Assignment 2

DATA420

Nicole Dunn | 75958138 | 20/10/2022

Data Processing

QUESTION ONE

We can see the directory of the dataset in Figure 1. Overall, the Million Song Dataset (MSD) is 12.9GB which comprises of 4 parts, Audio, Genre, Main and Taste Profile. These parts are of sizes 12.3GB, 30.1MB, 174.4MB and 490.4MB respectively, this makes it clear that Audio takes up the majority of the data within the dataset.

There were 13 CSV files within both the attributes and features folders under Audio. The structure and data type of the CSV files under attributes were the same and the information contained within the files consisted of a column name and data type. The CSV files within features were split into 8 partitions and compressed. Viewing the first 10 lines shows that files show the specific features of the songs in a numerical format.

Statistics contained one csv file that has been compressed. The file contains all the information about the songs within the data set such as the track ID, title and duration of the song.

```
.
└── /data/msd/
    ├── audio/
    │   ├── attributes/
    │   │   ├── msd-jmir-area-of-moments-all-v1.0.attributes.csv
    │   │   ├── msd-jmir-lpc-all-v1.0.attributes.csv
    │   │   ├── msd-jmir-methods-of-moments-all-v1.0.attributes.csv
    │   │   ├── msd-jmir-mfcc-all-v1.0.attributes.csv
    │   │   ├── msd-jmir-spectral-all-all-v1.0.attributes.csv
    │   │   ├── msd-jmir-spectral-derivatives-all-all-v1.0.attributes.csv
    │   │   ├── msd-marsyas-timbral-v1.0.attributes.csv
    │   │   ├── msd-mvd-v1.0.attributes.csv
    │   │   ├── msd-rh-v1.0.attributes.csv
    │   │   ├── msd-rp-v1.0.attributes.csv
    │   │   ├── msd-ssd-v1.0.attributes.csv
    │   │   ├── msd-trh-v1.0.attributes.csv
    │   │   └── msd-tssd-v1.0.attributes.csv
    │   ├── features/
    │   │   ├── msd-jmir-area-of-moments-all-v1.0.csv
    │   │   ├── msd-jmir-lpc-all-v1.0.csv
    │   │   ├── msd-jmir-methods-of-moments-all-v1.0.csv
    │   │   ├── msd-jmir-mfcc-all-v1.0.csv
    │   │   ├── msd-jmir-spectral-all-all-v1.0.csv
    │   │   ├── msd-jmir-spectral-derivatives-all-all-v1.0.csv
    │   │   ├── msd-marsyas-timbral-v1.0.csv
    │   │   ├── msd-mvd-v1.0.csv
    │   │   ├── msd-rh-v1.0.csv
    │   │   ├── msd-rp-v1.0.csv
    │   │   ├── msd-ssd-v1.0.csv
    │   │   ├── msd-trh-v1.0.csv
    │   │   └── msd-tssd-v1.0.csv
    │   └── 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
```

Figure 1: MSD Directory

Genre contained 3 TSV files, these files contained two columns, one listed the track ID which correlates to the one in Statistics, and the other column listed the genre of the song.

Under Main, a second file called Summary contains two compressed CSV files. The first one called Analysis contains the basic information regarding the interpretation of the song and the second file called Metadata contains what is named after.

Finally, the last file Taste Profile contains matches and mismatches TXT files and a TSV file. The TSV file is partitioned in the same was as the Features folder with 8 partitions for the compressed files. The expected level of parallelism is 8.

The repartition method is a way to increase or decrease the partitions. It evenly re-distributes the data from all the original partitions which leads to a full shuffle which can

be a very expensive operation. For this operation the repartition method will not be useful as the dataset only has 8 partitions which means it can be parallel processed on 8 cores within a cluster. When a cluster has more cores, the repartition method would be better suited as it can increase the level of parallelism.

To count the total number of rows in the datasets, the wc -l command was used which returned the below figure. As we can see there is some correlation between the total number of unique songs and the total number of rows in the datasets with the majority of them sitting below the one million mark.

