MSCA 31013 1 Big Data Platforms

Music Recommendation

Million Song Dataset

Sasha Roberts
Skye Zhang
Sowmya Nallapaneni
Vineela Potula

Agenda

1. Business Problem
2. Our Dataset + Preparation
3. Analysis & Visualization
4. Machine Learning Models
5. Next Steps

Business Problem

We are trying to build a recommendation engine for a music streaming app using:

>> Hotness: Recorded when downloaded (in December 2010)

>> MFCC: A mathematical representation of sound

Million Song Dataset

analysis_sample_rate	artist_7digitalid
artist_familiarity	artist_hotttnesss
artist_id	artist_latitude
artist_location	artist_longitude
artist_mbid	artist_mbtags
artist_mbtags_count	artist_name
artist_playmeid	artist_terms
artist_terms_freq	artist_terms_weight
audio_md5	bars_confidence
bars_start	beats_confidence
beats_start	danceability
duration	end_of_fade_in
energy	key
key_confidence	loudness
mode	mode_confidence
num_songs	release
release_7digitalid	sections_confidence
sections_start	segments_confidence
segments_loudness_max	segments_loudness_max_time
segments_loudness_start	segments_pitches
segments_start	segments_timbre
similar_artists	song_hotttnesss
song_id	start_of_fade_out
tatums_confidence	tatums_start
tempo	time_signature
time_signature_confidence	title
track_7digitalid	track_id
year	

Dataset Size: 280 GB
44, 745 unique artists 515, 576 dated tracks

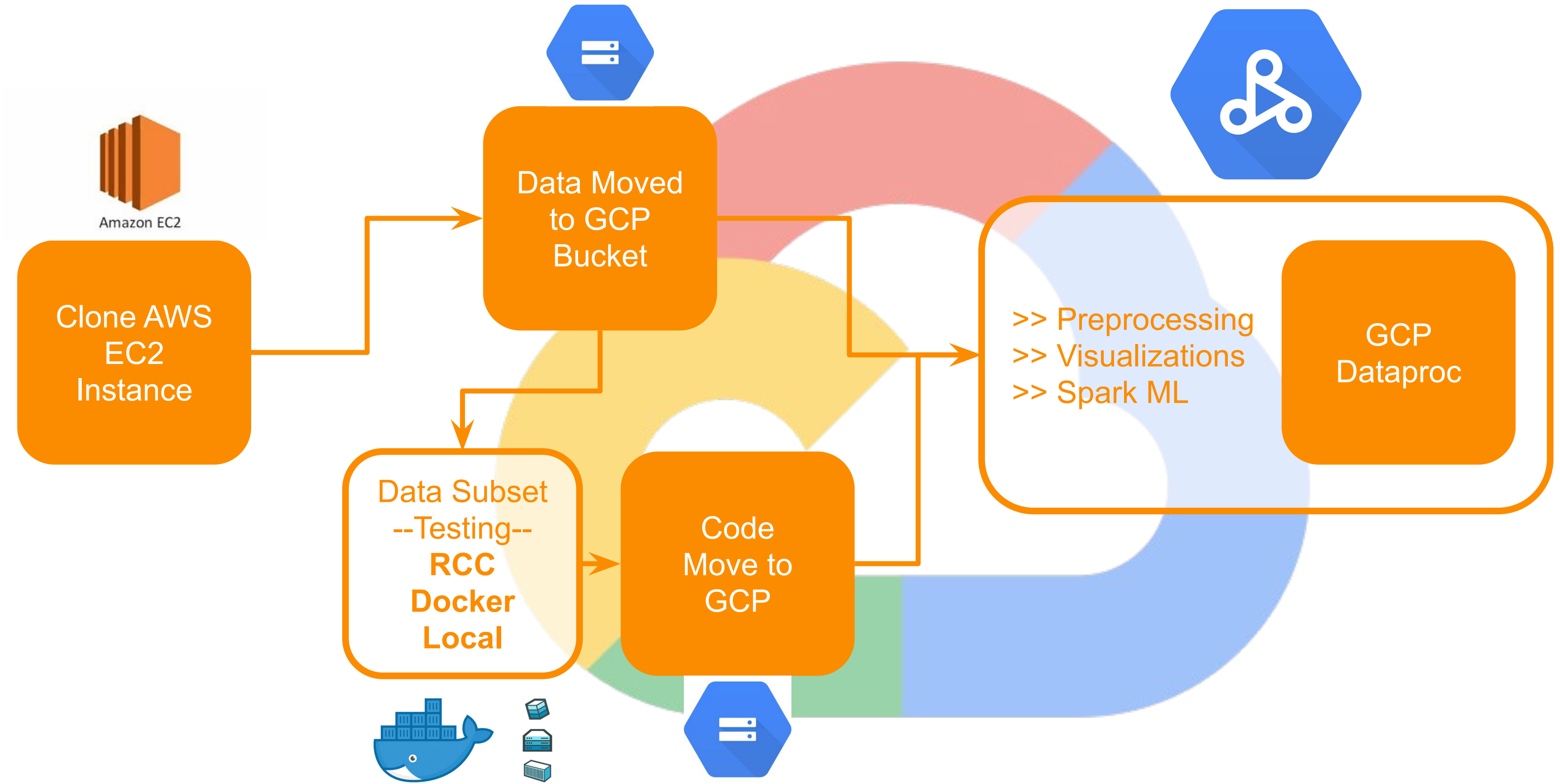
Years: 1926-2010 (we focused on 2000 to 2010)

Our Subset: 6.5+ GB
76,799 songs

Data Access: Amazon Public Dataset Snapshot

Data Format: Stored using HDF5 format.

Contains: Each file represents a Song that contains audio features and metadata



Amazon EC2
Clone AWS EC2 Instance
Data Moved to GCP Bucket
Data Subset
--Testing--
RCC
Docker
Local
Code Move to GCP
>> Preprocessing
>> Visualizations
>> Spark ML
GCP Dataproc

Data Preprocessing

Missing Values

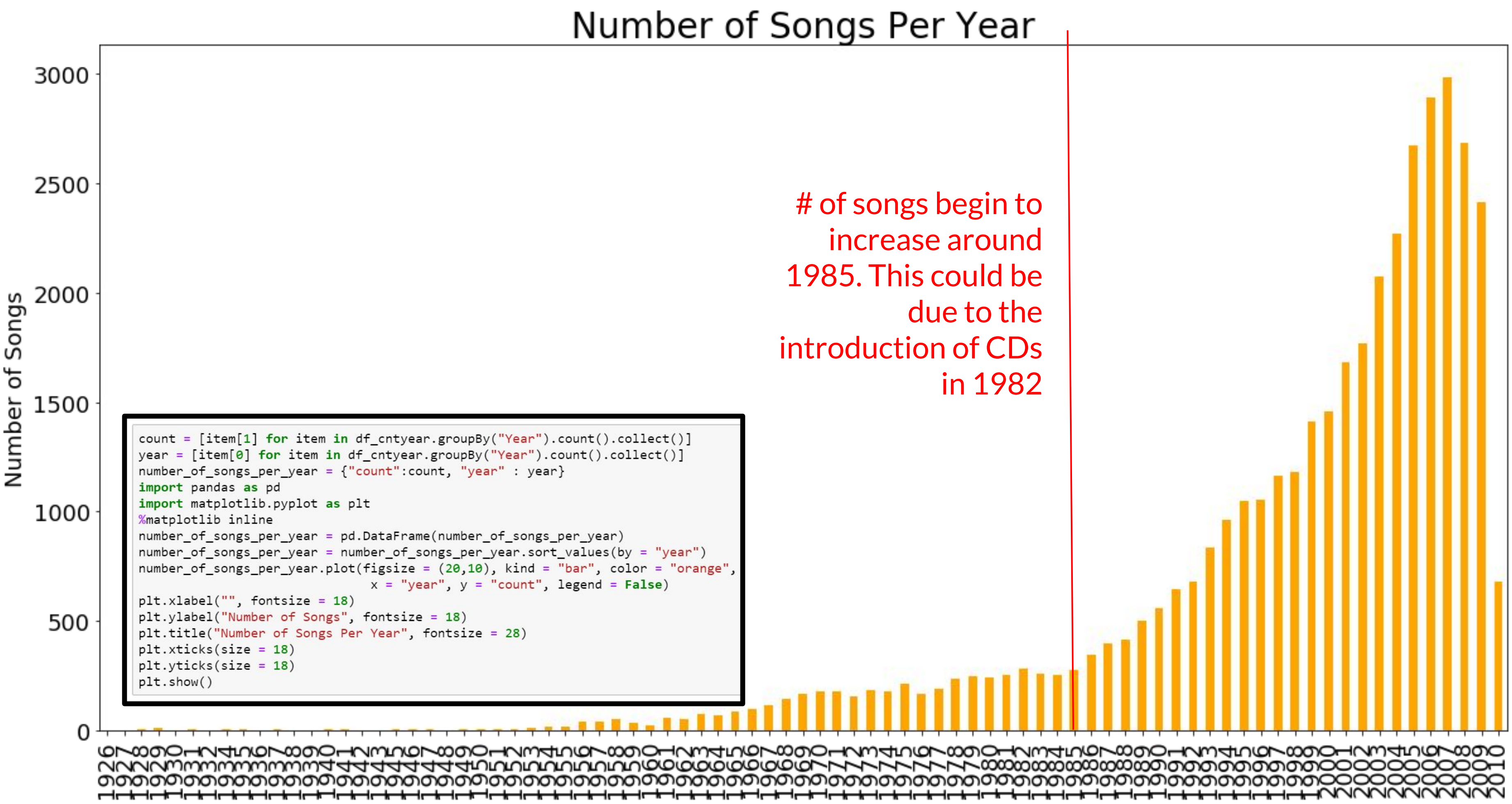
```
ArtistLatitude 49527
ArtistLocation 37631
ArtistLongitude 49527
```

Invalid Values

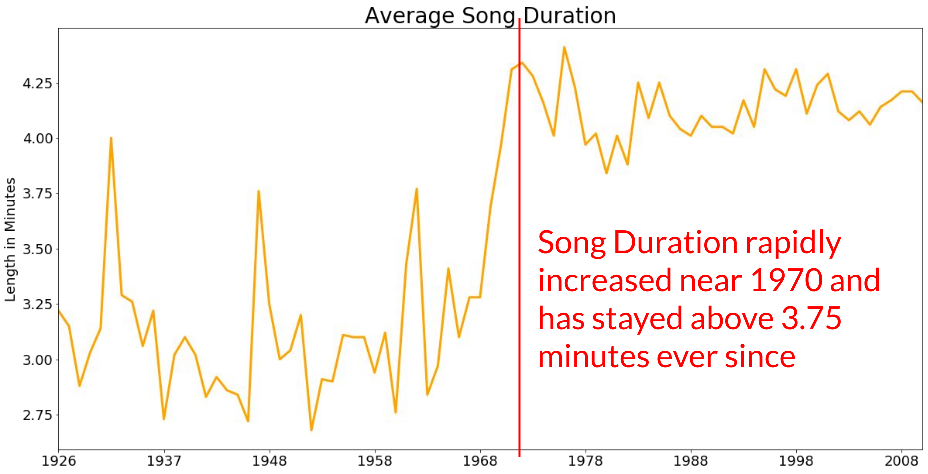
```
+------------------------------+
|count(DISTINCT song_danceability)|
+------------------------------+
|                             1|
+------------------------------+
```

```
correlation_matrix(df, colnames_int, method='pearson')
```

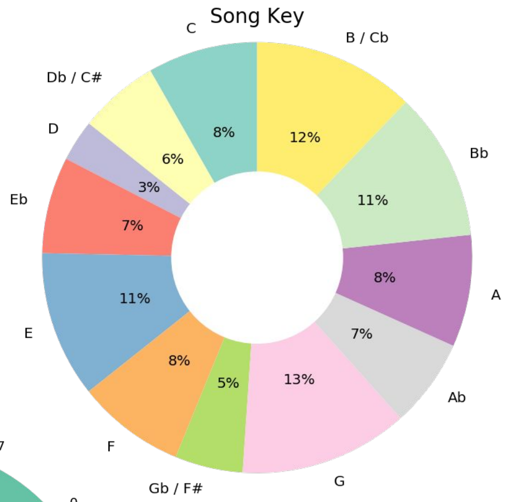
	Duration	KeySignature	KeySignatureConfidence	Tempo	TimeSignature	TimeSignatureConfidence
Duration	1.000000	0.027058	-0.008351	-0.022009	0.094642	0.036313
KeySignature	0.027058	1.000000	-0.012559	0.020896	0.011392	0.006313
KeySignatureConfidence	-0.008351	-0.012559	1.000000	0.006298	-0.035078	-0.005598
Tempo	-0.022009	0.020896	0.006298	1.000000	0.054549	-0.083065
TimeSignature	0.094642	0.011392	-0.035078	0.054549	1.000000	0.092429
TimeSignatureConfidence	0.036313	0.006313	-0.005598	-0.083065	0.092429	1.000000



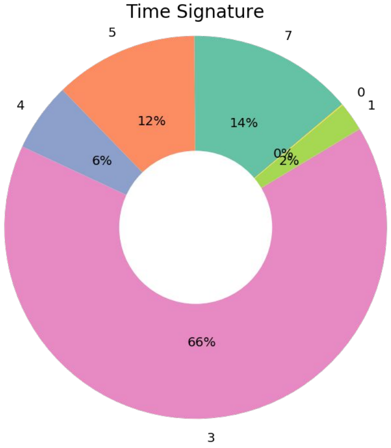

```python
count = [item[1] for item in df_cntyear.groupBy("Year").count().collect()]
year = [item[0] for item in df_cntyear.groupBy("Year").count().collect()]
number_of_songs_per_year = {"count":count, "year" : year}
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
number_of_songs_per_year = pd.DataFrame(number_of_songs_per_year)
number_of_songs_per_year = number_of_songs_per_year.sort_values(by = "year")
number_of_songs_per_year.plot(figsize = (20,10), kind = "bar", color = "orange",
                              x = "year", y = "count", legend = False)
plt.xlabel("", fontsize = 18)
plt.ylabel("Number of Songs", fontsize = 18)
plt.title("Number of Songs Per Year", fontsize = 28)
plt.xticks(size = 18)
plt.yticks(size = 18)
plt.show()
```



The most popular Song Key isn't clear but G, B/Cb, and E are popular in this dataset.



Most *Hot* Songs Have A Tempo of **122** Beats Per Minute



The most popular Time Signature was '3' but documentation was unclear what this maps to. The most common meter in music is 4/4.

