

Bertrand

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 [4]: ! 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 [2]: ! btc -measureonly bertrandtrain.csv -headerless
```

Brainome Daimensions(tm) 0.97 Copyright (c) 2019, 2020 by Brainome, Inc. All Rights Reserved.

Data:

Number of instances: 13187

Number of attributes: 10

Number of classes: 2

Class balance: 37.35% 62.65%

Learnability:

Best guess accuracy: 62.65%

Capacity progression (# of decision points): [13, 14, 15, 15, 16, 16]

Decision Tree: 5993 parameters

Estimated Memory Equivalent Capacity for Neural Networks: 157 parameters

Risk that model needs to overfit for 100% accuracy...

using Decision Tree: 90.91%

using Neural Networks: 100.00%

Expected Generalization...

using Decision Tree: 2.20 bits/bit

using a Neural Network: 83.99 bits/bit

Recommendations:

Note: Maybe enough data to generalize. [yellow]

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
```

Brainome Daimensions(tm) 0.97 Copyright (c) 2019, 2020 by Brainome, Inc. All Rights Reserved.

Time estimate for a Neural Network:

Estimated time to architect: 0d 0h 0m 14s

Estimated time to prime (subject to change after model architecting): 0d 0h 3m 23s

Note: Machine learner type NN given by user.

Model capacity (MEC): 49 bits

Architecture efficiency: 1.0 bits/parameter

Estimated time to prime model: 0d 0h 2m 30s

Priming model... done.

Estimated training time: 0d 0h 19m 34s

Training... done.

Classifier Type: Neural Network

System Type: Binary classifier

Best-guess accuracy: 62.65%

Model accuracy: 100.00% (13187/13187 correct)

Improvement over best guess: 37.35% (of possible 37.35%)

Model capacity (MEC): 49 bits

Generalization ratio: 269.12 bits/bit

Model efficiency:	0.76%/parameter
System behavior	
True Negatives:	37.35% (4925/13187)
True Positives:	62.65% (8262/13187)
False Negatives:	0.00% (0/13187)
False Positives:	0.00% (0/13187)
True Pos. Rate/Sensitivity/Recall:	1.00
True Neg. Rate/Specificity:	1.00
Precision:	1.00
F-1 Measure:	1.00
False Negative Rate/Miss Rate:	0.00
Critical Success Index:	1.00
Overfitting:	No

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