Machine learning - based recommendation model for bipartite networks

Project summary: Jan Majher
FMFI UK, Machine learning 2020/2021

2021-01-29

Contents

1 Introduction

1.1 Motivation

Main objective of this project is to create a recommedation model using structural properties of a bipartite network. Bipartite network models relationship between two classes of objects. Dataset, selected for this project, contains social networking, tagging, and music artist listening information from a set of 2K users from Last.fm online music system. Consequently, bipartite network will represent the relationship between the users and the artist, i.e. whether a user listens to a artist. Problem we are facing is link prediction, which is machine learning context binary classification. To solve it, we will apply various supervised learning methods, with intetion to obtrain satisfactory results. Main goal is to build model, that recommends artists to the user, based on his preferences, popular of artists, etc. This idea was inspired by: [1], where authors introduced approach of using biparite newtorks in recommender systems.

1.2 Dataset

We will take advantage of data structure called hetrec2011-lastfm-2k. Detailed description can be found at [2] and it is available to download at [3]. This data structure contains information about 1892 users and 17632 artists. We will be particulary using folowing parts of the data: 92834 user - listened artist relations, i.e. listenings, and 12717 user - friend relations.

1.3 Workflow

Firstly, we will construct biparite graph. Classes of vertices are the users and artists, respectively. We will begin with separating user - artist relations data and we will use first part to create edges between the verices, where the listening count represents weight of the edge. This will be graph G at time $T0$. For second part of data, we will consider this relations happening in future. Including them in $G[T0]$ we obtain $G[T1]$. As displayed in figure 1, if a non-edge in $G[T0]$, becomes an edge in $G[T1]$, it is positively labeled, otherwise negatively labeled. This represents target variable, for each user-artist pair.

First predictor, we will consider is similarity metric ItemRank. ItemRank is random-walk based ranking algorthm [4], which utilizes correlation between artist verices. We will use graph $G[T0]$ to calculate scores in order to obtain ranking of artists for each user according to their expected preferences. Listening count will be considered as rating from user, i.e user-artist listening relations are reviews. If the score is high, user would prefer artist with high probability. Consequently, user-artist connections that exists in $G[T0]$ will have high score. We will finish this feature engineering process by calculating, for each user-artist pair, how many of the user's friends also listen to the artist.

This concludes data preparation and now dataset is ready for traning and testing various models. As occurence of edge is our target variable we are

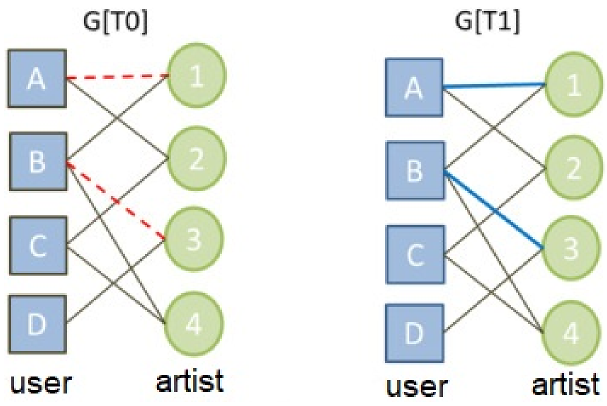


Figure 1: New edge occurences

engaging in supervised learning. Specifically, binary classification. We will train model using methods such as Logistic regression (LR), Support vector classification (SVC) and Random forest classifier (RFC). To get best results, we will attempt to tune parameters. Accuracy, AUC, specificity and sensitivity will be our main performance measures. In the end, we will compare each model and select finest one.

That concludes workflow of this project. In the next sections we will go futher into detail for each step of the process.

2 Data preparation

2.1 Data analysis and cleaning

We began with analysing nature of our data. First thing we noticed, was that number of the artists is far superior than number of the users. As shown in figure 2, there is roughly 10k artist with only one fan (one user listening to them). We considered these examples as artists with a low popularity, or samples with a low information value.

