LAST.FM SONG PLAY COUNT PREDICTION WITH SPOTIFY DATA - WITH MACHINE LEARNING APPROACHES

A PREPRINT

Jieyu Chen
jc7483@nyu.edu

Wujie Duan
wd633@nyu.edu

Xinyi Wang
xw1165@nyu.edu

December 9, 2019

ABSTRACT

The paper shows the machine learning approaches we took to make prediction on a song's play count given its audio features, album features and artist features. The data was crawled from Spotify API and the Last.fm API using snowball sampling. After preprocessing the data, we tried several machine learning models, including Linear Regression, SGDRegressor, LinearSVR, Basic Neural Network and Random Forest Regression model. Then we evaluated all the models with Mean Absolute Error (MAE). The methods we used to get the optimal hyper parameters are Grid Search and Randomized Search.

***K*eywords** Machine Learning · Prediction · Linear Regression · SGD Regression · SVR · LinearSVR · Deep Learning · Neural Network · Ensemble · Bagging · Random Forest

1 Introduction

Music streaming is a very popular Internet activity. In 2017, the Computer and Internet Use Supplement to the Current Population Survey done by US Census Bureau shows that 52.6% of the online adult population stream or download music on the Internet. With the growth of many music Apps and websites, music streaming has took the place of physical CDs and become the most favored way of music listening[1].

As music streaming is increasingly more into people's daily life, music producers will be wishing to launch music that are welcomed on music streaming services. A prediction of a song's popularity is a helpful solution to reduce the risk of loss when doing the business. In our research, popularity is quantified as a song's play count from Last.fm. With various features including audio features, track features, album features and artist features, we set up multiple regression models with machine learning. By comparing the performances of each models evaluated by MAE, we are able to get an optimal model of the song's play count prediction. With the result we have, we hope to help music producers make better business plans for the songs with the predicted play count, and in the end, to help with the development of the music service market.

2 Related Work

Rutger Nijkamp has tried to explain the success of a song by its audio features[2]. As a social science study, this paper inspired us on the social impact of our project. In his paper, he used the statistical approach to predict stream count by audio features with a linear regression model. Instead of involving the whole set of features provided by the Spotify API, this paper focused on 10 most popular genres and dug deeper into the explanation of a song's popularity from the social science perspective.

3 Dataset and Features

3.1 Spotify and Last.fm

Spotify is a very popular music APP originated in Sweden in 2006, which provides over 50 million tracks. It has an official API with different libraries such as Spotipy. Spotipy is the main tool we adopted to get access to the dataset.

Last.fm is an online music streaming platform founded in United Kingdom in 2002. It specializes in its music recommendation system. Last.fm provides an API which gives developers exposure to its data.

3.2 Dataset Component

There are 21 predictors in our dataset that were grabbed from Spotify. It contains various features, including audio features, track features, artist features and album features. Full description is in the appendix.

The dependent variable is designed to be a value that evaluates a song's popularity. Spotify API provides a feature called popularity. However, it is not clear how this number is calculated. Therefore, we turned to a more specific and reasonable value, play count. Since Spotify API doesn't offer the play count number of each song. We then use the play count data of Last.fm, which is another widely used music service.

4 Snowball Sampling

The dataset available on Spotify is huge. But it does not allow us to directly get a large group of songs with one function. Therefore, in order to get enough number of songs, we proceeded with an exponential non-discriminative sampling method called Snowball Sampling.

Spotify API provides a function called get_related_artists, which returns a list of 20 artists who are related to the given artist. Using this function iteratively, we are able to roll a large snowball of artists with one starting point.

Moreover, according to the research conducted by Julian Kerchherr, using multiple sample seeds can increase sample diversity, so we chose ten artists of different genres as the starting points of our snowball sampling. Thus, at last, we got 10 snowballs of artists. By extracting all the songs of each artist, we then got all the songs that belong to all the artists we got. The final total data contains more than 840,000 songs.

Furthermore, to look into the relationships among each snowball, we did further visualization on the networks of artists in the Figure 2. Its analysis is also written in the following paragraphs.

