DATA420-18S2 (C)

Assignment 2

The Million Song Dataset (MSD)

Due on Friday, October 19 by 5:00 PM, to be submitted as a zip file using LEARN.

If you have any questions please email me (james.williams@canterbury.ac.nz).

LEARN

https://learn.canterbury.ac.nz/mod/assign/view.php?id=734033

Instructions

- You are encouraged to work together to understand and solve each of the tasks, but you must submit your own work.
- You have the full resources of the internet at your disposal.
- All write up, code, and outputs should be submitted in a single zip file via LEARN.
- Any assignments submitted after the deadline will receive a 50% penalty.
- All data under `hdfs:///data` is read only. Please use your own home directory to store your outputs (e.g. `hdfs:///user/abc123/outputs`).
- Please be mindful that you are sharing cluster resources. Keep an eye on Spark using the web user interface (madslinux01:8080), and don't leave a shell running for longer than necessary. Feel free to change the number of executors, cores, and memory that you are using, but be prepared to scale down if others need to share those resources. If you need to ask someone to share their resources, email them using their user code (e.g. abc123@uclive.ac.nz). If they don't reply within a reasonable time and their shell has been running for more than 24 hours, you can kill their shell by going to the web user interface and selecting **kill**.

The Million Song Dataset (MSD)

In this assignment we will study a collection of datasets referred to as the Million Song Dataset (MSD), a project initiated by The Echo Nest and LabROSA. The Echo Nest was a research spin-off from the MIT Media Lab established with the goal of understanding the audio and textual content of recorded music, and was acquired by Spotify after 10 years for 50 million Euro.

- The Million Song Dataset (MSD)

The main dataset contains the song ID, the track ID, the artist ID, and 51 other fields, such as the year, title, artist tags, and various audio properties such as loudness, beat, tempo, and time signature. Note that track ID and song ID are not the same concept - the track ID corresponds to a particular recording of a song, and there may be multiple (almost identical) tracks for the same song. Tracks are the fundamental identifier, and are matched to songs. Songs are then matched to artists as well.

The Million Song Dataset also contains other datasets contributed by organisations and the community,

- SecondHandSongs (cover songs)
- musiXmatch dataset (song lyrics)
- Last.fm dataset (song-level tags and similarity)
- Taste Profile subset (user-song plays)
- thisismyjam-to-MSD mapping (user-song plays, imperfectly joined)
- tagtraum genre annotations (genre labels)
- Top MAGD dataset (more genre labels)

We will focus on the Taste Profile and Top MAGD datasets, but you are free to explore the other datasets on your own and as part of the challenges. There are many online resources and some publications exploring these datasets as well.

Taste Profile

The Taste Profile dataset contains real user-song play counts from undisclosed organisations. All songs have been matched to identifiers in the main million song dataset and can be joined with this dataset to retrieve additional song attributes. This is an implicit feedback dataset as users interact with songs by playing them but do not explicitly indicate a preference for the song.

The dataset has an issue with the matching between the Taste Profile tracks and the million song dataset tracks. Some tracks were matched to the wrong songs, as the user data needed to be matched to song metadata, not track metadata. Approximately 5,000 tracks are matched to the wrong songs and approximately 13,000 matches are not verified. This is described in their blog post in detail.

Audio Features (Vienna University of Technology)

The Music Information Retrieval research group at the Vienna University of Technology downloaded audio samples for 994,960 songs in the dataset which were available from an online content provider, most in the form of 30 or 60 second snippets. They used these snippets to extract a multitude of features to allow comparison between the songs and prediction of song attributes,

Rhythm Patterns
Statistical Spectrum Descriptors
Rhythm Histograms
Temporal Statistical Spectrum Descriptors
Temporal Rhythm Histograms
Modulation Frequency Variance

Marsyas
Timbral features

jMir
Spectral Centroid
Spectral Rolloff Point
Spectral Flux
Compactness
Spectral Variability
Root Mean Square
Zero Crossings
Fraction of Low Energy Windows
Low-level features derivatives
Method of Moments
Area of Moments
Linear Predictive Coding (LPC)
MFCC features

These features are described in detail on the million song dataset benchmarks downloads page and the audio feature extraction page, and the number of features is listed along with file names and sizes for the separate audio feature sets.

MSD AllMusic Genre Dataset (MAGD)

Many song annotations have been generated for the MSD by sources such as Last.fm, musiXmatch, and the Million Song Dataset Benchmarks by Schindler et al. The latter contains song level genre and style annotations derived from the AllMusic online music guide. We will use the MSD All Music Genre Dataset (MAGD) provided by the Music Information Retrieval research group at the Vienna University of Technology.

This dataset is included on the million song dataset benchmarks downloads page and class frequencies are provided on the MSD AllMusic Genre Dataset (MAGD) details page as well. For more information about the genres themselves have a look at the AllMusic genres page.

Data processing

The data for the assignment can be found under `hdfs:///data/msd`. The `main/summary` directory contains all the metadata for the main million song dataset but none of the audio analysis, similar artists, or tags (see getting the dataset).

The `tasteprofile` directory contains the user-song play counts from the Taste Profile dataset as well as logs identifying mismatches that were identified and matches that were manually accepted.

The `audio` directory contains audio features sets from the Music Information Retrieval research group at the Vienna University of Technology. The `audio/attributes` directory contains attributes names from the header of the ARFF, the `audio/features` directory contains the audio features themselves, and the `audio/statistics` directory contains additional track statistics.

Do not copy any of the data to your home directory.

Q1 Read through the documentation above and figure out how to read from each of the datasets. Make sure that you only read a subset of the larger datasets, so that you can develop your code efficiently without taking up too much of your time and the cluster resources.

(a) Give an overview of the structure of the datasets, including file formats, data types, and the expected level of parallelism that you can expect to achieve from HDFS.

(b) Look up the `repartition` method. Do you think this method will be useful?

(c) Count the number of rows in each of the datasets. How do the counts compare to the total number of unique songs?

Q2 Complete the following data preprocessing.

(a) Filter the Taste Profile dataset to remove the songs which were mismatched.

(b) Load the audio feature attribute names and types from the `audio/attributes` directory and use them to define schemas for the audio features themselves. Note that the attribute files and feature datasets share the same prefix and that the attribute types are named consistently. Think about how you can automate the creation of `StructType` by mapping attribute types to `pyspark.sql.types` objects.

Audio similarity

In this section you will explore using numerical representations of a song's audio waveform to predict its genre. If this is possible, it could enable an online music streaming service to offer a unique service to their customers. For example, it may be possible to compare songs based entirely on the way they sound, and to discover rare songs that are similar to popular songs even though they have no collaborative filtering relationship. This would enable users to have more precise control over variety, and to discover songs they would not have found any other way.

Q1 There are multiple audio feature datasets, with different levels of detail. Pick one of the small datasets to get started.

(a) The audio features are continuous values, obtained using methods such as digital signal processing and psycho-acoustic modeling.

Produce descriptive statistics for each feature column in the dataset you picked. Are any features strongly correlated?

(b) Load the MSD All Music Genre Dataset (MAGD).

Visualize the distribution of genres for the songs that were matched.

(c) Merge the genres dataset and the audio features dataset so that every song has a label.

Q2 First you will consider binary classification only.

(a) Research and choose three classification algorithms from the `spark.ml` library.

Justify your choice of algorithms, taking into account considerations such as explainability, interpretability, predictive accuracy, training speed, hyperparameter tuning, dimensionality, and issues with scaling.

Based on the descriptive statistics from Q1 part (a), decide what processing you should apply to the audio features before using them to train each model.

(b) Convert the genre column into a column representing if the song is "Electronic" or some other genre as a binary label.

What is the class balance of the binary label?

(c) Split the dataset into training and test sets. Note that you may need to take class balance into account using a sampling method such as stratification, subsampling, or oversampling. Justify your choice of sampling method.

(d) Train each of the three classification algorithms that you chose in part (a).

(e) Use the test set to compute the compute a range of performance metrics for each model, such as precision, accuracy, and recall.

(f) Use cross-validation to tune the hyperparameters for each model.

How has this changed your performance metrics?

(g) Comment on the relative performance of each model and of the classification algorithms overall, taking into account the performance metrics that you computed in parts (e) and (f).

How does the class balance affect the performance of the algorithms?

Q3 Next you will extend your work above to predict across all genres .

(a) Look up and explain the difference between the **one vs. one** and the **one vs. all** multiclass classification strategies, and describe how you would implement each of these strategies in Spark. Do you expect either one of these strategies to perform better than the other?

How do the classification algorithms that you chose earlier relate to these strategies?

(b) Convert the genre column into an integer index that represents the genre consistently. Find a way that requires the least amount of work by hand.

(c) Repeat Q2 parts (c) - (f) using one of the classification algorithms that you used for binary classification, choosing performance metrics that are relevant for multiclass classification instead. Make sure you take into account the class balance in your comments.

How has the performance of your model been affected by the inclusion of multiple genres?

Song recommendations

In this section you will use the Taste Profile dataset to develop a song recommendation system based on collaborative filtering.

Collaborative filtering describes algorithms that generate songs recommendations for specific users based on the combined user-song play information from all users. These song recommendations are generated by embedding users and songs as numerical vectors in the same vector space, and selecting songs that are similar to the user based on cosine similarity.

Q1 First it will be helpful to know more about the properties of the dataset before you being training the collaborative filtering model.

(a) How many unique songs are there in the dataset? How many unique users?

(b) How many different songs has the most active user played?

What is this as a percentage of the total number of unique songs in the dataset?

(c) Visualize the distribution of song popularity and the distribution of user activity.

What is the shape of these distributions?

(d) Collaborative filtering determines similar users and songs based on their combined play history. Songs which have been played only a few times and users who have only listened to a few songs will not contribute much information to the overall dataset and are unlikely to be recommended.

Create a clean dataset of user-song plays by removing songs which have been played less than N times and users who have listened to fewer than M songs in total. Choose sensible values for N and M and justify your choices, taking into account (a) and (b).

(e) Split the user-song plays into training and test sets. Make sure that the test set contains at least 20% of the plays in total.

Note that due to the nature of the collaborative filtering model, you must ensure that every user in the test set has some user-song plays in the training set as well. Explain why this is required and how you have done this while keeping the selection as random as possible.

Q2 Next you will train the collaborative filtering model.

(a) Use the `spark.ml` library to train an implicit matrix factorization model using Alternating Least Squares (ALS).

(b) Select a few of the users from the test set by hand and use the model to generate some recommendations. Compare these recommendations to the songs the user has actually played. Comment on the effectiveness of the collaborative filtering model.

(c) Use the test set of user-song plays and recommendations from the collaborative filtering model to compute the following metrics

- Precision @ 5
- NDCG @ 10
- Mean Average Precision (MAP)

Look up these metrics and explain why they are useful in evaluating the collaborate filtering model. Explore the limitations of these metrics in evaluating a recommendation system in general. Suggest an alternative method for comparing two recommendation systems in the real world.

Assuming that you could measure future user-song plays based on your recommendations, what other metrics could be useful?

Q3 The method used to train the collaborative filtering model above is one of many.

(a) Explain the limitations of using this model to generate recommendations for a streaming music service such as Spotify. In what situations would this model be unable to produce recommendations for a specific user and why?

In what situations would this model produce low quality recommendations and why?

(b) Suggest two other recommendation systems that could be used in each of these scenarios respectively.

(c) Based on the song metadata provided in the million song dataset summary, is there any other business logic that could be applied to the recommendations to improve the real world performance of the system?