Without loosing much user-artist relation information, we decided to ommit these data examples. By ommiting these artists we lost little over 10k reviews, but we also drastically reduced size of the bipartite graph. Reviews are now covering much more bigger portion of the graph. Futher on, we have considered following two datasets:

- **Dataset_1**: 1881 users, 2828 artists, 71426 listening relations
- **Dataset_2**: 1885 users, 6953 artists, 82155 listening relations

where **Dataset_1** and **Dataset_2** contains artists with 5 or more fans and 2 or more fans, respectively.

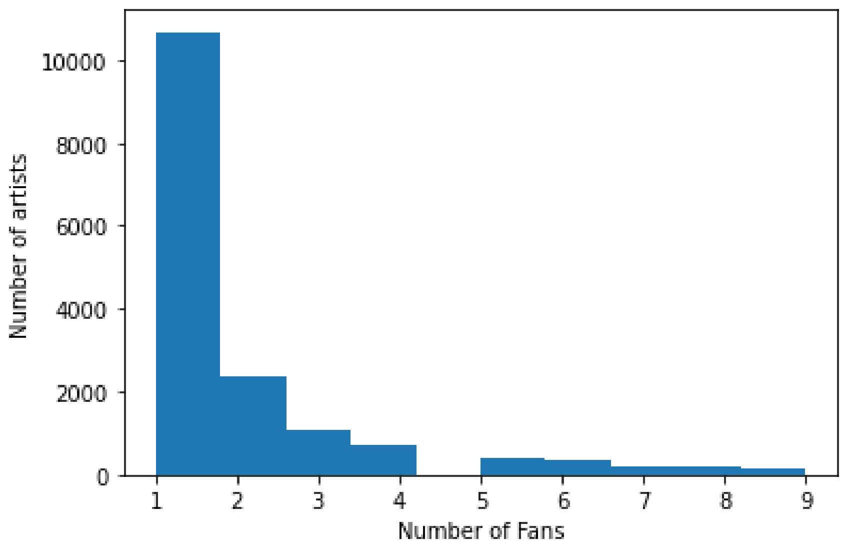


Figure 2: Histogram of artist's listeners

NOTE: Following Data Preparation 2 and Binary Classification 3 was firstly done on smaller dataset **Dataset__1**. We considered significant artists with 5 and more fans and also extraction was much simpler due to difficult calculation process of ItemRank. Nevertheless, we managed to obtain **Dataset__2** and execute similar data preparation process and apply same machine learning methods.

2.2 ItemRank and labeling

Next we divided listening counts into 10 quantiles with equal amount of sampel, this is called quantile binning. We then rated 10 % of samples with highest listening count with number 10. This represented artist's rating by the user, i.e. edges in the graph obtained weights on scale from 1 to 10.

Following this adjustment, we were ready to construct graph $G[T0]$ with randomly selected 40k samples from **Dataset__1** and 45k samples from **Dataset__2** and calculate ItemRank score for all user-artist pairs. These were training sets for ItemRank algorithm. To calculate ItemRank we used **ItemRank.py** available at [5] (**Dataset__2** - 8 hours running time).

Samples that were not used for the Ranking algorithm were labed positively. To balance things out, we also randomly selected 50k negatively labeled samples for **Dataset_1** and after some consideration 35k samples for **Dataset_2** (Remember, **Dataset_2** was processed later and we tried to apply better approach. Here for example to evenly matched positively labeled pairs). Label of the pair, i.e. 1 and 0 for positive and negative sample, respectively, is our response variable in the supervised learning models presented in section 3.

2.3 Feature engineering

As mentioned before, main feature is similarity metric ItemRank. Therefore we can assign particlar score to pairs occuring in our datasets. Another feature we added is number of the users friends, that listened to the correspoding artist. Engineering of this feature comes from intuition, that mutual friends would listen to the same artists, as they share same taste or recommend artists to each other. Values for this feature were extracted from $G[T0]$, as those are the user-artist relations we know of. This predictor has chaotic distribution. For example in (**Dataset_2** we have mean value of 0.79, even though maximum value is at 50, meaning most of samples are 0. After examening number of histograms, we decided to create rating scale from 0 to 5 as shown in table 1 below:

# of friends listening to artist	Friends Rating
0	0
1	1
$[2, 5)$	2
$[5, 11)$	3
$[11, 20)$	4
≥ 20	5

Table 1: Friends Rating

This distribution seemed reasonable as number of samples with higher counts of friends listening is decreasing rather steeply. Rating from 0 to 5 was choosen in order to have similar scale as ItemRank score. This way, normalization was not necessary. Nevertheless, the more friend are listening to the artist, the higher is value of Friend Rating.

