Artist Recognition

from Audio Features of Songs

An attempt to analyze Million Song Dataset using MapReduce

V 0.1	12.06.2015	Initial Draft
V 0.2	12.12.2015	Updated Tasks
V 1.0	12.13.2015	Final Draft

Project Team

Arulselvan Madhavan
Yogendra Miraje
Ameya Pandilwar

Table of Contents

1 Introduction

Science behind the music has been a topic of serious research recently. Analysing the huge data about songs and music composed for over a century is an exciting big-data topic and can provide some very interesting insights that have never been looked at before.

1.1 Problem Statement

The primary goal of our project is to predict the Artist who composed the song by using the audio features of the song.

An Artist can be a person or a band of musicians who composed the song. A unique ArtistID identifies each Artist or a band. The dataset we are using is 'Million Song dataset' (MSD) that contains metadata about 1 million songs prepared through collaboration of Columbia University and The Echo Nest.

1.2 Dataset Information

MSD has 273 GB of data comprising of 1 million songs/files(since 1922) from 44, 745 unique artists. The data is in HDF5 format.

The dataset has 42 fields for each song. You may find all the fields at http://labrosa.ee.columbia.edu/millionsong/pages/example-track-description

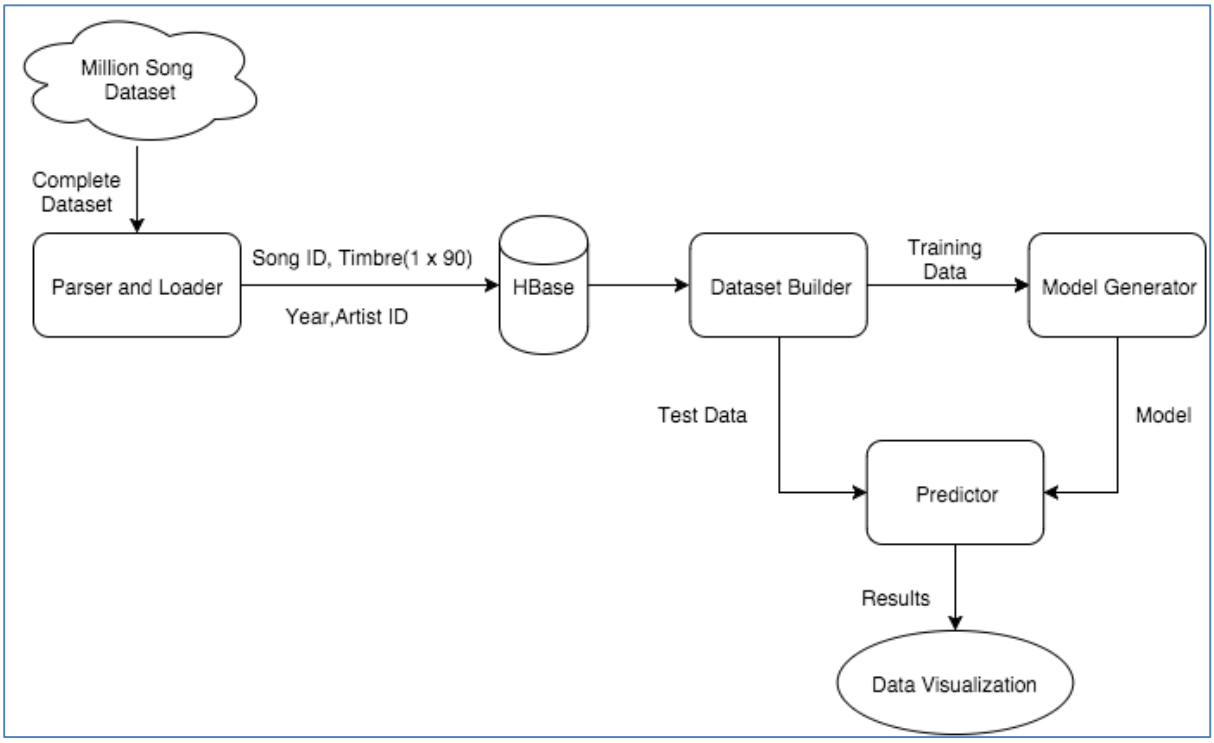
The fields for our analysis are given below.

Column Name	DataType	Description
Artist Id	String	ID of the Artist in the ECHO Nest database
Song Id	String	ID of the song in the ECHO Nest database
Timbre	2D Array of Float	Texture fields of the song.
Year	int	The year in which the song was released.

The timbre field is a vector that includes 12 unbounded values roughly centred around 0. Those values are high-level abstractions of the spectral surface, ordered by degree of importance. The first dimension represents the average loudness of the segment; second emphasizes brightness; third is more closely correlated to the flatness of a sound; fourth to sounds with a stronger attack; etc. After extraction of timbre data, we get 12 average values over the hundreds of segments of the song and 78 co-variances. We have added the released year of the song as a feature, Thus, for each song, we have 91[12 + 78+ 1] features.

