

Spotify Data Analysis

- B i g d a t a p r o j e c t

EGCO 466

Big Data Processing

Spotify-Data

Sample of data

	A	B	C	D	E	F	G	H	I	J
1	id	name	artists	duration_ms	release_date	year	acousticness	danceability	energy	instrumentalness
2	6KbQ3uYMLKb5	Singende Bataill	['Carl Woitschac	158648	1928	1928	0.995	0.708	0.195	0.563
3	6KuQTIu1KoTTk	Fantasiestücke,	['Robert Schuma	282133	1928	1928	0.994	0.379	0.0135	0.901
4	6L63VW0PibdM	Chapter 1.18 - Z	['Seweryn Goszc	104300	1928	1928	0.604	0.749	0.22	0
5	6M94FkXd15sO	Bebamos Juntos	['Francisco Cana	180760	9/25/28	1928	0.995	0.781	0.13	0.887
6	6N6tiFZ9vLTSo	Polonaise-Fanta	['Frédéric Chopir	687733	1928	1928	0.99	0.21	0.204	0.908
7	6NxAf7M8DNHC	Scherzo a capric	['Felix Mendelss	352600	1928	1928	0.995	0.424	0.12	0.911
8	6O0puPuyrxPjD	Valse oubliée No	['Franz Liszt', 'Vi	136627	1928	1928	0.956	0.444	0.197	0.435
9	6OJjveoYwJdl7	Per aspera ad as	['Carl Woitschac	153967	1928	1928	0.988	0.555	0.421	0.836
10	6OaJ8Bh7IsBeY	Moneda Corrient	['Francisco Cana	162493	10/3/28	1928	0.995	0.683	0.207	0.206
11	6PrZexNb16cab	Chapter 1.3 - Za	['Seweryn Goszc	111600	1928	1928	0.846	0.674	0.205	0
12	6QBInZBkQNIQ	Piano Sonata No	['Sergei Rachma	590293	1928	1928	0.994	0.376	0.0719	0.883
13	6QIONtzbQCbnr	Piano Sonata No	['Frédéric Chopir	85133	1928	1928	0.989	0.17	0.0823	0.911
14	6QgdUySTRGVI	Piano Sonata in	['Samuel Barber'	338333	1928	1928	0.99	0.359	0.0435	0.899

Data Cleansing

```
df.count()
```

```
169909
```

```
from pyspark.sql.functions import col, sum

null_counts = df.select([sum(col(c).isNull().cast("int")).alias(c) for c in df.columns])
null_counts.show()
```

```
df = df.dropna()
```

```
#No. of column after drop null
df.count()
```

```
168462
```

Extract artist name

```
from pyspark.sql.functions import split, regexp_replace, explode

df_exploded = df.withColumn("artist_array", split(regexp_replace("artists", r"\\[\\]'", ""), ", "))
df_exploded = df_exploded.withColumn("artist", explode("artist_array"))
```

```
df_exploded.select("artist_array", "artist").show()
```

```
+-----+-----+
|  artist_array|      artist|
+-----+-----+
|[Carl Woitschach]| Carl Woitschach|
|[Robert Schumann,...]| Robert Schumann|
|[Robert Schumann,...]| Vladimir Horowitz|
|[Seweryn Goszczyń...]| Seweryn Goszczyński|
|[Francisco Canaro]| Francisco Canaro|
|[Frédéric Chopin,...]| Frédéric Chopin|
|[Frédéric Chopin,...]| Vladimir Horowitz|
|[Felix Mendelssoh...]| Felix Mendelssohn|
|[Felix Mendelssoh...]| Vladimir Horowitz|
|[Franz Liszt, Vla...]| Franz Liszt|
|[Franz Liszt, Vla...]| Vladimir Horowitz|
|[Carl Woitschach]| Carl Woitschach|
|[Francisco Canaro...]| Francisco Canaro|
|[Francisco Canaro...]| Charlo|
|[Seweryn Goszczyń...]| Seweryn Goszczyński|
|[Sergei Rachmanin...]| Sergei Rachmaninoff|
|[Sergei Rachmanin...]| Vladimir Horowitz|
|[Frédéric Chopin,...]| Frédéric Chopin|
|[Frédéric Chopin,...]| Vladimir Horowitz|
|[Samuel Barber, V...]| Samuel Barber|
+-----+-----+
only showing top 20 rows
```