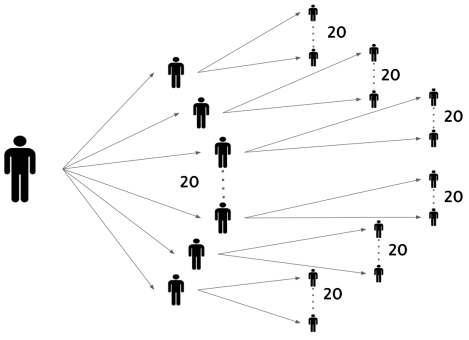

Figure 1: Process of Rolling One Snowball

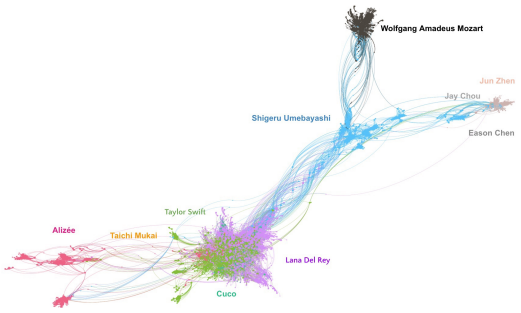

Figure 2: Visualization of Actual Snowballs

Figure 1 is a simple visualization of such process. We used one artist as a sample seed. With the Spotify API's get_related_artist function, we could get 20 related artists from 1 given artist. Then we did the same operation on the group of artists we got. After doing this operation recursively, we got a big group of artists which contains artists that are directly or indirectly related to the given artist. There are duplicated artists in the snowball since some artists are related to each other, so we need to drop the redundant artists.

4.1 Multiple Snowballs

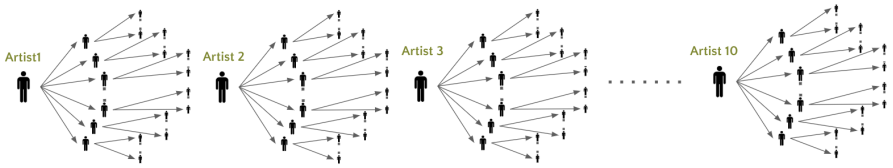

Figure 3: Sampling with 10 Snowballs

Julian Kerchherr suggests that using multiple sample seeds can increase sample diversity[3]. Thus, in order to make our dataset more diverse, we manually picked 10 artists with different backgrounds as sample seeds, just as what Figure 2 shows. We firstly rolled 10 independent snowballs. With the 10 groups of artists, we gather them together and eliminated duplicates. The iteration of function get_related_artist might cause overlap between the artist groups generated from two different sample seeds. Finally, we cleaned dataset of artist information generated from 10 snowballs. Then we were able to get the songs of each artist and put them into the final data set that we used for machine learning.

We visualized our snowball sampling result with Gephi in Figure 2. Different coloring represents snowballs generated from different sample seeds. Nodes represent artists, edges connect two related artists, and nodes with the same color means they are acquired by the snowball sampling of the same starting point. Before plotting this graph, we assumed that there may be some closed clusters. But as we can see from the graph, although several groups are formed, there are bridges between them. Therefore, we can assert that snowball sampling can traverse Spotify's artists using related artists theoretically.

5 Data Preprocessing

We dealt with missing value with two approaches. First, we deleted the whole data point if album release date or country was missing. Second, we imputed the data with the mean value. For the number of artist followers, we put in the mean value calculated from the datasets.

For categorical data, we conducted one-hot encoding. The variable "country" contains originally the two-character abbreviation for each country. "get_dummies" in sklearn allows us to turn this variable to over one hundred variables, with 1 indicating this data is in this country and 0 otherwise [7]. We turned the feature "explicit" to 0 or 1 in correspondence with False or True. Moreover, we decided to work on two data sets. One with countries that is one-hot encoded, with 126 features. We will later call it the dataset with countries for convenience in this paper. The other one already has the column "country" with 21 features. We will call it the dataset without countries. These two datasets are for our comparative experiments.

The album release date is originally a string containing the year, month and date. We first turned these strings into datetime type in Python, and then subtracted it by today. In this way, this data represents how long an album has been released. We sorted the whole dataset according to the new album release date in an descending order, meaning the older album came first.

As for data normalization, we applied MinMaxScaler in sklearn, a way of linear transformation, on the regressors. MinMaxScaler in sklearn adopts Min-Max normalization with the default upper bound of 1 and lower bound of 0, which has the advantage of preserving the relationships among the original data values [8].

We split the dataset with countries into train, development and test set with a ratio of 7:2:1, and the size of 585059, 166060, 85077. The same way as the dataset without countries, with the size of 586403, 167544, 83772 for train, development and test set.