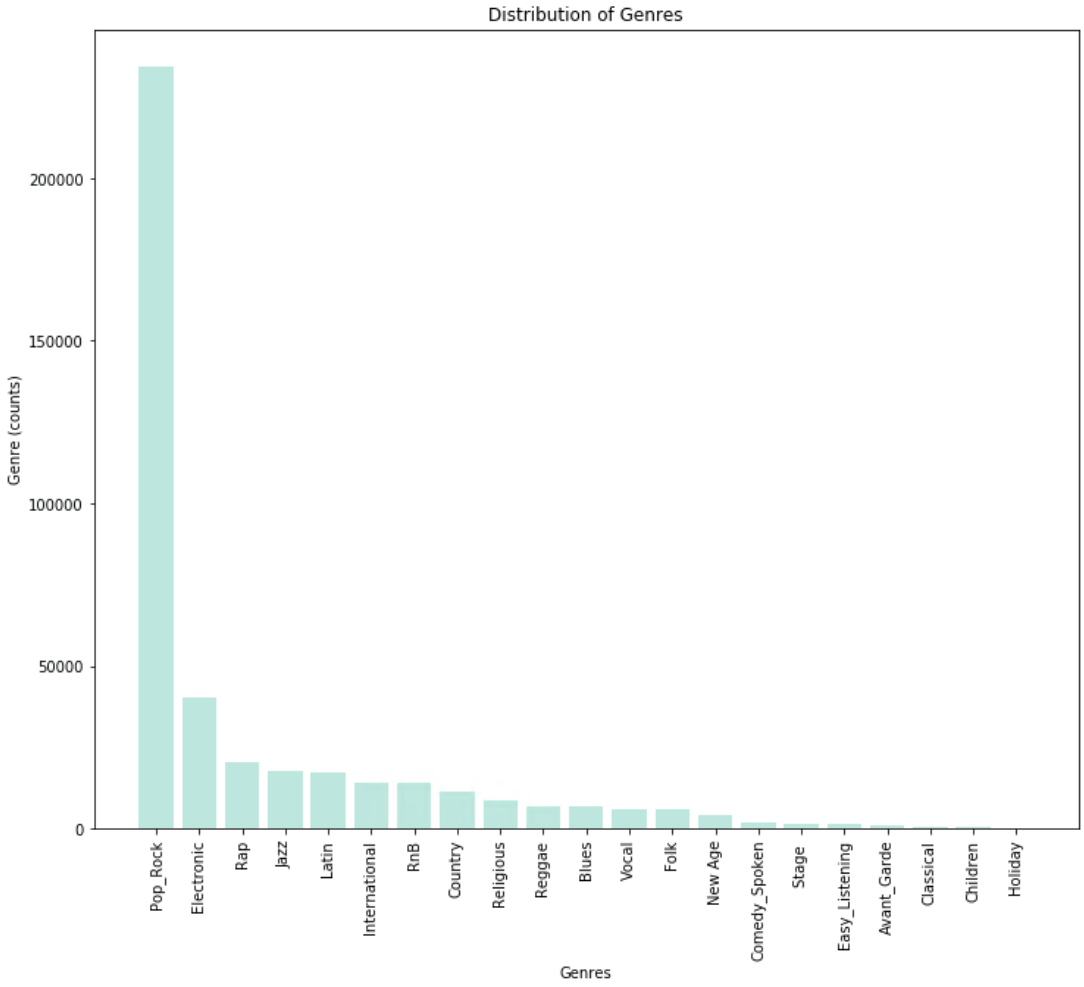
QUESTION TWO

To accomplish the task of removing the files with mismatched data, a schema was created to combine the two mismatched text files. Using the Triplets folder, a table was created to connect the mismatched songs to their song and user ID's. The total number of matches with mismatches removed was 45795111.

Audio Similarity

QUESTION ONE

The dataset used for this part of the assignment was the msd-jmir-methods-of-moments-all-v1.0.csv. There are 11 columns with the track ID being the only string type column. The other 10 columns contain numeric information. There were a few features with strong positive correlation such as 0.985 between Average 4 and Average 5, 0.903 between Average 2 and Average 3, 0.942 between Standard Deviation 5 and Standard Deviation 4 and 0.858 between Standard Deviation 3 and Standard Deviation 2. These positive correlations show that as these features increase, the one with the strong positive correlation will increase as well. There were no strong negative points of correlation between the features.



Visualising the Genres displayed that Pop_Rock had the largest presence amongst the dataset.

Merging genres into the audio features dataset using Track ID as an identifier provided a count of 413293.

QUESTION TWO

The three classification algorithms chosen were Naïve Bayes, Logistic Regression and Linear Support Vector Classification. These were chosen as methods that had quick training speeds and simple hyperparameter tuning. Naïve Bayes is one of the simplest classification methods so establishing a simple baseline was my main task, it also has very cheap computations which when using in conjunction with a remote connection was an important part of the consideration. Using a Gaussian method should be the most appropriate for this dataset.

Logistic Regression as chosen as it is a well-known method that has very understandable hyperparameters and is typically a relatively accurate model. It is also a method I feel very

comfortable using so the level of understanding is much higher than other methods. In this model I used lambda = 0.8 and alpha = 0.1.

Linear Support Vector Classification is a supervised classification method that is effective in high dimensional spaces and it is very memory efficient due to it using support vectors. It is also a method that is able to do both binary and multi-class classification. For this model regParam = 0.1.

Converting the genre data into a binary column representing whether the genre is electronic or not displayed a class balance as shown below. As expected, the dataset displays around 10% of the genres being Electronic – we can assume this would be low due to the previous graph that displayed that Pop_Rock had the highest representation – followed by Electronic.

```
+----------+------+
|Electronic| count|
+----------+------+
|         1| 40028|
|         0|373265|
+----------+------+
```

Figure 2: Electronic Class Balance

When splitting the dataset, it became immediately obvious that there was a severe imbalance. In order to preserve quality I chose to select a subsample instead of an oversample of the data that made up the majority of the larger side of the imbalance, this is because an oversample would be creating a lot of "fake" data compared to reducing the amount of real data. To select a stratified random sample, I used the lit() function to establish fractions to use with the .sampleBy() function to get a 70/30 train/test split. To get a random sample of the majority data I used the .sample() function with a ratio of 0.11 to get a similar amount of data that the minority data has.

Performance Metrics:

```
Test accuracy = 0.7218725856296678
Test recall = 0.7471685335186818
Test precision = 0.9312926227869471
```

Figure 3: Niave Bayes Model Metrics

```
Test accuracy = 0.8520473860417204
Test recall = 0.9186337672984557
Test precision = 0.917636167807518
```

Figure 4: Logistic Regression Model Metric

```
Test accuracy = 0.7005536955961885
Test recall = 0.7150265992995963
Test precision = 0.9388316231616142
```

Figure 5: LinearSVC Model Metrics

QUESTION THREE

The hyperparameters used for Logistic Regression are the alpha and lambda values. When the lambda value is zero there is no regularization and alpha is ignored, when lambda is greater than zero and alpha is greater than zero and less than 1 this establishes the elastic net penalty. When alpha is closer to zero the model prefers ridge regression whereas when it is closer to one is prefers LASSO. Using cross-validation, the model achieved worse performance metrics to the previous version, this seems to indicate that the data isn't very fit for this task.

```
Test accuracy = 0.6747720364741642
Test recall = 0.6802579515715574
Test precision = 0.9437312582092141
```

Figure 6: Logistic Regression Cross-Validated Model

QUESTION FOUR

As mentioned previously, Linear Support Vector Classification is capable of multiclass classification. It uses a method call One-to-One, which turns the multiclass problem into multiple binary classification problems. Another method it can use is the One-to-Rest approach, whereas it creates a binary classifier per each class. Including multiple genres has reduced the performance accuracy and precision and increased the recall. This could be because the balance is called into question with a multiclass problem. There will be a lot of values that are underrepresented and likewise for values that are then overrepresented so it is expected that there will be a lot more inaccurate predictions for this multiclass model.

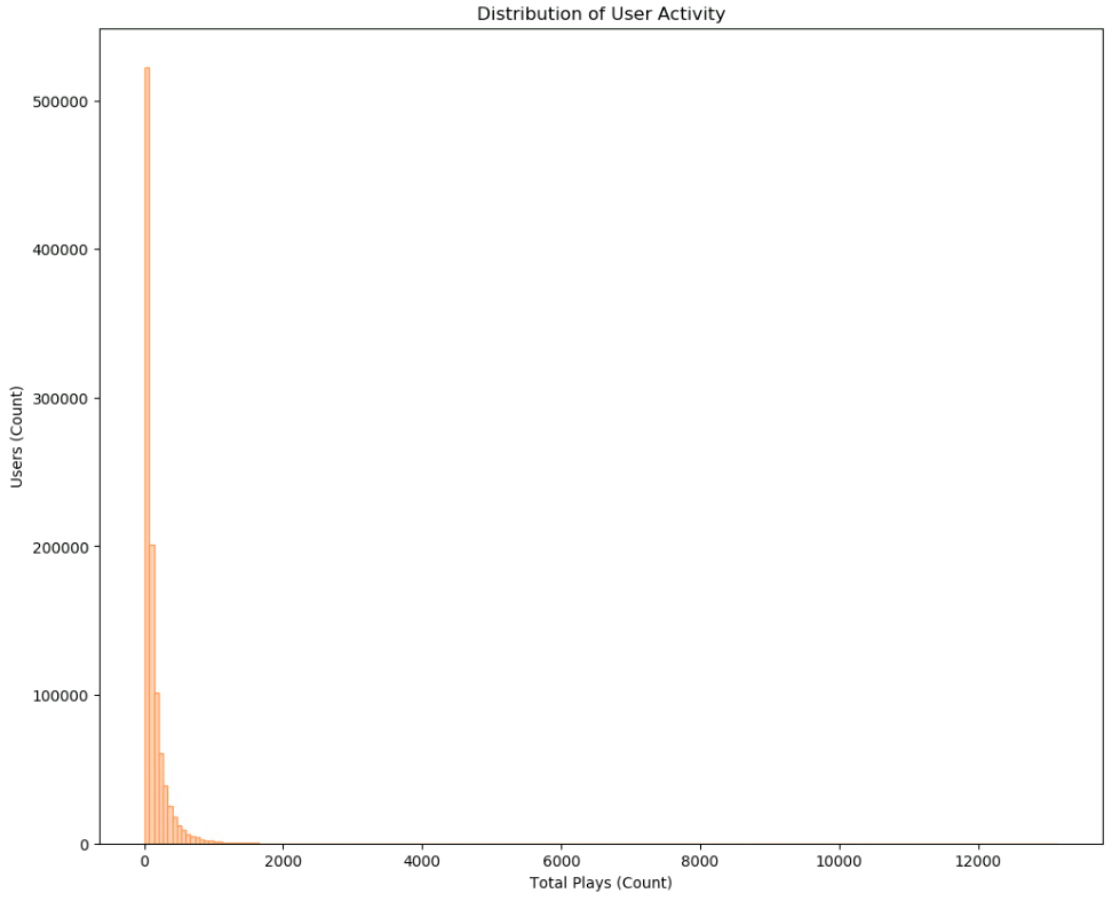
```
Test accuracy = 0.5628824587005071
Test recall = 0.9969280009144555
Test precision = 0.565587458050291
```

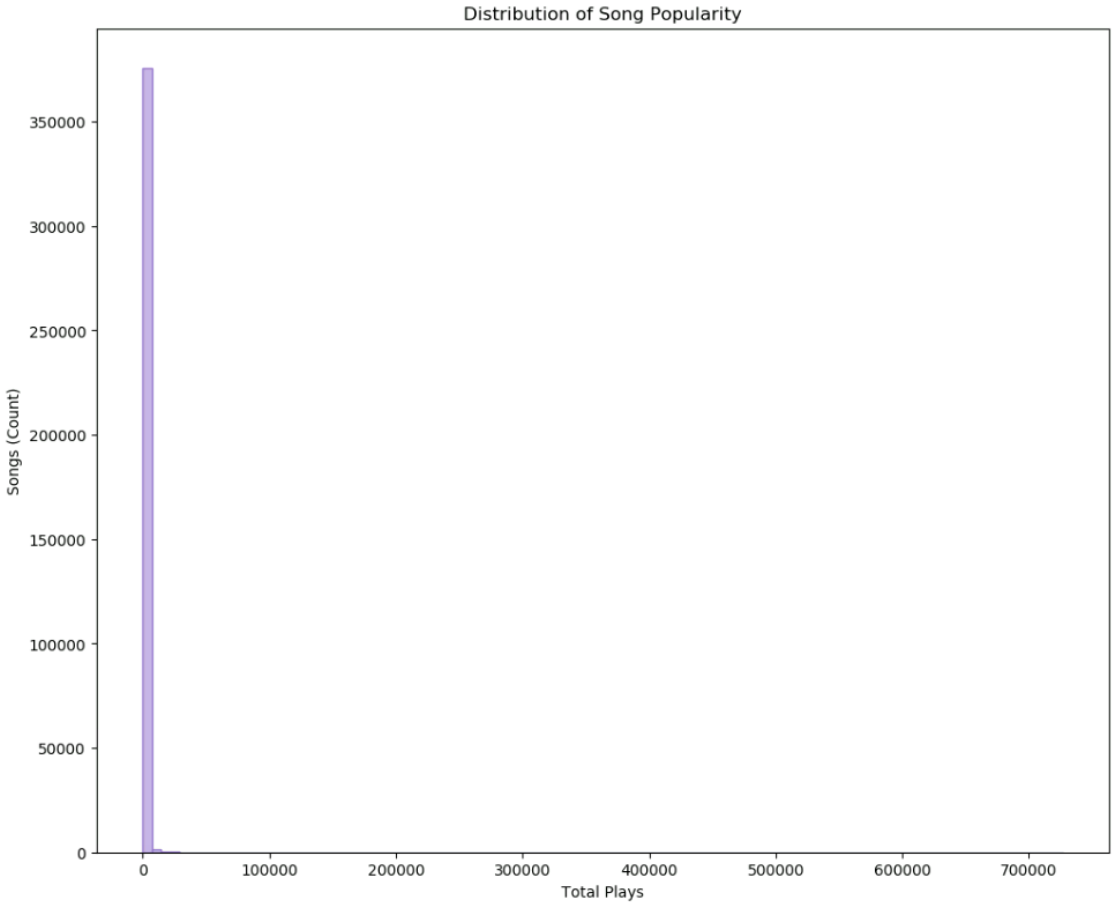
Figure 7: Multiclass Classification using LinearSVC (OnevRest method)

Song Recommendations

QUESTION ONE

There are 378,310 unique songs and 1,019,318 unique users in the data set. The most active user has listened to 13,132 streams which works out to be 195 unique songs which is 0.05% of the total unique songs in the data set.





As we can see from the above graphs, the distribution of song popularity (Total Plays) and user activity both show a very strong positive skew. With the majority of the plays sitting within the first 200 total plays.

Removing the bulk of these underplayed songs will help with future analysis so a decision was made to remove the lower quantile of active users and songs. This reduced the number of unique users to 755,896 and the number of unique songs to 279,978.

When creating the test/train split, it is essential to ensure the user is in the train data set if they appear in the test as otherwise it is not possible to create a calculation matrix to generate recommendations for the user.

QUESTION TWO

The ALS method was used to accomplish the tasks set in this question. The effectiveness of the model for the randomly selected user recommendations using the collaborative filtering method was an average-to-good performance. Due to limiting the number of iterations due to computer specifications I wasn't able to get the best results but with more time and resources I think it would be able to perform very well.

From this the following metrics were observed when applying the method to the rest of the test set of data were as follows:

Precision @ 10: 0.7560100934699033

NDCG @ 10: 0.8208473044410545

MAP: 0.2512344595147811

Precision@k corresponds to the number of relevant results within the top 'k' results retrieved from the dataset. One drawback is that it doesn't measure the positions of the relevant results among this top 'k', 'k' can also be considered a minimum number of results because if there are results less than 'k', the result will always be less than 1. It is a very easy to use metric though and can even be done by hand.

NDGC is a way of measuring how things are ordered. In comparison to other searching methods such as google, a user wants the most relevant search result to be at the top of the list and so NDGC measures how far away the predictions are from the perfect relevance. This is useful as it allows a way to measure relevancy accuracy which wasn't available with the prior metric. It is obviously not a useful metric if relevance is not a requirement for a model.

MAP is used for a dataset is the mean of the average precision scores for each query involved in the calculation, it also incorporates the precision and recall trade off.This does mean that if you are trying to improve both metrics with MAP it is impossible to do so as one cannot increase with the other and this may waste time and resources.

An alternative method of testing models in the real world would be to use an A/B test, which is a randomized test with the two models, and letting real users attempt to use the models and gather feedback. This would be the most accurate way to measure if a model is working appropriately.

Other useful metrics to establish would be the recall, RMSE, and F1 metrics. These all show different measures of accuracy depending on what we want our model to perform as.