Explore Data

```
#Top 20 artist by number of song
from pyspark.sql.functions import desc
df_exploded.groupBy('artist').count().orderBy(desc('count')).show()
```

artist	count
Francisco Canaro	2234
Эрнест Хемингуэй	1215
Frédéric Chopin	1033
Ludwig van Beethoven	935
Wolfgang Amadeus ...	904
Johann Sebastian ...	838
Эрих Мария Ремарк	781
Frank Sinatra	732
Billie Holiday	680
Arturo Toscanini	628
Igor Stravinsky	623
Ignacio Corsini	620
Vladimir Horowitz	612
Johnny Cash	589
New York Philharm...	556
Bob Dylan	553
The Rolling Stones	522
The Beach Boys	503
Lata Mangeshkar	502
Elvis Presley	501

only showing top 20 rows

```
#Top 20 year with highest song
df.groupBy('year').count().orderBy(desc('count')).show()
```

year	count
1973	2000
1978	1999
1976	1999
2017	1999
1977	1998
2018	1998
1988	1998
1979	1998
2019	1998
1982	1998
1970	1998
1987	1998
2005	1998
1984	1998
2010	1998
1975	1997
2006	1997
1992	1997
1969	1997
1965	1997

only showing top 20 rows

```
# Top 20 years with least song
df.groupBy('year').count().orderBy('count').show()
```

year	count
1922	72
1921	128
1923	168
1924	237
1925	262
1932	478
1934	550
1938	576
1927	594
1931	595
1937	596
1933	622
1943	628
1944	769
1926	874
1929	924
1941	956
1939	999
1936	1046
1928	1182

only showing top 20 rows

Explore Data

```
# Number of artist
df_exploded.select("artist").distinct().count()
```

```
27389
```

```
# Average songs of each artist
from pyspark.sql.functions import count, avg
artist_song_counts = df_exploded.groupBy("artist").agg(count("id").alias("num_songs"))
artist_song_counts.agg(avg("num_songs")).show()
```

```
+-----+
| avg(num_songs) |
+-----+
| 8.036730074117346 |
+-----+
```

```
# Average songs over the year
year_song_counts = df.groupBy("year").agg(count("name").alias("num_songs"))
year_song_counts.agg(avg("num_songs")).show()
```

```
+-----+
| avg(num_songs) |
+-----+
| 1684.62 |
+-----+
```

```
# Total song
df.count()
```

```
168462
```

```
# Song release in recent years
df.groupBy('year').count().orderBy(desc('year')).show()
```

```
+-----+-----+
| year | count |
+-----+-----+
| 2020 | 1752 |
| 2019 | 1998 |
| 2018 | 1998 |
| 2017 | 1999 |
| 2016 | 1963 |
| 2015 | 1929 |
| 2014 | 1996 |
| 2013 | 1995 |
| 2012 | 1997 |
| 2011 | 1993 |
| 2010 | 1998 |
| 2009 | 1991 |
| 2008 | 1993 |
| 2007 | 1981 |
| 2006 | 1997 |
| 2005 | 1998 |
| 2004 | 1989 |
| 2003 | 1994 |
| 2002 | 1990 |
| 2001 | 1992 |
+-----+-----+
```

```
only showing top 20 rows
```


Regression

▼ Attribute Selection

```
[ ] df_select = df.select('year','acousticness','danceability','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','mode','key','explicit','duration_ms','popularity')
```

```
[ ] #Check correlation
features = df_select.columns[:-1]

for col in features:
    corr = df.stat.corr(col, "popularity")
    print(f"Correlation between {col} and popularity: {corr:.4f}")
```

```
↩ Correlation between year and popularity: 0.8801
Correlation between acousticness and popularity: -0.5897
Correlation between danceability and popularity: 0.2142
Correlation between energy and popularity: 0.4937
Correlation between instrumentalness and popularity: -0.3030
Correlation between liveness and popularity: -0.0739
Correlation between loudness and popularity: 0.4637
Correlation between speechiness and popularity: -0.1375
Correlation between tempo and popularity: 0.1318
Correlation between valence and popularity: -0.0011
Correlation between mode and popularity: -0.0325
Correlation between key and popularity: 0.0099
Correlation between explicit and popularity: 0.2127
Correlation between duration_ms and popularity: 0.0650
```

