segments related data into three different subcategories. One related to the timbre, another one related with the starting time and a third one related to the loudness. Each subcategory is transformed into 400 features with each element multiplied by the segment confidence. The loudness features are computed obtaining the time of the loudness (maximum time - start time) multiplied by the maximum loudness value and multiplied again by the segment confidence.
- Tatums start: the smallest time interval between successive notes in a rhythmic phrase. We have transformed this feature into 500 features in which each value is the result of multiplying the tatum start by its confidence.
- Sections: the sections of a song are a complete, but not independent, musical idea. Types of sections include the introduction or intro, exposition, development, recapitulation, verse, chorus or refrain, conclusion, coda or outro, fade out, bridge or interlude. This time we have transformed the section vector into 10 features.

- Beats: the basic rhythmic unit of a measure. The transformation performed this time was taking out 300 features of the beats start time multiplied by its confidence.

- Bars: one small segment of music that holds a number of beats. For this feature we decided to transform the array data into a 100 features with the values of the starting time of the bar multiplied by its confidence.

After this whole processing we obtained a dataset with 10000 rows and 2119 columns.

B Model Architectures

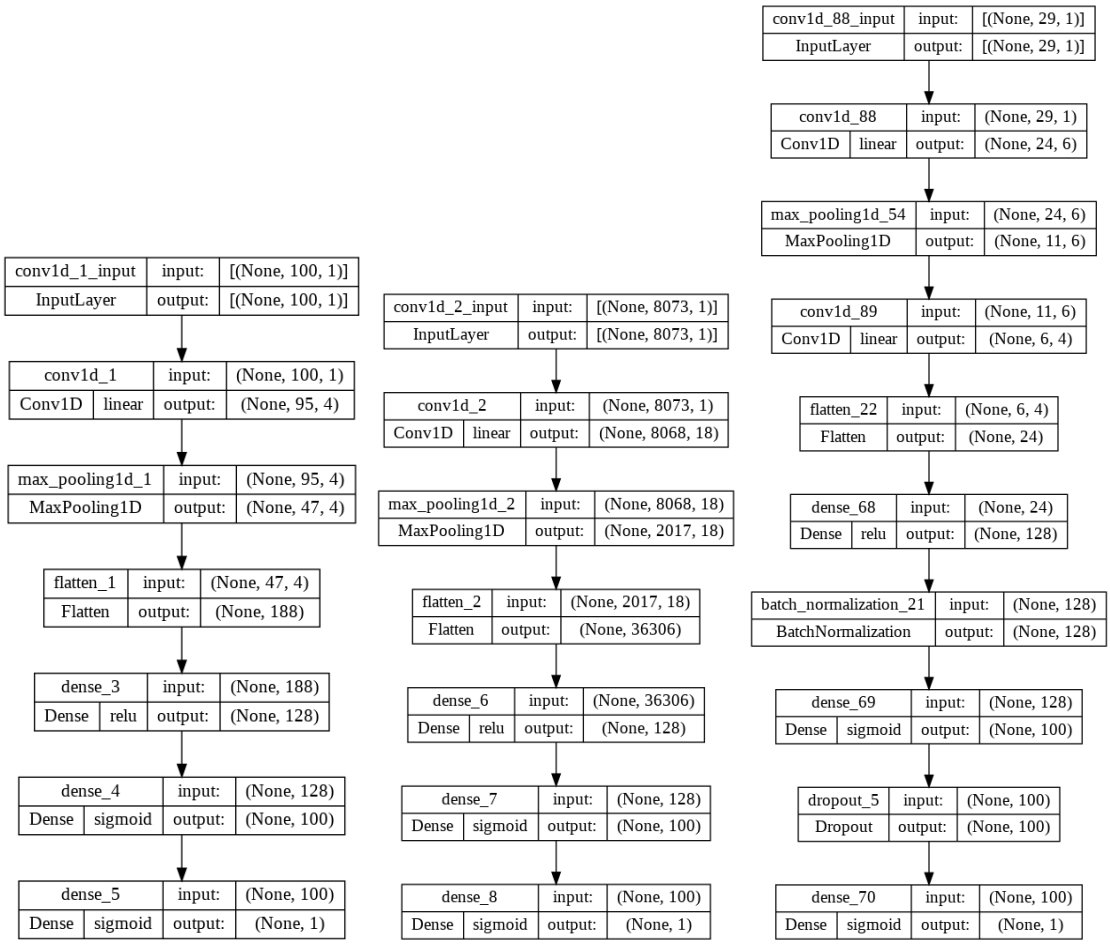
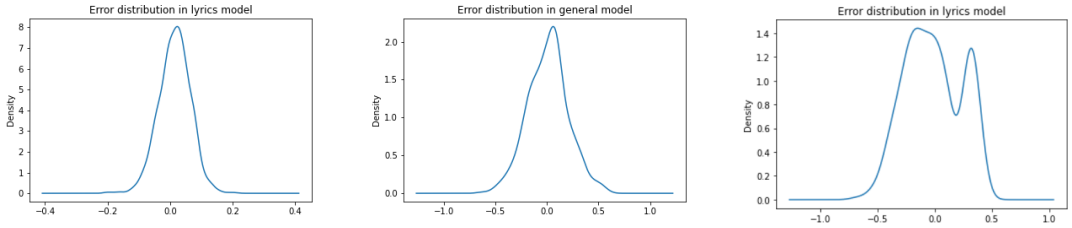


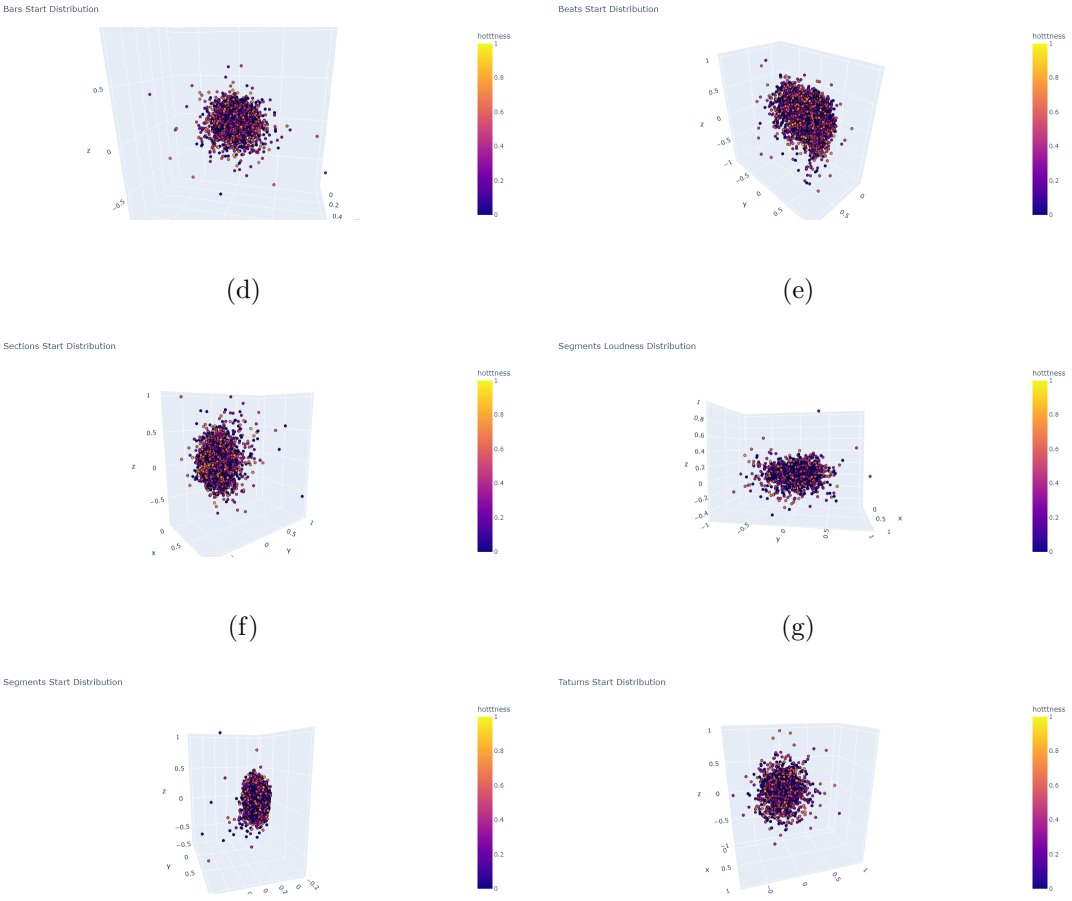
Figure 6: (a) Lyrics Data Model (b) General Data Model (c) Musical Data Model

C Models test error curves



(a) (b) (c)

D Musical Dataset Features Representation



(d) (e)

(f) (g)

(h) (i)

Figure 7: (a) Bars (b) Beats (c) Sections (d) Segments (e) Segments Loudness (f) Tatums