Cold-Start in Music Recommendation Systems

1st Yuval Dagan
Ben-Gurion University
yuvda@post.bgu.ac.il

2nd Yam Sharon
Ben-Gurion University
yamsha@post.bgu.ac.il

3rd Bracha Shapira
Ben-Gurion University
bracha.shapira@gmail.com

***Abstract*—Most music stream services contain a recommendation system that aims to predict whether a user would like to listen to a specific song. These systems should handle many challenges. One main challenge is the handle of new users which the system doesn't have previous information about - the 'cold-start' challenge. In our paper, we will examine a previous method to recommend different songs, its ability to handle cold-start cases and we will propose a feature engineering method that will aim to improve the accuracy in music recommendations for new users.**

I. Introduction

Spotify, Youtube, and Apple Music are just a few examples of music stream services that are used by millions of users. Any of these services aim to provide its users with the best music experience. As part of the experience, these music services aim to be able to predict whether a specific user would like to listen to a specific song at a certain time. Most of the routine recommendations of these services include a known user whose music preferences are known to the system. As part of Kaggle competitions, the music stream service KKbox made a music recommendation challenge based on real date from their application. Each example of their data includes user and song ID and a label that represents whether the certain user listened to the specific song twice within a month. They also provided basic information about the user.

In section IV we will review the paper of J. Zhang and F. Fogelman-Soulié [8], who proposed a unique method to engineer features which will be used as the input of the deep-learning model that will predict labels for new examples. In their paper, they emphasized the importance of feature engineering and reached great results - 6th place out of more than 1,000.

One main target of music stream services is to attract new users and to keep them. This goal creates a significant challenge to the recommendation system - there is no previous information about these users. To cope with this challenge, we will propose a new features engineering method that will gain more knowledge about new users. In section V we will explain the whole feature engineering process that contained removing specific features that caused over-fitting and adding new features from each song's audio and lyrics.

In our research, we reached great results and we improved the AUC score of the originally proposed method by about 0.03 for cold-start cases.

The remainder of this paper is organized as follows: Section II reviews the background of our work. Section III describes the data of our project. Section VI presents the experimental evaluation metrics and section VII concludes the results of our project.

II. Related Work

Over the years, the consumption of music becomes easier and easier. From the gramophones to cassette players and eventually music streaming services in our mobile phones, everyone has the ability now to access tens of millions of tracks, of any band and artist, at the tap of a touchscreen. To give the users the best streaming experience and to help the users to 'chase a needle in a haystack', these streaming services offer recommendations systems in an attempt to predict the users' musical taste and desire to hear a particular song.

A. Challenges

Choosing a song or a playlist to recommend is not an easy task. Music recommendation systems struggle with many challenges - some related to any recommendation

system and some are unique to music recommendation systems.

1) Amount of Data: As mentioned, the streaming services include tens of millions of tracks by different bands and artists, which relate to different genres and musical tastes.
As a result, it is very difficult to determine just a few songs to recommend out of all the songs.

2) Cold Start: Each music streaming service includes many users and artists. When each user or artist joins the system, the system yet to know much information about them.
As a result, the recommendation system has trouble making accurate and satisfying recommendations.

3) Unpredictability: Humans are different and difficult to predict. Even two people with almost the same musical taste may have a different opinion on a new recommended song. Therefore, no recommendation is certain which may cause losing the confidence of the users.

B. Methods

In this section, we will present existing methods of recommendation systems and its relation and context of music recommendation -

1) Collaborative Filtering: Collaborative filtering (CF) methods produce user-specific recommendations of items based on patterns of ratings or usage (e.g., listened songs) without the need for exogenous information about either items or users [7]. In the field of music recommendation, generating automatic predictions about the musical interests of a user by collecting preferences or taste information from many users.
This filtering technique uses user ratings for its recommendation. Ratings can be divided into two categories—explicit or implicit. Examples of explicit ratings provided by the users are a one-to-five-star rating. Implicit ratings can be obtained by interpreting user behavior. Play counts can be used for implicit rating [6]. The biggest drawback of this system is that at the early stages it provides a poor recommendation. This is known as the cold-start problem, which is part of a bigger problem of sparse data [5]. New users will not get accurate and custom songs and playlist, and on the other hand- new artists and songs will be recommended less than other popular artists and songs since their audience is still not well known yet.