Regression

```
[ ] df_select = df.select('year','acousticness','danceability','energy','instrumentalness','loudness','tempo','explicit','popularity')
```

```
[ ] featureColumns = df_select.columns[:-1]
```

▶ featureColumns

```
[ ] ['year',  
    'acousticness',  
    'danceability',  
    'energy',  
    'instrumentalness',  
    'loudness',  
    'tempo',  
    'explicit']
```

```
[ ] (trainData, testData) = df_select.randomSplit([0.8,0.2], seed = 13234 )
```

```
[ ] assembler = VectorAssembler(inputCols=featureColumns, outputCol="features")  
    scaler = StandardScaler(inputCol = 'features',outputCol='scaledFeatures',withStd=True,withMean=False)
```

```
[ ] lr = LinearRegression(featuresCol="scaledFeatures", labelCol="popularity")
```


Regression

```
[ ] from pyspark.ml import Pipeline
    pipeline = Pipeline(stages=[assembler, scaler, lr])

[ ] from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
    paramGrid = ParamGridBuilder() \
        .addGrid(lr.fitIntercept, [False, True]) \
        .addGrid(lr.maxIter, [5, 10, 20]) \
        .build()

[ ] from pyspark.ml.evaluation import RegressionEvaluator
    crossval = CrossValidator(estimator=pipeline,
                             estimatorParamMaps=paramGrid,
                             evaluator=RegressionEvaluator(labelCol="popularity", metricName="rmse"),
                             numFolds=5)
    cvModel = crossval.fit(trainData)

[ ] predictions = cvModel.transform(testData)

[ ] rmse_evaluator = RegressionEvaluator(labelCol="popularity", metricName="rmse")
    r2_evaluator = RegressionEvaluator(labelCol="popularity", metricName="r2")
    rmse = rmse_evaluator.evaluate(predictions)
    r2 = r2_evaluator.evaluate(predictions)

    # Print the results
    print(f"Root Mean Squared Error (RMSE): {rmse}")
    print(f"R-squared (R2): {r2}")

⇒ Root Mean Squared Error (RMSE): 10.040384945080456
   R-squared (R2): 0.7813629999625085
```

Classification

Attribute Selection

```
[ ] df_select = df.select('year','acousticness','danceability','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','key','duration_ms','popularity','mode','explicit')
features = df_select.columns[:-1]

for col in features:
    corr = df.stat.corr(col, "explicit")
    print(f"Correlation between {col} and explicit: {corr:.4f}")
```

```
↔ Correlation between year and explicit: 0.2440
Correlation between acousticness and explicit: -0.2524
Correlation between danceability and explicit: 0.2410
Correlation between energy and explicit: 0.1410
Correlation between instrumentalness and explicit: -0.1387
Correlation between liveness and explicit: 0.0401
Correlation between loudness and explicit: 0.1513
Correlation between speechiness and explicit: 0.4141
Correlation between tempo and explicit: 0.0105
Correlation between valence and explicit: -0.0251
Correlation between key and explicit: 0.0086
Correlation between duration_ms and explicit: -0.0443
Correlation between popularity and explicit: 0.2127
Correlation between mode and explicit: -0.0832
```

Classification

```
[ ] df_select = df.select('year','acousticness','danceability','energy','instrumentalness','loudness','speechiness','popularity','explicit')
```

```
[ ] featureColumns = df_select.columns[:-1]
```

```
[ ] featureColumns
```

```
⇒ ['year',  
   'acousticness',  
   'danceability',  
   'energy',  
   'instrumentalness',  
   'loudness',  
   'speechiness',  
   'popularity']
```

```
[ ] (trainData, testData) = df_select.randomSplit([0.8,0.2], seed = 13234 )
```

```
[ ] assembler = VectorAssembler(inputCols=featureColumns, outputCol="features")  
   scaler = StandardScaler(inputCol = 'features',outputCol='scaledFeatures',withStd=True,withMean=False)
```

```
[ ] lr = LogisticRegression(featuresCol="scaledFeatures", labelCol="explicit")
```

Classification

```
[ ] from pyspark.ml import Pipeline
    pipeline = Pipeline(stages=[assembler, scaler, lr])
```

```
[ ] paramGrid = ParamGridBuilder() \
    .addGrid(lr.fitIntercept, [False, True]) \
    .addGrid(lr.maxIter, [5, 10, 20]) \
    .build()
```

```
[ ] from pyspark.ml.evaluation import MulticlassClassificationEvaluator
    crossval = CrossValidator(estimator=pipeline,
                              estimatorParamMaps=paramGrid,
                              evaluator = MulticlassClassificationEvaluator(labelCol="explicit", predictionCol="prediction", metricName="accuracy"),
                              numFolds=5)
    cvModel = crossval.fit(trainData)
```

```
[ ] predictions = cvModel.transform(testData)
```

