Titanic Using Daimensions

This notebook uses data from the Titanic competition on Kaggle (https://www.kaggle.com/c/titanic/overview).

Kaggle's description of the competition: "The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered 'unsinkable' RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. In this challenge, we ask you to build a predictive model that answers the question: 'what sorts of people were more likely to survive?' using passenger data (ie name, age, gender, socio-economic class, etc)."

Goal: Make a predictor of survival from Titanic training data. We'll do this by using Daimensions to measure, build, and validate a predictor.

0. Getting Started

Because this is the very first tutorial, we'll go over how to install btc and get started. You can also see how to setup btc in the Daimensions Quickstart guide.

First, use the following link to download the installation script: https://download.brainome.net/btc-cli/btc-setup.sh. From the download directory, run the following bash command.

In [ ]:

```
! sh btc-setup.sh
```

The script will check that your operating system is supported, download the latest btc client to your machine and install it in /usr/local/bin. You will be prompted to enter the administrator password to install the software. *NOTE: After installation, make sure that "/usr/local/bin" is in your search path.*

Next, run the following command to wipe all cloud files. You will need your user credentials to login to DaimensionsTM. The first time you login, your license key will be downloaded automatically. Please use the default password that was provided to you.

In [ ]:

```
! btc WIPE
```

To change your password, use the following bash command.

In [ ]:

```
! btc CHPASSWD
```

1. Get Measurements

Measuring our data before building a predictor is important in order to avoid mistakes and optimize our model. If we don't measure our data, we have no way of knowing whether the predictor we build will actually do what we want it to do when it sees new data that it wasn't trained on. We'll probably build a model that is much larger than it needs to be, meaning our training and run times will probably be much longer than they need to be. We could end up in a situation where we just don't know whether we have the right amount or right type of training data, even after extensive training and testing. Because of these reasons, it's best to measure our data beforehand. Not to mention, Daimensions will tell us about learnability, the generalization ratio, noise resilience, and all the standard accuracy and confusion figures. For more information, you can read the Daimensions How-to Guide and Glossary.

In [8]:

In [8]:

```
# Below is a clip of the training data:
! head titanic_train.csv
# For Windows command prompt:
# type titanic_train.csv | more
```

As you can see from above, the target column (Survived) isn't the last column on the right. Because of this, we need to use '-target' so that Daimensions is looking at the correct target column for measuring and building a predictor.