2) Content-Based Filtering: In a content-based filtering technique, songs are recommended based on the comparison done by the system between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document [7]. Acoustic features of the song like loudness, tempo, rhythm, and timbre are analyzed to recommend songs. [6]
The user's preference model does not necessarily contain any information about specific songs that the user does or does not like. Instead, it records the user's response to each of the components in the item model, for example - female vocals [6].

3) Metadata-Based Filtering: This method uses the metadata of the songs to recommend it to the user. Examples of metadata- artist name, genre, album name, year. It is a basic and traditional form of filtering technique, and using it purely (without combining other methods) will lead to poor results, since it can only recommend music based on editorial metadata, and none of the user's information has been considered [6].

4) Emotion-Based Filtering: Music and human emotions are closely connected. Therefore, there is a complete technique designed to filter according to the current emotions and mood experienced by the user. In recent years, the connection between music and emotions has been a topic that has been widely studied. Music evokes different emotions in the listener, which depend on different acoustic features of the song. Research has also shown that a user's mood also plays a key role in selecting the songs. This important conclusion led H. James et al. [1] to perform research that matched custom playlists based on the listener's current mood and emotion, which is measured by image processing of his face.

5) Context-Based Filtering: The context-based model uses public perception of a song in its recommendation. It uses social media sites like Facebook, Twitter, Reddit, and video platforms like YouTube to gather information about the public perception of a song and recommend them accordingly to the listeners. The context-based model also uses the location of the user to recommend songs.

This model can behave efficiently with a small amount of data. With the gathered information, a context-based model can create a For You section for the user based on users listening history and social media engagement of different songs.

C. Existing Systems

Most of the music streaming services use 3 kinds of recommendation systems [6]:

1) User-Centric Experience: This kind of music recommendation is based on each user experience. It tries to learn their habits and recommend accordingly.

2) Listening History: Users' history gives a lot of information about the user such as their overall musical taste, favorite bands, and artists' most loved genres. for example, I Kamehkhosh et al [2] attempt to generate a playlist using a user's recent history. They propose a model which combines -

- **Short-term profile** - the last songs that the user listened to.
- **Personalization Components** - conclusions made from the user's overall listening history such as favorite genres and musical preferences.

3) Hybrid Systems: Hybrid models use a combination of a different method to recommend music.
one example of an implementation of such a model is the proposed model by A. Neil Arnold et al. [3]. They present a model which aims to predict the listener's musical taste by combining collaborative filtering using meta-data features with features extracted from the album picture by the YOLO detection algorithm.
Another example of a hybrid model is the proposed model by S. Oramas et al. [4]. In their paper, they aim to improve the recommendations of new artists. They separated this problem into artist and song levels -

- **Artist Biography** - they used the written biography of each new artist to gain as much information as possible before the artist takes part in the system.
- **Songs' Audio Stream** - they extracted information from the artists' first song to predict their genre and musical area to characterize the artist.

III. Data

The data is a collection of real information taken from users of the well-known KKBOX music streaming service developed in 2005, which includes over 10 million users. Each example of the dataset is the first observable listening event for each unique user-song pair within a specific time duration. As mentioned before, this data was given as part of a Kaggle contest.
The original data contained millions of example for a large number of users and many songs from all over the world. In order to execute our suggested improvement, we had to take only songs with English lyrics. Furthermore, in order to have a full dataset with no missing values, we had to further reduce our dataset. Our final dataset summary of the unique values is presented in figure 1. The data is well balanced as it may be seen in figure 2.