Classification

```
[ ] def evaluate(result):  
    predictionAndLabels = result.select("prediction", "explicit")  
    metrics = ["f1", "precisionByLabel", "recallByLabel", "weightedPrecision", "weightedRecall", "accuracy"]  
    for m in metrics:  
        evaluator = MulticlassClassificationEvaluator(labelCol="explicit", predictionCol="prediction", metricName=m)  
        print(f"{m}: {evaluator.evaluate(predictionAndLabels):.4f}")
```

```
[ ] evaluate(predictions)
```

```
⇒ f1: 0.9334  
precisionByLabel: 0.9522  
recallByLabel: 0.9834  
weightedPrecision: 0.9325  
weightedRecall: 0.9395  
accuracy: 0.9395
```

Model Summary

พบว่าผลลัพธ์ที่ได้จาก Regression Model ในการทำนายค่า Popularity ของเพลงด้วยการใช้ค่าจาก year, acousticness, danceability, energy, instrumentalness, loudness, tempo, explicit, popularity โมเดลให้ค่าผลลัพธ์ค่า RMSE 10.04 และ R-squared 0.78 ซึ่งเป็นผลลัพธ์ที่ดีที่บ่งบอกว่าโมเดลมีประสิทธิภาพในการทำนายค่า Popularity ได้และจากผลลัพธ์ที่ได้จาก Classification Model ในการทำนายค่า Explicit ของเพลงด้วยการใช้ค่าจาก year, acousticness, danceability, energy, instrumentalness, loudness, speechiness, popularity, explicit พบว่าโมเดลให้ค่าผลลัพธ์ค่า Accuracy 0.9395, F1-score 0.9334, Precision (by label) 0.9522, Recall (by label) 0.9834, Weighted Precision 0.9325, และ Weighted Recall 0.9395 ซึ่งบ่งบอกว่าโมเดลมีประสิทธิภาพที่ดีในการ predict ข้อมูลทั้ง 2 class

Append Spotify Data

bigdata_demo_pro2

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Sources Enter Data Data source settings Manage Parameters Refresh Preview Properties Advanced Editor Choose Columns Remove Columns Keep Rows Remove Rows Sort Split Column Group By Data Type: Text Use First Row as Headers Replace Values Merge Queries Append Queries Combine Files Text Analytics Vision Azure Machine Learning AI Insights

Queries [6]

- part-00000-4f1fdade-08...
- part-00001-4f1fdade-08...
- Append_spotify
- part-00000-25f6ffa1-fc6...
- part-00001-25f6ffa1-fc6...
- Append_artist

Table.TransformColumnTypes(Source,{{"year", type text}})

