# ASSIGNMENT 2 (Chen Liang, 46275313)

## Data Processing

### QUESTION1

(a)

Using `hdfs dfs -ls -R hdfs:///data/shared/msd | awk '{print $8}' | sed -e 's/[^-][^\/]*\//--/g' -e 's/^/ /' -e 's/-/|/'` command to export the directory of the whole dataset, it could be seen that the `msd` folder contains `audio`, `genre`, `main` and `tasteprofile`.

Audio includes the folder attributes that contains the attribute for each feature, which are saved as csv format. The following part is the various feature files. Because of the large amount of data, each feature file is split into several partitions in gz format. In addition, audio

```
 |---------audio
 |----------attributes
 |------------msd-jmir-area-of-moments-all-
v1.0.attributes.csv
                ...
 |------------msd-tssd-v1.0.attributes.csv
 |----------features
 |------------msd-jmir-area-of-moments-all-v1.0.csv
 |--------------part-00000.csv.gz
                  ...
 |--------------part-00007.csv.gz
 |------------msd-jmir-lpc-all-v1.0.csv
                  ...
 |------------msd-tssd-v1.0.csv
 |--------------part-00000.csv.gz
                  ...
 |--------------part-00007.csv.gz
 |----------fn.txt
 |----------fn1.txt
 |----------statistics
 |------------sample_properties.csv.gz
 |---------genre
 |----------msd-MAGD-genreAssignment.tsv
 |----------msd-MASD-styleAssignment.tsv
 |----------msd-topMAGD-genreAssignment.tsv
 |---------main
 |----------summary
 |------------analysis.csv.gz
 |------------metadata.csv.gz
 |---------tasteprofile
 |----------mismatches
 |------------sid_matches_manually_accepted.txt
 |------------sid_mismatches.txt
 |----------triplets.tsv
 |------------part-00000.tsv.gz
                  ...
 |------------part-00007.tsv.gz
```

**Figure 1 Directory of msd**

also contains the statistics folder, which is also stored in a compressed file as gz format, providing information of all sample properties, such as track_id, artist_name and so on.

Genre mainly contains the genre and style of the song. A file containing three tsv(using a tab (Tab '\t') as a separator for values) files.

Tasteprofile contains two sets in total. Mismatches contains mismatch data and mismatch data that is acceptable, both in txt format. Triplets provides user, songs and play counts, and files are stored in compressed in gz files.

(b)

Repartition is to repartition RDD by shuttle and get a new RDD with more partitions and better parallelism. However, in this dataset, larger files like features have been partitioned for storage and can be processed in parallel, thus there is no need to repartition.

(c)

The number of rows in each dataset is calculated using MapReduce defined by linecount.jar, and the result is collected as followed. The number of unique song is 384546.

**Audio:**
**attributes(3929)**
**features:**
**msd-jmir-area-of-moments-all-v1.0.csv(994623)**
**msd-jmir-lpc-all-v1.0.csv(994623)**
**msd-jmir-methods-of-moments-all-v1.0.csv(994623)**
**msd-jmir-mfcc-all-v1.0.csv(994623)**
**msd-jmir-spectral-all-all-v1.0.csv(994623)**
**msd-jmir-spectral-derivatives-all-all-v1.0.csv(994623)**
**msd-jmir-spectral-derivatives-all-all-v1.0.csv(994623)**
**msd-marsyas-timbral-v1.0.csv(995001)**
**msd-mvd-v1.0.csv(994188)**
**msd-rh-v1.0.csv(994188)**
**msd-rp-v1.0.csv(994188)**
**msd-ssd-v1.0.csv(994188)**
**msd-tssd-v1.0.csv (994188)**
**fn.txt (13)**
**fn1.txt (0)**
**statistics (992866)**
**genre:**
**msd-MAGD-genreAssignment.tsv (422714)**
**msd-MASD-styleAssignment.tsv (273936)**
**msd-topMAGD-genreAssignment.tsv (406427)**
**main:**
**summary (2000002)**
**tasteprofile:**
**mismatches (20032)**
**triplets.tsv (48373586)**