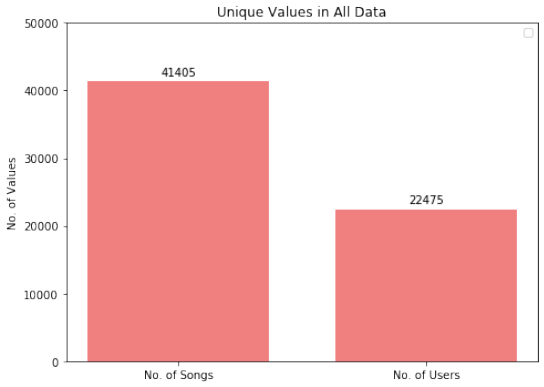

Figure 1. Number of Unique Values of Songs and Users.

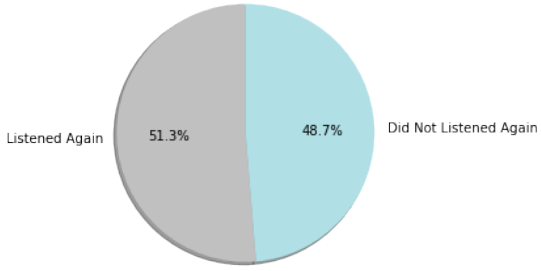

Figure 2. Split of Positive and Negative Labels in the Dataset.

In order to examine the cold start cases, we split the users so that 0.8 of the users are the train set and 0.2 of the users are the test set. Eventually, the train set

contains 632,393 examples while the test set contains 157,866 examples.

IV. Original Paper

The chosen paper presents one of the top solutions of the KKbox's Music Recommendation Challenge.

In their challenge, J.Zhang and F.Fogelman-Soulié [8] aimed to build a recommendation system that can predict whether a user will listen again to a song within one month after the user's very first observable listening event in KKbox. This solution is mostly based upon systematic and extensive feature engineering and an ensemble of simple boosting tree classification algorithms, both of which could easily be used in industry.

A. Features

Feature engineering is a critical step in the data science process, which comes right before the modeling stage. Its purpose is to design, from the raw data, features that will make the model easier and faster to train, and increase its performances.

Feature engineering involves deep data exploration to understand their specificities, as well as domain knowledge to create meaningful and helpful features. In the original paper, J. Zhang and F. Fogelman-Soulié [8] used four types of features:

- **Song Feature**s: Language, Year of publication and Country.
- **User-Artist Features**: the User-Artist SD score.
- **User Features**: Age, Registration & Expiration Info, Number of songs listened to before.
- **User-Song Features**: Age Gap between the song and the user, User-Song similarity (using similarity measured e.g. cosine similarity, Jaccard similarity, etc.), User-Song SVD Score.

.

Parameter	LightGBM 1	LightGBM 2
learning_rate	0.05	0.1
max_depth	15	15
num_leaves	2**8	2**8
application	binary	binary
colsample_bytree	0.7	0.9
subsumple	0.7	0.9
num_boost_round	3100	3100
early_stopping_rounds	10	10

Table I
Split of Positive and Negative Labels in the Dataset.

B. Models

The model used in the original paper is an ensemble of two LightGBM (version 2.0.10) models, a gradient boosting framework that uses tree based learning algorithm, with different sets of parameters. The only preprocessing applied to the features was filling missing values, using Label-Encoder (sklearn version 0.19.1) to map string variables to numeric ones. Missing categorial features values were replaced with new "others" category, and missing numeric values were filled with "-1". The models were trained using 10-fold cross-validation. Each fold has almost equal numbers of positive and negative examples. Then, a LightGBM model was trained on the training part of the cross-validation fold, and evaluate it on the validation part. The next table describes the hyper-parameters used in LightGBM1 and LightGBM2 models,the other hyperparameters that are not listed here, they simply use default values.

C. Evaluation

The original paper's models were evaluated by Time to Train measure and by AUC metric, which will be further explained in section VI. In this subsection, we will present the original paper's results.
Figure 3 presents the AUC results, followed by model type. As can be seen in the graph, the results of the AUC metric for the different types of models are very similar. Yet, the ensemble model gave the best result (72.93).
Figure 4 presents the Time to Train results. In this graph, in opposite to the AUC measure, there is a big difference between the models. LightGBM 1 and LightGBM 2 were trained in the shortest time, and the ensemble model's training was the longest.