id	name	artists	duration_ms
1	Singende Bataillone 1. Teil	['Carl Woitschach']	158648
2	Fantasiestücke, Op. 111: Più tosto lento	['Robert Schumann', 'Vladimir Horowitz']	282133
3	Chapter 1.18 - Zamek kaniowski	['Seweryn Goszczyński']	104300
4	Bebamos Juntos - Instrumental (Remasterizado)	['Francisco Canaro']	180760
5	Polonaise-Fantaisie in A-Flat Major, Op. 61	['Frédéric Chopin', 'Vladimir Horowitz']	687733
6	Scherzo a capriccio: Presto	['Felix Mendelssohn', 'Vladimir Horowitz']	352600
7	Valse oubliée No. 1 in F-Sharp Major, S. 215/1	['Franz Liszt', 'Vladimir Horowitz']	136627
8	Per aspera ad astra	['Carl Woitschach']	153967
9	Moneda Corriente - Remasterizado	['Francisco Canaro', 'Charlo']	162493
10	Chapter 1.3 - Zamek kaniowski	['Seweryn Goszczyński']	111600
11	Piano Sonata No. 2 in B-Flat Minor, Op. 36: I. Allegro agitato - Meno m...	['Sergei Rachmaninoff', 'Vladimir Horowitz']	590293
12	Piano Sonata No. 2, Op. 35: IV. Finale. Presto	['Frédéric Chopin', 'Vladimir Horowitz']	85133
13	Piano Sonata in E-Flat Minor, Op. 26: III. Adagio mesto	['Samuel Barber', 'Vladimir Horowitz']	338333
14	Nachtstücke, Op. 23: No. 4 in F	['Robert Schumann', 'Vladimir Horowitz']	167333
15	Symphony No. 5 in C Minor, Op. 67: 3. Allegro	['Ludwig van Beethoven', 'Staatskapelle Berlin', 'Richard Strauss']	276563
16	A Shropshire Lad: Is My Team Ploughing?	['George Butterworth', 'John Cameron']	184840
17	Sonata No. 3, Op. 23 in F-Sharp Minor: IV. Presto con fuoco; Meno mo...	['Alexander Scriabin', 'Vladimir Horowitz']	326067
18	Invocación al Tango - Remasterizado	['Francisco Canaro', 'Luis Scalón']	167107
19	Where the Bee Sucks	['Thomas Arne', 'John Heddle Nash']	122533
20	Tendrás Que Llorar Conmigo - Instrumental (Remasterizado)	['Francisco Canaro']	173707
21	Etude in A-Flat, Op. 72, No. 11	['Moritz Moszkowski', 'Vladimir Horowitz']	80800
22	Quisiste Cachar un Gil - Instrumental (Remasterizado)	['Francisco Canaro']	176000
23	Andante spianato in E-Flat Major, Op. 22	['Frédéric Chopin', 'Vladimir Horowitz']	245440
24	La Recova - Instrumental (Remasterizado)	['Francisco Canaro']	169720
25	El Espejito - Remasterizado	['Francisco Canaro', 'Charlo']	166000
26	It Was a Lover and His Lass	['Roger Quilter', 'John Heddle Nash']	152600
27	Kız Saçları	['Hafız Yaşar']	162197
28	Scherzo No. 1 in B Minor, Op. 20	['Frédéric Chopin', 'Vladimir Horowitz']	590293

19 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 4:55 PM

Query Settings

PROPERTIES

Name

Append_spotify

All Properties

APPLIED STEPS

Source

Changed Type

Append Artists Data

bigdata_demo_pro2

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Sources Enter Data Data source settings Manage Parameters Refresh Preview Properties Advanced Editor Manage Choose Columns Remove Columns Keep Rows Remove Rows Sort Split Column Group By Data Type: Text Use First Row as Headers Replace Values Merge Queries Append Queries Combine Files Text Analytics Vision Azure Machine Learning AI Insights

Queries [6]

- part-00000-4f1fdade-08...
- part-00001-4f1fdade-08...
- Append_spotify
- part-00000-25f6ffa1-fc6...
- part-00001-25f6ffa1-fc6...
- Append_artist

Table.Combine({#"part-00000-25f6ffa1-fc63-4dcb-9d73-5f2b53905a71-c000", #"part-00001-25f6ffa1-fc63-4dcb-9d73-5f2b53905a71-c000"})

	1.2 valence	1.2 mode	1.2 key	1.2 popularity	1.2 explicit	artist
1	118.469	0.779	1	10	0	Carl Woitschach
2	83.972	0.0767	1	8	0	Robert Schumann
3	83.972	0.0767	1	8	0	Vladimir Horowitz
4	107.177	0.88	0	5	0	Seweryn Goszczyński
5	108.003	0.72	0	1	0	Francisco Canaro
6	62.149	0.0693	1	11	1	Frédéric Chopin
7	62.149	0.0693	1	11	1	Vladimir Horowitz
8	63.521	0.266	0	6	0	Felix Mendelssohn
9	63.521	0.266	0	6	0	Vladimir Horowitz
10	80.495	0.305	1	11	0	Franz Liszt
11	80.495	0.305	1	11	0	Vladimir Horowitz
12	123.31	0.857	1	1	0	Carl Woitschach
13	119.833	0.493	0	9	0	Francisco Canaro
14	119.833	0.493	0	9	0	Charlo
15	81.249	0.759	1	9	0	Seweryn Goszczyński
16	141.39	0.0393	0	10	0	Sergei Rachmaninoff
17	141.39	0.0393	0	10	0	Vladimir Horowitz
18	85.989	0.346	0	10	1	Frédéric Chopin
19	85.989	0.346	0	10	1	Vladimir Horowitz
20	96.645	0.042	1	7	0	Samuel Barber
21	96.645	0.042	1	7	0	Vladimir Horowitz
22	78.98	0.216	1	5	0	Robert Schumann
23	78.98	0.216	1	5	0	Vladimir Horowitz
24	80.204	0.406	0	5	0	Ludwig van Beethoven
25	80.204	0.406	0	5	0	Staatskapelle Berlin
26	80.204	0.406	0	5	0	Richard Strauss
27	79.831	0.169	0	7	0	George Butterworth
28	79.831	0.169	0	7	0	John Cameron