2.4 Final product

Throughout this process lost negligable amount of the samples. For example, if some artists were not in $G[0]$ after spliting data, we did not obtain their ItemRank score. Also after randomly selecting samples, we had to make sure there are no duplicates or already labeled pairs from $G[T0]$. At the end we acquired two final datasets with following characteristics:

- **Dataset_1**
  - 1868 users, 2770 artists
  - 80 137 samples: 31349 positive, 48788 negative
- **Dataset_2**
  - 1880 users, 5426 artists
  - 71 430 samples: 35768 positive, 35662 negative

3 Binary classification

In this section we present methods and stategy, in order to obtaining the best possible binary classifier for the link prediction problem.

3.1 Patterns in data

Before fully engaging in parameter tuning and model selection, we visualized our data in order to have initial sense of what should we expect. On next two page we can observe various figures. Figure 3 showcases distribution of ItemRank scores depending on label. We can notice that positively labeled pairs tend to have higher scores. Higher number of user's friends also indicates to be characteritic of positively labeled samples. Figure 5 showcases both features together, where we can observe positive impact from both of them. Same figures for **Dataset_2** are displayed on next page, where result of observation came to same conclusion.

All this visualization gives us certain amount of confidence, that binary classifiers, could be useful link prediction models, that will eventually offer users good recommendations.