2 Approach to solve the problem

The first step towards analyzing the data is to store this data in some readable and understandable format. So, we parsed the complete dataset, extracted the key features required and loaded them into a distributed database. Having access to the key features in a distributed database enabled us to prepare our train and test data using Map Reduce. We chose a Multi-Class Machine Learning algorithm for training our model and predicting artists for the test songs. We prepared Confusion Matrix by year and used data visualization tools to analyze the variations in our predictions.



3 Technology Stack

1. Hadoop 2.6
2. Hbase 0.98.15
3. Anaconda Python 2.7
4. Thrift 0.9.3
5. Spark - 1.5.0
6. Scala 2.10
7. Amazon Web Services (EC2, EMR, S3)

4 Data Cleansing

4.1 Accessing the Dataset

The Dataset is available as an AWS Snapshot. We mounted the AWS Snapshot into a M3.xlarge EC2 machine, which served as our access point for data throughout our project.

4.2 Reading HDF5 Formatted Files

Our Dataset was written in HDF5 format. Out of all high level programming languages, only Python had a well-written library for parsing HDF5 content. So, we decided to use Python to access the different features of the song.

4.3 Feature Selection

The key factor to the success of any prediction algorithm is feature selection. The key audio feature for predicting the artists is timbre. Timbre is a 2D array of float values representing the spectral surface of the song. Given below is a sample timbre value for a song.

```
[[  1.623  14.133  24.345 ...,  28.915  27.059  19.852]
 [ 147.921  62.654  15.674 ...,  71.085   9.634  53.185]
 [ -18.608  55.373  53.274 ...,  -5.287  27.235 -49.148]
 ...,
 [  12.15   29.04   10.286 ..., -11.659   1.134 -23.767]
 [ -17.023  26.954   2.878 ...,  -9.901   1.608 -58.072]
 [ -11.293 -20.522  -4.528 ...,   4.693  -2.655   1.341]]
```

There are 12 rows and hundreds of columns (1 column for each song segment[1]). We compute the average of each row and calculate the first 12 features. We compute the co-variance of this 2D array and calculate 78 features. And finally the last feature we choose is the 'Release Year' of the song. Given below is a sample feature vector for a song

```
SongID Timbre Year ArtistID
TRAQIED128F42915CC      47.0137585302    62.8697716535    2.07508530184    ……. -
2.02984514436         -0.841545931759  -4.00789107612               1999 ARPFQSQ1187FB5284B
```

[1] Segment is a set of sound entities (typically under a second) each relatively uniform in timbre and harmony.

4.4 Loading Data Into HBase

Since we use Python for extracting the features and HBase is written in Java, we needed 'Thrift' to translate python objects into HBase rows and columns. We used a library called 'HappyBase' to create HBase connection via thrift and post data to the thrift server.

4.5 Map Reduce Jobs

Mapper
Input1: HDF5 File Location(Files Containing Song)
Input2: Thrift Server IP address and Port

Map Only Job using Map Partitions
We managed our database connections by using Map Partitions and specifying the number of partitions. The entire input files were split into x partitions and each partition established a connection with Hbase and re-used that connection for all the records that were sent to that partition.

5 Test and Train Data Preparation

5.1 Filtering the Data

5.1.1 By Year

One of the features that we use for predicting the artist is the year in which the song was released. The Dataset did not have 'release year' for some songs. As a first step, we filtered all the songs that had no information about their 'release year'.

5.1.2 By Artists with more than 50 songs

In Supervised learning, the most important factor for building a good prediction model is choosing the data on which the model will get trained. We decided to train our model with artists who has more than 50 songs. We filtered out the artists who did not have more than 50 songs in the dataset.

5.1.3 Map Reduce Jobs

Mapper
Input1: HBase IP address and Port
Input2: HBase Table Name
Task: Scan entire HBase Table

Mapper Side Filter
Filter Songs which don't have information about release year.

Mapper Output Format: (ArtistID, (SongId,Year))
Key: ArtistID
Value: (SongId,Year)

Reducer
Key: (ArtistID,[(SongId,Year),(SongId,Year)...]

Reduce Side Filter
Filter Artists who don't have more than 50 songs.

Task
For each artist who have more than 50 songs and for each song that belong to this artist, get the row from HBase.

The last 5 songs of the artists go to the Test Dataset.
All the remaining songs goto the training dataset.

Reducer Output Format (LibSVM Format)
ArtistID 1:Feature1,2:Feature2...90:Feature3,91:Year

6 Applying the Machine Learning Algorithm

6.1 Deciding the ML Algorithm

Since we needed to do multi-class prediction, we needed a Multi-Class ML algorithm that works well on distributed machines. After considering KNN, PCA and Naive Bayes, we decided that Logistic Regression best fits our needs.