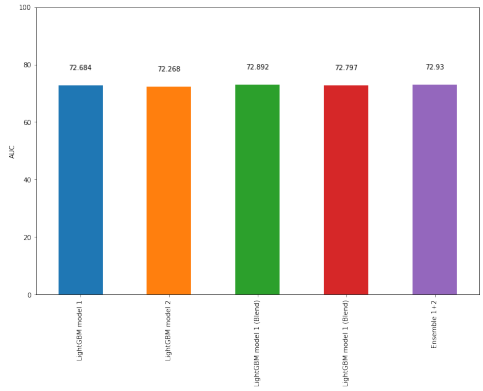

Figure 3. **Evaluation- AUC by model type**

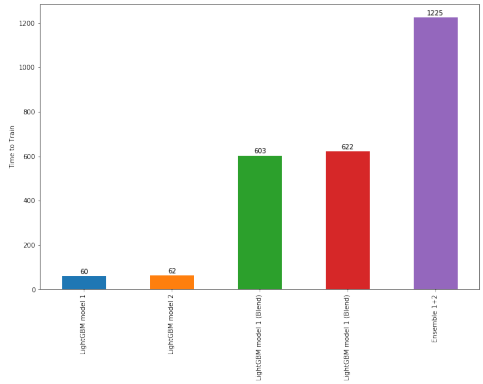

Figure 4. **Evaluation- Time to Train by model type**

D. Drawbacks

In their paper, J.Zhang and F.Fogelman-Soulié [8] did not gave any special attention to new members. So, when a new user is identified, the algorithm does not have any relevant information about this user and his listening habits. Despite this, there is no reference to this case. In other words, they did not handle the cold-start problem.

Another drawback is that in the original paper, they only used basic and superficial information about songs. There is no use in features like audio and lyrics, which in our opinion may be very relevant for the prediction.

E. Our Recreation

In order to recreate the original paper's results, we extracted the original features from the given data. Then, we built the models with the matching parameters, and created the pipeline as mentioned in their paper. Eventually, we received the following results:

Model	**AUC**
LightGBM 1	0.67093
LightGBM 2	0.66754
Ensemble	0.67363

V. Improvement

In our paper, we faced the problem of cold start in music recommendation. As a new user reaches the system, there is not much information about his music preferences which makes the recommendation problem more difficult than it usually is. To solve this problem, we had to figure out features that are not related to the user's history but features that are related to some users and some songs in general.

First, we examined the features presented in the original paper. We assumed that features which express some relation between users and songs or artist are not relevant in this case and may only decrease the result due to the fact that Singular value decomposition (SVD) models don't deal well with users that they have no information about. Moreover, features that express such relations will probably have a relatively large weight in the training step, as these scores have large meaning for users that the system has information about, while at the test set, these features may not be accurate, which will decrease the model's performances.

Additionally, we thought that we still need some features which will describe the user and some relation between the user and the song. To do so, we used the feature which represent the age gap between the user and the song and some information about the user's registration and expiration date as we saw that it achieved good results in the original paper.

Our idea was to describe each song as best as possible due to the fact that these features, combined with some user's features, may give us knowledge without previous knowledge about the user's music preferences. In order to gain as much information as possible, we used features from each song's lyrics and song's audio.

We separated the two groups of the song's features - lyrics and audio, and created three different datasets to examine the effect of each group of features. each dataset also included the remaining features from the original dataset -

- **Lyrics features** - We aimed to describe each song's lyrics so the model will be able to figure the connection between the user's features. We collected song's lyrics from 'Songs Lyrics' database in Kaggle and the PyLyrics Python library. To extract the features, we used two Natural Language Programming (NLP) tools -
  - **Doc2Vec** - Further development of Word2Vec. Using this tool, we transformed each lyrics to a 5 features vector.
  - **Sentiment Analysis** - Using NLTK's sentiment analysis, we added 3 features for each song - negative sentiment, neutral sentiment and positive sentiment.