In [9]:

```
# Measuring the training data:
! btc -measureonly titanic_train.csv -target Survived
```

```
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 -measureonly titanic_train.csv -target Survived

Start Time:                    03/05/2021, 17:44


Data:
    Input:                          titanic_train.csv
    Target Column:                  Survived
    Number of instances:            891
    Number of attributes:           11
    Number of classes:              2
    Class Balance:                  0: 61.62%, 1: 38.38%

Learnability:
    Best guess accuracy:              61.62%
    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:                       419 parameters
    Neural Networks:                      39 parameters
    Random Forest:                       106 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                    99.42%
    Neural Networks:                 100.00%
    Random Forest:                    59.89%
```

```
Expected Generalization using...
    Decision Tree:                2.04 bits/bit
    Neural Network:               8.10 bits/bit
    Random Forest:                8.41 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.



Time to Build Estimates:
    Decision Tree:           a few seconds
    Neural Network:            3 minutes


[+] Building 0.0s (0/1)
[+] Building 0.1s (1/1)
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
[+] Building 0.3s (1/2)
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
[+] Building 0.4s (2/2)
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
 => [internal] load .dockerignore                                         0.1s
 => => transferring context: 2B                                           0.0s
[+] Building 0.5s (4/7)
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
 => [internal] load .dockerignore                                         0.1s
 => => transferring context: 2B                                           0.0s
 => [internal] load metadata for docker.io/brainome/btc_local_cpu:alpha   0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 1.37kB                                       0.0s
 => [1/3] FROM docker.io/brainome/btc_local_cpu:alpha                     0.0s
[+] Building 0.6s (7/8)
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
 => [internal] load .dockerignore                                         0.1s
 => => transferring context: 2B                                           0.0s
 => [internal] load metadata for docker.io/brainome/btc_local_cpu:alpha   0.0s
 => [internal] load build context                                         0.1s
 => => transferring context: 1.37kB                                       0.0s
 => [1/3] FROM docker.io/brainome/btc_local_cpu:alpha                     0.0s
 => CACHED [2/3] RUN adduser --disabled-password --gecos '' --uid 501 --g 0.0s
 => CACHED [3/3] COPY --chown=501:20 .daimensions.key /btc-alex           0.0s
 => exporting to image                                                    0.1s
 => => exporting layers                                                   0.0s
 => => writing image sha256:43dd56f18fcffcdd02c290bfcce87c7a6cd0b9ccf4db3 0.0s
[+] Building 0.7s (8/8) FINISHED
 => [internal] load build definition from btc-dockerfile.12070            0.0s
 => => transferring dockerfile: 239B                                      0.0s
 => [internal] load .dockerignore                                         0.1s
 => => transferring context: 2B                                           0.0s
 => [internal] load metadata for docker.io/brainome/btc_local_cpu:alpha   0.0s
 => [internal] load build context                                         0.1s
 => => transferring context: 1.37kB                                       0.0s
 => [1/3] FROM docker.io/brainome/btc_local_cpu:alpha                     0.0s
 => CACHED [2/3] RUN adduser --disabled-password --gecos '' --uid 501 --g 0.0s
 => CACHED [3/3] COPY --chown=501:20 .daimensions.key /btc-alex           0.0s
 => exporting to image                                                    0.1s
 => => exporting layers                                                   0.0s
 => => writing image sha256:43dd56f18fcffcdd02c290bfcce87c7a6cd0b9ccf4db3 0.0s
 => => naming to docker.io/library/btc-alex:latest                        0.0s
Docker image btc-alex:latest updated successfully.
```

2. Build the Predictor

Because the learnability of the data (based on capacity progression and risk) is yellow, the how-to guide recommends to choose predictor with higher generalization and increase effort for best results. This means using a neural network with effort should work best. Here, I'm using '-f NN' to make the predictor a neural network. I'm also using '-o predict.py' to output the predictor as a python file. To increase the effort, I'm using '-e 10' for 10 times the effort. Again, we have to use '-target Survived' because the target column isn't the last one.

In [13]:

```
# Building the predictor and outputting it to 'titanic_predict.py':
! btc -v -v -f NN titanic_train.csv -o titanic_predict.py -target Survived --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 -v -v -f NN titanic_train.csv -o titanic_predict.py -target Survived --yes

Start Time:                 03/05/2021, 17:48







Data:
    Input:                        titanic_train.csv
    Target Column:                Survived
    Number of instances:          891
    Number of attributes:         11
    Number of classes:            2
    Class Balance:                0: 61.62%, 1: 38.38%

Learnability:
    Best guess accuracy:            61.62%
    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:                    419 parameters
    Neural Networks:                   39 parameters
    Random Forest:                    106 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                 99.42%
    Neural Networks:              100.00%
    Random Forest:                 59.89%

Expected Generalization using...
    Decision Tree:                  2.04 bits/bit
    Neural Network:                 8.10 bits/bit
    Random Forest:                  8.41 bits/bit
```

```
Recommendations:

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


Time to Build Estimates:

    Neural Network:              2 minutes




Model created:
Sequential(
  (0): Linear(in_features=11, out_features=2, bias=True)
  (1): ReLU()
  (2): Linear(in_features=2, out_features=1, bias=True)
)



System Meter:                             titanic_predict.py
    Classifier Type:                      Neural Network
    System Type:                          Binary classifier
    Training/Validation Split:              60% : 40%
    Accuracy:
        Best-guess accuracy:                  61.61%
        Training accuracy:                    63.67% (340/534 correct)
        Validation Accuracy:                  58.54% (209/357 correct)
        Overall Model Accuracy:               61.61% (549/891 correct)
        Improvement over best guess:           0.00% of possible  38.39%

    Model Capacity (MEC):                        28 bits
    Generalization Ratio:                      11.48 bits/bit
    Model Efficiency:                           0.00 /parameter
    Generalization Index:                       5.65
    Percent of Data Memorized:                 17.70%

    Training Confusion Matrix (count):
                   0 |  340    0
                   1 |  194    0

    Validation Confusion Matrix (count):
                   0 |  209    0
                   1 |  148    0

    Full Confusion Matrix (count):
                   0 |  549    0
                   1 |  342    0

    Accuracy by Class:
               class |   TP   FP   TN   FN     TPR      TNR      PPV      NPV       F1
TS
                   0 |  549    0    0  342   61.62%    0.00%  100.00%    0.00%   76.25%
61.62%
                   1 |    0  342  549    0     nan%  100.00%    0.00%  100.00%    0.00%
0.00%

End Time:
Runtime Duration:

Output: titanic_predict.py
READY.
```