6.2 Training the model and Predicting

We used the Logistic Regression that comes as part of Apache Spark's MLlib library. It uses multinomial logistic regression to train and predict multiclass classification problems. For example, for K possible outcomes, one of the outcomes can be chosen as a "pivot", and the other K-1 outcomes can be separately regressed against the pivot outcome. In MLlib, the first class 0 is chosen as the "pivot" class.
For multiclass classification problems, the algorithm will output a multinomial logistic regression model, which contains K-1 binary logistic regression models regressed against the first class. Given a new data points, K-1 models will be run, and the class with largest probability will be chosen as the predicted class.

6.3 Modeling performance comparison

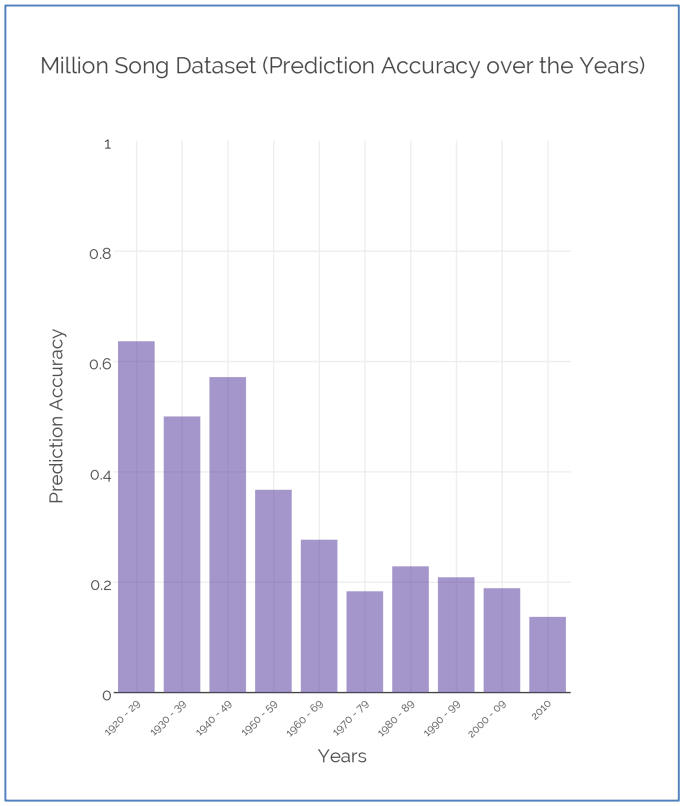
We trained our model on AWS instances. Below is the performance comparison for running modeling on AWS instances with different configuration.

No.	Instance Type	Configuration	Run time
1	m3.xlarge	1 Master, 2 Core	2 hour, 39 minutes
2	m3.xlarge	1 Master, 5 Core	1 hour, 17 minutes
3	m3.xlarge	1 Master, 10 Core	1 hour, 18 minutes

7 Analysis of Prediction Results

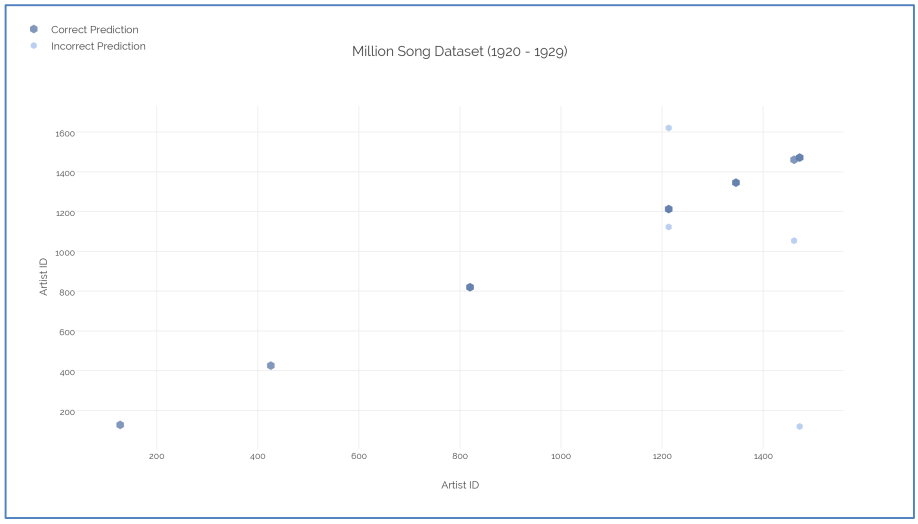
7.1 Confusion Matrix By Year

We decided to do a confusion matrix by year to analyze if there have been any significant correlation between the timbre vectors(produced by old musical instruments) in the 1920s to 1980s vs 1990s to 2010s(modern musical instruments). We found out that our model performed equally well on old songs and new songs.

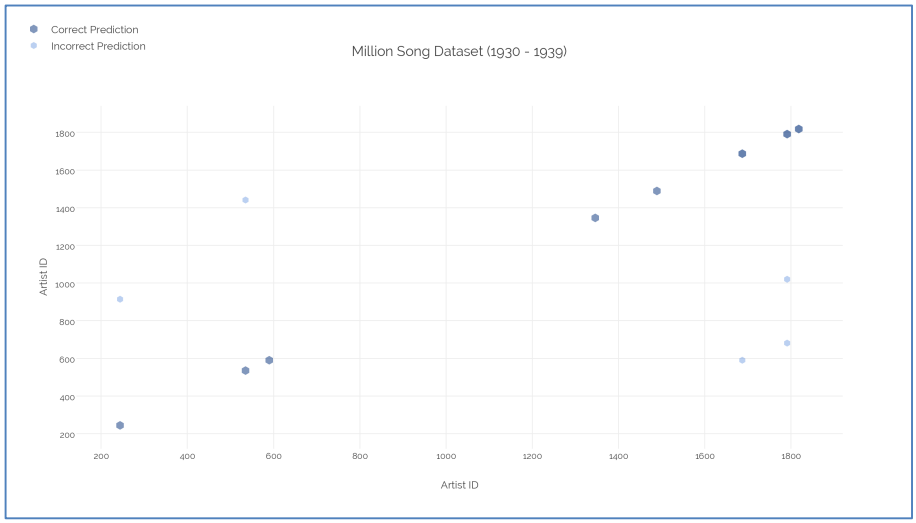


7.1.1 Breakdown of Prediction Results by Decade

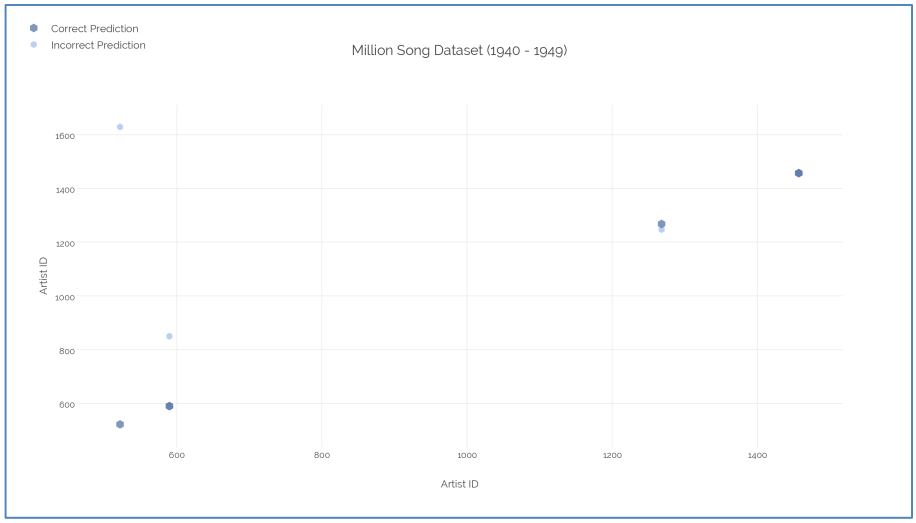
The scattered plot of the actual vs predicted artists are categorized per decade.