20 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 4:46 PM

Query Settings

PROPERTIES

Name

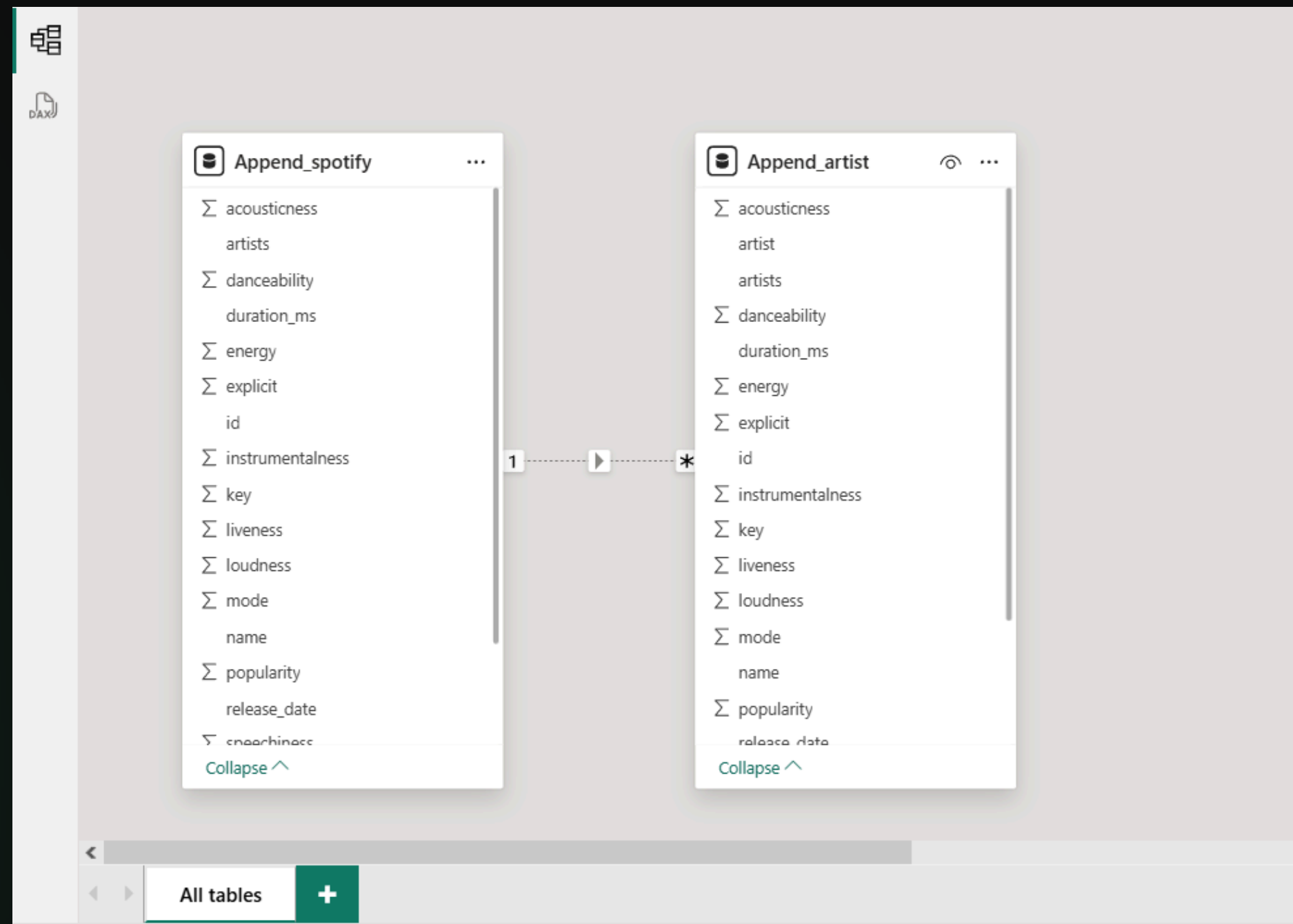
Append_artist

All Properties

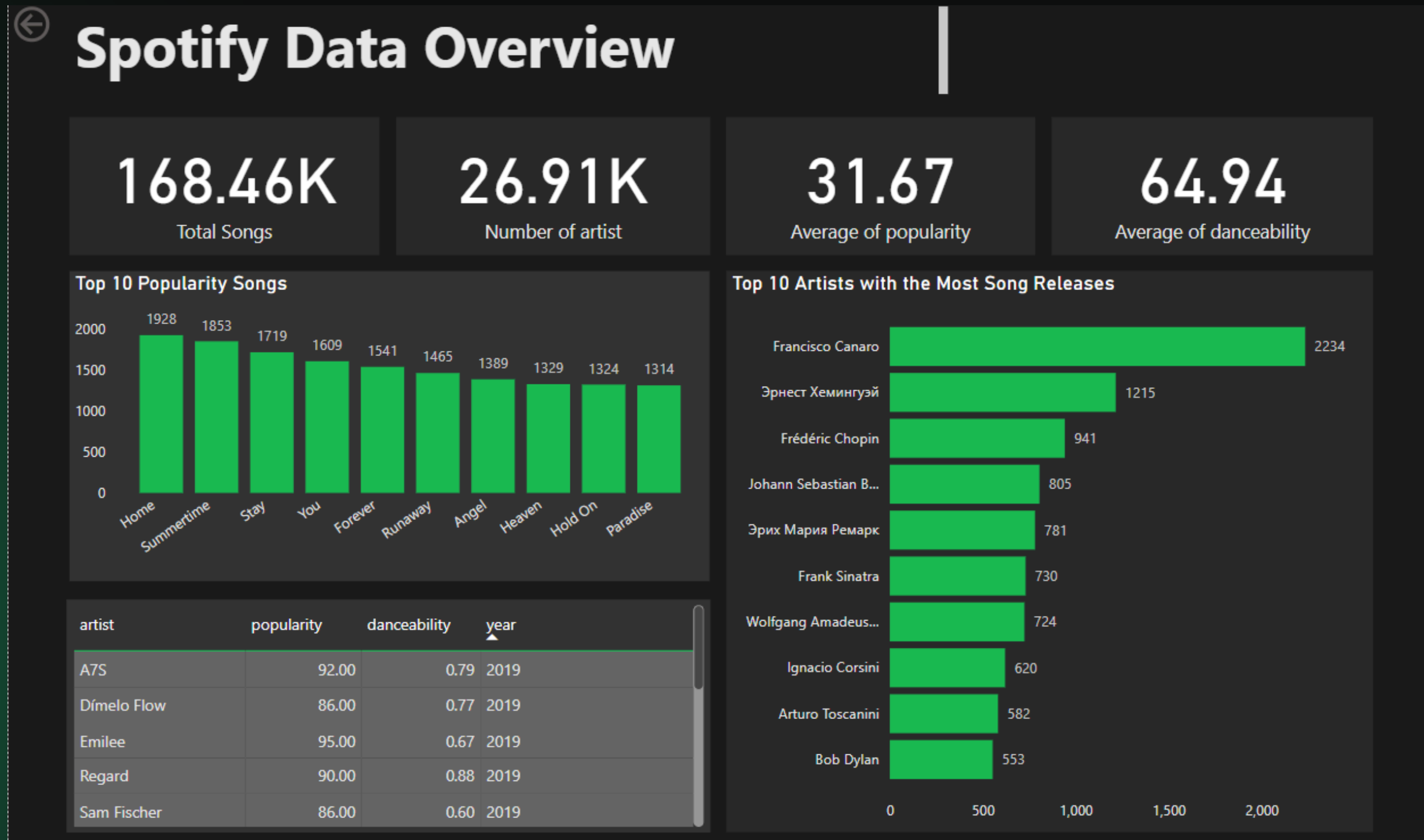
APPLIED STEPS

Source

Visualization



Visualization by Power BI



Team members

6513114 Nipatsa Chainiwattana

6513134 Puthipong Yomabut

6513170 Patiharn Kamenkit

6513172 Phattaradanai Sornsawang

Thank You

