Song recommender to sail you down the lane of nostalgia

SpotFlix

SCALABLE SONG RECOMMENDATION SYSTEM

Team Members

- Ashwani Rajan
- Sunil Kumar J S
- Vishwas Prabhu
- Melvin Vellera
- Shubham Thakur

Analytical Goals

Objective : Build a scalable song recommendation tool.

1. **Customized Song Recommendation** : Given user, recommend customized songs based on his/her preferences

2. **Song Genre Prediction** : Predict Genre of the songs

3. **Song Clustering based on Lyrics**: Based on Lyrics , group the songs

4. **Song Popularity Prediction** : Predict popularity of the songs

Datasets

Play Count	Contains information of play count for each user and song combination	52.7 Million Rows	Playcount
One Million Songs	Song level information	1 Million Rows	One Million Songs
Lyrics	Lyrics of the song	210K Rows	Lyrics
Genre	Genre of the song	423K Rows	Genre

Customized Song Recommendation

Data Description

Data Description Encoded each user and song with unique

Size 52.7 Million rows

Sample Data

User ID	Song ID	Play Count
Diane	Lovin' The Fool Out Of Me	25
Diane	It's Not Hard To Love You	3
Terence	Money Comes Fast	4

Modelling

Preprocessing Encoded each user and song with unique

Algorithm Alternative Least Square with target variable as "play count"

Cross Validation

- Performed 5 fold cross validation with 5% of training data because of computation cost and time on fitting cv on entire training data.
- Hyperparameters search space : : Regularization parameter - [10,1,0.1,0.001], embedding size :[10,20]. Time taken : ~ 44.53 minutes with cluster i3.xlarge with v.10.3 (Spark 3.2.1, Scala 2.12)
- Metrics : RMSE
- Best Hyper parameters : Regularization parameter -1 , embedding size : 10

Results

Final Model

- RMSE on test data : 4.32
- Time taken to fit on 8.54 minutes with cluster i3.xlarge with v.10.3 (Spark 3.2.1, Scala 2.12)

Song Genre Prediction

- **Analytic Goal:** Predict the **genre** of a song using the song's features
- **21 Genres:** Pop rock, International, Jazz, Classical, Rap, R&B and others
- **Features:**
  - Song characteristics: duration, key, loudness, mode, tempo, time signature
  - Decade in which song was released
  - Lyrics embeddings using Word2Vec
- **Preprocessing algorithms:** StringIndexer, OneHotEncoder, VectorAssembler
- **Machine Learning algorithms:** Weighted/Unweighted Random Forest, Logistic Regression
- **Total time taken:** 308 seconds using cluster i3.xlarge with v.10.3 (Spark 3.2.1, Scala 2.12)
- **Cross validation** for finding optimal hyperparameters took around 2 minutes

Song Genre Prediction - Results

Algorithm	F1 Score	Accuracy	Time Taken
Random Forest	0.6492	0.7463	7 seconds
Logistic Regression	0.6482	0.7445	26 seconds
Weighted Random Forest	0.6336	0.6486	9 seconds

- **Model Parameters:**
  - **Random Forest:** Max Depth: 7, Min Instances Per Node: 3
  - **Logistic Regression:** Reg Param: 0.01, Max Iterations: 1000
- **Machine Specs**:
  - **Type**: i3.xlarge
  - **Cores**: 4, **Disk**: 950 Gb, **Memory**: 30.5Gb,
  - **Min** workers: 2, **Max** workers: 8

Song Lyrics clustering based on Word2Vec and KMeans

Analytic Goal: Cluster song lyrics based on embeddings generate using Word2Vec

Features: Word frequency data for song lyrics. Embedding of size 10 generated from Word2Vec for KMeans generated.

Preprocessing algorithms used included Word2Vec. Vector assembler not required because output of word2vec is a dense vector which can be directly used in KMeans.

Machine Learning algorithms: KMeans

Total time taken: Around 40 seconds with cluster: 3 Nodes i3.xlarge with v.10.3 (Spark 3.2.1, Scala 2.12)

Cross validation for finding optimal hyperparameters took around 4 minutes

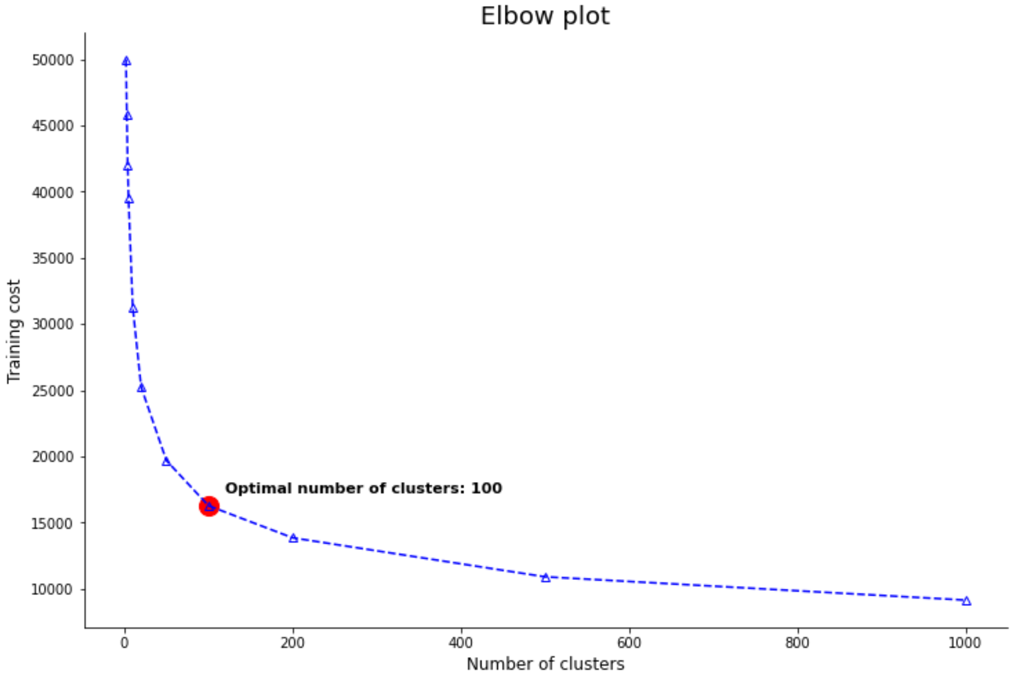
Song Lyrics clustering K-Means: Hyperparameter tuning and results

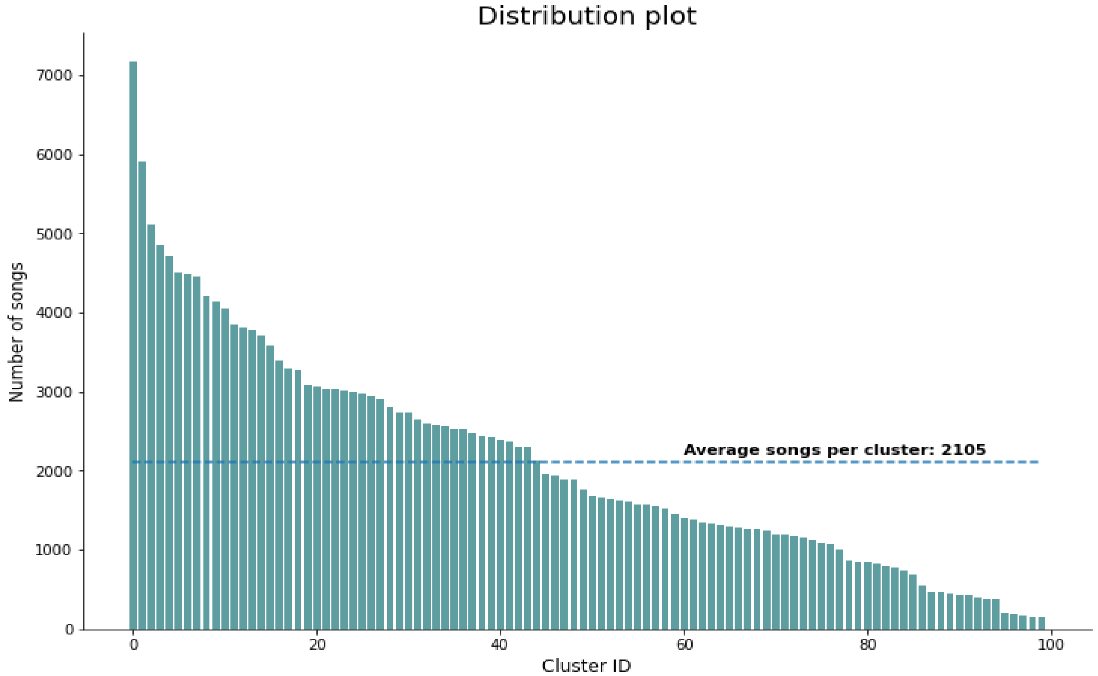
Metric: KMeans training cost which is sum of squared distances to the nearest centroid for all points in the training dataset

Elbow plot to decide optimal K: Training cost for each K plotted with the number of clusters.

Optimal K comes to be 100 since curve starts getting flat (Later decreases in a linear fashion) after cluster size of 100.

Distribution of songs across each cluster is shown below for cluster size of 100.





Song Popularity Prediction - Linear Model

Analytic Goal: To predict the song popularity using song based features

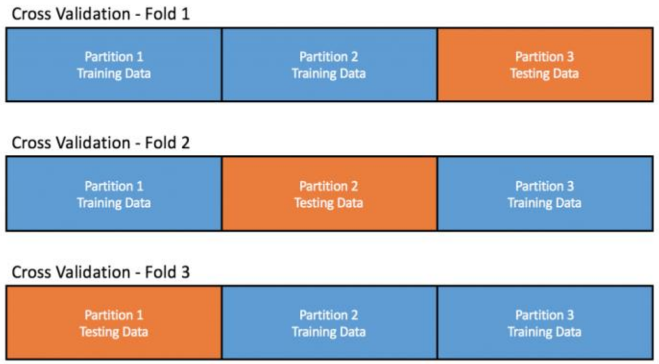
Features: Duration, Key, Loudness, Mode, Year, Time Signature, Artist name, start of fade out

Preprocessing algorithms: Missing Value Imputation, We used string indexer, One hot encoding, Vector Assemble

Machine Learning algorithms: Linear Regression with 3 fold Cross Validation

Performance Analysis

Estimator/Transformers	Execution Time(sec)
String Indexer	4.0
One hot Encoding	1.34
Vector Assembler	0.94
Linear Regression	160
Cross Validation	459



$$Y_i = f(X_i, \beta) + e_i$$

Metric: RMSE

Best Regularization Parameter: 0.01

Best Model Performance: 0.157

https://brettromero.com/wp-content/uploads/2016/05/cross-validation-1024x561.png

Song Popularity Prediction – Non Linear Models

Analytics Goal: Song popularity prediction using Non-Linear Models

Features: Duration, Key, Loudness, Mode, Year, Time Signature, start of fade out

Machine Learning algorithms: Decision Tree Regressor with Cross-Validation, Random Forest

Cross Validation: 3-fold CV with Max-depth: [5, 10, 15, 20, 25, 30] and Max-bins [16,32,48]

Metric: RMSE

Time efficiency

Estimator/Transformers	Execution Time(sec)
String Indexer	2 seconds
Vector Assembler	637 microsecs
Decision Tree Regressor + CV	21 min 45 secs
Random Forest Regressor (maxdepth = 20)	5.14 Minutes

Decision Tree:

Best RMSE: 0.2128
Cross Validation Best value for Max Depth: 5
Cross Validation Best value for Max Bins: 32

Random Forest:

Best RMSE: 0.2162
Max Depth: 20
Number of Trees: 20

Bringing It All Together

Customized Song Recommendations

Corrido de Boxeo
by Ry Cooder
Popularity: 0.55

Caned
by Markus
Popularity: 0.29

These are songs recommended by collaborative filtering by taking dot product between user embedding and item embedding

Collaborative Filtering

Select User Type

Lyrics Based Song Recommendations

Tighten Up
by The Black Keys
Popularity: 1.00

Tu Guardian
by Juanes
Popularity: 0.57

These songs are recommended based on the word2vec embedding closeness to the favorite song lyrics.

Content-based (Lyrics) Recommendations

The Electronic fan

Favourite Songs
Exit Music (For A Film)
Kids In America
Restless

The Country person

Favourite Songs
Yellow

Genre

Popularity

Lessons Learned:

1. Handling large amount of data using tools such as Pyspark, Databricks and databases such as Amazon S3, MongoDb, HDFS tables.

2. How to build distributed and scalable recommendation system to recommend songs to users using machine learning pipelines.

3. Apply Spark ML library to build user-item based collaborative filtering model using ALS

4. Use Spark ML library to build multiclass classification model such as logistic regression and random forest.

5. Use Spark ML library to build regression models such as Linear regression and Tree based algorithms

6. Use Spark ML library to generate word2Vec embeddings and cluster using KMeans.

Conclusions:

1. Large amounts of data can be stored easily using Amazon S3, MongoDB, and HDFS file systems

2. The task of processing big data using distributed clusters becomes easier and more efficient using Databricks rather than using AWS directly

3. MLlib and H2O are mature distributed machine learning frameworks for building scalable machine learning pipelines

4. Algorithms optimized for distributed computing such as ALS, Random Forest, Linear/Logistic Regression, and K-Means can be used to build a scalable recommender system that can give accurate predictions with low latency