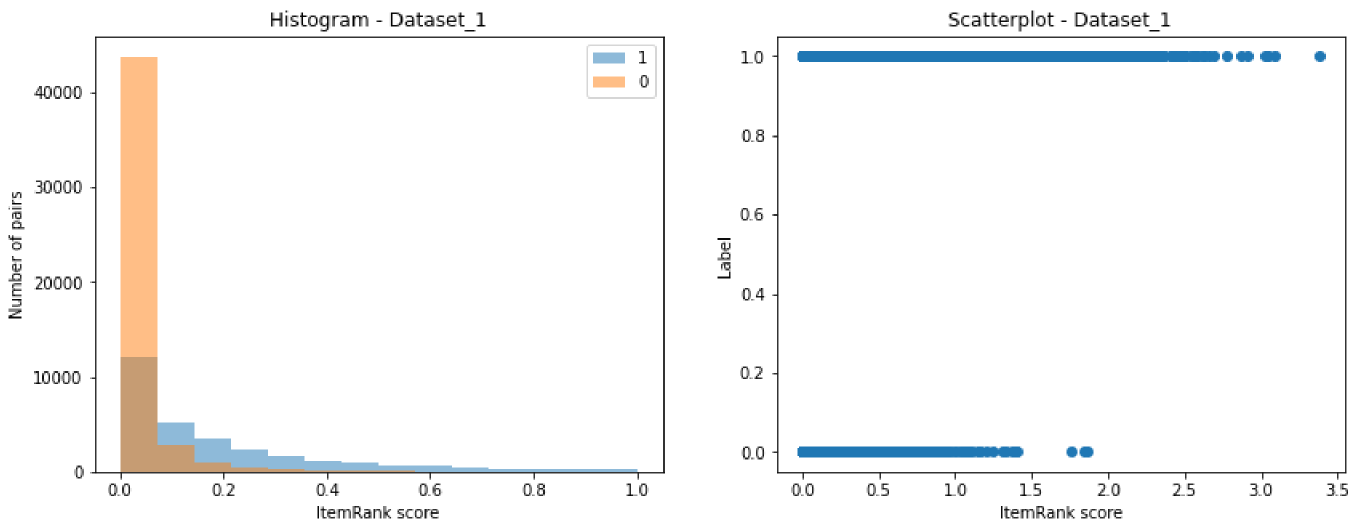


Figure 3: ItemRank score

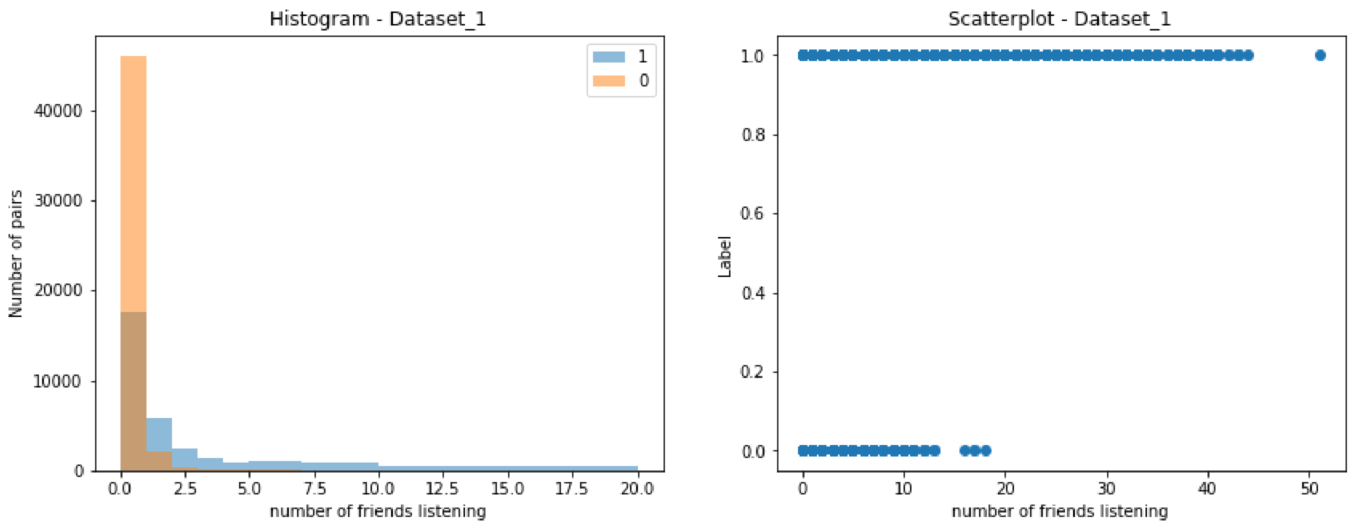


Figure 4: Number of listening friends

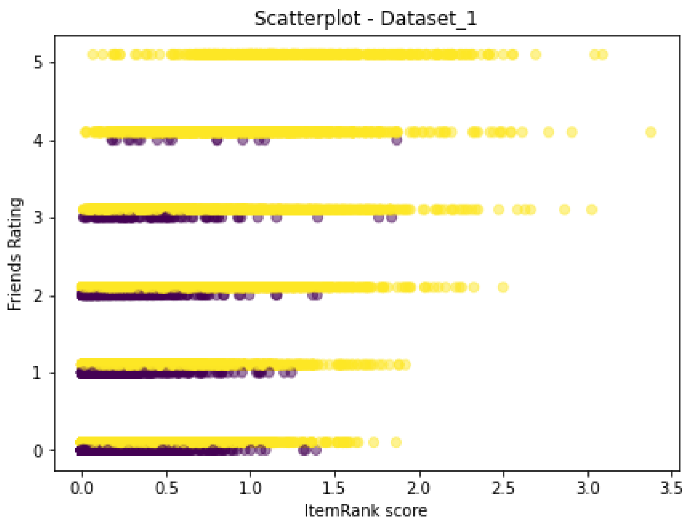
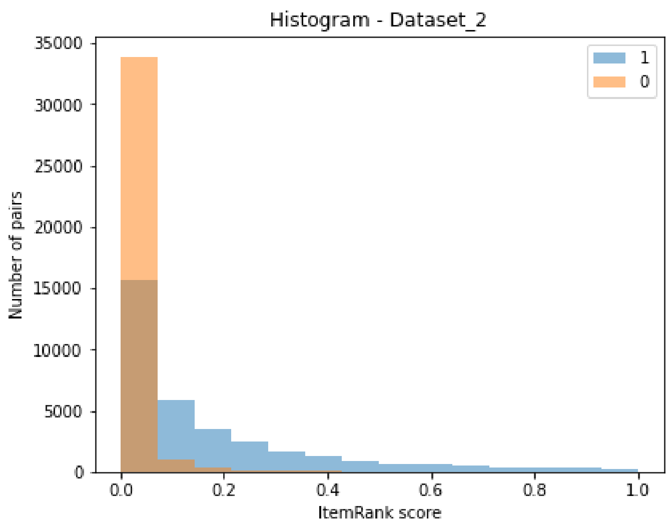


Figure 5: Friends Rating & ItemRank score



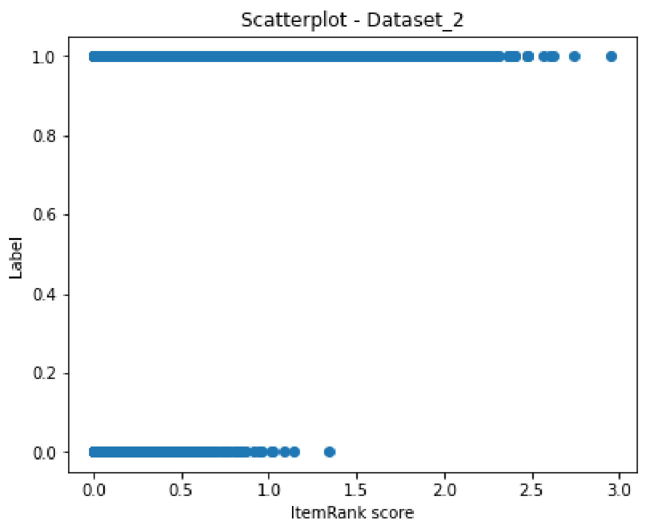
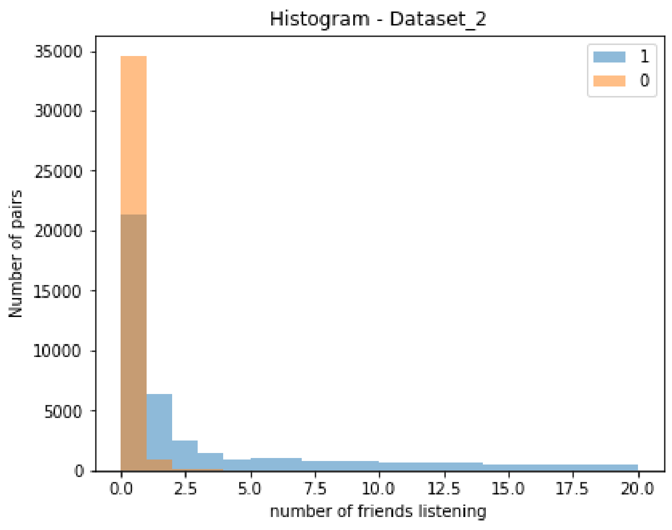