Y in our model, the play count, has a maximum of 15351835, a minimum of 0, an average of around 98362, and a median of 2222.

6 Machine Learning Models

After the data sampling and data processing, we proceeded with our prediction. The predictor in our case is play count, a continuous variable. Thus, a regression model is the most suitable model to use. In total, we used linear regression, SGD regressor, linera SVR, basic neural network and LSTM to predict the song play count.

6.1 Evaluation Criterion

$$MeanAbsoluteError = \frac{1}{n}\sum_{i=1}^{n}|Actual - Predict| \tag{1}$$

We use Mean Absolute Error (MAE) (i.e. the difference between two continuous variables) as the main evaluation criterion for the models. This is because the Y in our model is continuous. In addition, MAE is intuitive and straightforward in its interpretation. It can describe the average error.

6.2 Baseline

First of all, we used the mean of Y in train set to set a baseline model. We applied mean_absolute_error(y_test, y_predict) from sklearn. Specifically speaking, y_predict is the mean of Y in the train set. y_test is Y in the test set.

6.3 Linear Regression

First of all, we set a basic linear regression model. We started with the linear regression with polynomial feature to be 1, then the one with polynomial feature of 2. We picked polynomial feature with degree of 2 to avoid the case where the ratio between the number of observations and the number of features is less than 10. The number of parameters in the dataset with countries is $126^2 + 126 + 1 = 16003$. The size of the train set is 585059. The evaluation at this stage is MAE.

6.4 SGD Regressor

$$b' = b - \eta\frac{\partial L}{\partial b} \tag{2}$$

- b - current value b' - value after update
- η - learning rate, set to 0.05 $\frac{\partial L}{\partial b}$ - gradient i.e. partial differential of L w.r.t b

Then we applied stochastic gradient descent regression. We used Random Search to tune the parameters. We set the stopping condition, namely tolerance, in order to stop the training when the model could not improve the result more than tolerance. This speeds up model building and is great for prototyping. We also tried polynomial features with the degree of 2.

	loss	**penalty**	**max_iter**	**tol**	**alpha**	**random_state**	**fit_intercept**
SGD Regressor(Base)	squared_loss	l2	3000	1e-3	0	0	True
SGD Regressor(Poly=2)	squared_loss	l2	3000	1e-4	0.0001	0	True

6.5 LinearSVR

We chose primal instead of dual because the number of observations far outnumbered features. The loss function is squared_epsilon_insensitive, which is L2 loss. We performed Randomized Search for hyper-parameter tuning. The dataset with countries had a higher MAE in the test set than the train set. The dataset without countries had an MAE error in the test set that was more than two times of that in the train set. This shows that linear support vector regression had a higher variance on the dataset without countries than the datset with countries.

	C	**dual**	**epsilon**	**fit_intercept**	**intercept_scaling**	**loss**	**max_iter**	**random_state**	**tol**	**verbose**
Linear SVR(with country)	1.0	False	0.2	True	1.0	squared_epsilon_insensitive	5000	None	0.01	0
Linear SVR(without country)	0.2	False	0.0	False	1.0	squared_epsilon_insensitive	15000	None	1e-06	0