3. Validate and Make Predictions

We've built our first predictor! Now it's time to put it to use. In the case of Titanic, we are given test data from

Kaggle, where it's different from the training data and doesn't include 'Survival'. We can use the model we built to make predictions for the test data and submit it to Kaggle for its competition. In the following code, I'll save the model's prediction in 'titanic_prediction.csv'. You will see that the predictor appended the model's prediction of survival as the last column.

In [14]:

```
# Using predictor on test data and saving it to 'titanic_prediction.csv':
! python3 titanic_predict.py titanic_test.csv > titanic_prediction.csv
! head titanic_prediction.csv
```

```
PassengerId,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked,Prediction
892,3,"Kelly, Mr. James",male,34.5,0,0,330911,7.8292,,Q,0
893,3,"Wilkes, Mrs. James (Ellen Needs)",female,47,1,0,363272,7,,S,0
894,2,"Myles, Mr. Thomas Francis",male,62,0,0,240276,9.6875,,Q,0
895,3,"Wirz, Mr. Albert",male,27,0,0,315154,8.6625,,S,0
896,3,"Hirvonen, Mrs. Alexander (Helga E Lindqvist)",female,22,1,1,3101298,12.2875,,S,0
897,3,"Svensson, Mr. Johan Cervin",male,14,0,0,7538,9.225,,S,0
898,3,"Connolly, Miss. Kate",female,30,0,0,330972,7.6292,,Q,0
899,2,"Caldwell, Mr. Albert Francis",male,26,1,1,248738,29,,S,0
900,3,"Abrahim, Mrs. Joseph (Sophie Halaut Easu)",female,18,0,0,2657,7.2292,,C,0
```

If you have validation data, or data that has the target column but wasn't used for training, you can use it to validate the accuracy of your predictor, as we will do. For this particular instance, I found an annotated version of the Titanic test data, 'titanic_validation.csv', and used it to validate our model.

In [15]:

```
# To validate:
! python3 titanic_predict.py -validate titanic_validation.csv
```

```
Classifier Type:                    Neural Network
System Type:                        Binary classifier
Best-guess accuracy:                62.20%
Model accuracy:                     62.20% (260/418 correct)
Improvement over best guess:        0.00% (of possible 37.8%)
Model capacity (MEC):               28 bits
Generalization ratio:               8.88 bits/bit
Model efficiency:                   0.00%/parameter
System behavior
True Negatives:                     62.20% (260/418)
True Positives:                     0.00% (0/418)
False Negatives:                    37.80% (158/418)
False Positives:                    0.00% (0/418)
True Pos. Rate/Sensitivity/Recall:  0.00
True Neg. Rate/Specificity:         1.00
F-1 Measure:                        0.00
False Negative Rate/Miss Rate:      1.00
Critical Success Index:             0.00
Confusion Matrix:
 [62.20% 0.00%]
 [37.80% 0.00%]
```

From validating the predictor, we can see that it has 74.64% accuracy, 12.44% better than best-guess accuracy (which classifies all data points as the majority class).