**Figure 2 Row Count for each file**

### QUESTION2

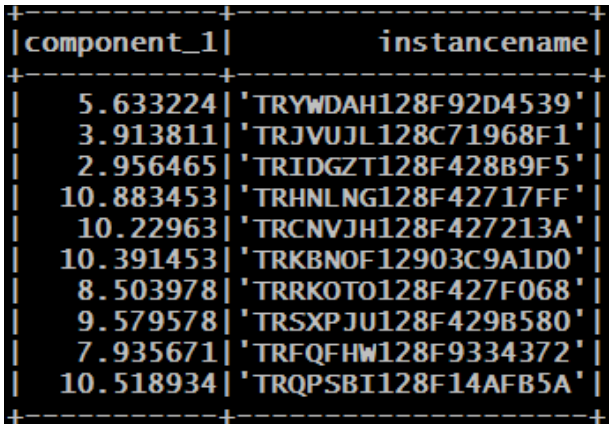
(a)

The mismatches folder contains two files, one is a file of mismatching songs and the other is an acceptable mismatching songs file, which is due to mismatches caused by errors such as uppercase and lowercase in the matching process, and the two files do not have any overlap. If the triplets file is leftjoin with the mismatched file using the song name, then 'track' which is brought by the mismatches file will be null if the song is not mismatched.

If the acceptable mismatches songs are taken into account and removed, the number of remaining observations after removing the mismatched song is 45785819. If the acceptable mismatches songs are not treated as mismatching songs, the number of remaining observations is 45785100.

(b)

First, identify the data type of feature involved in attribute, which are string and numeric in featurerh. Build a dictionary to map string and numeric to StringType () and DoubleType () in spark respectively. Then the attribute file is turned into one column through RDD, and the column is traversed with StructField (), all type is transformed into pyspark.sql.types and the schema of the feature is defined with StructType ().Next, I use the featurerh dataset and set the data type with the defined schema.



| component_1 | instancename |
|---|---|
| 5.633224 | 'TRYWDAH128F92D4539' |
| 3.913811 | 'TRJVUJL128C71968F1' |
| 2.956465 | 'TRIDGZT128F428B9F5' |
| 10.883453 | 'TRHNLNG128F42717FF' |
| 10.22963 | 'TRCNVJH128F427213A' |
| 10.391453 | 'TRKBNOF12903C9A1D0' |
| 8.503978 | 'TRRKOTO128F427F068' |
| 9.579578 | 'TRSXPJU128F429B580' |
| 7.935671 | 'TRFQFHW128F9334372' |
| 10.518934 | 'TRQPSBI128F14AFB5A' |

**Figure 3 featurerh**

## Audio similarity

### QUESTION1

(a)

First, the numeric variables in the data set are extracted, and then describe () is implemented to calculate the descriptive statistics of each variable, shown in Figure 4.

| summary | component_0 | component_1 | component_2 | component_3 | component_4 | component_5 |
|---|---|---|---|---|---|---|
| count | 994185 | 994185 | 994185 | 994185 | 994185 | 994185 |
| mean | 10.747208130044188 | 8.804710263856345 | 7.545513634260266 | 6.632828271028025 | 6.130460207571026 | 5.86765469807729 |
| stddev | 4.745722803168785 5 | 3.623742452542451 | 2.990341180856473 | 2.773977960095909 | 2.634144727349877 | 2.4497105599761326 |
| min | 2.23E-4 | 2.23E-4 | 2.23E-4 | 2.23E-4 | 2.23E-4 | 2.23E-4 |
| max | 73.000064 | 50.474817 | 52.279847 | 63.014065 | 38.099166 | 33.066823 |

**Figure 4 Descriptive Statistics of featurerh**

The Statistics.corr () function is mainly used when calculating the correlation among the features. The function of compute_correlation_matrix (DF, method='pearson') is defined to realize the whole computing process. In the function, the data frame is converted to a matrix first, and then the correlation of all the features in the matrix is calculated using Statistics.corr (), and a correlation data frame is output as the result (shown in Figure 5).