6.6 Basic Neural Network

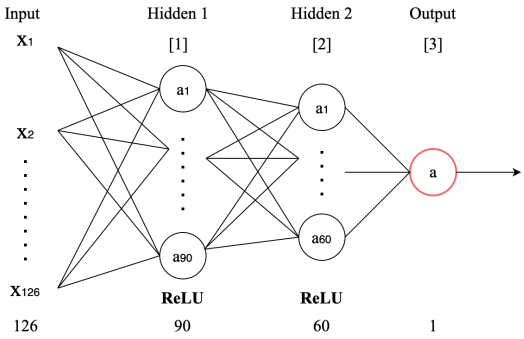

Figure 4: Neural Network Layers with Country

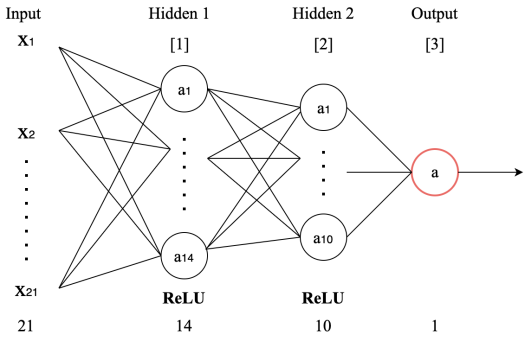

Figure 5: Neural Network Layers without Country

We implemented a three-layered basic neural network. Since the number of input features of the no_country dataset and with_country dataset is 21 and 126, we decide to use hidden layers of size 90, 60 and 14, 10 after manually tuning the network, which are approximately 2/3 of the input size. The activation function we used on hidden layers was ReLU and we did not apply activation function on the output layer since this was a regression problem. The optimizer we used was Adam. In order to adjust the learning rate effectively, we used a learning rate scheduler StepLR with step size of 100 and gamma of 0.85. After several experiments, we found that using L1 loss function would achieve best result.

	Optimizer	Scheduler	Loss Function	Epochs
Basic Neural Network(with country)	Adam(loc_model,parameters(), 3e-1)	StepLR(optimizer, step_size=100, gamma=0.85, last_epoch=-1)	L1 loss(reduction="sum")	2000
Basic Neural Network(without country)	Adam(loc_model,parameters(), 3e-1)	StepLR(optimizer, step_size=100, gamma=0.85, last_epoch=-1)	L1 loss(reduction="sum")	3000

6.7 Random Forest Regression

$$f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(X) \tag{3}$$

Random Forest is an ensemble machine learning technique capable of performing regression tasks using multiple decision trees and a statistical technique called bagging [5]. Bagging aims to tackle high variance and high bias [5]. In bagging, prediction is given based on the aggregation of predictions from all the models [6], as is shown in the formula 3. We performed Grid Search on Random Forest, and got the best performance out of all the models we tried. The best parameters after cross validation are shown below in the table.

	boostrap	criterion	max_depth	max_features	min_impurity_decrease	min_samples_leaf	min_samples_split	min_weight_fraction_leaf
RandomForestRegressor	True	mse	10	auto	0.0	5	5	0.0

6.8 Result

	With Country Features		Without Country Features	
	Train MAE	**Test MAE**	**Train MAE**	**Test MAE**
Baseline	103168.42	-	98362.37	-
Linear Regression(poly=1)	41944.49	43114.19	39493.78	91996.27
Linear Regression(poly=2)	29380.57	518590172693.81	30695.32	30819.11
SGD Regressor(poly=1)	41202.81	40841.74	39587.93	65708.15
SGD Regressor(poly=2)	32511.59	32327.99	35362.17	35369.02
Linear SVR	41890.47	43057.26	39424.01	91885.48
Basic Neural Network	26136.77	26288.02	24755.06	24423.62
Random Forest Regression	14039.89	19864.32	7469.46	15982.22

7 Conclusion

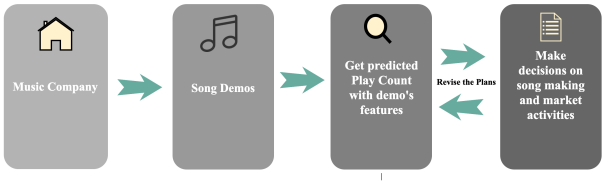

Figure 6: Schematic Diagram of a Reallife Workflow with Song Play Count Predictor

Music streaming services are growing rapidly together with the whole music industry according to IFPI's Global Music Report 2019 [1]. Streaming revenue grows by 34.0% and accounts for almost half (47%) of global revenue, driven by a 32.9% increase in paid subscription streaming. Such growth seems to be very reasonable as technology enables music streaming services to be more functional and more interactive. Searchable music collections and interactions such as recommendation system play a big role in the booming of music streaming. The usage of data has proved its importance in the prosperity of the music streaming And along with the rise of consumer activity in digital music, there are obviously new opportunities for research into using large music collections for discovering trends and patterns in music [9].

In this project, we got predictors (audio features, album features and artist features) from Spotify API and depend variable (play count of each song) from Last.fm API. We used snowball sampling with ten snowballs to crawl data from these two websites. Then we preprocessed raw data (844194 observations) and split them into train, development and test set. Then we tried different machine learning models, including simple linear regression, SGDRegressor, LinearSVR, basic neural network, and random forest regression on data using Grid Search and Randomized Search with cross validations to avoid overfitting and evaluated the trained model with mean absolute error.

From Table 1 we can see that the model with the best performance on with_country dataset is Random Forest Regression, which has an MAE of around 19864. It greatly outperformed the baseline, which has an MAE of 103168 on the dataset with countries. The model with the best performance on without_country dataset is also Random Forest Regression, which has an MAE of around 15982. Random Forest Regression, as a subcase of bagging, handles variance well.