4. Improving Our Model

Our model did pretty well, but let's see if we can improve it. A column that contains a unique value in each row (for example a database key) will never contribute to generalization, so we shouldn't include database keys or other unique ID columns. We can remove these columns by using '-ignorecolumns'. We'll try ignoring columns: PassengerId, Name, Ticket, Cabin, Embarked, because they're all unique ID columns. We could also use '-rank' to rank columns by significance and only process contributing attributes.

Ignorecolumns vs Rank:

There may be situations where domain knowledge suggests a better choice of features than -rank. If we know the data generative process, we can do better with -ignorecolumns than with -rank. Rank is also optimizing for quick clustering/decision tree. For neural networks, we may still wish to reduce input features, which can be done with pca, but at the cost of interpretability. Some applications may require the original features are used in which case pca isn't viable. Ignorecolumns can reduce features while maintaining interpretability and work better for neural networks than -rank may, but the burden of choosing the right columns to keep is now on us.

Using -ignorecolumns:

In [17]:

```
# Using -ignorecolumns to make a better predictor:
! btc -v -v -f NN titanic_train.csv -o titanic_predict_igcol.py -target Survived -ignore
columns PassengerId,Name,Ticket,Cabin,Embarked -e 10 --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 -v -v -f NN titanic_train.csv -o titanic_predict_igcol.py -target Survived -ignor
ecolumns PassengerId,Name,Ticket,Cabin,Embarked -e 10 --yes

Start Time:                 03/05/2021, 17:56







Data:
    Input:                      titanic_train.csv
    Target Column:              Survived
    Number of instances:        891
    Number of attributes:       6
    Number of classes:          2
    Class Balance:              0: 61.62%, 1: 38.38%

Learnability:
    Best guess accuracy:          61.62%
    Data Sufficiency:            Not enough data to generalize. [red]

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


Estimated Memory Equivalent Capacity for...
    Decision Tree:                  224 parameters
    Neural Networks:                  1 parameters
    Random Forest:                  101 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                55.32%
    Neural Networks:               7.86%
    Random Forest:                56.11%

Expected Generalization using...
    Decision Tree:                 3.67 bits/bit
```

```
    Neural Network:               163.00 bits/bit
    Random Forest:                  8.82 bits/bit


Recommendations:


    Note: Machine learner type NN given by user.


Time to Build Estimates:

    Neural Network:              3 minutes






Model created:
Sequential(
  (0): Linear(in_features=6, out_features=2, bias=True)
  (1): ReLU()
  (2): Linear(in_features=2, out_features=1, bias=True)
)



System Meter:                              titanic_predict_igcol.py
    Classifier Type:                       Neural Network
    System Type:                           Binary classifier
    Training/Validation Split:               50% : 50%
    Accuracy:
        Best-guess accuracy:                   61.61%
        Training accuracy:                     78.87% (351/445 correct)
        Validation Accuracy:                   82.06% (366/446 correct)
        Overall Model Accuracy:                80.47% (717/891 correct)
        Improvement over best guess:           18.86% of possible  38.39%

    Model Capacity (MEC):                           1 bit
    Generalization Ratio:                      337.31 bits/bit
    Model Efficiency:                           18.86 /parameter
    Generalization Index:                      166.08
    Percent of Data Memorized:                   0.60%

    Training Confusion Matrix (count):
                   0 |  244   30
                   1 |   64  107

    Validation Confusion Matrix (count):
                   0 |  251   24
                   1 |   56  115

    Full Confusion Matrix (count):
                   0 |  495   54
                   1 |  120  222

    Accuracy by Class:
               class |   TP   FP   TN   FN     TPR      TNR      PPV      NPV       F1
TS
                   0 |  495   54  222  120   80.49%   64.91%   90.16%   64.91%   85.05%
73.99%
                   1 |  222  120  495   54   80.43%   90.16%   64.91%   90.16%   71.84%
56.06%

End Time:
Runtime Duration:

Output: titanic_predict_igcol.py
READY.
```

