Music Preference Prediction Using Random Forest Model

This dataset was made by our external consultant Alexander Makhratchev. He found about 500 songs he liked and 500 songs he disliked, and downloaded information about them through the Spotify API. There are 16 attribute columns, such as song name, danceability, and speechiness. You can see a sample of the dataset below.

In [1]:

```
%%bash
head spotify.csv
```

```
,artist,album,track_name,track_id,danceability,energy,key,loudness,mode,speechiness,instr
umentalness,liveness,valence,tempo,duration_ms,time_signature,like
0,Tiësto,BLUE (Remixes),BLUE - Mike Williams Remix,10WrTQMhZu2gocF8UB6obr,0.644,0.9209999
999999999,11,-3.201,1,0.045,0.000371,0.355,0.5770000000000001,128.015,191733,4,True
1,RICCI,Whistle,Whistle,0s7TF4xqdNcXn8U8cWXrhC,0.7120000000000001,0.934,2,-4.769,1,0.0595
,6.22e-06,0.0542,0.53,120.024,196030,4,True
2,Harris & Ford,"Freitag, Samstag","Freitag, Samstag",35WEFAhw47XLju1gu40cjT,0.638,0.961,
10,-3.8280000000000003,0,0.121,0.0055899999999999995,0.0663,0.353,138.034,156356,4,True
3,ILLENIUM,Awake,Feel Good,0e0UxWGgjXoYAYUFhJgwji,0.625,0.7070000000000001,2,-4.761,1,0.0
337,0.0,0.213,0.479,138.064,248156,4,True
4,TJR,Bounce Generation,Bounce Generation - Radio Edit,3l3wjXneWieRL0yKd4Tihf,0.687,0.998
,2,-1.304,1,0.29100000000000004,0.0431,0.32899999999999996,0.129,128.007,152813,4,True
5,Rush,Moving Pictures (2011 Remaster),Tom Sawyer,3QZ7uX97s82HFYSmQUAN1D,0.536,0.90099999
99999999,9,-7.211,1,0.0374,0.0186,0.06,0.6659999999999999,87.559,276880,4,False
6,Tina Turner,Tina!,River Deep - Mountain High,19jo0UT2vqD4pNVfIqTy4R,0.621,0.972,8,-3.79
10000000000004,1,0.0724,0.000585,0.195,0.866,155.113,244160,4,False
7,Drake,Dark Lane Demo Tapes,Toosie Slide,466cKvZn1j45IpxDdYZqdA,0.83,0.49,1,-8.82,0,0.20
9,3.04e-06,0.113,0.845,81.604,247059,4,True
8,Axel Rudi Pell,Wings of the Storm,Wings of the Storm,0oPS4seBoSMmz0M1OnL0l1,0.492000000
00000005,0.836,8,-6.922999999999999,0,0.0356,0.00043,0.128,0.384,82.07600000000001,347720
,4,False
```

In this notebook our goal is to simply demonstrate the Random Forest model. Using the -f RF command we can specify the exact predictor we would like to use.