In addition, the features with strong correlation need to be identified, so the high_correlation (cor_matrix) function is defined to identify the data in the calculated correlation data frame and get a list of all the feature pairs with the correlation greater than 0.7 (shown in Figure 6). A total of 36 pairs of features have strong correlation.

| | component_0 | component_1 | component_2 | component_3 | component_4 | \ |
|---|---|---|---|---|---|---|
| component_0 | 1.000000 | 0.750977 | 0.673178 | 0.628888 | 0.573104 | |
| component_1 | 0.750977 | 1.000000 | 0.751841 | 0.759427 | 0.689649 | |
| component_2 | 0.673178 | 0.751841 | 1.000000 | 0.721451 | 0.707931 | |
| component_3 | 0.628888 | 0.759427 | 0.721451 | 1.000000 | 0.735750 | |
| component_4 | 0.573104 | 0.689649 | 0.707931 | 0.735750 | 1.000000 | |

| | component_5 | component_6 | component_7 | component_8 | component_9 | \ |
|---|---|---|---|---|---|---|
| component_0 | 0.499907 | 0.518697 | 0.519017 | 0.445463 | 0.452813 | |
| component_1 | 0.592217 | 0.648734 | 0.657164 | 0.581596 | 0.574251 | |
| component_2 | 0.716099 | 0.676746 | 0.643414 | 0.605237 | 0.621948 | |
| component_3 | 0.626658 | 0.706648 | 0.774696 | 0.687388 | 0.638198 | |
| component_4 | 0.637212 | 0.695184 | 0.687289 | 0.705349 | 0.771830 | |

| | ... | component_50 | component_51 | component_52 | \ |
|---|---|---|---|---|---|
| component_0 | ... | 0.090859 | 0.152777 | 0.180875 | |
| component_1 | ... | 0.145577 | 0.230543 | 0.267657 | |
| component_2 | ... | 0.242243 | 0.301876 | 0.324356 | |
| component_3 | ... | 0.186955 | 0.288761 | 0.336784 | |
| component_4 | ... | 0.222771 | 0.322656 | 0.362837 | |

**Figure 5 Correlation Matrix**

```
[('component_0', 'component_1'),
 ('component_1', 'component_0'),
 ('component_1', 'component_2'),
 ('component_1', 'component_3'),
```

**Figure 6 Feature Pairs with strong correlation**

(b)

Figure 7 is the distribution of genre labels. As can be seen from the figure, the distribution of genre varies much, the number of tracks belonging to Pop_Rock type is the largest, exceeding 200000, followed by electronic music, about 40000 in total.

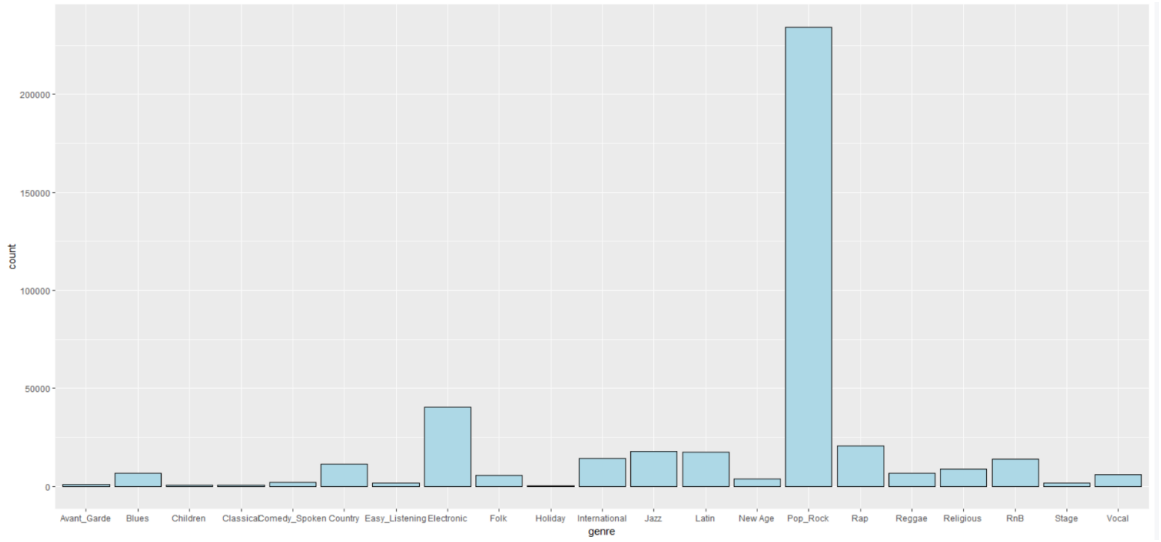



**Figure 7 Distribution of genre**

(c)

After merging genre dataset and audio features dataset, the new dataset is obtained, since the number of variables contained in features is large, so the track ID and genre label are selected to display the result in Figure 8.