In [18]:

```
# Using the ignorecolumns predictor on test data and saving it to 'titanic_prediction_igcol.csv':
! python3 titanic_predict_igcol.py titanic_test.csv > titanic_prediction_igcol.csv
! head titanic_prediction_igcol.csv
```

```
PassengerId,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked,Prediction
892,3,"Kelly, Mr. James",male,34.5,0,0,330911,7.8292,,Q,0
893,3,"Wilkes, Mrs. James (Ellen Needs)",female,47,1,0,363272,7,,S,1
894,2,"Myles, Mr. Thomas Francis",male,62,0,0,240276,9.6875,,Q,0
895,3,"Wirz, Mr. Albert",male,27,0,0,315154,8.6625,,S,0
896,3,"Hirvonen, Mrs. Alexander (Helga E Lindqvist)",female,22,1,1,3101298,12.2875,,S,1
897,3,"Svensson, Mr. Johan Cervin",male,14,0,0,7538,9.225,,S,0
898,3,"Connolly, Miss. Kate",female,30,0,0,330972,7.6292,,Q,1
899,2,"Caldwell, Mr. Albert Francis",male,26,1,1,248738,29,,S,0
900,3,"Abrahim, Mrs. Joseph (Sophie Halaut Easu)",female,18,0,0,2657,7.2292,,C,1
```

As we wanted, -ignorecolumns removed the PassengerId, Name, Ticket, Cabin, and Embarked attributes. Next, we can use -validate to check the accuracy of our new predictor.

In [19]:

```
# Validating the -ignorecolumns predictor
! python3 titanic_predict_igcol.py -validate titanic_validation.csv
```

```
Classifier Type:                    Neural Network
System Type:                        Binary classifier
Best-guess accuracy:                62.20%
Model accuracy:                     77.27% (323/418 correct)
Improvement over best guess:        15.07% (of possible 37.8%)
Model capacity (MEC):               1 bits
Generalization ratio:               308.99 bits/bit
Model efficiency:                   15.06%/parameter
System behavior
True Negatives:                     52.87% (221/418)
True Positives:                     24.40% (102/418)
False Negatives:                    13.40% (56/418)
False Positives:                    9.33% (39/418)
True Pos. Rate/Sensitivity/Recall:  0.65
True Neg. Rate/Specificity:         0.85
Precision:                          0.72
F-1 Measure:                        0.68
False Negative Rate/Miss Rate:      0.35
Critical Success Index:             0.52
Confusion Matrix:
 [52.87% 9.33%]
 [13.40% 24.40%]
```

Using -ignorecolumns has improved our accuracy to 77.27% from 74.64% originally.

Using -rank:

In [20]:

```
# Using -rank to make a better predictor:
! btc -v -v -f NN titanic_train.csv -o titanic_predict_rank.py -target Survived -rank --yes -e 10
```

```
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 Reserved.
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 -v -v -f NN titanic_train.csv -o titanic_predict_rank.py -target Survived -rank -
-yes -e 10

Start Time:                 03/05/2021, 18:09





Attribute Ranking:
    Important columns:           Sex, SibSp, Parch, Pclass
    Overfit risk:                    0.0%
    Ignoring columns:            PassengerId, Name, Age, Ticket, Fare, Cabin, Embarked
    Test Accuracy Progression:      Sex :   78.68%
                                    SibSp :   79.69%  change   +1.01%
                                    Parch :   80.36%  change   +0.67%
                                    Pclass :   80.92%  change   +0.56%






Data:
    Input:                       titanic_train.csv
    Target Column:               Survived
    Number of instances:        891
    Number of attributes:       4
    Number of classes:          2
    Class Balance:               0: 61.62%, 1: 38.38%

Learnability:
    Best guess accuracy:          61.62%
    Data Sufficiency:            Maybe enough data to generalize. [yellow]

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


Estimated Memory Equivalent Capacity for...
    Decision Tree:                    1 parameters
    Neural Networks:                 16 parameters
    Random Forest:                   32 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                 0.29%
    Neural Networks:             100.00%
    Random Forest:                16.49%

Expected Generalization using...
    Decision Tree:                692.67 bits/bit
    Neural Network:                17.62 bits/bit
    Random Forest:                 27.84 bits/bit


Recommendations:


    Note: Machine learner type NN given by user.


Time to Build Estimates:

    Neural Network:              2 minutes
```

```
Model created:
Sequential(
  (0): Linear(in_features=4, out_features=4, bias=True)
  (1): ReLU()
  (2): Linear(in_features=4, out_features=1, bias=True)
)



System Meter:                          titanic_predict_rank.py
    Classifier Type:                   Neural Network
    System Type:                       Binary classifier
    Training/Validation Split:           50% : 50%
    Accuracy:
        Best-guess accuracy:             61.61%
        Training accuracy:               78.42% (349/445 correct)
        Validation Accuracy:             82.28% (367/446 correct)
        Overall Model Accuracy:          80.35% (716/891 correct)
        Improvement over best guess:     18.74% of possible  38.39%

    Model Capacity (MEC):                     1 bit
    Generalization Ratio:                335.39 bits/bit
    Model Efficiency:                     18.73 /parameter
    Generalization Index:                165.13
    Percent of Data Memorized:             0.61%

    Training Confusion Matrix (count):
                   0 |  254   20
                   1 |   76   95

    Validation Confusion Matrix (count):
                   0 |  259   16
                   1 |   63  108

    Full Confusion Matrix (count):
                   0 |  513   36
                   1 |  139  203

    Accuracy by Class:
              class |   TP   FP   TN   FN     TPR      TNR      PPV      NPV       F1
TS
                  0 |  513   36  203  139   78.68%   59.36%   93.44%   59.36%   85.43%
74.56%
                  1 |  203  139  513   36   84.94%   93.44%   59.36%   93.44%   69.88%
53.70%

End Time:
Runtime Duration:

Output: titanic_predict_rank.py
READY.
```