In [8]:

```
! btc spotify.csv -f RF --yes
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:              Alexander Makhratchev  (Evaluation)
Expiration Date:          2021-04-30   56 days left
Number of Threads:        1
Maximum File Size:        30 GB
Maximum Instances:        unlimited
Maximum Attributes:       unlimited
Maximum Classes:          unlimited
Connected to:             daimensions.brainome.ai  (local execution)



Command:
    btc spotify.csv -f RF --yes

Start Time:                 03/05/2021, 17:32


Data:
    Input:                      spotify.csv
    Target Column:              like
    Number of instances:        1224
```

```
    Number of attributes:       17
    Number of classes:          2
    Class Balance:              0: 43.38%, 1: 56.62%

Learnability:
    Best guess accuracy:         56.62%
    Data Sufficiency:            Not enough data to generalize. [red]

Capacity Progression:            at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:           6,   7,   8,   9,   9,  10


Estimated Memory Equivalent Capacity for...
    Decision Tree:                   607 parameters
    Neural Networks:                   1 parameters
    Random Forest:                    84 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:               100.95%
    Neural Networks:               1.60%
    Random Forest:                29.27%

Expected Generalization using...
    Decision Tree:                 1.99 bits/bit
    Neural Network:              318.00 bits/bit
    Random Forest:                14.57 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type RF given by user.


System Meter:                           a.py
    Classifier Type:                    Random Forest
    System Type:                        Binary classifier
    Training/Validation Split:            50% : 50%
    Accuracy:
        Best-guess accuracy:               56.61%
        Training accuracy:                100.00% (612/612 correct)
        Validation Accuracy:               84.96% (520/612 correct)
        Overall Model Accuracy:            92.48% (1132/1224 correct)
        Improvement over best guess:       35.87% of possible  43.39%

    Model Capacity (MEC):                     13 bits
    Generalization Ratio:                  46.65 bits/bit
    Model Efficiency:                       2.75 /parameter
    Generalization Index:                  23.21
    Percent of Data Memorized:              4.31%

    Training Confusion Matrix (count):
                True |   272     0
               False |     0   340

    Validation Confusion Matrix (count):
                True |   220    39
               False |    53   300

    Full Confusion Matrix (count):
                True |   492    39
               False |    53   640

    Accuracy by Class:
               class |    TP    FP    TN    FN     TPR      TNR      PPV      NPV
F1       TS
                True |   492    39   640    53   90.28%   92.35%   92.66%   92.35%   91.
45%   84.25%
               False |   640    53   492    39   94.26%   92.66%   92.35%   92.66%   93.
29%   87.43%
```

```
End Time:
Runtime Duration:

Messages:
Warning: Remapped class labels to be contiguous. Use -cm if DET/ROC-based accuracy measur
ements are wrong.
```

From the measurements, we can see that the random forest predictor has better generalization and memory equvalent capacity than the decision tree. However, it is still fairly far off from the neural network. The accuracy of the random forest predictor on the validation set is 84.96%, which is about a 30% improvement on best guess.

Using Attribute Ranking

Now we will use the -rank command in order to select the attributes that are most correlated to the target class. This will allow us to find the needle in the haystack.

In [6]:

```
! btc spotify.csv -f RF -rank --yes
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:               Alexander Makhratchev  (Evaluation)
Expiration Date:           2021-04-30   56 days left
Number of Threads:         1
Maximum File Size:         30 GB
Maximum Instances:         unlimited
Maximum Attributes:        unlimited
Maximum Classes:           unlimited
Connected to:              daimensions.brainome.ai  (local execution)



Command:
    btc spotify.csv -f RF -rank --yes

Start Time:                 03/05/2021, 16:56

Attribute Ranking:
    Important columns:          artist, , loudness
    Overfit risk:                   0.0%
    Ignoring columns:           album, track_name, track_id, danceability, energy, key, m
ode, speechiness, instrumentalness, liveness, valence, tempo, duration_ms, time_signature
    Test Accuracy Progression:     artist :   98.28%
                                    :  100.00%  change   +1.72%
                                   loudness :  100.00%  change   +0.00%




Data:
    Input:                      spotify.csv
    Target Column:              like
    Number of instances:        1224
    Number of attributes:       3
    Number of classes:          2
    Class Balance:              0: 43.38%, 1: 56.62%

Learnability:
    Best guess accuracy:          56.62%
    Data Sufficiency:            Maybe enough data to generalize. [yellow]

Capacity Progression:            at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:           6,   7,   8,   8,   9,   9
```

```
Estimated Memory Equivalent Capacity for...
    Decision Tree:                  333 parameters
    Neural Networks:                 13 parameters
    Random Forest:                  218 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                55.38%
    Neural Networks:             100.00%
    Random Forest:                76.22%

Expected Generalization using...
    Decision Tree:                 3.63 bits/bit
    Neural Network:               20.92 bits/bit
    Random Forest:                 5.61 bits/bit


Recommendations:


    Note: Machine learner type RF given by user.


System Meter:                           a.py
    Classifier Type:                    Random Forest
    System Type:                        Binary classifier
    Training/Validation Split:            60% : 40%
    Accuracy:
        Best-guess accuracy:               56.61%
        Training accuracy:                100.00% (734/734 correct)
        Validation Accuracy:               66.73% (327/490 correct)
        Overall Model Accuracy:            86.68% (1061/1224 correct)
        Improvement over best guess:       30.07% of possible  43.39%

    Model Capacity (MEC):                     15 bits
    Generalization Ratio:                  48.42 bits/bit
    Model Efficiency:                       2.00 /parameter
    Generalization Index:                  24.09
    Percent of Data Memorized:              4.15%

    Training Confusion Matrix (count):
                True |   323     0
               False |     0   411

    Validation Confusion Matrix (count):
                True |   126    82
               False |    81   201

    Full Confusion Matrix (count):
                True |   449    82
               False |    81   612

    Accuracy by Class:
               class |    TP    FP    TN    FN     TPR      TNR      PPV      NPV
F1       TS
                True |   449    82   612    81   84.72%   88.31%   84.56%   88.31%   84.
64%   73.37%
               False |   612    81   449    82   88.18%   84.56%   88.31%   84.56%   88.
25%   78.97%

End Time:
Runtime Duration:

Messages:
Warning: Remapped class labels to be contiguous. Use -cm if DET/ROC-based accuracy measur
ements are wrong.
```

