Million Song Database Analysis

Justin Eugenio, Eric Pham, Jimmy Sisenglath

Data Warehousing and Data Mining, California State University, Sacramento
6000 J Street
Sacramento, CA 95819
[1]jimmysisenglath@gmail.com
[2]erichungpham@gmail.com
[3]justinbrianeugenio@gmail.com

***Abstract*— Our final project for Data Warehousing and Data Mining involves analyzing Spotify's Million Song Database. This document is to give a brief overview on our implementations, observations, and an analysis of a dataset provided by Spotify. The objective of this project is to demonstrate relevant skills in the field of Data Sciences.**

***Keywords*— Database, Song, Spotify,**

I. Introduction

Spotify released a dataset named The Million Playlist Dataset (MPD) as challenge for users to create and extend their own playlists. This dataset will be used to train, test, and observe models that we have implemented. For our purposes, we will be using a subset of the MPD called the Million Song Dataset (MSD). The objective of this project is to take our dataset, set our songs' genre as ground truths, train a few different data models, and see how accurate our data models are to predict the genre of the songs.

II. Development Tools

While implementing our data models, we used a number of tools to implement our models with Jupyter-Notebook. The technologies that were used are as follows:

A. Python

Python is a popular programming language used by the data science community. Python has lots of technologies that appeal to the data science such as graphics and machine learning. Following that, Python also has packages such as NumPy, SciPy, and Pandas that are relevant for this project.

B. NumPy

NumPy is a package that is used for scientific computing used in Python. NumPy provides an effective means of storage along with providing a means of handling mathematical operations. The biggest appeal of NumPy to us are NumPy arrays.

C. Scipy

Scipy is a library that uses the NumPy package for mathematical functions. Scipy applies models from NumPy for tasks such as matrices and optimization.

D. Sklearn

Sklearn is a tool used for machine learning in Python. Sklearn will help train our models using algorithms such as Logistic Regression, Neural Network, and Decision Tree.

E. Pandas

Pandas is a library primarily used for data manipulation. Pandas is based on Numpy and has tools that will allow us to access, index, merge, and group data.

F. Matplotlib

Matplotlib is a library that is used for creating graphics that helps users visualize data.

III. Data Models

There are a variety of different data models in the field of data science. Each model is a machine learning algorithm that takes a series of data points as an input. We used the following models:

A. Logistic Regression Model

Logistic Regression Model classifies data with a binary output. It is used to describe data and explain the significant between one binary variable to another variable. We use this as opposed to linear regression because our response variable is not continuous (height, weight).

B. Decision Tree Model

Decision Tree Model classifies data based on a series of inputs. Decision trees will typically look to ask for an input and then branch off to ask for another input up until a conclusion is made. Decision trees are easy to understand and are visually simple to follow.

C. Support Vector Machine

Support Vector Machine classifies data using a hyperplane to classify its data points. In terms of our project, our SVM

will draw a “line” on a 2D plane containing our data points that will classify each side as a class.

D. Neural Network

Neural Network classifies data using a supervised learning algorithm called Multi-layer Perceptron. Neural Networks take an input, passes the input through the specified amount of neuron layers that modify the passed value, and outputs the latest value from the last layer.

E. K-Nearest Neighbor Model

K-Nearest Neighbor (K-NN) Model classifies data based on the proximity of a set of data points. In the K-NN model, there is usually a centerpoint for the data and everything within a radius of that data point is classified in that class.

IV. Data Description

Our data set we will be using is called the Million Song Dataset (MSD). The MSD is a subset of a of Spotify’s Million Playlist Dataset (MPD).

A. Data Sources

The Million Song Dataset is a freely-available collection of audio features and metadata for a million contemporary music tracks. It was created under a grant from the National Science Foundation, project IIS-0713334.*[3]* The original data set was published by The Echo Nest as part of an NSF-sponsored GOALI collaboration. The Million Song Dataset is a collaboration between the Echo Nest and LabROSA, a laboratory working towards intelligent machine listening.

B. Data Description

The original data set (Million Playlist Dataset) is 280 GB. However, we used the subset provided by the CORGIS Dataset Project. The Million Song Dataset file (subset will be referred to as MSD) is represented as a “MSD_data.csv” with 10,000 songs and 35 attributes with a estimated file size of 33MBs.