In [21]:

```
# Using the rank predictor on test data and saving it to 'titanic_prediction_rank.csv':
! python3 titanic_predict_rank.py titanic_test.csv > titanic_prediction_rank.csv
! head titanic_prediction_rank.csv
```

```
PassengerId,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked,Prediction
892,3,"Kelly, Mr. James",male,34.5,0,0,330911,7.8292,,Q,0
893,3,"Wilkes, Mrs. James (Ellen Needs)",female,47,1,0,363272,7,,S,0
894,2,"Myles, Mr. Thomas Francis",male,62,0,0,240276,9.6875,,Q,0
895,3,"Wirz, Mr. Albert",male,27,0,0,315154,8.6625,,S,0
896,3,"Hirvonen, Mrs. Alexander (Helga E Lindqvist)",female,22,1,1,3101298,12.2875,,S,0
897,3,"Svensson, Mr. Johan Cervin",male,14,0,0,7538,9.225,,S,0
898,3,"Connolly, Miss. Kate",female,30,0,0,330972,7.6292,,Q,1
899,2,"Caldwell, Mr. Albert Francis",male,26,1,1,248738,29,,S,0
```

```
900,3,"Abrahim, Mrs. Joseph (Sophie Halaut Easu)",female,18,0,0,2657,7.2292,,C,1
```

You can see that -rank decided to only look at the columns 'Sex','Parch' (Parent/child), and 'Fare'. This makes a lot of sense that the determining factors for survival on the Titanic were sex, how many parents or children they had on board, and how much their fare was. Seeing what attributes -rank chooses gives us powerful insight into understanding our data and its correlations.

In [22]:

```
# Validating the -rank predictor
! python3 titanic_predict_rank.py -validate titanic_validation.csv
```

```
Classifier Type:                    Neural Network
System Type:                        Binary classifier
Best-guess accuracy:                62.20%
Model accuracy:                     76.79% (321/418 correct)
Improvement over best guess:        14.59% (of possible 37.8%)
Model capacity (MEC):               1 bits
Generalization ratio:               307.07 bits/bit
Model efficiency:                   14.59%/parameter
System behavior
True Negatives:                     54.55% (228/418)
True Positives:                     22.25% (93/418)
False Negatives:                    15.55% (65/418)
False Positives:                    7.66% (32/418)
True Pos. Rate/Sensitivity/Recall:  0.59
True Neg. Rate/Specificity:         0.88
Precision:                          0.74
F-1 Measure:                        0.66
False Negative Rate/Miss Rate:      0.41
Critical Success Index:             0.49
Confusion Matrix:
 [54.55% 7.66%]
 [15.55% 22.25%]
```

With -rank, our accuracy is 76.79%, again, an improvement over our original 74.64%.

5. Next Steps

Success! We've built our first predictor and used it to make predictions on the Titanic test data. From here, we can use our model on any new Titanic data or use other control options to try to improve our results even more. To check out some of the other control options, use '-h' to see the full list. You can also check out Brainome's How-to Guide and Glossary for more information.

In [23]:

```
! btc -h
```

```
usage: btc [-h] [-hh] [-version] [-headerless] [-target TARGET]
           [-ignorecolumns IGNORECOLUMNS] [-rank [ATTRIBUTERANK]]
           [-measureonly] [-f FORCEMODEL] [-nosplit] [-nsamples NSAMPLES]
           [-ignoreclasses IGNORELABELS] [-biasmeter]
           [-usecolumns IMPORTANTCOLUMNS] [-o OUTPUT] [-v] [-q] [-y]
           [-e EFFORT] [-balance] [-nopriming] [-novalidation] [-O OPTIMIZE]
           [-nofun] [--forcemodel FORCEMODEL]
           input [input ...]

Brainome Table Compiler

Required arguments:
  input                 Table as CSV files and/or URLs.
                        Alternatively, one of: {TERMINATE, WIPE, CHPASSWD, LOGOUT}
                        TERMINATE: Terminate all cloud processes.
                        WIPE: Delete all files in the cloud.
                        CHPASSWD: Change password.
                        LOGOUT: Force relogin.

Optional arguments:
```

```
  -h                    show this help message and exit
  -hh                   show advanced help message
  -version              show program's version number and exit

CSV input file details:
  -headerless           Headerless CSV input file.
  -target TARGET        Specify target column by name or number. Default: last column of
table.
  -ignorecolumns IGNORECOLUMNS
                        Comma-separated list of columns to ignore by name or number.

Basic options:
  -rank [ATTRIBUTERANK]
                        Select the optimal subset of columns for accuracy on held out dat
a
                        If optional parameter N is given, select the optimal N columns.
Works best for DT.
  -measureonly          Only output measurements, no predictor is built.
  -f FORCEMODEL         Force model type: DT, NN, RF  Default: RF
  -nosplit              Use all of the data for training. Default: dataset is split betwe
en training and validation.

Intermediate options:
  -nsamples NSAMPLES    Train only on a subset of N random samples of the dataset. Defaul
t: entire dataset.
  -ignoreclasses IGNORELABELS
                        Comma-separated list of classes to ignore.
  -biasmeter            Measure model bias
  -usecolumns IMPORTANTCOLUMNS
                        Comma-separated list of columns by name or number used to build t
he predictor.
  -o OUTPUT, --output OUTPUT
                        Predictor filename. Default: a.py
  -v                    Verbose output
  -q                    Quiet operation.
  -y                    Answers yes to all overwrite questions.

Advanced options:
  -e EFFORT, --effort EFFORT
                        Increase compute time to improve accuracy. 1=<effort<100. Default
: 1
  -balance              Treat classes as if they were balanced (only active for NN).
  -nopriming            Do not prime the model.
  -novalidation         Do not measure validation scores for created predictor.
  -O OPTIMIZE, --optimize OPTIMIZE
                        Maximize true positives towards a single class.
  -nofun                Stop compilation if there are warnings.

  --forcemodel FORCEMODEL
                        Force model type: DT, NN, RF  Default: RF

Examples:
---------
Measure and build a predictor for titanic
 btc https://download.brainome.net/data/public/titanic_train.csv

Build a better predictor by ignoring columns:
 btc titanic_train.csv -ignorecolumns "PassengerId,Name" -target Survived

Automatically select the important columns by using ranking:
 btc titanic_train.csv -rank -target Survived

Measure headerless dataset:
 btc https://download.brainome.net/data/public/bank.csv -headerless -measureonly

Full documentation can be found at https://www.brainome.ai/documentation/howto
```

In [ ]: