Titanic

Titanic Using Daimensions

This notebook uses data from the Titanic competition on Kaggle (link).
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.

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 [1]: # Below is a clip of the training data:
        ! head titanic_train.csv
```

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
```

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 [2]: # Measuring the training data:
        ! ./btc_linux -measureonly titanic_train.csv -target Survived
```

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

Data:
Number of instances: 891
Number of attributes: 11
Number of classes: 2
Class balance: 61.5% 38.5%

Learnability:
Best guess accuracy: 61.50% /* Classifies everything as whichever class is the
    majority */
Capacity progression (# of decision points): [8, 9, 10, 10, 11, 11] /* Takes
    increasing sizes of sampled data and counts number of threshold changes, should
     be convergent */
Decision Tree: 422 parameters /* Expected that DT will branch 422 times */
Estimated Memory Equivalent Capacity for Neural Networks: 118 parameters /* Based
    on estimated number of neurons used */

Risk that model needs to overfit for 100% accuracy...
using Decision Tree: 94.73%
using Neural Networks: 100.00%

Expected Generalization...
using Decision Tree: 2.11 bits/bit
using a Neural Network: 7.55 bits/bit /* Has the higher generalization */

Recommendations:
Note: Maybe enough data to generalize. [yellow]

Time estimate for a Neural Network:
Estimated time to architect: 0d 0h 0m 1s
Estimated time to prime (subject to change after model architecting): 0d 0h 3m 58s

Time estimate for Decision Tree:
Estimated time to prime a decision tree: a few seconds
```

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 [3]: # Building the predictor and outputting it to 'titanic_predict.py':
        ! ./btc_linux -v -v -f NN titanic_train.csv -o titanic_predict.py  -target
         ↪  Survived -e 10 --yes
```

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

Input: titanic_train.csv
Note: Machine learner type NN given by user.

Model capacity (MEC):   27 bits
Architecture efficiency: 1.0 bits/parameter

Classifier Type:                 Neural Network
System Type:                     Binary classifier
Best-guess accuracy:             61.61%
Model accuracy:                  81.59% (727/891 correct)
Improvement over best guess:     19.98% (of possible 38.39\%)
Model capacity (MEC):            27 bits
Generalization ratio:            26.92 bits/bit
Model efficiency:                0.74%/parameter
System behavior
True Negatives:                  54.32% (484/891)
True Positives:                  27.27% (243/891)
False Negatives:                 11.11% (99/891)
False Positives:                 7.30% (65/891)
True Pos. Rate/Sensitivity/Recall: 0.71
True Neg. Rate/Specificity:      0.88
Precision:                       0.79
F-1 Measure:                     0.75
False Negative Rate/Miss Rate:   0.29
Critical Success Index:          0.60
Overfitting:                     No

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 [4]: # 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,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
```

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 [9]: # 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:                     74.64% (312/418 correct)
Improvement over best guess:        12.44% (of possible 37.8%)
Model capacity (MEC):               27 bits
Generalization ratio:               11.55 bits/bit
Model efficiency:                   0.46%/parameter
System behavior
True Negatives:                     54.31% (227/418)
True Positives:                     20.33% (85/418)
False Negatives:                    17.46% (73/418)
False Positives:                    7.89% (33/418)
True Pos. Rate/Sensitivity/Recall: 0.54
True Neg. Rate/Specificity:         0.87
Precision:                          0.72
F-1 Measure:                        0.62
False Negative Rate/Miss Rate:      0.46
Critical Success Index:             0.45
```

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 [5]: # Using -ignorecolumns to make a better predictor:
        ! ./btc_linux -v -v -f NN titanic_train.csv -o titanic_predict_igcol.py
         ↪  -target Survived -ignorecolumns PassengerId,Name,Ticket,Cabin,Embarked
         ↪  -e 10
```

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

Input: titanic_train.csv

Learnability:
Best guess accuracy: 61.62%
Capacity progression (# of decision points): [7, 9, 10, 11, 11, 13]
Decision Tree: 285 parameters
Estimated Memory Equivalent Capacity for Neural Networks: 73 parameters

Risk that model needs to overfit for 100% accuracy...
using Decision Tree: 63.97%
using Neural Networks: 100.00%

Expected Generalization...
using Decision Tree: 3.13 bits/bit
using a Neural Network: 12.21 bits/bit

Classifier Type:                Neural Network
System Type:                    Binary classifier
Best-guess accuracy:            61.61%
```

```
Model accuracy:                 81.48% (726/891 correct)
Improvement over best guess:    19.87% (of possible 38.39%)
Model capacity (MEC):           17 bits
Generalization ratio:           42.70 bits/bit
Model efficiency:               1.16%/parameter
System behavior
True Negatives:                 58.02% (517/891)
True Positives:                 23.46% (209/891)
False Negatives:                14.93% (133/891)
False Positives:                3.59% (32/891)
Overfitting:                    No

Output: titanic_predict_igcol.py
READY.
```

```
In [7]: # Using the ignorecolumns predictor on test data:
        ! python3 titanic_predict_igcol.py titanic_test.csv > titanic_prediction_igcol.csv
        ! head titanic_prediction_igcol.csv
```

```
Pclass,Sex,Age,SibSp,Parch,Fare,Prediction
3,male,34.5,0,0,7.8292,0
3,female,47,1,0,7,0
2,male,62,0,0,9.6875,0
3,male,27,0,0,8.6625,0
3,female,22,1,1,12.2875,0
3,male,14,0,0,9.225,0
3,female,30,0,0,7.6292,0
2,male,26,1,1,29,0
3,female,18,0,0,7.2292,1
```

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 [2]: # Validating the -ignorecolumns predictor
        ! python3 titanic_predict_igcol.py -validate titanic_validation.csv
```

```
Model accuracy:                 77.75% (325/418 correct)
Improvement over best guess:    15.55% (of possible 37.8%)
Model capacity (MEC):           17 bits
```

Using -ignorecolumns has improved our accuracy to 77.75% from 74.64% originally.

Using -rank:

```
In [6]: # Using -rank to make a better predictor:
        ! ./btc_linux -v -v -f NN titanic_train.csv -o titanic_predict_rank.py
         ↪  -target Survived -rank --yes -e 10
```

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

Input: titanic_train.csv

Attribute Ranking:
Using only the important columns: Sex, Fare, Parch

Learnability:
Best guess accuracy: 61.50%
Capacity progression (# of decision points): [11, 11, 12, 12, 14, 14]
Decision Tree: 226 parameters
Estimated Memory Equivalent Capacity for Neural Networks: 41 parameters

Risk that model needs to overfit for 100% accuracy...
using Decision Tree: 50.73%
using Neural Networks: 89.13%

Expected Generalization...
using Decision Tree: 3.94 bits/bit
using a Neural Network: 21.73 bits/bit

Classifier Type:                Neural Network
System Type:                    Binary classifier
Best-guess accuracy:            61.61%
Model accuracy:                 79.34% (707/891 correct)
Improvement over best guess:    17.73% (of possible 38.39%)
Model capacity (MEC):           16 bits
Generalization ratio:           44.18 bits/bit
Model efficiency:               1.10%/parameter
System behavior
True Negatives:                 53.09% (473/891)
True Positives:                 26.26% (234/891)
False Negatives:                12.12% (108/891)
False Positives:                8.53% (76/891)
Overfitting:                    No

Output: titanic_predict_rank.py
READY.
```

```
In [8]: # Using the rank predictor on test data:
        ! python3 titanic_predict_rank.py titanic_test.csv > titanic_prediction_rank.csv
        ! head titanic_prediction_rank.csv
```

```
Sex,Parch,Fare,Prediction
male,0,7.8292,0
female,0,7,1
male,0,9.6875,0
```

```
male,0,8.6625,0
female,1,12.2875,1
male,0,9.225,0
female,0,7.6292,1
male,1,29,0
female,0,7.2292,1
```

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 [3]: # Validating the -rank predictor
        ! python3 titanic_predict_rank.py -validate titanic_validation.csv
```

```
Model accuracy:                    76.79% (321/418 correct)
Improvement over best guess:       14.59% (of possible 37.8%)
Model capacity (MEC):              16 bits
Generalization ratio:              20.06 bits/bit
```

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 [1]: ! ./btc_linux -h
```

```
usage: btc [-h] [-o [OUTPUT]] [-headerless] [-cm CLASSMAPPING] [-nc NCLASSES]
           [-l LANGUAGE] [-target TARGET] [-nsamples NSAMPLES]
           [-ignorecolumns IGNORECOLUMNS] [-ignorelabels IGNORELABELS]
           [-rank [ATTRIBUTERANK]] [-v] [--quiet] [-biasmeter] [-measureonly]
           [-Wall] [-pedantic] [-nofun] [-f FORCEMODEL] [-O OPTIMIZE]
           [-e EFFORT] [--yes] [-stopat STOPAT] [-modelonly] [-riskoverfit]
           [-nopriming] [-novalidation]
           input [input ...]

Brainome Daimensions(tm) Table Compiler
```