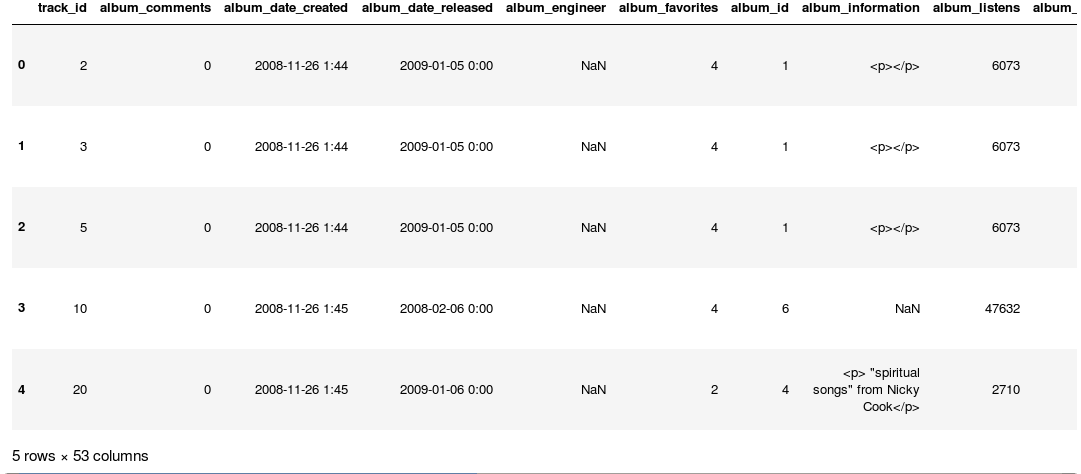
**Problem definition**

* Use a music track dataset to perform several statistics in Music industry
* To understand which track’s feature(s) contribute in its ranking
* Challenges:
  + Many empty columns and rows
  + Price not included
  + Non-linear data
  + Broad range of values for several features such as: interest, quality, listens…

**Dataset description**

1. music\_tracks.csv:

* Includes more than 100,000 audio tracks names with almost 50 different features.
* These features include information about artist, track’s specifications, album, genre…
* The following specific features are considered for this project:
  + Track\_bit\_rate (track quality)
  + Track\_listens (number of times this track has been listened)
  + Track\_interest (number of times this track has been favourite)
  + Track\_date (date of track or album released)
  + genre
  + Artist



1. Genres.csv

* Used to extract genres title.

**Solution**

* Clean up and remove empty columns and rows
* Delete tracks with very low quality
* ranking and quality are six digits, so calculated ratios for each one to be used in later evaluation and calculations
  1. track\_bit\_rate coefficient (Track Quality)

tracks['bit\_rate\_factor'] = np.round(tracks['track\_bit\_rate']/max\_bit\_rate, decimals=2)

* 1. track\_interest coefficient (Ranking)

# 1: Top Chart (Interest factor > 0.09) x =1 - more than 290,000 interests

# 0.8: between 230,000 and 290,000

# 0.7: between 165,000 and 230,000

# 0.6: between 98,000 and 165,000

# 0.5: between 65,000 and 98,000

# 0.3: between 30,000 and 65,000

# 0.2: below 30,000 would be low interest

**tracks['interest\_factor'] = np.round(tracks['track\_interest'] / max\_interest\_rate , decimals = 4) \* 100**

* Divide some features to few categories since they have broad range
  1. **tracks['interest\_factor'] = tracks['interest\_factor'].apply(lambda x: 1 if x >= 0.09 else 0.8 if 0.07 <= x < 0.09 else 0.7 if 0.05 <= x < 0.07 else 0.6 if 0.03 <= x < 0.02 else 0.5 if 0.02 <= x < 0.03 else 0.3 if 0.01<= x <= 0.02 else 0.2 if 0.001<= x <= 0.01 else 0.1)**
  2. **tracks['track\_length'] = pd.to\_numeric(tracks['track\_duration']).apply(lambda x: 'less than 3 minutes' if x < 180 else 'between 3 and 5 minutes' if 180<= x < 300 else 'above 7 minutes')**
* Calculate price based on ranking and quality factor

**highest\_track\_price = 4.99**

**tracks['track\_price'] = np.round(highest\_track\_price \* tracks['bit\_rate\_factor']\*tracks['interest\_factor'], decimals=2)**

**Results**

1. **Classification using the original data columns**

First classification was run considering the following features as X columns:

* album\_listens
* track\_bit\_rate
* artist\_id
* track\_listens

and track\_interest (Ranking) as target

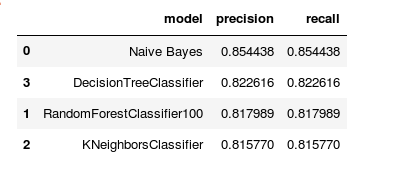
**Issues**:

* model runs very slow
* Random Forest regression did not run due to low memory
* Very low accuracy (accuracy score : 0.02)
* Achieved highest accuracy of 0.18 by using only Track\_year and Track\_listens

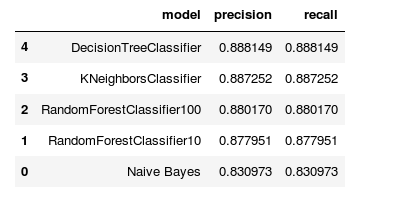
|  |  |
| --- | --- |
| **Features** | **Accuracy Score** |
| album\_listens,track\_bit\_rate,artist\_id,track\_listens | 0.02 % |
| track\_bit\_rate , track\_listens | 0.05 % |
| Track\_listens | 0.13 % |
| track\_listens , track\_year\_created | 0.18 % |
| Track\_Bit\_rate | 0.0 % |

1. **Classification using ratios**

* All model run very fast
* High accuracy
* Different classifications:
  + Accuracy achieved by using only track\_listens



* Track\_listens and Track\_year\_created



* Track\_listens, track\_year\_created, artist\_id, bit\_rate\_factor