Anatomy of a Song


```
#!pip install wordcloud
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
```

```
words = df.select("Title").rdd.flatMap(lambda x: x).collect()
```

```
stopwords = ['Album Version','Version','Remix','LP',
             'Digital','Remaster', 'Album','Digital Remaster', 'Live']

STOPWORDS.update(stopwords)
```

```
text = words
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'white',
    stopwords = STOPWORDS.update(stopwords)).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

What Should You Name Your Song?

La De Love

—

Pipeline:

- EDA to understand the data and identify the important features
- Input all the features to the vector assembler
- Create a vector from the features
- Use the feature vector for scaling
- Input scaled features to the ML model

Building a pipeline for ML

```python
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = col_int, outputCol = 'features')
df_lin_model = vectorAssembler.transform(df_final_v2)
df_lin_model = df_lin_model.select(['features', 'song_hotness','song_year','song_title','song_id'])
df_lin_model.show(3)
```

features	song_hotness
[284.44689,5.0,0....	0.35590136322473975
[217.67791,2.0,0....	0.3458699096807943
[294.94812,4.0,0....	0.35147900521285574

only showing top 3 rows

features	song_hotness	song_year	song_title	song_id	scaledFeatures
[112.63955,9.0,0....	0.3665278019554403	0	El Coco	SOHYCYE12A58A799B6	[0.86965412273356...
[117.26322,2.0,0....	0.34413271125964556	1996	Ease Up	SOYBAEE12A8C13EF2D	[0.90535200751435...

only showing top 2 rows

Prediction Techniques:

- Linear Regression
- Decision Tree
- Gradient Boosting

With 3-fold cross validation

Method	RMSE (test data)
Linear Regression	0.125
Linear regression- CV	0.122
Decision Tree	0.124
Decision Tree-CV	0.121
Gradient Boosting	0.125

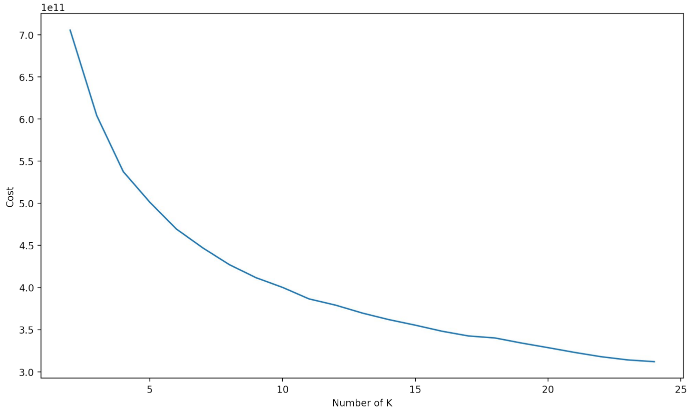
```
lr_predictions = lr_model.transform(test_df)
lr_predictions.select("features","prediction","song_hotness",'song_id',"song_title","song_year")
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="song_hotness",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))
```

```
dt_model = dt.fit(train_df)
dt_predictions = dt_model.transform(test_df)
dt_evaluator = RegressionEvaluator( labelCol="song_hotness", predictionCol="prediction", metricName="rmse")
rmse = dt_evaluator.evaluate(dt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
from pyspark.ml.regression import GBTRegressor
gbt = GBTRegressor(featuresCol = 'scaledFeatures', labelCol = 'song_hotness', maxIter=10)
gbt_model = gbt.fit(train_df)
gbt_predictions = gbt_model.transform(test_df)
gbt_predictions.select("features","prediction","song_hotness",'song_id',"song_title","song_year").show(5)
```

Clustering on Spark:

1. K-Means
2. Gaussian Mixture Models (GMM)
3. Bisecting K-Means



K should 16

In [44]:
```
kmeans = KMeans().setK(16).setFeaturesCol("features")
KM = kmeans.fit(df_model)
resultsKM = KM.transform(df_model)
resultsKM.toPandas().head(10)
```

Out[44]:

	features	song_hotness	song_id	song_title	song_year	prediction
0	[818.4930000000002, 4310.520999999999, -4150.2...	0.355901	SOPRYBN12AB0188F46	Del Miravad	2007	10
1	[634.8789999999999, 3489.2280000000005, -9213....	0.347873	SOGJOLI12AB0187011	House Carpenter	2005	15
2	[549.418, 4907.5999999999985, 4424.64100000000...	0.423428	SORPZGC12A6D4F83B5	Estrella De Mar (Estrella Fugaz)	2002	8
3	[1210.3079999999995, 4622.834, -1570.700000000...	0.353943	SOFMUOF12A6D4F81C2	A	2000	0
4	[-742.981, 4831.004, -1722.768, 2058.725999999...	0.199940	SODVFBO12AAA8C7AD2	Podzim	2007	5
5	[422.9800000000001, 4482.146000000001, -3013.0...	0.396267	SOBNTAP12A6D4FBECD	Affections the Pay	2005	5
6	[396.8369999999999, 4675.497999999998, 1170.23...	0.446194	SOKSWAF12A58A7C2E1	Apple Pie Bed	2009	14
7	[-898.6479999999997, 4530.546999999999, 4838.3...	0.335156	SOBRZNR12AB017CD10	Peanuts Clu	2009	2
8	[22.027000000000037, 3708.5579999999995, 6849....	0.425941	SOZZCYM12AF72A224A	Yaletown	2000	2
9	[219.65500000000003, 5162.152, 4683.5110000000...	0.345624	SOOVQJT12A8C1454A3	Creatures	2009	3

Gaussian Mixture Model

In [45]:
```
gm = GaussianMixture().setK(16).setFeaturesCol("features").setPredictionCol("Prediction").setProbabilityCol("Probabil
gmm = gm.fit(df_model)
```

In [46]:
```
resultsGMM= gmm.transform(df_model)
resultsGMM.toPandas().describe()
```

Out[46]:

	song_hotness	song_year	Prediction
count	17346.000000	17346.000000	17346.0

Bisecting KMeans

We used the previous number of K, 16

In [47]:
```
bkm = BisectingKMeans().setK(16).setFeaturesCol("features")
BKM = bkm.fit(df_model)
resultsBKM = BKM.transform(df_model)
resultsBKM.toPandas().head(10)
```

Out[47]:

	features	song_hotness	song_id	song_title	song_year	prediction
0	[818.4930000000002, 4310.520999999999, -4150.2...	0.355901	SOPRYBN12AB0188F46	Del Miravad	2007	2
1	[634.8789999999999, 3489.2280000000005, -9213....	0.347873	SOGJOLI12AB0187011	House Carpenter	2005	5
2	[549.418, 4907.5999999999985, 4424.64100000000...	0.423428	SORPZGC12A6D4F83B5	Estrella De Mar (Estrella Fugaz)	2002	10
3	[1210.3079999999995, 4622.834, -1570.700000000...	0.353943	SOFMUOF12A6D4F81C2	A	2000	12
4	[-742.981, 4831.004, -1722.768, 2058.725999999...	0.199940	SODVFBO12AAA8C7AD2	Podzim	2007	6
5	[422.9800000000001, 4482.146000000001, -3013.0...	0.396267	SOBNTAP12A6D4FBECD	Affections the Pay	2005	3
6	[396.8369999999999, 4675.497999999998, 1170.23...	0.446194	SOKSWAF12A58A7C2E1	Apple Pie Bed	2009	13
7	[-898.6479999999997, 4530.546999999999, 4838.3...	0.335156	SOBRZNR12AB017CD10	Peanuts Clu	2009	8
8	[22.027000000000037, 3708.5579999999995, 6849....	0.425941	SOZZCYM12AF72A224A	Yaletown	2000	8
9	[219.65500000000003, 5162.152, 4683.5110000000...	0.345624	SOOVQJT12A8C1454A3	Creatures	2009	10

Based on the results of 3 clustering algorithms, KMeans performed the best. Hence, we'll use the result of KMeans to create the playlist.

DEMO

Next Steps

- Combine **Last.fm data** with Million Song Dataset to gain access to *tags* and *similar songs*
- Plot spectrograms of the MFCCs and classify songs using CNN
- Improve GUI