Song Hotness Prediction

By :
Chirag Gupta
chirag.15je001713@am.ism.ac.in
IIT (ISM) Dhanbad

Approach

The aim of the task is to predict song hotness, which ranges between 0 to 1, of a list of songs. The training dataset consists of several interesting audio features of songs like *duration of song, end_of_fade_in, tempo, mode, key, start_of_fade_out* etc. along with some basic information of the artists who have delivered the songs like *artist_familiarity, artist_hotttnesss* etc. The evaluation metric used for the evaluation of prediction of *song_hotttnesss* is ***mean absolute error***.

Initially, I have analysed the training data and looked for statistics of data like range of values of each feature, is every feature have correct data type or not, missing or incorrect values etc. I found several discrepancies in data, removed them and pre-process the data to train on with several machine learning algorithms.

Further, I tried to approach the problem in two ways: Regression and Classification methods.

1) Regression approach - As the target (*song_hotttnesss*) values are in range of 0 to 1 (instead of only 0 and 1), so initial approach was to use regression technique to predict the value. I have implemented and tested main regression methods which can perform better on large dataset: Linear Regression, K nearest neighbor regression, Random Forest Regression and Gradient Boosting Regression.
2) Classification approach – As the target (*song_hotttnesss*) values are only in the range of 0 to 1 (not greater than 1), so I tried to convert regression problem in to classification problem by defining partitions, and assigning each partition a class. For example, I splited range 0 to 1 in 10 partitions as 0-0.05, 0.05-0.15, 0.15-0.25 etc. Then, I implemented and trained main classification methods: Logistic regression, SVM classifier, Random forest classifier etc. Further, the predicted values by classifiers are categorical values, then I transform them again in range of 0 to 1 by dividing *predicted label by total no. of classes*, so that I can evaluate my model using mean absolute error using exact song hotness value instead of class label assigned.

I tried on several partitions 20, 100, 1000 etc. Also, tried binary classification approach and used probability of prediction of "1" label as song hotttnesss predicted value.

Quality checks performed / Errors found

- Duplicate rows found in training data.
- 25% of data in *"Year"* column are 0 and 1 values. (which is not possible)
- *"Duration"* column have string data type instead of float data type (seconds) and around 0.8% values are either **'y'** or **NaN** (missing).
- Similarily, *"Key_confidence"* column have string data type instead of float data type (value between 0 to 1) and around 0.5% values are either **'z'** or **NaN** (missing).

Data preprocessing steps and Feature generation

- Merged the artist features with the training as well as testing song feature data.
- Removed unnecessary columns like song_id, title, analysis_sample_rate, audio_md5 artist_latitude and longitude etc.
- Duplicate rows are removed as data points should be independent and identically distributed, so duplicate data might effect accuracy of model.
- As dataset is too large (56k values and 18 features) and total *"Duration"* and *"Key_confidence"* incorrect/missing values are around 1.5% only and most importantly inaccurate rows *song_hotttnesss* value is normally distributed over 0-1 instead of belonging to some particular partition range. So removing them will not effect much in accuracy of model.
- Convert each feature variable into float64 datatype.
- Spitted total data into training data(75%) and testing data(25%) for training and evaluating the models.
- Finally after all data preprocessing steps, I have 55395 data values with 13 features (i.e. feature matrix of 55395x13) and *song_hotttnesss* values array of size 55395.

Key observations and significant variables

Artist_familiarity and artist_hotttnesss have significant positive correlation with *song_hotttnesss* which might suggest these value have much importance than all other features in prediction. For predicting important of features, I used "Random forest feature

importance" which calculates feature importance as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. *The higher the value the more important the feature.*

Descending order of feature importance :- *'artist_familiarity', 'artist_hotttnesss', 'loudness', 'tempo', 'mode_confidence', 'end_of_fade_in', 'key_confidence', 'start_of_fade_out', 'duration', 'time_signature_confidence', 'key', 'time_signature', 'mode'.*

Hence, most significant feature in the dataset is *artist_familiarity.*

Model Selection and Results

As the dataset is too large and have lots of features i.e multi-dimensional data, so the chance of good prediction by linear models (linear regression/) is very low. In such case, Ensemble models perform much better than single estimators (which combine the predictions of several base estimators built with a given learning algorithm in order to improve generality / robustness over a single estimator) like Random forest, XGBoost etc.

I compared several machine learning algorithms using both regression and classification approach and evaluated models on splited test data using mean absolute error metric.

Regression Model	**Mean absolute error (MAE)**
Linear Regression	0.12238
KNeighbors Regression	0.13549
Random forest regression	**0.10143**
XGBoost regression	0.10223

Classification Model (taking 10 partions/ 11 classes)	**Mean absolute error (MAE)**
Logistic Regression	0.12694
SVM Classifier	0.12744
KNN Classifier	0.13448
Random forest classifier	0.12021

Hence, Best estimator is random forest regressor with **0.10143** MAE and final submission score (on the task website) corresponding to the model is **0.90236**.