TABLE I

MSD_Data CSV File Example

Attributes	**Description**	**Values**
artist_hotness	algorithmic estimation	0.553072
artist_id	Echo Nest ID	ARIK43K1187 B9AE54C
artist_name	artist name	Lionel Richie
artist_mbtags	tags from musicbrainz.org	soul and reggae
artist_mbtags_count	tag counts for musicbrainz tags	1.0
bars_confidence	confidence measure	0.191
bars_start	beginning of bars, usually on beat	0.82842
beats_confidence	confidence measure	1.000
duration	song length in seconds	307.39240
end_of_fade_in	seconds before song starts	0.612
familiarity	artist familiarity	0.776676
key	key the song is in	3.0
key_confidence	confidence measure	0.524
latitude	artist latitude	37.157357
location	location name	Beverly Hills, CA
longitude	artist longitude	-63.933358
loudness	overall loudness in dB	-8.346
mode	major or minor	1
mode_confidence	confidence measure	0.533
release_id	ID from 7digital.com	25811
release_name	album name	Dancing On The Ceiling
similar	Echo Nest artist IDs	ARVP1TO118 7B9A4A7A
song_hotness	algorithmic estimation	NaN
song_id	Echo Nest song ID	SOBONFF12A 6D4F84D8
start_of_fade_out	time in sec	296.658
tatums_confidence	confidence measure	0.389
tatums_start	smallest rhythmic element	0.58901
tempo	estimated tempo in BGM	125.197
terms	song genre	quiet storm
terms_freq	genre tags freqs	1.000000
time_signature	confidence measure	3.0
title	song title	Tonight Will Be Alright

year	song release year from MusicBrainz	1986

V. Initial Observations

Before we data cleaned, we wanted to explore our dataset named "df_msd". We did that by seeing the size of the array we are dealing with. Our result came out with 10,000 rows and 35 columns. We listed our data in terms of years and took the mean of values in each column. The only significant observation we had in that data frame is that as time went on, the average song duration increased.

Following that, we created a bar graph to show song release dates. We observed that most of our data contains songs between the years of 1990 and 2010.

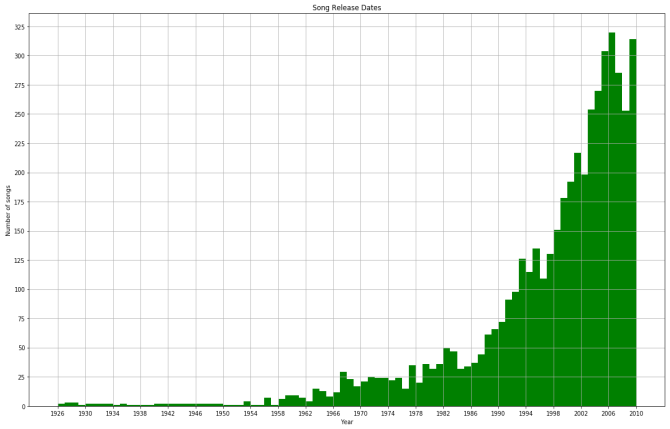

Fig. 1 A sample histogram showing the song release trend over the years. The x-axis represents years and the y-axis represents number of songs in the data set.

The primary objective of this project is to predict genres of songs, therefore we will observe our genre feature. We created a dataframe called "top_genres" and examined what the top genres were.

	genre	count
36	blues-rock	346
215	hip hop	346
60	ccm	255
69	chanson	208
102	country rock	156
253	latin jazz	150
328	post-grunge	146
106	dance pop	141
179	gangster rap	134

Fig. 2 A look into the most popular genres in our dataset.

Our top genres include "blues-rock", "hip hop", and "ccm" in the as our top 3 genres at 346, 346, and 255 counts respectively.

VI. Preprocessing Data

Looking into our initial dataset called "df_msd", we observed a couple things that could be adjusted. To begin preparing our data, we created two datasets called "df_msd_vect" and "df_msd_num". In our data set, we can categorize our data points as either strings or numbers. The "df_msd_vect" will contain data points containing strings so that we will vectorize them, and our "df_msd_num" will contain numeric values that we will normalize.

Next, we vectorized the features "artist_id", "artist_name", "artist_mbtags", "location", "similar", "song_id", and "genre". For vectorizing, we set our max features to 2000 and our "min_df" to 1. After we finished vectorize those features, we created a dataframe for each of those features and concatenated them with the main dataframe to produce our "vect_merge" dataframe. Because we vectorized some features, we created a new dataframe called "df_msd_vect" that dropped those prevectorized features because those are no longer relevant.

Before we normalized data, we had to perform the a standard scaler operation, fit operation, and transform operation on our "df_msd_merge_arr" so that we can have our "X_std" to be used for training. From there, we created a new dataframe called "norm_df" that contains our normalized data and takes in our "X_std" data frame. After that, we concatenated our vectorized and our normalized data frame to produce our "norm_vect_merge" data frame. We further cleaned our "norm_vect_merge" data frame by dropping all records that contains "NaN" in our "song_hotness" column.

Then, we created another data frame called "top_genre" that keeps count of how many times each genre appears in our data frame.

	genre	genre_count
34	blues-rock	239
191	hip_hop	201
53	ccm	176
293	post-grunge	110
89	country_rock	93
60	chanson	89
227	latin_jazz	85
158	gangster_rap	79
93	dance_pop	77

Fig. 3 The genre count after we cleaned our data.

After that, we created a new dataframe called "new_top_genres" containing only genres that appear more than thirty times. Finally, we merged the datasets "norm_vect_merge" and "new_top_genres" to a dataframe named "final_merge".

	artist_hotness_norm	artist_mbtags_count_norm	bars_confidence_norm	bars_start_norm	beats_confidence_norm
0	0.114490	-0.593501	1.399524	-0.278543	0.682443
1	4.319056	0.537655	2.405614	0.072621	0.242031
2	0.751981	0.537655	-0.630002	1.095586	-0.108438
3	0.188801	0.537655	-0.813873	-0.379896	1.197290
4	0.180283	-0.593501	-0.362868	-0.600515	-1.904203
5	0.126929	-0.593501	0.043038	1.969165	-0.241802

Fig. 4 A preview of our final_merge ready to be trained.

Now that we have everything needed to train, test, and split our data, we split our dataset into a train and test data. This will allow our models to predict our ground truths. Our ground truths for this will be genres. We assigned each genre a number and that will be what our models will be trying to guess. Now we are ready to test our models.

VII. Model Training and Output

For our models, we are going to get the confusion matrix, precision, recall, and F-measure for each. This will tell us how well we trained our models to predict our ground truths (genre).

1) *Decision Tree:* Our first output from decision tree is the precision which is 1.0. The next output is our recall which came out to be 1.0. Our last output came out to be the F-measure which was also 1.0.

2) *Support Vector Machine (SVM):* Our first output here is precision which came out to be 0.9496. The next output is our recall which is 0.9514. Our last output is F-measure which was 0.9438.

3) *Logistic Regression:* First output is our precision which is 0.9977. Our next output is our recall which is 0.9975. The last output is our F-measure which came out to also be 0.9975.

4) *Neural Network:* Our first output for Neural Network is precision which came out to be 1.0. Our next output is recall which is 1.0. Our last output is F-measure which was also 1.0.

5) *k - Nearest Neighbor (k-NN):* First output for our k-NN model is our precision which came out to be 0.9965. Our next output is recall which came out to be 0.9963. Finally, our F-measure outputted 0.9963.

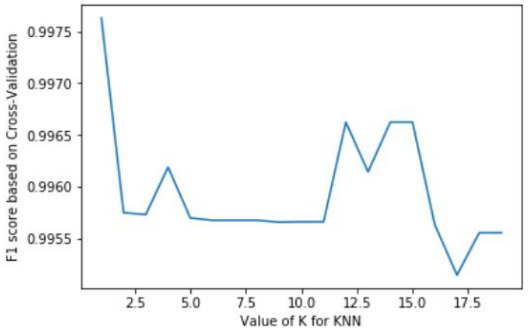

Fig. 5 F1 score graph of k-NN.

VIII. Initial Analysis

The objective of this project is to train data science models and see how well our models are at predicting our ground truths, which would be our genres. Looking at our output, it seems as though we were really successful in training our models. We have a total of 43 genres that our machine can guess. This means that if one were to randomly guess our genre, there would be a 2.33% chance of guessing correctly. When we tested our models, all of our models correctly predicted with over 97% accuracy. Our models Decision Tree, Logistic Regression, and Neural Network both had an outstanding prediction success rate at 100%.

IX. Parameter Tuning and Analysis

Our models are fairly accurate, but there is still more that we can do with our data. With took our data and used the Recursive Feature Elimination to figure out our 10 best features in our data set. From there, we are going to use that information to train and test our data once more.

Our new Decision Tree model named "dtree_tune", outputted 1.0 for the precision, recall, and F-measure. There is no change in this output and our previous output. The tuned SVM model named "svm_clf_tune" outputted 1.0 for the precision, recall, and F-measure. Compared to our initial test, we went from an average of 0.9483 to 1.0, which is an improvement. With our Logistic Regression model named "lr_clf_tune", outputs for our precision, recall, and F-measure resulted in 0.3625, 0.5524, and 0.4227 respectively. For our Logistic Regression model, we went from a 99% success rate to about a 45% success rate. And lastly, our Neural Network model outputted 1.0 for all three accuracy measurements, compared to our initial output of 1.0.

Our initial models were exceptionally accurate with all of them being over 95% successful prediction rate. After we parameter tuned, our SVM model improved to a flawless success rate and our Logistic Regression model lost more than half of its success rate. Even though we improved one of our models to be flawless, another one of our models took a huge hit resulting in a pretty poor accuracy.

X. Work Division

The division of work for our group is as follows. Jimmy was in charge of vectorizing, normalizing, training and testing the models, and annotating the code with comments.. Eric was involved in the analysis, report and presentation, and data cleaning. Justin was in charge of exploring out data, testing the models, graphics, powerpoint, and partial fo the report.

XI. Learning Experiences

There are a few things that we learned from this project. Firstly, we learned of Pandas' "get_dummies()" function. Basically, our implementation of "get_dummies()" was to vectorize each of our eight features, then concatenate each of those it to our main data frame. It took several lines of copy and pasted code that could've been implemented with Pandas' built in function. Another thing we learned is how to use the Recursive Feature Elimination function from sklearn. The Recursive Feature Elimination function gave us the best features in our data set and allowed us to retrain our models with said features. When we trained our models, our models

became more accurate or less accurate depending on what models you are using and how you implement those models.

XII. Conclusions

The intent of this project is to demonstrate what we've learned in our Data Science course. To show that we are capable of such, we took the Million Song Dataset and enacted several skills that we've learned. Firstly, we were able to clean our data set with things that are irrelevant to training our models. Then we were able to vectorize and normalize our data. From there, we were able to train, test, and observe our data. Finally, we had the opportunity to analyze our findings and explain the significance of our observations.

From our analysis, it seems that our data models are very accurate. We had three models that had a 100% prediction rate which is pretty outstanding in our opinion. Furthermore, the lowest prediction rate we observed was at 97.98% which compared to a random guess of 2.33% is also excellent.

The objective of this project is to show what we have learned in this Data Science class. We did this by taking a data set, cleaning that data set, then training, testing, and analyzing our results. We feel that if you are able to do this, then we have displayed a firm understanding of Data Sciences. Following that, our models were able to predict at near flawless and flawless success rates which means we were successful in training our models. Overall, our group collectively has learned necessary elements of Data Sciences that we will be able to use in the future.

References

[1] Maruti Techlabs. (2019). *Is Python the most popular language for data science? - Maruti Techlabs.* [online] Available at: https://www.marutitech.com/python-data-science/ [Accessed 24 Apr. 2019].

[2] Reitz, K. and Python, R. (2019). *Scientific Applications — The Hitchhiker's Guide to Python.* [online] Docs.python-guide.org. Available at: https://docs.python-guide.org/scenarios/scientific/ [Accessed 25 Apr. 2019].

[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. *The Million Song Dataset*. In proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.