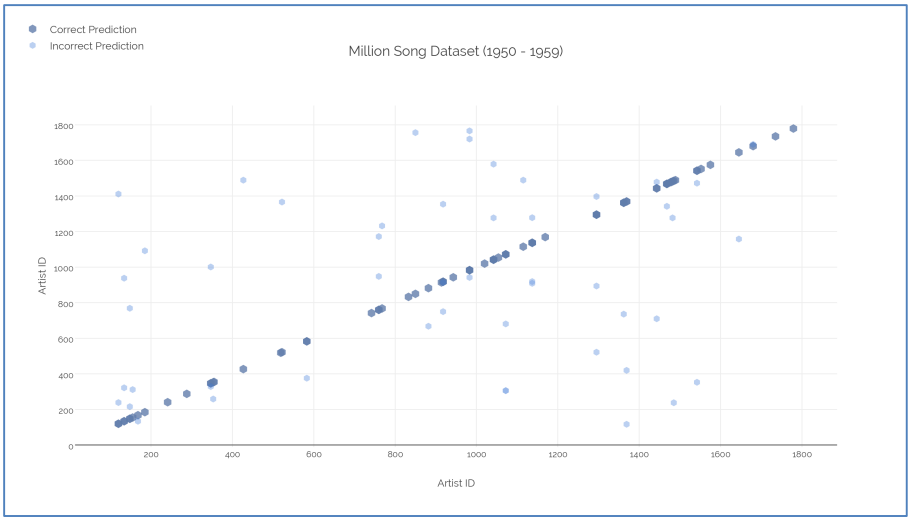
Year 1920 – 1929



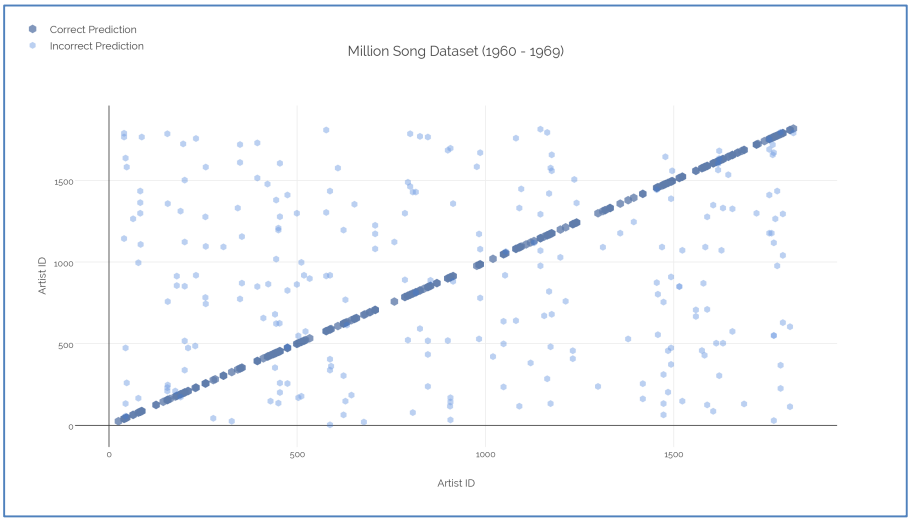
Year 1930 – 1939



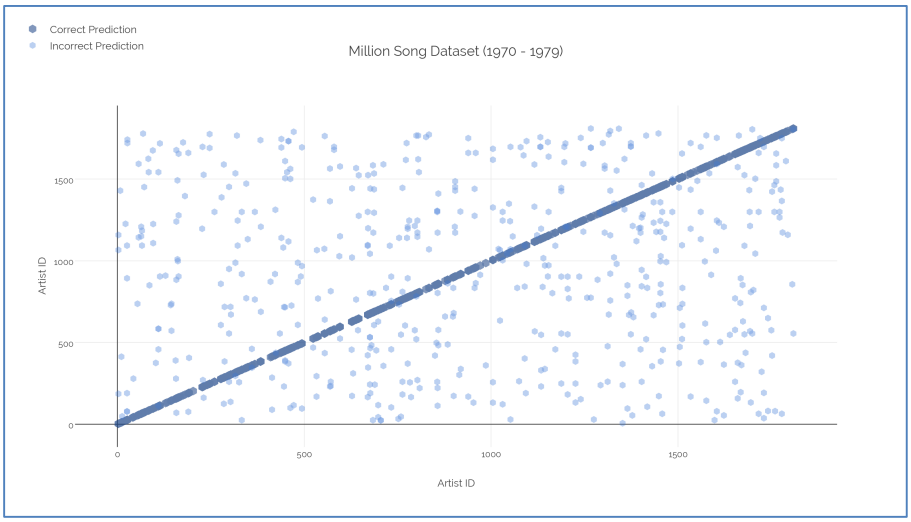
Year 1940 – 1949



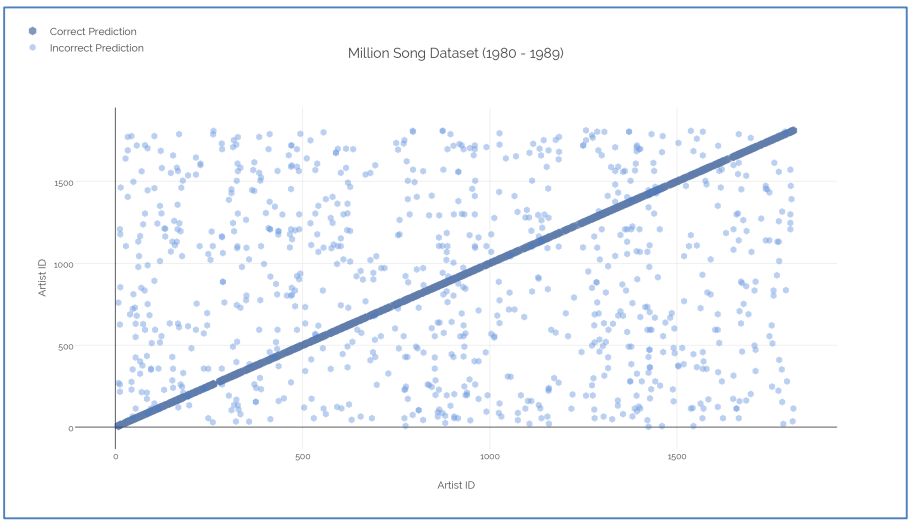
Year 1950 – 1959



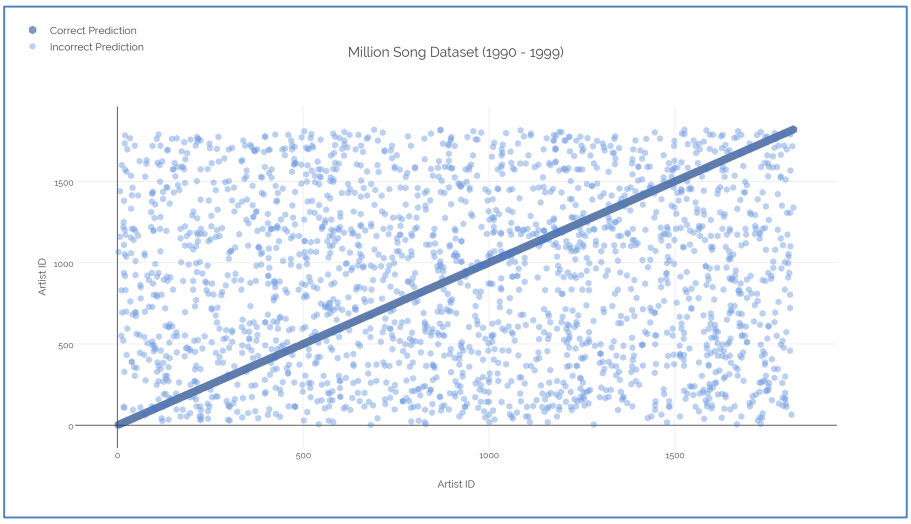
Year 1960 – 1969



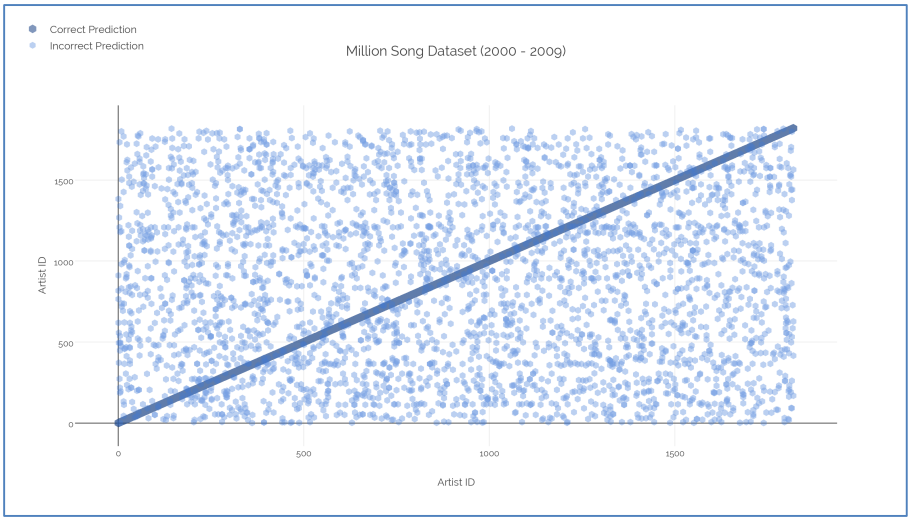
Year 1970 – 1979



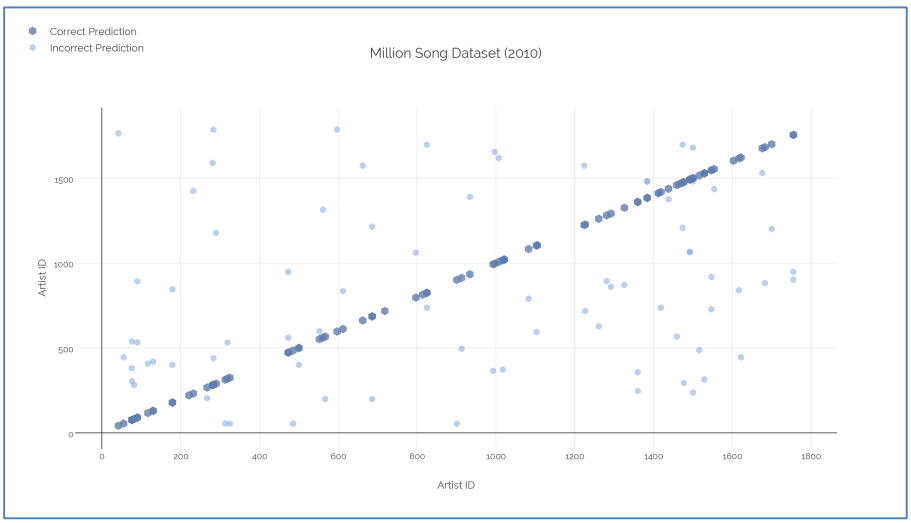
Year 1980 - 1989



Year 1990 – 1999



Year 2000 - 2009



Year 2010

7.2 Predictability of Artists

In order to evaluate the predictability of artists with our model, we found out the artists who were predicted most accurately and also those who were totally unpredictable. This may give us some of the insights about the artists who are producing songs of similar type and who are producing varying kind of music.

7.2.1 Most Predictable Artists

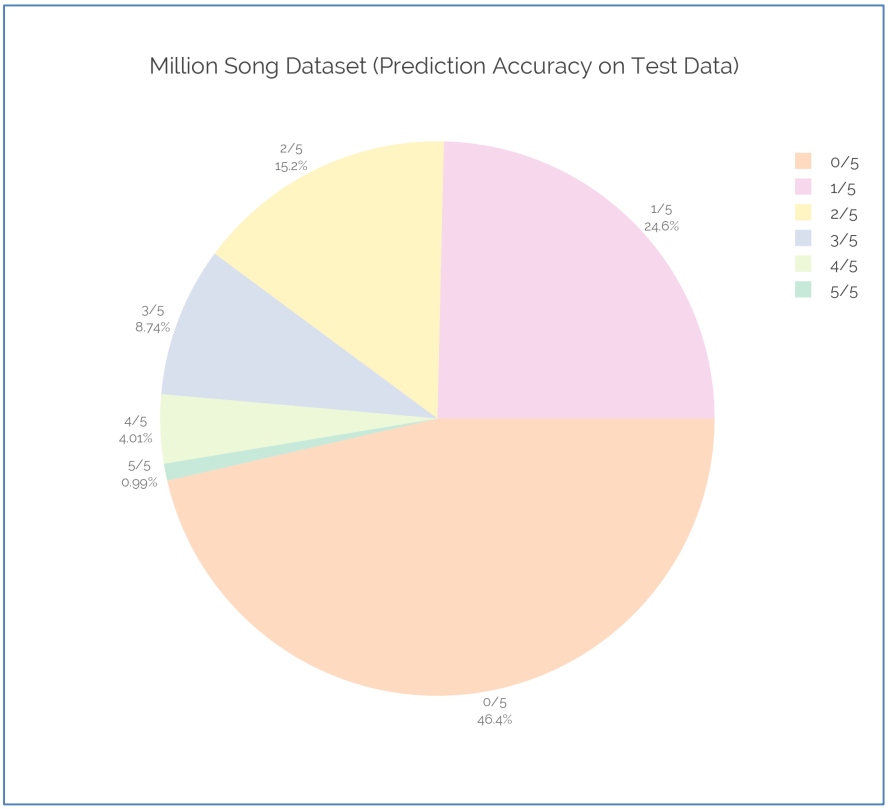
Below are the some of the artists that were predicted with 100% accuracy in our tests:

ARYF20K1187B9B76BD:	George Lopez
ARQJR4T1187FB3D259:	Yonder Mountain String Band
ARMGO6W1187FB3CEB9:	Down to the Bone

7.2.2 Least Predictable Artists

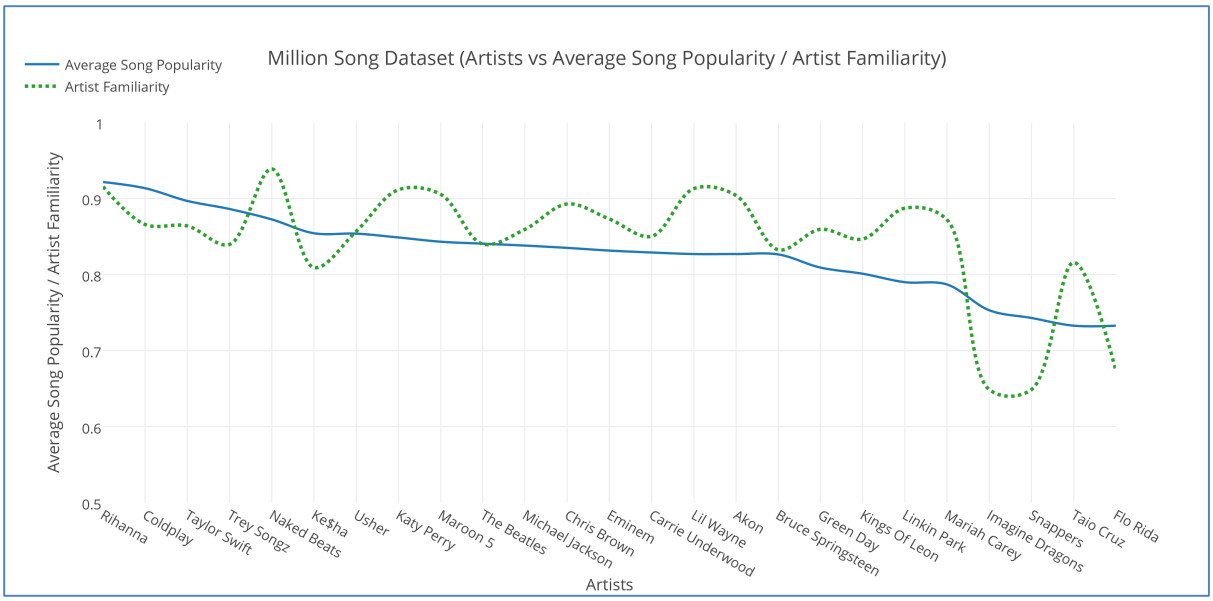
Below are the some of the artists that we could not predict accurately even once:

ARIGI7G1187B9A6D83:	Jimmy LaFave
ARFN3551187FB4C930:	The Turtles
ARLVX371187B9AF852:	Peter Dennis Blandford Townshend



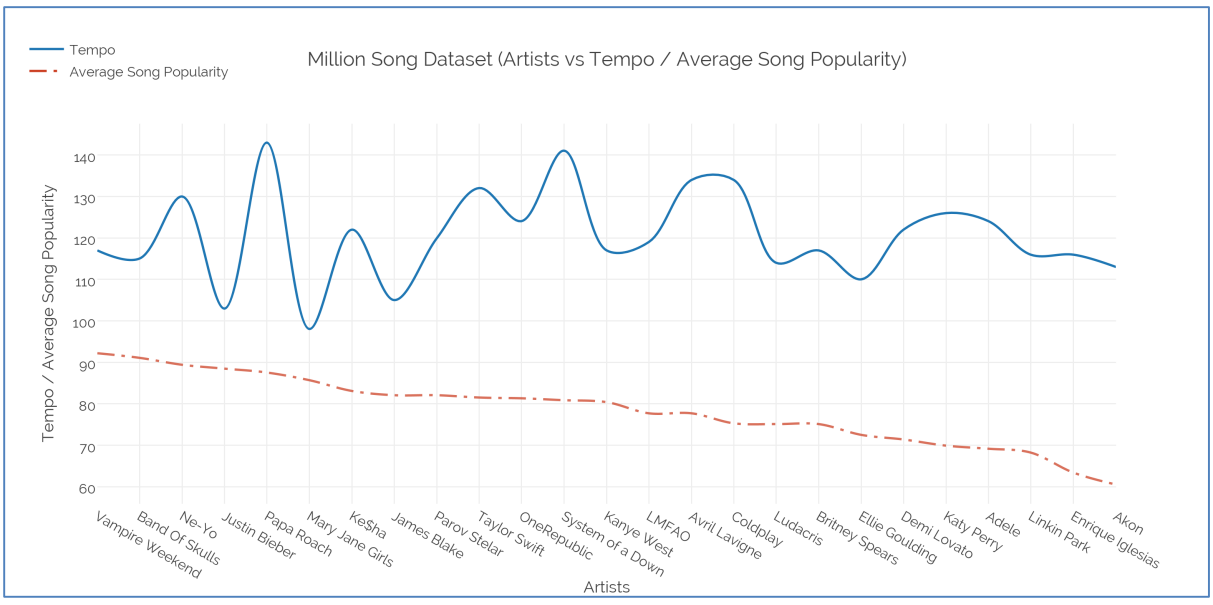
7.2.3 Insights

Trend 1 - Song Popularity & Artist Familiarity



The song popularity affects artist's familiarity.

Trend 2 - Tempo & Song Popularity



The tempo doesn't affect the song popularity.

8 References

1. Million Song Dataset http://labrosa.ee.columbia.edu/millionsong/
2. Github project: https://github.com/tbertinmahieux/MSongsDB
3. Scala tutorial: http://ampcamp.berkeley.edu/big-data-mini-course/introduction-to-the-scala-shell.html
4. HBase configuration: http://hbase.apache.org/book.html#distributed
5. Thrift Interface: http://thrift.apache.org/docs/install/
6. AWS data load: https://gist.github.com/bwhitman/130c6290514fe4d877ff
7. Spark MLlib logistic regression: http://spark.apache.org/docs/latest/mllib-linear-methods.html#logistic-regression
8. Confusion Matrix: https://www.safaribooksonline.com/library/view/advanced-analytics-with/9781491912751/ch04.html