Our project can be very beneficial to music producers by helping them predict the play count which reflects the popularity of a song. If the music producers have made a plan to produce a song, they can use the audio features of the drafted song, the features of the intended artist and the features of the planned album to predict its possible future play count. Thus, they can decide whether they will actually release the song. With further tuning, the music producers can even adjust some specific features of the planned song to achieve higher possible play count. For example, the producers can pick the release date among several possible dates that can result in the highest predicted play count. For another example, the producers can adjust duration time or tempo of the song to get higher possible popularity. In all, our project will benefit the music producers by helping them to do better product development and avoid the loss of product failure.

8 Future Work

We intend to add more regional information to the play count. An extra column of region information can be added. For instance, China, Japan and South are categorized as East Asia. Furthermore, the Y we are using now is limited to the platform of Last.fm, so we think that we can do separate models using different platforms or create a comprehensive model with a Y that is the average of various platforms. Moreover, we can add a column about the frequency of the buzzwords in the lyrics. That means a research into the methodology and engineering work of matching the buzzwords in the buzz words and the popular words in different regions. This is also the place where the techniques of data mining and natural language processing can come into play.

References

[1] IFPI Global Music Report 2019: Global Music Market Grows for Fourth Consecutive Year. (n.d.). Retrieved from https://musicbiz.org/news/ifpi-global-music-report-2019-global-music-market-grows-for-fourth-consecutive-year/.

[2] Nijkamp, R. (2018). Prediction of product success: explaining song popularity by audio features from Spotify data (Bachelor's thesis, University of Twente)

[3] Kirchherr, J., & Charles, K. (2018). Enhancing the sample diversity of snowball samples: Recommendations from a research project on anti-dam movements in Southeast Asia. PloS one, 13(8), e0201710. doi:10.1371/journal.pone.0201710

[4] Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges, 668-669.

[5] Drakos, G. (2019, June 12). Random Forest Regression model explained in depth. Retrieved from https://gdcoder.com/random-forest-regressor-explained-in-depth/.

[6] Shubham,J.(2018, July 6). Ensemble Learning - Bagging and Boosting. Retrieved from https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e.

[7] learn: machine learning in Python - scikit-learn 0.16.1 documentation. (n.d.). Retrieved from https://scikit-learn.org/.

[8] Singh, B.K., Verma, K., Thoke, A.S. (2015). Investigations on Impact of Feature Normalization Techniques on Classifier's Performance in Breast Tumor Classification.

[9] Clement, J.(2018, November 7). U.S.: Internet acitivities. Retrieved from https://www.statista.com/statistics/183910/internet-activities-of-us-users/.

[10] Spotify API. (n.d.). Retrieved from https://developer.spotify.com/documentation/web-api/.

Appendix: Feature Descriptions

Audio Features	
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
Liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
Tempo	The overall estimated tempo of the section in beats per minute (BPM). Tempo is the speed or pace ofa given piece and derives directly from the average beat duration.
Speechiness	Speechiness detects the presence of spoken words in a track.
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic., 1.0 represents highconfidence the track is acoustic.
Instrumentalness	Predicts whether a track contains no vocals. Rap or spoken word tracks are clearly “vocal".
Time Signature	An estimated overall time signature of a track. The time signature (meter) is a notationalconvention to specify how many beats are in each bar (or measure).
Danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements.
Key	The estimated overall key of the section. If no key was detected, the value is -1.
Loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entiretrack and are useful for comparing relative loudness of tracks.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with highvalence sound more positive (e.g. happy, cheerful).
Mode	Indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived.
Track Features	
Duration ms	The duration of the track in milliseconds.
Disc number	The disc number (usually 1 unless the album consists of more than one disc).
Explicit	Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown).
Artist Features	
Artist followers	Information about the followers of the artist.
Artist popularity	The value will be between 0 and 100, with 100 being the most popular. The artist’s popularity is calculated from the popularity of all the artist’s tracks.
Album Features	
Album release date	The date the album was first released, for example "1981-12-15".
Album total tracks	The number of the track. If an album has several discs, the track number is the number on the specified disc.
Album available markets	The markets in which the album is available: ISO 3166-1 alpha-2 country codes.
Listener count	Total number of listeners.
Country	The country of the album.