| instanceName | GENRELABEL |
|---|---|
| 'TRAAAAV128F421A322' | Pop_Rock |
| 'TRAAABD128F429CF47' | Pop_Rock |
| 'TRAAADT12903CCC339' | Easy_Listening |
| 'TRAAAED128E0783FAB' | Vocal |
| 'TRAAAEF128F4273421' | Pop_Rock |

**Figure 8 Genre Label for Tracks**

## QUESTION2

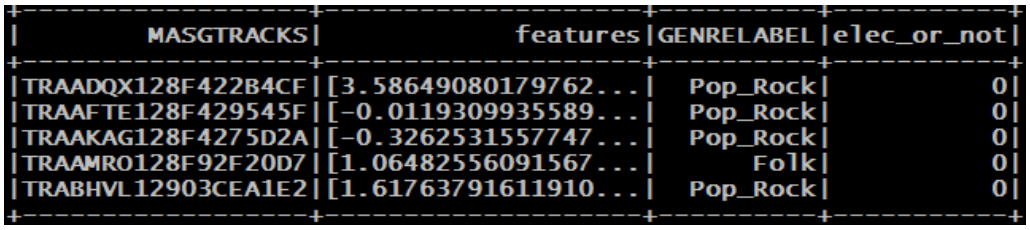
(a)

Since the response variable are binary and many of the variables in the dataset have strong collinearity, therefore, logistic regression, random forest and support vector machine can be used to fit the model. Before implementing logistic regression and support vector machine, we need to firstly standardize the data. However, random forest could randomly select variables to get decorrelated trees, so the original dataset could be used directly.

(b)

The function elec_or_not (songtype) is defined to mark the tracks with genre ‘electronic’ as class 1, marking tracks belonging to other genres as class 0. The total number of class 1 is

40048, and the total number of class 0 is 372982. In order to conduct the model fitting conveniently in the following step, the original data and standardized data are all marked in this section.



| MASGTRACKS | features | GENRELABEL | elec_or_not |
|---|---|---|---|
| TRAADQX128F422B4CF | [3.58649080179762... | Pop_Rock | 0 |
| TRAAFTE128F429545F | [-0.0119309935589... | Pop_Rock | 0 |
| TRAAKAG128F4275D2A | [-0.3262531557747... | Pop_Rock | 0 |
| TRAAMRO128F92F20D7 | [1.06482556091567... | Folk | 0 |
| TRABHVL12903CEA1E2 | [1.61763791611910... | Pop_Rock | 0 |

**Figure 9 Classification assignment**


**Figure 10 Classification Count**

(c)

Since in the data set, class 1 accounts for only about 1/10 of the total data set. Therefore, we should pay attention to the sampling balance of two classes in the training set and test set. Stratified sampling is used here (the sampleBy method will flip a coin to decide whether an observation will be sampled or not, therefore requires one pass over the data, and provides an expected sample size[1]). We use 70% of the data set as training set, which contains 289010 observations and the rest observations (124021) as test set.

(d, e, f)

Logistic Regression

When fitting logistic regression models, the standardized data is used. In the process of model fitting, elasticnetparam is set to 0.5, so that the penalty items of lasso and ridge regression are introduced to the model and do parameter shrinkage and variable selection. In addition, the threshold is decreased to 0.3, since the proportion of electronic types is too low, then the model tends to predict all the results as 0 to improve the accuracy and stability of the model.

The following figure shows the calculated metric, including 'weightedPrecision', 'weightedRecall' and 'accuracy'.

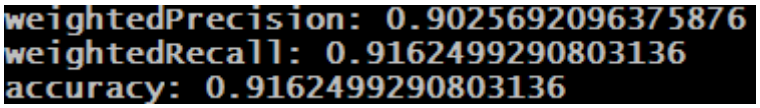



**Figure 11 Evaluation for Logistic Regression Model**

Random Forest

Original data set is used in Random Forest buiding, because random forests can solve the

problem of high collinearity by selecting variables at random for each node building. In the fitting process, numtree is set to 20, being used to balance the high variance of each large tree. The evaluation results of the model are shown in the Figure 12.

