Artificially-Created Data Prediction Using Daimensions

This dataset was artificially created with a specific rule in mind. The goal of this notebook is to show how Daimensions handles data created by a specified rule. Bertrand, the cofounder of Brainome, made this dataset, so the csv's are named after him.

In [5]:

```
%%bash
head bertrandtrain.csv
```

```
1,1,0,0,0,0,0,0,0,1,1
1,1,1,0,0,1,0,1,0,1,0
0,1,0,1,0,1,0,0,0,1,0
0,0,1,1,0,1,0,1,0,1,1
0,0,1,0,0,0,1,0,1,1,1
0,0,1,1,1,1,0,0,1,1,0
0,0,1,1,1,1,0,0,0,0,0
0,0,1,0,0,1,1,0,1,1,1
1,1,1,0,1,1,1,1,0,0,1
0,0,0,0,1,0,1,0,0,1,1
```

As you can see from above, this data doesn't have column names. Because of this, we have to use -headerless when measuring our data and building our model.

1. Get Measurements

We always want to measure our data before building our predictor in order to ensure we are building the right model. For more information about how to use Daimensions and why we want to measure our data beforehand, check out the Titanic notebook.

In [6]:

```
! btc -measureonly bertrandtrain.csv -headerless
```

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

Brainome Table Compiler 0.991
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                  Alexander Makhratchev  (Evaluation)
Expiration Date:              2021-04-30   45 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 -measureonly bertrandtrain.csv -headerless

Start Time:                 03/16/2021, 22:01 UTC




Pre-training Measurements
Data:
    Input:                      bertrandtrain.csv (headerless csv)
    Target Column:              target
    Number of instances:      13187
    Number of attributes:        10
    Number of classes:            2
```

```
Class Balance:
                              0: 37.35%
                              1: 62.65%

Learnability:
    Best guess accuracy:          62.65%
    Data Sufficiency:             Not enough data to generalize. [red]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              9,  10,  11,  12,  12,  13



Expected Generalization:
    Decision Tree:              7952.45 bits/bit
    Neural Network:             6593.00 bits/bit
    Random Forest:             13187.00 bits/bit

Expected Accuracy               Training           Validation
    Decision Tree:                63.26%               63.25%
    Neural Network:              100.00%              100.00%
    Random Forest:               100.00%              100.00%

Recommendations:


Time to Build Estimates:
    Decision Tree:                less than a minute    Neural Network:
3 minutes


End Time:           03/16/2021, 22:01 UTC
Runtime Duration:   30s
```

2. Build the Predictor

Based on our measurements, Daimensions recommends we use a neural network, which has 83.99 bits/bit of expected generalization for this dataset. Don't forget to use -headerless.

In [7]:

```
! btc -f NN bertrandtrain.csv -o bertrand_predict.py -headerless -e 10 --yes
```

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

Brainome Table Compiler 0.991
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   45 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 -f NN bertrandtrain.csv -o bertrand_predict.py -headerless -e 10 --yes

Start Time:                03/16/2021, 22:01 UTC




Pre-training Measurements
Data:
    Input:                     bertrandtrain.csv (headerless csv)
    Target Column:             target
    Number of instances:      13187
    Number of attributes:        10
```

```
    Number of classes:            2

Class Balance:
                              0: 37.35%
                              1: 62.65%

Learnability:
    Best guess accuracy:          62.65%
    Data Sufficiency:             Not enough data to generalize. [red]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              9,  10,  11,  12,  12,  13



Expected Generalization:
    Decision Tree:              7952.45 bits/bit
    Neural Network:             6593.00 bits/bit
    Random Forest:             13187.00 bits/bit

Expected Accuracy               Training           Validation
    Decision Tree:                63.26%               63.25%
    Neural Network:              100.00%              100.00%
    Random Forest:               100.00%              100.00%

Recommendations:
    Note: Model type NN given by user.

Time to Build Estimates:
    Neural Network:                2 minutes
```

Predictor: bertrand_predict.py

```
    Classifier Type:              Neural Network
    System Type:                  Binary classifier
    Training / Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:        62.65%
      Training accuracy:         100.00% (6593/6593 correct)
      Validation Accuracy:       100.00% (6594/6594 correct)
      Combined Model Accuracy:   100.00% (13187/13187 correct)

    Model Capacity (MEC):         49    bits
    Model Capacity Utilized:       1    bits
    Generalization Ratio:        128.06 bits/bit
    Generalization Index:         62.87
    Percent of Data Memorized:     1.59%

    Training Confusion Matrix:
              Actual | Predicted
                   0 |   2449      0
                   1 |      0   4144

    Validation Confusion Matrix:
              Actual | Predicted
                   0 |   2476      0
                   1 |      0   4118

    Combined Confusion Matrix:
              Actual | Predicted
                   0 |   4925      0
                   1 |      0   8262

    Training Accuracy by Class:
               class |     TP     FP     TN     FN     TPR      TNR      PPV      NPV
F1       TS
                   0 |   2449      0   4144      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%
```

```
100.00%  100.00%
                1 |   4144      0   2449      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%

    Validation Accuracy by Class:
             class |     TP     FP     TN     FN     TPR      TNR      PPV      NPV
F1       TS
                0 |   2476      0   4118      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%
                1 |   4118      0   2476      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%

    Combined Accuracy by Class:
             class |     TP     FP     TN     FN     TPR      TNR      PPV      NPV
F1       TS
                0 |   4925      0   8262      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%
                1 |   8262      0   4925      0  100.00%  100.00%  100.00%  100.00%
100.00%  100.00%



End Time:           03/16/2021, 22:03 UTC
Runtime Duration:   1m 16s
```

3. Make a Prediction

Hooray! Our model has 100% accuracy. Now we can use our model to make predictions on test data, a separate set of data that wasn't used for training.

In [8]:

```
! python3 bertrand_predict.py bertrandtest.csv > bertrand_prediction.csv
! head bertrand_prediction.csv
```

```
0,0,1,1,0,1,1,0,0,0,Prediction
1,0,0,0,1,0,0,1,1,0,0
0,0,1,1,1,0,0,1,0,0,0
1,0,0,0,1,1,0,0,1,1,0
1,1,0,1,1,0,1,1,1,1,1
0,0,0,0,0,1,1,0,0,1,0
0,0,0,0,1,1,1,0,1,1,1
0,1,0,0,0,0,1,0,1,0,1
0,1,1,1,1,1,0,0,1,0,0
0,1,0,1,0,0,0,1,0,1,0
```