  We created a model which uses the features from the original model and the lyrics features.
- **Audio features** - We used the 'Spotify Tracks DB' database to get audio features such as popularity, acousticness etc. for our songs to gain as much information as possible about each song. We created a model which uses the features from the original model and the audio features.
- **Combined** - We created a model which combines all features - original, lyrics, and audio,

Table II presents all features and the group it relates to. In conclusion, we will examine 4 approaches -

- **Base Model** - Contains only the features from the 1st group.
- **+ features** - Contain features from the 1st and the 2nd groups.
- **+ lyrics** - Contain features from the 1st and the 3rd groups.
- **All improvement** - Contain all extracted features.

	Name	Range
1st Features from original paper	exp_DM	[0.0 - 102.0]
	exp_MY	[101.0 - 3112.0]
	exp_Y	[12010.0 - 122019.0]
	exp_YMD	[2004.0 - 2020.0]
	reg_DM	[20041016 - 20201017]
	reg_MY	[101.0 - 3112.0]
	reg_Y	[12005.0 - 122016.0]
	reg_YMD	[2004.0 - 2017.0]
	age_diff	[20040326 - 20170131]
	song_year	[1919.0 - 2018.0]
	user_age	[0.0 - 92.0]
2nd Audio feautes	popularity	[0.0 - 93.0]
	acousticness	[1.39e-06 - 0.996]
	danceability	[0.0596 - 0.986]
	duration_ms	[22750.0 - 1412451.0]
	energy	[0.00264 - 0.999]
	instrumentalness	[0.0 - 0.996]
	liveness	[0.00967 - 0.99]
	loudness	[-42.402 - 0.892]
	speechiness	[0.0225 - 0.96]
	tempo	[36.71 - 239.848]
	valence	[0.0176 - 0.989]
3rd Lyrics features	Doc2Vec_0	[-4.4392 - 4.4657]
	Doc2Vec_1	[-4.5004 - 3.7398]
	Doc2Vec_2	[-5.0709 - 3.9588]
	Doc2Vec_3	[-4.5176 - 4.8583]
	Doc2Vec_4	[-2.8271 - 4.2038]
	negative	[0.0 - 0.603]
	neutral	[0.265 - 1.0]
	positive	[0.0 - 0.735]

Table II
ALL FEATURES, THE GROUP THEY RELATE TO AND THEIR RANGE.

		Predicted	
		1	0
Actual	1	True Positive (TP)	False Negative (FN)
	0	False Positive (FP)	True Negative (TN)

Table III
CONFUSION MATRIX

VI. EVALUATION

Any binary classifier labels any given example with 0 or 1. Therefore, there are a few possible cases for each classification of any example - true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

In our research, we will evaluate all models using the receiver operating characteristic (ROC) curve and more specifically by the Area Under the ROC Curve (AUC), as evaluated in the original challenge. the ROC curve is a graph which evaluates the classification model all classification thresholds. This curve depends in two parameters - True Positive Rate (TPR) and False Positive Rate (FPR).

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

VII. Results

In this section, we will present the results of the different models and each feature engineering. As mentioned before, our goal is to find the best solution for cold-start cases for new users. Figure 5 presents the results of each model (X-axis) based on each group of features. As shown in the graph, the original set of features achieved the best results on the train set - around 0.88, while our 3 different sets of features reached a maximum of 75.022. These results support our hypothesis as we assumed that the original set of features (which contains SVD features) will create a large over-fit, and will lead to bad results on the test set which is separated from the train set.

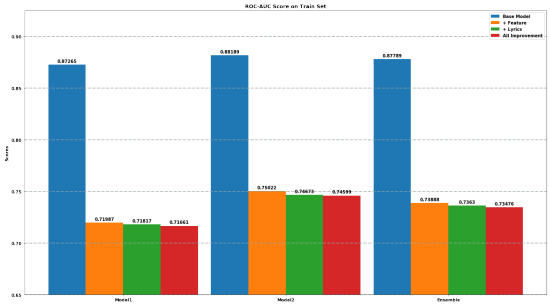

Figure 5. Results on train set.

Table IV and figure 6 presents the final results on the test set. As shown, the base model achieved the best score of 0.58685. It is possible to see that there is a big difference in the score of those feature group in the train and test sets, as we anticipated.

We may see that each improvement reached different results while the audio features achieved better results. We believe that this fact is a result of the direct connection between audio attributes and users' music preferences. Moreover, the lyrics features extracted using different NLP tools may not be accurate enough to characterize the lyrics as they should. The best result on the features set which included the base features and the audio features is 0.61448 which was the result of the ensemble model while the ensemble model reached the score of 0.61303 with the ensemble model.

Eventually, we examined that we achieved the best results using a combination of all features as each gives more knowledge on the relation between the song and the user.

Our best result is **0.61733** which we achieved with all the features and the ensemble model. We improved the cold-start cases by 0.03048 which is a significant improvement. Figure 7 presents the RoC graph of the base model while figure 8 presents the RoC graph of the best result.

Improvement \Model	**Model1**	**Model2**	**Ensemble**
Base Model	0.58685	0.57893	0.58340
With Features	0.61446	0.61022	0.61448
With Lyrics	0.61437	0.60797	0.61303
All improvement	0.61707	0.61379	**0.61733**

Table IV
Final Results

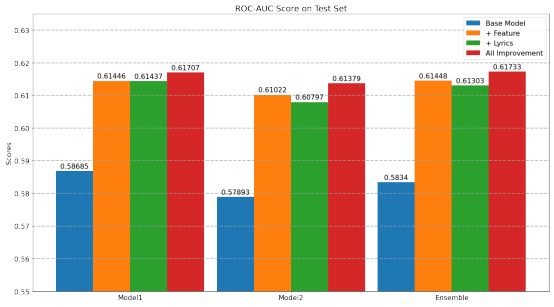

Figure 6. Results on test set.

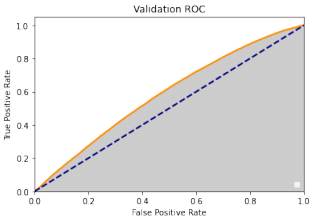

Figure 7. RoC curve of the base model's best result.

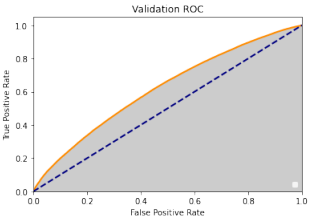

Figure 8. RoC curve of the Our best result.

VIII. Discussion

We set a goal to take special care of recommendations for new users which the systems have no previous information about. We presented the method of feature engineering in order to improve the results for cold-start cases. We used audio features and lyrics features, in addition to features describing the user. We evaluated

our models using AUC score, as used in the original challenge. The results matched our base hypothesis - the original solution over-fitted to the data and did not handled cases of cold-start.
Eventually, our method gained AUC score of 0.61733, which is an improvement of 0.03048 from the original solution's results.

IX. Code

The full code files are in the following github repository - https://github.com/yamsharon102/Cold-Start-Music-Recommendation-System.git

References

[1] H Immanuel James, J James Anto Arnold, J Maria Masilla Ruban, M Tamilarasan, and R Saranya. Emotion based music recommendation system. *EMOTION*, 6(03), 2019.

[2] Iman Kamehkhosh, Dietmar Jannach, and Lukas Lerche. Personalized next-track music recommendation with multi-dimensional long-term preference signals. In *UMAP (Extended Proceedings)*, 2016.

[3] Anand Neil Arnold and S. Vairamuthu. Music recommendation using collaborative filtering and deep learning. In *international Journal of Innovative Technology and Exploring Engineering*, 2019.

[4] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. A deep multimodal approach for cold-start music recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, pages 32–37, 2017.

[5] Jacob O'Bryant. A survey of music recommendation and possible improvements, 2017.

[6] Dip Paul and Subhradeep Kundu. A survey of music recommendation systems with a proposed music recommendation system. In *Emerging Technology in Modelling and Graphics*, pages 279–285. Springer, 2020.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[8] Jianyu Zhang and Françoise Fogelman-Soulié. Kkbox's music recommendation challenge solution with feature engineering. In *11th ACM International Conference on Web Search and Data Mining WSDM*, 2018.