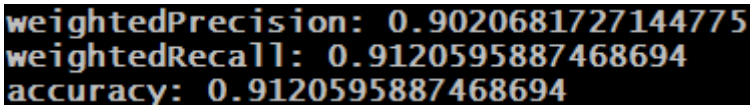



**Figure 12 Evaluation for Random Forest Model**

Support Vector Machine

SVM uses the inner function kernel to replace the nonlinear transformation to the high dimensional space. When the size of dataset is huge, the calculation of the matrix will consume a large amount of machine memory and time. Therefore, when we execute SVM, the sample containing 30 features after PCA being complemented. The following figure shows the performance of the SVM model.

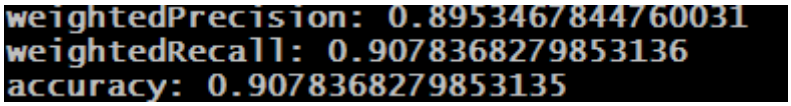



**Figure 13 Evaluation for SVM Model**

From the evaluation of each model, it can be seen that on matter for recall, which is the accuracy of 1 prediction or the precise, the best one of the three models seems the logistic regression model, followed by the random forest model. however, what we have to consider is that during the process of building the model, the logistic regression parameters are adjusted so that the model avoids the trend of predicting only 0. SVM is more sensible to the outlier of the sample. Since the distribution of the sample is very unbalanced, the performance of SVM performance is poor. Therefore, in a comprehensive way, random forests perform best in this sample.

## Question3

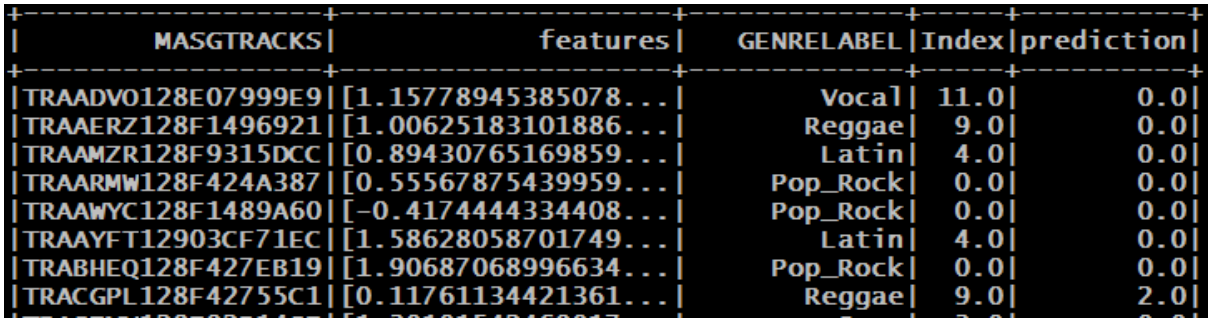
(a)

OneVsOne trains K (K - 1) / 2 two - element classifiers for a multiple K classification problem; each classifier receives a pair of class samples from the initial training set and must learn to distinguish between the two classes [1]. When prediction, there will be a vote: all K (K - 1) / 2 interpreters are applied to an unknown sample, and the class that gets the most "+1" prediction will be the prediction result of the combined classifier. OneVsAll needs to establish a unique classifier for each class. All samples belong to this category are positive, and the rest are negative cases [1]. When executing OneVsAll, we first establish a binary classifier and bring it into and build a onevsall classifier.

In most cases, onevsone performs better, because onevsall is easy to produce bias. However, when data sets are large, onevsone is computationally expensive.

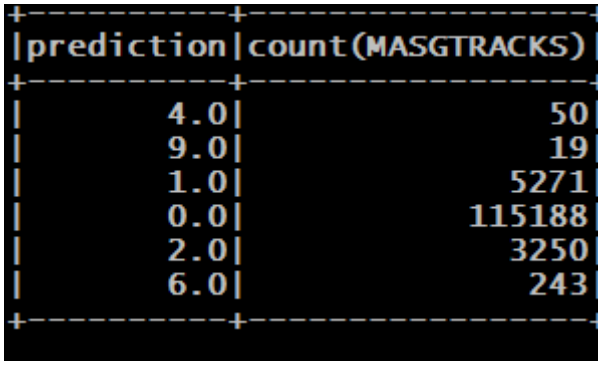
(b)~(c)

Use StringIndex() to set an integer label for each category to represent this genre label. Then SVM is used for multi-classification. The following Figure shows the results of classification and the distribution for different genres in the results. As can be seen from the figure, only 6 categories are contained in the result. The calculation shows that the accuracy of the classification is only 59%. The reason may be due to the large variance in the distribution of genre, which might produce high error rate when using OneVsAll.



| MASGTRACKS | features | GENRELABEL | Index | prediction |
|---|---|---|---|---|
| TRAADVO128E07999E9 | [1.15778945385078... | Vocal | 11.0 | 0.0 |
| TRAAERZ128F1496921 | [1.00625183101886... | Reggae | 9.0 | 0.0 |
| TRAAMZR128F9315DCC | [0.89430765169859... | Latin | 4.0 | 0.0 |
| TRAARMW128F424A387 | [0.55567875439959... | Pop_Rock | 0.0 | 0.0 |
| TRAAWYC128F1489A60 | [-0.4174444334408... | Pop_Rock | 0.0 | 0.0 |
| TRAAYFT12903CF71EC | [1.58628058701749... | Latin | 4.0 | 0.0 |
| TRABHEQ128F427EB19 | [1.90687068996634... | Pop_Rock | 0.0 | 0.0 |
| TRACGPL128F42755C1 | [0.11761134421361... | Reggae | 9.0 | 2.0 |

**Figure 14 Prediction for SVM Model for Multiclassification**



| prediction | count(MASGTRACKS) |
|---|---|
| 4.0 | 50 |
| 9.0 | 19 |
| 1.0 | 5271 |
| 0.0 | 115188 |
| 2.0 | 3250 |
| 6.0 | 243 |

**Figure 15 Count for Prediction of SVM Model**

## Song recommendations

### Question1

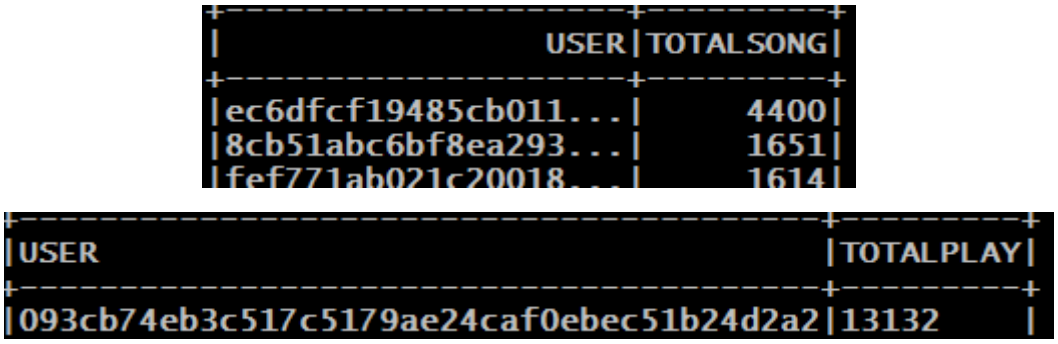
(a)

Using distinct () and count () commands, the result shows that there are 378309 unique songs and 1019318 unique users.

(b)

Calculate the total number of play count for each user, then sort it, and get the most active users. As can be seen from the Figure, the most active users with most songs played have played 4400 songs which account for 1.1%, and the user with most play count have played

202 unique songs in the original database, which accounts for around 0.05% of the total number of unique songs.



| USER | TOTALSONG |
|---|---|
| ec6dfcf19485cb011... | 4400 |
| 8cb51abc6bf8ea293... | 1651 |
| fef771ab021c20018... | 1614 |

| USER | TOTALPLAY |
|---|---|
| 093cb74eb3c517c5179ae24caf0ebec51b24d2a2 | 13132 |

**Figure 16 Most Active User**

(c)

In this section, the two distributions are shown in boxplots, and the total number of play count is used to measure the popularity of songs and if the user is activity. As can be seen from the figure, the two distributions both show obvious right skewed, that is, the median is obviously larger than the mean and mode.

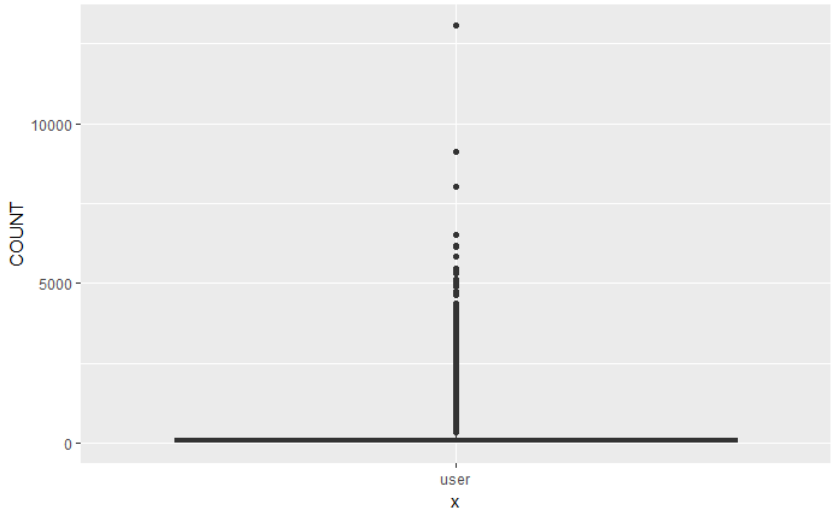



**Figure 17 Distribution of playcount of user**

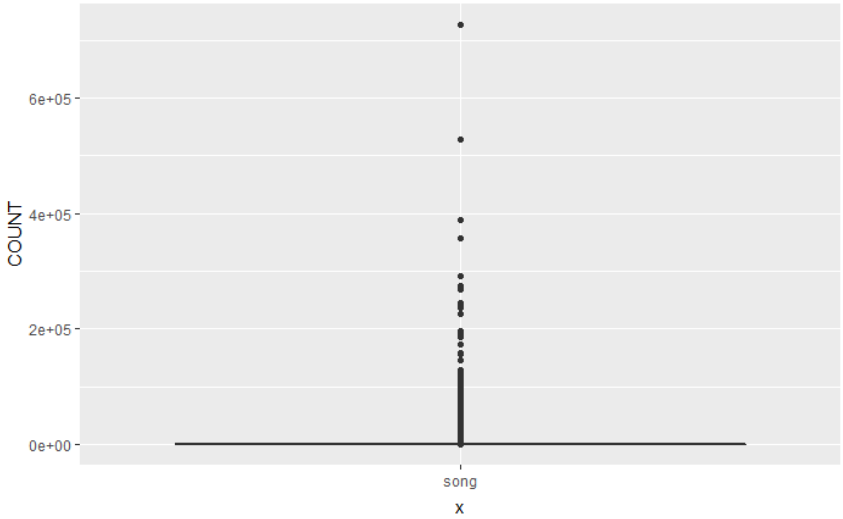



**Figure 18 Distribution of playcount of song**

(d)

ApproxQuantile () is used to obtain the 1/4 quantiles of the two distributions, which are 16 and 8, respectively. Therefore, M is set as 8 and N is 16. Then the users with less than 8 songs and the songs with less than 16 play counts are deleted as less significant observations. After that, the data sets of useful users and useful songs are inner joined with tasteprofile are inner join, and the remaining dataset contains 44615167 observations in total.

(e)

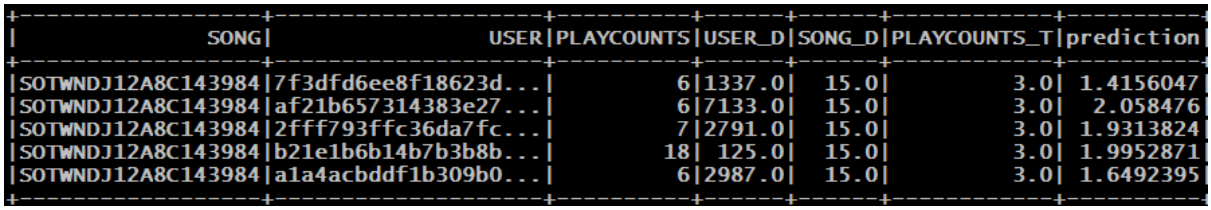
When splitting training set and test set, firstly, compose the data with uniqueuser and uniquesong are both 1 as dataset 1, and then the data set 2 is composed of data with both of them are 0. After that, randomly extract 80% from each of the two sets as the training set, and the rest as the test set.

**Question2**

(a)

When model is built, dummy variables are set for user ID and song ID, and playcount is used as a rating standard for user preferences (use QuantileDiscretizer () to convert playcount to categorical features and set numBuckets as 6). Setting the parameter rank as default value. Then the dataset is split into training set, which contains 35692141 observations, accounting 80% of the dataset, and test sets, which takes 20%.

The test set is predicted using the obtained model, and the predicting result is as follows. In the fitting process, the system could just identify 3 buckets, which is 1, 2 and 3.

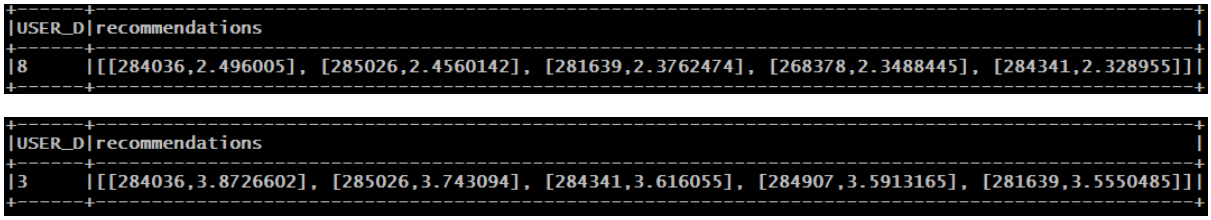


| SONG | USER | PLAYCOUNTS | USER_D | SONG_D | PLAYCOUNTS_T | prediction |
|---|---|---|---|---|---|---|
| SOTWNDJ12A8C143984 | 7f3dfd6ee8f18623d... | 6 | 1337.0 | 15.0 | 3.0 | 1.4156047 |
| SOTWNDJ12A8C143984 | af21b657314383e27... | 6 | 7133.0 | 15.0 | 3.0 | 2.058476 |
| SOTWNDJ12A8C143984 | 2fff793ffc36da7fc... | 7 | 2791.0 | 15.0 | 3.0 | 1.9313824 |
| SOTWNDJ12A8C143984 | b21e1b6b14b7b3b8b... | 18 | 125.0 | 15.0 | 3.0 | 1.9952871 |
| SOTWNDJ12A8C143984 | a1a4acbddf1b309b0... | 6 | 2987.0 | 15.0 | 3.0 | 1.6492395 |

**Figure 19 Prediction of Recommendation Model**

(b)

The following two groups are the predictions made by the models I obtained. The calculation results show that the accuracy of the model is 65.26%.



| USER_D | recommendations |
|---|---|
| 8 | [[284036,2.496005], [285026,2.4560142], [281639,2.3762474], [268378,2.3488445], [284341,2.328955]] |



| USER_D | recommendations |
|---|---|
| 3 | [[284036,3.8726602], [285026,3.743094], [284341,3.616055], [284907,3.5913165], [281639,3.5550485]] |

**Figure 20 Recommendations by the Model**

(c)

Precision:

If a relevant document never gets retrieved, we assume the precision corresponding to that relevant doc to be zero

MAP:

Now perhaps most commonly used measure in research papers
assumes user is interested in finding many relevant documents for each query
requires many relevance judgments in text collection

NDCG:

Normalize DCG at rank n by the DCG value at rank n of the ideal ranking
The ideal ranking would first return the documents with the highest relevance level, then the next highest relevance level, etc
NDCG is now quite popular in evaluating Web search [2]

**Question3**

(a)

The model obtained is only trained on the existing data. Therefore, when we encounter the

cold start problem, that is, when new users appear, because of the lack of information, it may not be able to get the desired result, and we cannot have data to get the recommendation for the user. At the same time, since no information about the recommended songs themselves are involved, so popular songs which have more data are more likely to be recommended This kind of recommendation is hard to surprise the users.

(b)

First, if content based recommendation is used, this method needs more metadata. If more metadata is involved, the model based on songs aiming to distinguish the content of a single song would get more accurate results in recommending songs.

In addition, the mixed recommendation model is used to solve the cold start problem. It includes content based and collaborative filtering recommendation methods. Another way is to mine some distinctive songs to provide ratings to new users, so as to quickly learn user preferences.

(c)

In practical applications, the songs being recommended to user could belong to the song list from other users that have high similarity with the user according to the collaborative filtering model based on users. At the same time, what we can do is that build a content based filtering model using the big amount metadata about songs in the database. This can more accurately recommend songs that users like.

## Reference,

[1] http://mlwiki.org/index.php/One-vs-All_Classification

[2] https://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf