DATA420 Assignment 2

Yiwei Chen

Table of Contents

Data Processing

Q1 Document overview

(a) Over view of the dataset.

File size of each main directory, the largest folder “audio” is 12Gb which contains audio features and audio attributes. Majority of data are stored in compressed csv or tsv format, by default when we load compressed gz file, we get 1 partition for each gz file, level of parallelism is low.

```
13167872421   52671489684   /data/msd/audio                   12Gb
31585889      126343556     /data/msd/genre                   30Mb
182869445     731477780     /data/msd/main                    174Mb
514256719     2057026876    /data/msd/tasteprofile            490Mb
```

Structure of the MSD as below, with count lines result followed by file names.

```
/data/msd/
       ├─audio
       │  ├──attributes
       │  │          └─msd-jmir-area-of-moments-all-v1.0.attributes.csv                  21
       │  │          └─msd-jmir-lpc-all-v1.0.attributes.csv                              21
       │  │          └─msd-jmir-methods-of-moments-all-v1.0.attributes.csv               11
       │  │          └─msd-jmir-mfcc-all-v1.0.attributes.csv                            27
       │  │          └─msd-jmir-spectral-all-all-v1.0.attributes.csv                    17
       │  │          └─msd-jmir-spectral-derivatives-all-all-v1.0.attributes.csv        17
       │  │          └─msd-marsyas-timbral-v1.0.attributes.csv                          125
       │  │          └─msd-mvd-v1.0.attributes.csv                                      421
       │  │          └─msd-rh-v1.0.attributes.csv                                       61
       │  │          └─msd-rp-v1.0.attributes.csv                                       1441
       │  │          └─msd-ssd-v1.0.attributes.csv                                      169
       │  │          └─msd-trh-v1.0.attributes.csv                                      421
       │  │          └─msd-tssd-v1.0.attributes.csv                                     1177
       │  ├──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-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-trh-v1.0.csv                                                994188
       │  │          └─msd-tssd-v1.0.csv                                               994188
       │  └─statistics
       │             └─sample_properties.csv.gz                                       992866
       ├─genre
       │    ├─msd-MAGD-genreAssignment.tsv                                            422714
       │    ├─msd-MASD-styleAssignment.tsv                                            273936
       │    └─msd-topMAGD-genreAssignment.tsv                                         406427
       ├─main
       │    └─summary
       │      ├─analysis.csv.gz                                                       1000001
       │      └─metadata.csv.gz                                                       1000001
       └─tasteprofile
                  ├─mismatches
                  │         ├─sid_matches_manually_accepted.txt                       938
                  │         └─sid_mismatches.txt                                      19094
                  └─triplets.tsv                                                      19094
```

(b) Repartition

Spark splits data into partitions and executes computations on the partitions in parallel. As our datasets are in compressed file, dataset will be loaded with low number of partitions, by using repartition method, for example we can increase a dataset from 8 partitions to 32 partitions (or even more partitions based on number of executors and cores), that will increase to 4 times parallelism than of before. It can effectively reduce processing time by using optimal partitions. Repartition will shuffle the data frame, it could take some time for large file.

(c) Count lines

The dataset contains 1,000,001 tracks in metadata, audio features datasets have more than 990,000 lines, which means each audio feature dataset covers most of tracks. Genre assignment dataset have less than 430,000 lines, indicate less than half of tracks are used in genre assignment data.

Q2, Data processing

(a) Remove mismatched songs in taste profile.

Use left anti join takes out acceptable mismatch, and then anti join to remove mismatched files.

(b) Load audio feature attributes

Create a list for the file names then read attribute files to fit schema format, the schemas will be used for loading audio features datasets.

Audio similarity

Q1, Audio features

(a) Audio feature dataset

Pick the 3rd dataset in audio features, i.e. “msd-jmir-methods-of-moments-all-v1.0.csv”, it is the one with smallest file size, also has less columns (variables), size of audio features datasets varies from file to file, a small file means lesser processing time.

There are 11 columns in the dataset, 5 of them are “standard deviation” values for each track, another 5 of them are “average value” of some features for each track, and last column is track id.

Mean of the "standard deviation" columns range from 0.015 to 5297870, mean of the average" columns range from 0.35 to 143165, there are large variations between columns.

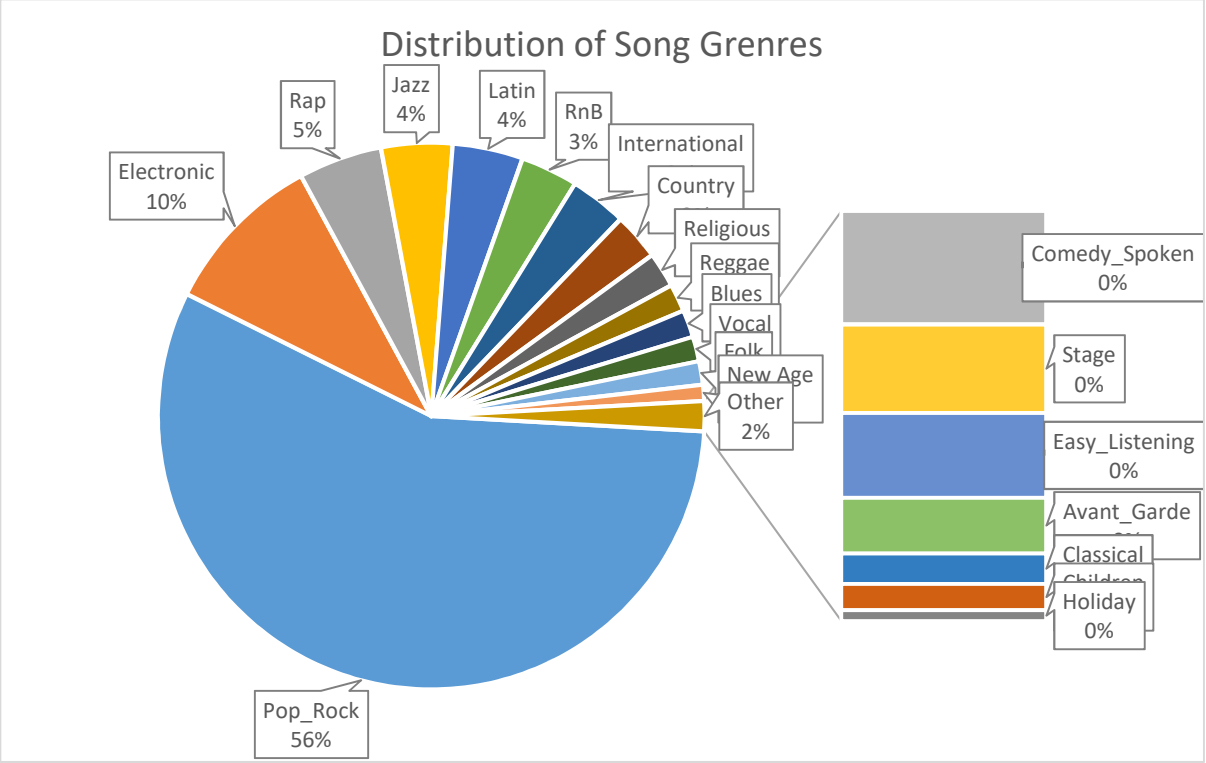
Correlation matrix shows there are a few variables with high correlations, such as column 2 and 3, column 3 and 4, column 4 and 5, column 7 and 8, column 9 and 10, the correlation coefficients are all more than 0.8 between each pair of columns.

	1	2	3	4	5	6	7	8	9	10
1	1	0.426	0.296	0.061	-0.055	0.754	0.498	0.448	0.167	0.1
2	0.426	1	0.858	0.61	0.434	0.025	0.407	0.396	0.016	-0.041
3	0.296	0.858	1	0.803	0.683	-0.082	0.126	0.185	-0.088	-0.135
4	0.061	0.61	0.803	1	0.942	-0.328	-0.223	-0.158	-0.245	-0.221
5	0.055	0.434	0.683	0.942	1	-0.393	-0.355	-0.286	-0.26	-0.212
6	0.754	0.025	-0.082	-0.328	-0.393	1	0.549	0.519	0.347	0.279
7	0.498	0.407	0.126	-0.223	-0.355	0.549	1	0.903	0.516	0.423
8	0.448	0.396	0.185	-0.158	-0.286	0.519	0.903	1	0.773	0.686
9	0.167	0.016	-0.088	-0.245	-0.26	0.347	0.516	0.773	1	0.985
10	0.1	-0.041	-0.135	-0.221	-0.212	0.279	0.423	0.686	0.985	1

(Correlation Matrix)

(b) MSD All Music Genre Dataset

There are 422714 songs in All Music Genre Dataset (MAGD), majority of songs are Pop_Rock, Pop_Rock takes up 56%, and the second largest genre group is Electronic which takes up 10%, followed by Rap and Jazz.



(C) Merge the genres dataset and audio features dataset by track id.

Choose the 3rd audio feature dataset. Inner join the column 'MSD_TRACKID' and 'track_id', there is comma for string value in column 'MSD_TRACKID', remove the comma then join on track id to get the data frame with audio features and track_id.

Q2, Binary classification

(a) Classification algorithms

There are a few options in spark.ml library for binary classification, includes linear SVMs, logistic regression, decision trees, random forests, gradient-boosted trees, naive Bayes. Linear SVMs and Logistic regression are linear models; decision trees, random forests, gradient-boosted trees are tree-based models.

The dataset has 10 input variables and 420,620 observations, pervious analysis shows that mean value and standard deviation varies greatly between variables, it would be good to normalize dataset. We also find there are some high correlations between some pairs of variables, we could also do PCA to reduce variables, but it is considered optional because 10 variables are not a big number and we have large number of observations. PCA may be necessary if we choose another bigger audio feature dataset with much more columns.

Since we don't know the details how does the input variables are generated, therefore we choose to try out a few different classification models to see which is better fit.

For linear models, we choose Logistic regression as it is popular method for binary classification, it works better with independent variables, it is relatively fast algorithm with easy interpretable results. In logistic regression, we can adjust the prediction threshold to get different prediction results.

Then we choose tree (CART) model, tree model is kind of universal model for all kinds of problems with classification or regression. The process of tree model is easy to understand, and in most of cases, tree model can produce accurate, maybe not the best, but reasonably good

result for predictions. There are some improvement models based on tree model, such as random forest and gradient-boost tree classifier, those improved tree based models aim to reduce over fitting and increase test accuracy, since our dataset has large number of observations and a few correlated variables, it would be beneficial to try gradient boost tree classifier. The one major draw back of those improved tree models are that they are slow to train.

Naïve Bayes is a fast model, with less interpretability, it works well on a small amount of training set. Sometimes it works well on some particular data.

Linear Support Vector Machine is also a linear method, it is effective in high dimensional space, and it is memory efficient.

So we start with Logistic regression, since it is an popular model, it could be used as a baseline for other models in terms of time and accuracy. The try gradient boost tree classifier model, if it has good accuracy, then Random forest should fit the data as well. Then pick one from Naïve bays or SVM to try out the performance.

(b) Convert to binary

Use withColumn function to convert genre column value into “Electronic” and “Other” genres. There are 40,666 counts of “Electronic” and 379,954 of “Other’. Since we know from above Q1, electronic music is about 10% of total data frame, that means number of “Other” genres is about 9 times of “electronic’. The two classes are extremely imbalance, it may impact on model accuracy.

(c) Train test split and sampling

Since the dataset is imbalanced, at approximately 1: 9 ratios, consider this imbalance will impact on model accuracy, we will try to compare difference of balanced and imbalanced training set. There are two sampling methods to balance dataset, one is subsampling, the other is oversampling. However, to ensure the test credibility, we need to split dataset into training and testing sets before any pre-processing, this is to keep the original form of dataset as test set, i.e. test set is supposed to be imbalanced.

We choose to split dataset at 0.7 to 0.3 ratio as training and testing sets, the split used stratified sampling, with 70% from “Electronic” and 70% from “Other” for training set, and the rest 30% of each class for testing sets. Nevertheless, because the dataset has quite large number of observations in each class, simple random 7:3 split will achieve the similar results. After the split, we get training set with 294,505 observations, testing set with 126,115 observations. The training and testing sets are then processed through pipeline for string Index (convert “Electronic” to binary 1, “Other” to 0), Vector Assembler and scaled.

The inherent training set has about 10% being “electronic” class, and about 90% being “other” class, to balance two classes in training set, we performed subsampling and oversampling. By subsampling, random pick equivalent number of observations of “other” to “electronic”, it is a process of scaling down the bigger class to the smaller class level. In contrast, oversampling takes a process of scaling up the smaller class to the bigger class level by repeat sampling from the smaller class.

Through subsampling and oversampling, we get two additional training sets:

Stratified training set:

“Electronic” class: 28, 440 observations.

“Other” class: 265,952 observations.

Subsampling training set:

"Electronic" class: 28,559 observations.

"Other" class: 28,440 observations.

Oversampling training set:

"Electronic" class: 266,285 observations.

"Other" class: 265,952 observations.

Compare three sampling models, by subsampling, it loses about 90% of observations

Due to the fact that subsampling lose about 90% of training observations in "Other" class, it has risk of under fit because of smaller training set, on the other hand, smaller training set means less computation time to fit model, since the subsampling training set has 56,999 observations for 10 variables, we believe the risk cause by losing observations is small. Instead of losing training observations, oversampling keeps all training inputs, because of the up scaling, the training data size is nearly 10 times as subsampling training set. This will cause significant more modeling fitting time. Oversampling also has risks of overfitting, because of sample observations were picked many times, especially when using oversampling in cross validation models, it could happen that the same observations in validation set are in training set as well, thus the cross validation model will consider it find the best model, but model turns out to be overfitted, however this could be improved with methods like SMOTE(Synthetic Minority Over-sampling Technique).

There for we proceed with 3 training sets of original stratified training set, subsampling training set and oversampling set, use all three datasets to fit model and compare the outcomes in next questions.

(d) Train model

Fit three models with 3 training sets, and use fitted model to predict testing set, examine and evaluate production results.

(e) Performance

Because of the input dataset is imbalanced, the testing dataset is imbalanced as well. In testing data, we have 113,956 count of "Other" and 12,272 count of "Electronic", assume that we have a classification model does nothing but predict all test data as "Other", then the model will get 113,956 correct and 12,272 wrongs, so the do-nothing-classification achieves prediction accuracy of 113956/(113956+12272) = 90.3%, that is a rather good prediction result (especially for a model does no calculation).

If the task is to distinguish Electronic from all other tracks, we need a more useful model. To measure the performance, we should use recall and precision.

$$\text{Recall} = \frac{True\ Positive}{\text{True Positive } + \text{ False Negative}} = \frac{\text{True Positive}}{\text{All Actual Electronic}}$$

$$\text{Precision} = \frac{True\ Positive}{\text{True Positive } + \text{ False Positive}} = \frac{\text{True Positive}}{\text{All Predicted Electronic}}$$

Recall means the percentage of real "Electronic" tracks that the model can find from all 'Electronic' tracks. Precision means the percentage that the model classified as "Electronic" are correct.

Another important measurement is AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve. ROC is a probability curve and AUC represent degree or measure of

separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. AUC ROC is an overall effective measurement of model performance.

Then we can look at the model evaluations results, results contain cross validation result.

Logistic Regression	Unbalanced Traning set	Subsampling	Oversampling	Cross Validation
Area under ROC	0.696	0.701	0.701	0.732
Accuracy	90.2%	67.3%	66.8%	67.7%
Precision	5.0%	17.5%	17.3%	18.4%
Recall	0.0%	63.5%	63.8%	67.6%

(Logistic Regression)

Gradient-boosted tree classifier	Unbalanced Traning set	Subsampling	Oversampling	Cross Validation
Area under ROC	0.807	0.810	0.810	0.810
Accuracy	90.8%	74.9%	75.1%	75.1%
Precision	62.1%	23.6%	23.8%	23.8%
Recall	12.5%	70.6%	70.9%	70.9%

(Gradient-boosted Tree)

LinearSVC classifier	Unbalanced Traning set	Subsampling	Oversampling	Cross Validation
Area under ROC	0.319	0.627	0.612	0.676
Accuracy	90.3%	58.4%	56.1%	67.2%
Precision	0.0%	12.9%	12.6%	16.4%
Recall	0.0%	57.2%	59.1%	57.9%

(Linear Support Vector Machine)

Naïve Bayes	Unbalanced Traning set	Subsampling
Area under ROC	0.345	0.345
Accuracy	90.3%	62.8%
Precision	0.0%	12.9%
Recall	0.0%	49.1%

(Naïve Bayes)

Judging from Area under ROC curve, performance of Logistic regression is about 0.7 level, Gradient-boosted tree is the best model among the four, its AUC value is about 0.8.

(f) Cross Validation

Since we notice from the result in question (e), subsampling and oversampling produce near identical results, therefore we only use subsampling training set for cross validation, since subsampling training set is about 10% of the size of oversampling training set, so cross validation with subsampling with greatly save model fitting time. The cross-validation results are included in above table for question (e).

Results show cross-validation can improve performance of classification model, but because of our training set has large number of observations, the effect of cross-validation is not significant. Cross validation with Logistic regression improved 0.03 AUC, and 0.05 AUC with Linear SVM.

Best models with hyperparameters

Cross Validation with Logistic regression

regParam = 0.1, elasticNetParam = 0.00, maxIter = 20

Cross Validation with Gradient-Boost Tree model

stepsize = 0.2, maxDepth = 5

Cross Validation with Linear SVM

regParam = 1, maxIter = 20

(g) Binary classification results

Logistic regression classifier achieved around 0.7 Area under ROC curve, it is a good model overall, with unbalanced training set, its accuracy is 90.2% which is same as the "do-nothing-classification", i.e. put all predictions to "Other" class, so Recall value is 0. With balanced training set, either with subsampling or oversampling set, Area under ROC value is improved a little to 0.701, but Recall value improved significantly to 63.5%. Cross validation with the best hyperparameters can further improve AUC to 0.732 and Recall to 67.6%.

With Logistic regression model, we can adjust prediction threshold, default threshold is 0.5, for the subsampling training model, if we lower down threshold to 0.4, then we get Recall value increase from 63.5% to 89.2%, but this is a trade off with Precision and Accuracy, as Precision drops from 17.5% to 11.5%, accuracy drops from 67.5% to 31.9%.

Gradient-boosted tree classification is the best performance model, its Area under ROC is at 0.8 level, like in Logistic regression, balanced class in training set also helps improve recall value, from 12.5% to 70.9%, the improvement with cross validation is little to the model. The major drawback of gradient boost tree model is training time, as it is much slower model to fit.

We also try out Random forest model, Random forest model is susceptible to unbalanced class, Random forest model on unbalanced training set yield AUC at 0.5 which poor performance, however, with balanced training data, Random forest produces 0.765 AUC and up to 68.3% Recall. Recall for oversampling is 6% higher than subsampling.

Linear Support Vector Machine is susceptible to unbalanced class as well, its initial AUC is only 0.319. With balanced training set, its performance improved, but not as good as Logistic regression. It also turns out Naïve Bayes does not fit this training set well, for neither balanced nor unbalanced data.

Q3 Multi-class classification

(a) One vs. one and one vs. all

For Multiclass classification, there are two approaches, one is one vs. all, the other is one. vs. one.

Assume that a dataset has n classes, for one vs. all classification, it builds a multiway classifier with a series of binary classifiers, it uses n binary classifiers, to train a classifier for class k, all observations belong to class is treated positive and the rest as negative. One vs. one approach have to train a separate classifier for each pair of classes. This leads to n(n-1)/2 classifiers, this is less sensitive to unbalanced class but takes much more computational resources. Generally, one vs. one approach should have better performance than one vs. all, but again that depends on how much computation power you can allocate.

In our data set, there are 21 unbalanced classes, with the largest class of Pop_Rock with 237,649 counts, while the smallest class has only 463 counts. Even with one vs. one approach, we still face serious class imbalanced problems.

```
+--------------+------+
|        genres| count|
+--------------+------+
|      Pop_Rock|237649|
|    Electronic| 40666|
|           Rap| 20899|
|          Jazz| 17775|
|         Latin| 17504|
|           RnB| 14314|
| International| 14194|
|       Country| 11691|
|     Religious|  8780|
|        Reggae|  6931|
|         Blues|  6801|
|         Vocal|  6182|
|          Folk|  5789|
|       New Age|  4000|
| Comedy_Spoken|  2067|
|        Stage |  1613|
|Easy_Listening|  1535|
|   Avant_Garde|  1012|
|     Classical|   555|
|      Children|   463|
+--------------+------+
```

Our models used in Q2, among the good models, Logistic regression and Random forest can perform multiclass classification, but Gradient-boosted tree can only work on binary classification. In Q2 (g), we learnt that AOC value for unbalanced training set on Random forest is only 0.5. So, we decide to fit logistic regression for this multiclass classification.

(b) Index classes

Use StringIndexer to convert genre category into integer index, for classification.

(c) Classification

Fit logistic regression on multiclassification, we get accuracy of 0.56, F1 value is 0.410, with cross-validation F1 improves to 0.4235. F1 is considered as efficient metric for multi-class classification, because F1 is overall average of Precession and Recall, as for multi-class evaluation, we cannot simply look at one or two classes.

Because training set is extremely imbalanced, so the classifier put majority of predictions into the biggest class, i.e.Pop_Rock. Recall value for Pop_Rock class is 0.997, Electronic class is 0.022, for the rest of classes Recall value is close to Zero. The class unbalance problem can be improved with subsampling and oversampling, one idea is to down sample big classes such as Pop_Rock and Electronic to ten thousand level, then over sample small classes observations to similar scale. F1 score is a better metric for multiclassification, average F1 score is the unweighted average of the F1-score over all the classes in the multiclass case.

Song Recommendations

Q1, Data preparation

(a) User song play dataset

Triplets dataset after clean mismatches.

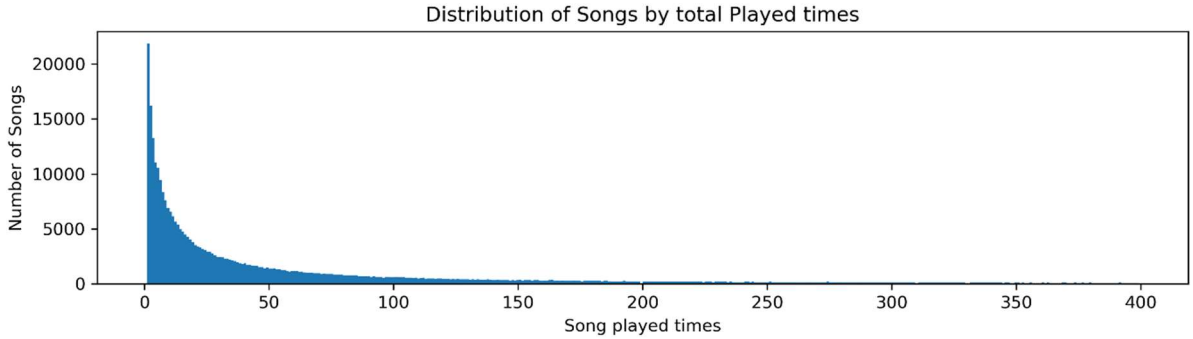
Unique songs: 378,310

Unique Users: 1,019,318

(b) Most active user

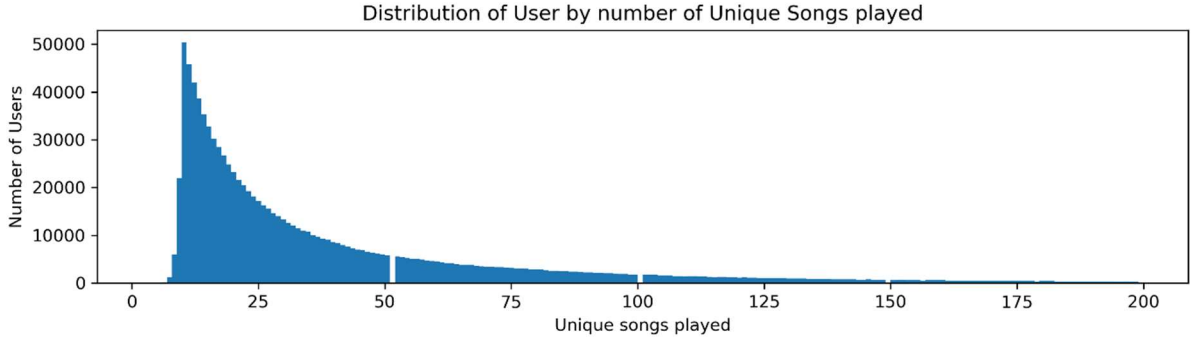
The most active user played 13,074 times, but the user had played the same song for many times. The user played most number of unique songs played 4,316 different songs, in terms of 378,310 unique songs, its only 1.14%.

(c) Distributions



(Distribution of Songs by times played, show only songs played less than 400 times.)

There are more than 378 thousand songs in dataset, very few of them are deemed as extremely popular songs, the most popular song was played more than 70,000 times, but majority of songs were played for less than 200 times.



(Distribution of users by number of unique songs played, show only users played less than 200 songs)

Two distributions show both are heavily left shewed and long right tail, very little percentage of songs were played for more than 200 times, and very small fraction of users have played more than 200 different songs. Most of users played between 10 to 100 unique songs.

(d) Collaborative filtering

Through distribution plot in question (c), we find large portion of songs were played less than 50 times, and large portion of users played less than 50 songs.

For users, 1% of users played less than 9 songs, 5% users played 10 songs, and 10% of users played less than 12 songs, 50% users played less than 26 songs.

For songs, 50% songs were played less than 32 times, like we can see from the first plot in question (c), the popular songs are played hundreds or even thousands of times. For 70% songs were played less than 94 times. The user who played the greatest number of songs only played 1.14% of total song pool, so even if we drop the 70% of songs, we still have large number of songs to recommend for uses.

So, we take n = 94, m = 26, that is to drop 50% of less active users and 70% of less played songs. By removing less active users and less popular songs, we also have a smaller dataset to train on.

(e) Train test split

It is required to have every user who is in test set has some user-song plays in the training set. This is to avoid cold-start problem, collaborative filtering recommendation is based on user actions, that can be either explicit or implicit, the more the user actions the model has, the easier it is to make recommendations. So, if the model has no information about the user, model couldn't make recommendations. If cold-start problem occurred, the model is optioned to drop the observations.

In order to avoid cold-start problem, we applied stratified sampling method for every single user that have listened to 5 and more songs in the cleaned user-song play dataset. Manually filter out the users who played less than 5 songs in cleaned user-song dataset, there are 54 users in the list, so we don't lose much of observations. Then use stratified sampling for each user_id, randomly take 21% of song plays data for each user. then the rest 79% for training dataset. Since we dropped users played less than 5 songs already, then take 21% for testing, for any users we have, at less 1 user-song play observation is selected in testing set, and the rest go to training set.

Q2 Train and test ALS model

(a) Fit model

Fit ALS model, pass training set and testing set through StringIndex pipe line, get numeric index for user_id and song_id, then fit ALS model with implicitPrefs = True, this is important, as we train the model on user-song plays, not the user rating on songs.

Use the fitted ALS model to predict testing set, we get root mean squared error of 6.53 for deviation between predicted user plays and actual plays.

(b) Recommendation review

Pick users in test set, for the convenience of analysis, random take 10 users played 30 more songs. Generate 10 recommendations with 'alsModel.recommendForUserSubset' function for each user. The 10 recommended songs are in a list, expand the list then use rdd flatMap to extract the recommended song_id. We now have a data frame with user_index and recommendation song_index. Join the data frame with our training set, to convert user_index, song_index to user_id and song_id, then join song_id with song_id in metadata to get song information.

For user index 38795, we get following recommendations.

```
+--------------+------------------+-----------------+-----+--------------------+
|   artist_name|   song_hotttnesss|            title|genre|             release|
+--------------+------------------+-----------------+-----+--------------------+
|  3 Doors Down|0.9151810636939853|       Kryptonite| null|           Total 90s|
| Guns N' Roses|              null|    Paradise City| null|       Greatest Hits|
|Counting Crows|               1.0|        Mr. Jones| null|Films About Ghost...|
```

```
|     Asia 2001|0.2998774882739778|          Epilogue| null|             Amnesia|
|      Bon Jovi|               1.0|Livin' On A Prayer| null|          Cross Road|
|Lynyrd Skynyrd|              null|Sweet home Alabama| null|           To Die For|
|     Metallica|0.9655075340177467| Master Of Puppets| null|   Master Of Puppets|
|     Radiohead|              null|  Creep (Explicit)| null|         Pablo Honey|
|        Eagles|0.9406301589603857|  Hotel California| null|Hotel California ...|
|    Nickelback|              null| How You Remind Me| null|FETENHITS - New P...|
+-------------+------------------+------------------+-----+--------------------+
```

Get the songs the user have listened in test data, join with meta data, we get a list of songs user have played.

```
+-----+--------------------+------------------+--------------------+--------------------+
|plays|         artist_name|    song_hotttnesss|               title|             release|
+-----+--------------------+------------------+--------------------+--------------------+
|   20|      C.V. Jørgensen|              null|Pak Dit Grej (199...|         De 2 Første|
|   10|        Billy Squier|              null|          The Stroke|Let's Go To Priso...|
|   10|        Billy Squier| 0.7646204000035967|         In The Dark|        Don't Say No|
|    8|          Nickelback|  0.936575245043614|Never Gonna Be Al...|          Dark Horse|
|    5|        Damn Yankees| 0.8158882089961863|High Enough (Albu...|Rhino Hi-Five: Da...|
|    4|       Robert Palmer|0.26586104921065007|Your Mother Shoul...|The Essential Sel...|
|    4|           Bon Jovi|               1.0|You Give Love A B...|Slippery When Wet...|
|    4|         Miley Cyrus|               1.0| Party In The U.S.A.| Party In The U.S.A.|
|    4|           Bon Jovi| 0.8709776265271216|        It's My Life|               Crush|
|    4|           Metallica|               1.0|       Enter Sandman|           Metallica|
|    4|    Billy Currington| 0.7268112207729489|That's How Countr...|Little Bit Of Eve...|
|    4|           Colt Ford| 0.4368224699038839|   Huntin' The World|Country Is As Cou...|
|    3|          Nickelback| 0.7830859657398406|         Never Again|      Silver Side Up|
|    3|          Toby Keith| 0.5944639098781762|      She's A Hottie|Toby Keith 35 Big...|
|    3|           Colt Ford|0.46417399167415674|             Like Me|Ride Through The ...|
|    3|           Colt Ford| 0.3972653548644609|             Twisted|Ride Through The ...|
|    2|      Britney Spears|               1.0|               Toxic|The Singles Colle...|
|    2|           Liz Phair|  0.519832962528063|Why Can't I? (Exp...|           Liz Phair|
|    2|        3 Doors Down| 0.8175613303456302|    Let Me Be Myself|        3 Doors Down|
|    2|            Alabama| 0.6448037051731974|My Home's In Alabama|        Alabama Live|
|    2|           Colt Ford| 0.4191634755594802|            Buck 'Em|Country Is As Cou...|
|    2|                Kix|              null|Don't Close Your ...|                 Kix|
|    1|           Kid Rock| 0.6072560951352716|So Hott (Amended ...|   Rock N Roll Jesus|
|    1|       Joe Christmas| 0.2998774882739778|       Bedroom Suite|Upstairs Overlooking|
|    1|         Miley Cyrus| 0.7865534054973415|Kicking And Screa...|The Time Of Our L...|
|    1|Usher Featuring L...| 0.7813811676365547|               Yeah!|         Confessions|
|    1|     The Wallflowers| 0.8696612354539689|       One Headlight|Bringing Down The...|
|    1|             Journey| 0.5456765460250699|If He Should Brea...|       Trial By Fire|
|    1|         Linkin Park| 0.8722290226088538|Crawling (Album V...|       Hybrid Theory|
+-----+--------------------+------------------+--------------------+--------------------+
```

