# Song Hotttnesss Prediction at Scale

**Qiqi Xu**
Department of Machine Learning
Carnegie Mellon University
qiqixu@andrew.cmu.edu

**Wenhuan Sun**
Department of Mechanical Engineering
Carnegie Mellon University
wenhuans@andrew.cmu.edu

**Yansheng Cao**
Department of Machine Learning
Carnegie Mellon University
yanshenc@andrew.cmu.edu

## 1 Introduction

### 1.1 Motivation and Problem Formulation

The music industry has experienced massive growth in the recent years due to digitization and streaming services that made the distribution of musical contents much easier and more readily available for consumers. In the 2019 global music report issued by the International Federation of the Phonographic Industry, the total revenues for the global recorded music market grew by 8.2% to $20.2 billion. Streaming revenue grew by 22.9% to $11.4 billion. For our project A, we worked and analyzed the data from the Million Song dataset [1,2] to predict the song popularity, also known as song hotness, from various categorical and numerical features such as audio features, artist related features and song related features; and we applied various machine learning models to predict the popularity of a song based on both song characteristics and metadata about a song from the features in the original dataset. We also examined which machine learning models are suitable for working with large-scale dataset. Through this modelling pipeline, we seek to understand whether we can predict the popularity of future songs using machine learning models, while also seeking to gain deep insight into features that are predictive of song popularity.

### 1.2 Dataset

The dataset that we used for this project is the Million Song Dataset (MSD), which is a a freely-available collection of audio features and metadata for a million contemporary popular music tracks.[1] The raw dataset is 280 GB and can be accessed as a public dataset snapshot from Amazon Web Services. In addition to the full raw dataset, a ten thousand song subset is also provided on the Million Song dataset homepage and can be downloaded via torrent as a tar.gz file that is about 2 GB. The dataset contains 54 fields, and we chose "song hotttness", represented as float, as our outcome variable, given that we are interested in predicting the popularity of a song.

## 2 Methodology

### 2.1 Project Pipeline

The entire modeling pipeline of this project, including the local test-run using a subset of the data and at-scale modeling using the entire data set, is illustrated in Figure 1. Firstly, we used instances on local machines and free-tier Databricks instances for script development, which performs feature extraction from the raw Hdf5 file into CSV files, data preprocessing, feature engineering, and model

Preprint. Under review.

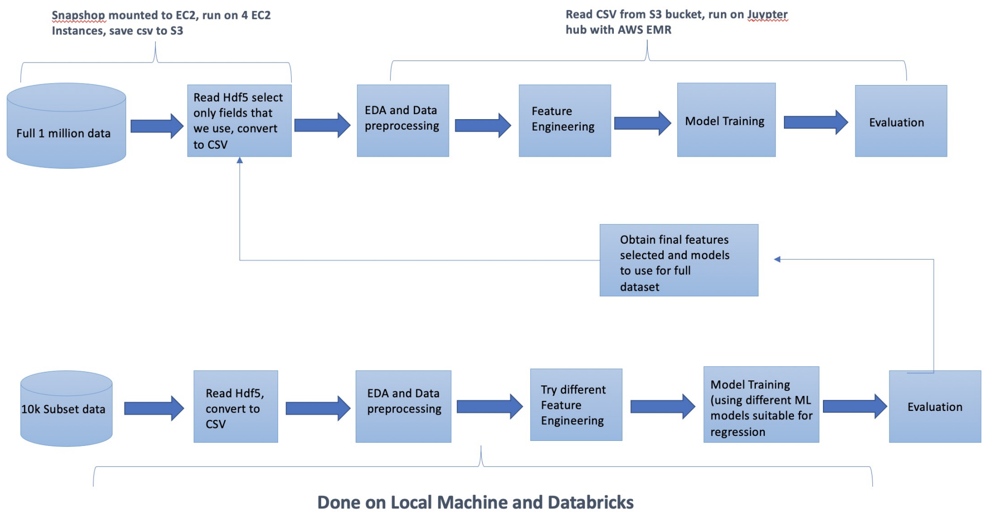
Snapshop mounted to EC2, run on 4 EC2 Instances, save csv to S3
Read CSV from S3 bucket, run on Juypter hub with AWS EMR
Full 1 million data
Read Hdf5 select only fields that we use, convert to CSV
EDA and Data preprocessing
Feature Engineering
Model Training
Evaluation
Obtain final features selected and models to use for full dataset
10k Subset data
Read Hdf5, convert to CSV
EDA and Data preprocessing
Try different Feature Engineering
Model Training (using different ML models suitable for regression
Evaluation
Done on Local Machine and Databricks

Figure 1: Entire Modelling Pipeline

training using different machine learning models deemed suitable for regression. Upon evaluation of the test-run, we selected a subset of the original features and identified models to be used at scale.