The two most important columns that were selected by attribute ranking were artist and loudness. Surprisingly, the -rank command lowered out validation accuracy to 66.73%. This might suggest that the target class is dependent on many more columns, and a neural network might be most effective here as indicated by the measurements.

Using -ignorecolumns

One of the attributes that the BTC identified and was important to the target class was the artist. However, we do not want our predictor to use that column, so we can utilize the -ignorecolumns command in order.

In [3]:

```
! btc spotify.csv -f RF --yes -ignorecolumns artist
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Table Compiler 0.99
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   53 days left
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)

Command:
    btc spotify.csv -f RF --yes -ignorecolumns artist

Start Time:                 03/08/2021, 23:31 UTC

Data:
    Input:                     spotify.csv
    Target Column:             like
    Number of instances:       1224
    Number of attributes:        16
    Number of classes:            2
    Class Balance:                0: 43.38%, 1: 56.62%

Learnability:
    Best guess accuracy:          56.62%
    Data Sufficiency:             Maybe enough data to generalize. [yellow]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              6,   7,   8,   9,  10,  10

Estimated Memory Equivalent Capacity for...
    Decision Tree:               603 parameters
    Neural Networks:               1 parameters
    Random Forest:                97 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:               100.29%
    Neural Networks:               1.26%
    Random Forest:                32.55%

Expected Generalization using...
    Decision Tree:                 2.00 bits/bit
    Neural Network:              337.00 bits/bit
    Random Forest:                12.62 bits/bit


Recommendations:
    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type RF given by user.

System Meter:                     a.py
    Classifier Type:              Random Forest
    System Type:                  Binary classifier
    Training/Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:        56.61%
      Training accuracy:         100.00% (612/612 correct)
```

```
      Validation Accuracy:          84.31% (516/612 correct)
      Overall Model Accuracy:       92.15% (1128/1224 correct)

    Architecture Capacity (MEC):  12    bits
    Generalization Ratio:         50.54 bits/bit
    Architecture Efficiency:            /parameter
    Generalization Index:         25.15
    Percent of Data Memorized:     3.98%

    Training Confusion Matrix (count):
                True |   272     0
               False |     0   340

    Validation Confusion Matrix (count):
                True |   216    43
               False |    53   300

    Full Confusion Matrix (count):
                True |   488    43
               False |    53   640

    Accuracy by Class:
                class |    TP    FP    TN    FN     TPR      TNR      PPV      NPV
F1       TS
                 True |   488    43   640    53   90.20%   92.35%   91.90%   92.35%   91.
04%   83.56%
                False |   640    53   488    43   93.70%   91.90%   92.35%   91.90%   93.
02%   86.96%


Messages:
Warning: Remapped class labels to be contiguous. Use -cm if DET/ROC-based accuracy measur
ements are wrong.
End Time:           03/08/2021, 23:32 UTC
Runtime Duration:   25s
```

We get 84.31% accuracy on the validation set, which is almost identical to the first time we ran the compiler on the dataset. Also, our accuracy is similar across classes, beacuse the dataset is balanced.