Figure 6: ItemRank score



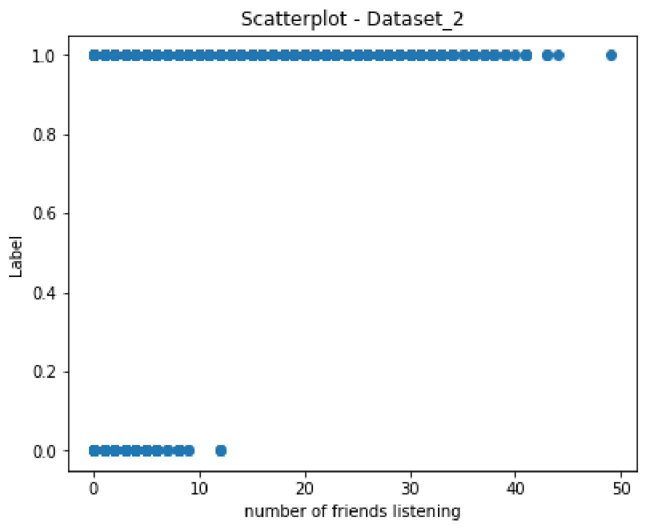


Figure 7: Number of listening friends

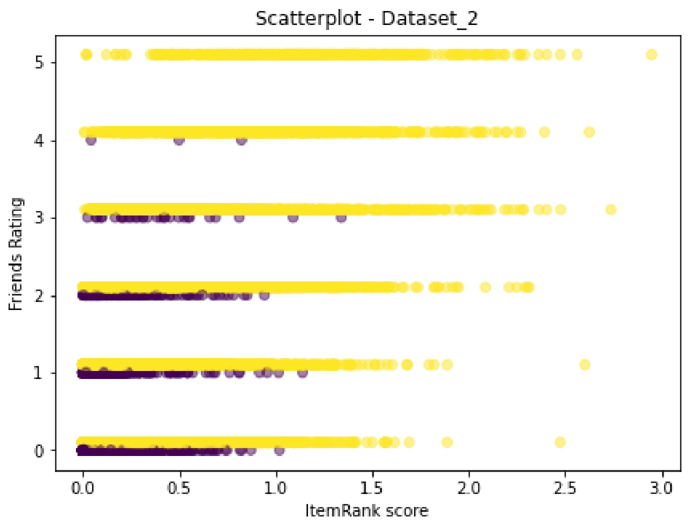


Figure 8: Friends Rating & ItemRank score

3.2 Model selection strategy

First we were required to split our data into the training and the test sets. For **Dataset_1**, which is little bigger, we used 65k samples for training and remaining 15 137 served as test set. In case of **Dataset_2** 55k samples were used for training and 16 430 for testing. Overall we were trying to find 3 best models for 3 approaches presented here:

- **Approach_1**
  - data: **Dataset_1**
  - features: ItemRank
- **Approach_2**
  - data: **Dataset_1**
  - features: ItemRank, Friends Rating
- **Approach_3**
  - data: **Dataset_2**
  - features: ItemRank, Friends Rating

NOTE: This is to show how we approched problem of selecting best model. We started with **Dataset_1** and only using the ItemRank feature and we built our way up to **Dataset_2** and utilizing both features.

As we mentioned before, we attemped to predict link occurance using LR, SVC and RFC. For each method we selected several parameters, that we tried to optimize for best performance. We decided to execute grid search for hyper-parameter tunning. On some occasions, we zoomed in on smaller interval of parameter values. We executed 5-fold cross validation, to measure accuracy. Based on this results we selected parameters used in the models.

Logistic regression

For LR we experimented with value of parameter C. Parameter C controls the penalty for misclassified training examples. In all 3 approaches, cross validation result pointed out that default value for $C = 1$ is sufficient. By incresing penalty we would only get sligthly better results, but it is not worth it, as it is better to keep models simple.

Support vector classification

In case of the SVC we also experimented with parameter C and selected values in range from 1 to 10. In addition, Gaussian kernel proved to be better fit then linear kernel, in all approaches. Lastly we attemped to optimize parameter γ.

Random forest classifier

Hyper-parameter tunning for RFC suggested that trees should not go deep to reache leaves, which comes from the fact that we have only two features. Also our forests were not in need of big number of the trees to get good results, i.e. small number of the estimators.

To sum it up, models we trained were not that complex. And that is all right, when we realize that pattern we were trying to capture is not complex either.

4 Evaluation and interpretation of results

4.1 Evaluation results

After we trained models on training set and selected best ones, we evaluate their performarce using test set. We used confusion matrix and ROC curve to calculate following measures.

- **Accuracy** $= \frac{\text{True positives + True negatives}}{\text{Total number of samples}}$
- **Sensitivity** $= \frac{\text{True positives}}{\text{True positives + False negatives}}$
- **Specificity** $= \frac{\text{True negatives}}{\text{True negatives + False positives}}$
- **AUC** - Area under ROC curve, that plots true positive rate (sensitivity) against false positive rate(1 - specificity)

Results are displayed in folowing tables 2,3 and 4 for all approaches, respectively. We can notice that models performed better when we added another feature and also when we broaden our data with more artists in **Approach_3**. But there are some slight differences, for example in **Approach_3**, LR and RFC have same AUC, but Specificity differs. Acurracy is also different for the benefit of RFC. SVC was sligthly behind in terms of AUC, but matched RFC in accuracy.

Approach_1			
Classification models	LR	SVC	RFC
Accuracy	0.769	0.785	0.786
Sensitivity	0.510	0.612	0.677
Specificity	0.937	0.897	0.858
AUC	0.843	0.820	0.842

Table 2: Evaluation of **Approach_1**

Approach__2			
Classification models	LR	SVC	RFC
Accuracy	0.786	0.797	0.799
Sensitivity	0.550	0.660	0.661
Specificity	0.938	0.886	0.889
AUC	0.857	0.829	0.857

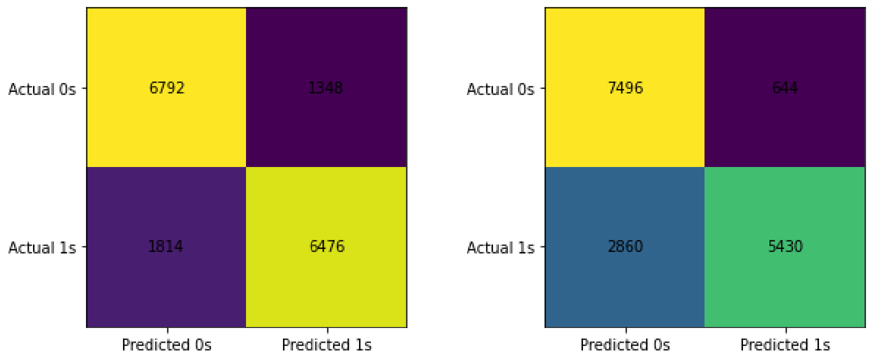
Table 3: Evaluation of **Approach__2**

Approach__3			
Classification models	LR	SVC	RFC
Accuracy	0.787	0.804	0.808
Sensitivity	0.655	0.733	0.781
Specificity	0.921	0.875	0.834
AUC	0.877	0.863	0.877

Table 4: Evaluation of **Approach__3**

4.2 Discussion

So how to choose between these models ? Going back to what this project is about, we would like to recommend artists to users, and we want recommendations to be best as possible. In approach **Approach__3** we considered 6953 artist, which is a lot. However in real-life scenario we would recommend up to 50 artists to the user at most. Therefore, we are looking for high specificity, so that there is smallest number as possible of falsely labeled negatives, i.e bad artist recommended. While still considering **Approach__3**, model with highest specificity is LR (9a,9b).



(a) Random Forest

(b) Logistic Regression

Figure 9: Confusion matrices

Unfortunately LR and RFC have the same ROC curve, as displayed in figure 10, where curves are overlapping each other. That means they have same true positive rate (sensitivity) against false positive rate(1 - specificity), just for different tresholds. And results in table 4 are no the same, because default threshold for all presented models is 50%.

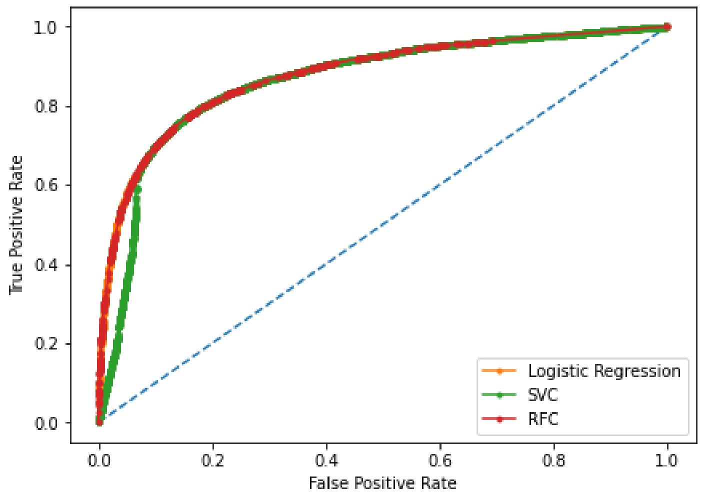


Figure 10: ROC curve

In conclusion, we can choose either Random forest or Logistic regression. Nevertheless, it is in our best interest to set higher thresholds, to obtain best options for recommendation.

NOTE: All datasets and source code is available at [6]

References

1. https://www.sciencedirect.com/science/article/pii/S0378437120300844?via%3Dihub#b18
2. http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-readme.txt
3. https://github.com/yaxue1123/data-analysis-of-last.fm
4. https://www.aaai.org/Papers/IJCAI/2007/IJCAI07-444.pdf
5. https://github.com/arashkhoeini/itemrank
6. https://github.com/JanMajher/ML_Project