Then, using AWS EMR, standalone EC2 instances and an S3, we ported the entire 1 million song data from the offical data repository to the S3. Meanwhile, features extraction was performed to collect features that were used in modeling from the Hdf5 files into a single CSV file. The script for local test-run was implemented on Jupyter Hub with AWS EMR to read the CSV file from S3 bucket to perform data processing, feature engineering, modeling training and evaluation.

## 2.2 Data Preprocessing and Pipeline Test-drive on the Subset

The original 1-million-song dataset contains a plethora of features. Including the entirety of such features will lead to exccesive data storage, computational cost and complexity. In addition, some features may not contribute to the regression model, for example, some features may exhibit zero variance, and should be excluded in further analysis. To identify such redundant features, a data preprocessing script was implemented with a subset of 10,000 song data on Google Colaboratory. First, non-numerical features were removed, after which data entries with invalid feature or label values were also excluded, for example, songs with NaN labels. A basic statistical analysis was performed to obtain the mean and standard deviation of each remaining features. Zero variance was observed in three features: 'danceability', 'energy', and 'analysis-rate'. Therefore they are also excluded. As a result of the presented steps, 14 features were included: 'artist hotttness', 'familiarity', 'duration' , 'loudness', 'year', 'tempo', 'end of fade in', ' key', 'key-confidence', 'mode', 'mode-confidence', 'start-of-fade-out', 'time-signature', and 'time-signature-conf'.

The above feature pruning procedure was integrated into a PySpark script for pipeline test-drive on the 10,000 subset on the Databricks platform. Briefly, the data was preprocessed as explained above with all features normalized and divided into training (80%) and test (20%) data. With feature-normazlied subset data, a naive model, one that outputs the averaged training song-hotttness for every prediction, a linear regression model with L1 and L2 regularization, Random Forest, and Gradient Boosting Trees were constructed, trained and evaluated. Grid searches were conducted to identify optimal hyperparameters associated with each proposed methods.

## 2.3 Techniques

**Linear regression with regularization** Our baseline is a linear regression model with regularization. We tried both L1 and L2 regularization on the subset data to compare performance and select features that we would work with on AWS. The hyperparameters for the linear regression with L1 and L2 regularization are $\lambda_1$ and $\lambda_2$ respectively. These regularization hyperparameters were chosen via grid search.

**Random Forest and Gradient Boosting Trees** For our more sophisticated models, we chose random forest and gradient boosting tree, which are powerful off the shelf models that does not require an enormous amount of hyperparameter tuning, while still capable of delivering good prediction results. The tree-based ensemble models also provide variable importance that help us understand what features are important and should be included. The hyperparameters that we tuned for random forest are the number of trees and the maximum depth of each tree; while the hyperparameter for the GBDT model is the maximum number of iterations. These hyperparameters are tuned via grid search and cross validation.

### 2.4 Evaluation metrics

Our two main metrics to evaluate and compare the techniques outlined in the above section are, Root Mean Squared Error, and training time.

$$RMSE(h(x), y) = \sqrt{\frac{\sum_{i=1}^{N}(y_i - h(x_i))^2}{N}}$$

We chose to evaluate the performances of the various models using RMSE because we formulated the problem of predicting song hotness score as a regression problem. And we also compared the training time of different models since we are working with a large-scale dataset with a fixed budget, and computational time is tied to practicality issues.

By performing and comparing the above three machine learning models on our extracted 1 million song data, we are able to not only compare the performance of different models but also study how suitable each model is under constrained time and capital budget. Furthermore, we are able to develop a better understanding of what features/characteristics are useful for predicting the popularity/hotness of a song. Our data preprocessing is suitable as it is important to select 10-20 features that we think are important to our problem from the original features in the raw dataset, as this would drastically reduce the size of the data and make the computation tractable given the budget constraint. Furthermore, the models that we utilized in the final pipeline are the ones that we selected from various models we experimented on the subset. They offered some levels of explainability and are computationally efficient with distributed machine learning framework.

## 3 Computation

### 3.1 Computational challenges

The major source of the computational challenge is the data conversion process for the full Million Song Dataset. The 280 GB raw data is stored as .h5 file which is not supported by spark. Therefore we have to extract the fields for each song (stored as .h5 file) and write them to a large CSV file. We estimated that extracting every field from the full dataset takes 40 hours on a single machine without multiprocessing or distributed computation. Therefore we plan to select only a subset of the fields. We ran experiments on the 10,000 subset. We tried different feature engineering for certain categorical variable and tested out different machine learning models. Afterward, we finalized our modelling pipeline and the 19 data fields that we need. This way, we drastically reduce the amount of read and write for each song when we move to AWS EC2 instances. Given the folder structure of the full Million Song Dataset, another thing we did is launching 8 instances to extract the fields from each subdirectory separately and write to its own CSV file. Once all 26 subdirectories have been parsed, we wrote a short concatenate CSVs code in python to combine the CSV files together as the column name for each CSV file matches. Concatenating the CSVs only took around 1 minutes. More details of the process will be explained in section 3.1.1 and 3.1.2.

For computation time, we will discuss each part of the pipeline for the full model separately. Firstly, we spent about 12 hours experimenting with the subset data to identify which set of features we will need to extract and use. Secondly, converting the full raw data to CSV file with fields that we need took around 12 hours. thirdly, 2 - 3 hours were spent on experimenting with finding suitable number and type of EC2 instances that we need. Lastly, the data preprocessing and applying feature engineering takes 40 minutes and running grid search on our models took 3 hours.

### 3.1.1 Resources and Tools

To identify and eliminate song features that do not contribute to regression model, a preliminary data processing and feature engineering procedure was performed on the subset of 10,000 songs using Google Colaboratory with a free CPU instance.

The data processing pipeline was developed and tested using the same subset of song data with a free-tier Databricks cluster. Upon the establishment of a streamlined pipeline, we scaled up to the full one-million-song data set. An Amazon EMR with one master machine (instance type: m5.xlarge, 4 vCore, 16 GB memory, EBS Storage: 64 GB) and up to six worker nodes (instance type: m5.xlarge, 4 vCore, 16 GB memory, EBS Storage: 64 GB) were created. To transfer the 300 GB of song data from the public repository and to perform data preprocessing (extracting song features from .h5 files to one .csv file), another EC2 intance was created. The original song data as well as the processed .csv file were then transfered to an AWS S3 container (size: 500 GB). We spent 19.44$ in total.

### 3.1.2 Language and Dependencies

We used Python, Pandas, Scikit-learn and Tensorflow Keras to perform the preliminary data processing, and feature engineering procedure was performed on the subset of 10,000 songs. Then we move our chosen pipeline with select data fields to Databricks, where we worked in Pyspark and leveraged the Spark MLlib. For our work with the full dataset, we extracted key fields from the .h5 files with a parallel script using the "tables" library, and we used glob and Pandas to concatenate individual CSV files that extracted certain subdirectories of the data to obtain a final CSV file with 1000000 song data entries with selected variables/fields. We utilized the same Pyspark notebook that we wrote earlier to run experiments on the full dataset and obtain the results.

## 4 Results

RMSE captures a regression model's ability to make accurate predictions based on input features. Using a naive model which constantly outputs the average song hotttnesss calculated from the training data regardless of the input features, the RMSE was 0.17. All of the three regression models implemented in this project exhibited significant improvement in prediction accuracy. Among the three models, Random Forest model yielded a test RMSE of 0.12996, which slightly outperformed the Gradient Boosting model (Table 1). In comparison, the linear regression model with L2 regularization achieved a higher RMSE of 0.1354. Using test RMSE as the metrics, Random Forest and Gradient Boosting models appear to be more desirable for song hotttnesss prediction.

Table 1: Table 1. Test set model performance metrics

| **Model** | **Test RMSE** | **Training Time** |
|---|---|---|
| Linear Regression with L2-Reg | 0.1354 | 2 mins for grid search on single EC2 instance |
| Random Forest (best) | 0.12996 | 25 mins for grid search with 6 EC2 instance |
| Gradient Boosting (best) | 0.13045 | 20 mins for grid search with 3 EC2 instance |

However, it is worth noticing that, the two 'best' models incurred significantly computation cost than the Linear Regression Model whose training with grid search can be completed within 2 mins with just one EC2 instance as the worker node. In contrast, the two more complex models both required over 20 minutes for grid search with 3+ EC2 instances as worker nodes.

In conclusion, for this specific task, the Random Forest and Gradient Boosting models achieved a slightly lower test RMSE at a cost of significantly higher computation cost. This trade-off between model performance and training cost should be considered for future applications.

## References

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[2] Million Song Dataset, official website by Thierry Bertin-Mahieux, available at: http://millionsongdataset.com/