Compare two tables, we can conclude that:

Recommended songs have high song_hotttnesss value, it's hard to blame it when the system recommends popular songs, most of songs are new to the user, but a few of songs the user had played before.

It recommends some songs by the artists who are familiar to the user, such as Bon Jovi and 3 Doors Down, the user have played those artistes before.

The user often plays Pop_rock music, and the recommendations seems are all in the category.

Look in to the user played songs in training set, we find the user have records of listening to Bon Jovi as well. We can potentially compare more users to check songs recommendations and songs users have played. So far, we can consider the model does make reasonable recommendations to users.

(c) Recommendation metrics

Fit model on test set, then calculate following metrics:

Precision@5: 0.889

NDCG@10: 0.917

Mean Average Precision (MAP) : 0.769

Precision at k is a measure of how many of the first k recommended documents are in the set of true relevant documents averaged across all users. In this metric, the order of the recommendations is not taken into account.
NDCG at k is a measure of how many of the first k recommended documents are in the set of true relevant documents averaged across all users. In contrast to precision at k, this metric takes into account the order of the recommendations (documents are assumed to be in order of decreasing relevance)
MAP is a measure of how many of the recommended documents are in the set of true relevant documents, where the order of the recommendations is taken into account (i.e. penalty for highly relevant documents is higher).

All three metrics use recommended songs to compare with songs have played by the user in test set, Precision takes in top 5 recommended songs without consider the order of recommendations, NDCG take top 10 recommended songs and consider the importance of ordering. Mean Average Precision is based on Precision, take into account the order, but not limited by number of recommendations.

Those metrics use recommendations based on historical play data, the results look quite good, but we need to rethink about this recommendation, as most of them are the songs the user have played before, if the user is kind of “old school” person, then he or she may enjoy the recommendation, but what if some users don’t like to hear a few songs over and over again, what if users want to explorer new songs.

If we use recommendation from part (b), those recommendations are not based on user play historical data, the above three metrics value will be much lower than current ones. The recommendations are more of news songs to users.

An alternative method for comparing two recommendation systems, can be based on song clusters, group songs by audio features into a number of genres, or even more smaller clusters, then evaluate if recommended songs are in the same cluster with the songs users have interested in. If the recommendations fit in the cluster, then we consider it is right recommendation.

If we can measure future user-song plays based on our recommendations, we can effectively know the user’s preference by checking times of played, whether the user skip the song during play, whether the user mark the song as favourite, whether the user replay the song once it finished. Those are important real-world feedbacks from users.

Q3, Options and other models.

(a) Limitations of ALS

One limitation is the cold-start problem, the model can not make recommendations if the model don’t have song play data for the user. Because ALS model works on clustering similar users together based on the songs users played, so without user song play data, the model can not cluster the user, then there will be no recommendations.

Another limitation is about popularity bias, since the model recommend based on similar user groups, then system trend to recommend songs are popular in the group, that makes popular songs get even more popular, it will end up with a few numbers of popular songs being play hundreds of thousands of times, but rest of songs ignored.

Situation like when a user who listen to songs in very specialized domains, or songs in some minority languages, the model will not produce high quality recommendations because of lacking similar user profile.

(b) Potential solutions.

For the cold-start problems, we can use a model like Content-based filtering, for new users, ask users to select interested genres, artists and songs, then the model can recommend based on the key words.

To take care of the popularity bias and specialize user good problem, we can use Item-Item Collaborative Filtering, that is to cluster songs instead of users. To cluster songs, we have many approaches, such as plays, audio features, genres, artists' similarities etc. Then we can recommend user songs based on user's preference.

(c) Other considerations.

For a real-world music streaming service such as Spotify, a few things should be specially considered,

1. First recommendation is very important, as music streaming is not like movie recommendation, users don't necessary actively choose from top 10, but stream music automatically.

2. Need consider the balance of new songs and songs users have played.

3. Negative feedback such as skip songs should be considered seriously.

4. Consider the popularity bias problem, should the system offer users more variety of songs to try out.