Plotting Titanic Prediction

This notebook is a tutorial of how we can modify the predictor python script to show a visualization of the data in the hidden space and how the predictor classifies each point. Each dimension of the hidden space corresponds to a neuron from the predictor. Because only two neurons are used for the titanic prediction model, we can plot it in the hidden space. ***Note: This isn't part of Daimensions. We are going to manually edit the python script to output a png of the model's results in the hidden space.***

1. Build the Predictor

In [5]:

```
! btc -f NN titanic_train.csv -o vis_titanic.py -target Survived -rank -e 5
```

```
Brainome Daimensions(tm) 0.97 Copyright (c) 2019, 2020 by Brainome, Inc. All Rights Reser
ved.
Licensed to: Ariana Park
Expiration date: 2020-11-30 (98 days left)
Number of threads: 1
Maximum file size: 4GB
Connected to: https://beta.brainome.ai:8080

Running btc will overwrite existing vis_titanic.py. OK? [y/N] yes
Ranking attributes...done.

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

Data:
Number of instances: 891
Number of attributes: 3
Number of classes: 2
Class balance: 61.62% 38.38%

Learnability:
Best guess accuracy: 61.62%
Capacity progression (# of decision points): [11, 11, 12, 14, 14, 14]
Decision Tree: 225 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

Recommendations:
Note: Enough data to generalize. [green]
Warning: Cannot find numpy. The output predictor may not run on this machine.

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 14s
Note: Machine learner type NN given by user.
Model capacity (MEC):     21 bits
Architecture efficiency:   1.0 bits/parameter

Estimated time to prime model: 0d 0h 2m 21s

Priming model...done.
Estimated training time: 0d 0h 6m 60s

Training...done.
Classifier Type:                      Neural Network
```

```
Classifier Type:                        Neural Network
System Type:                            Binary classifier
Best-guess accuracy:                    61.61%
Model accuracy:                         78.78% (702/891 correct)
Improvement over best guess:            17.17% (of possible 38.39%)
Model capacity (MEC):                   11 bits
Generalization ratio:                   63.81 bits/bit
Model efficiency:                       1.56%/parameter
System behavior
True Negatives:                         52.53% (468/891)
True Positives:                         26.26% (234/891)
False Negatives:                        12.12% (108/891)
False Positives:                        9.09% (81/891)
True Pos. Rate/Sensitivity/Recall:      0.68
True Neg. Rate/Specificity:             0.85
Precision:                              0.74
F-1 Measure:                            0.71
False Negative Rate/Miss Rate:          0.32
Critical Success Index:                 0.55
Overfitting:                            No
```

2. Make Changes to Python Script

The previous line of code should output a python script 'vis_titanic.py' with the model's predictor. Next, we'll manually edit the code by adding a few lines so that it will output the desired png when we run -validate. I'll show you the section of code we'll change with the modified lines highlighted.

In [ ]:

```
# Helper (save an import)
def argmax(l):
    f = lambda i: l[i]
    return max(range(len(l)), key=f)
# Classifier
def single_classify(row, ax):
    #inits
    x = row
    o = [0] * num_output_logits


    #Nueron Equations
    h_0 = max((((-0.23418278 * float(x[0]))+ (0.2055268 * float(x[1]))+ (-3.683856 * flo
at(x[2]))) + -0.5669335), 0)
    h_1 = max((((3.1762295 * float(x[0]))+ (0.29285425 * float(x[1]))+ (-0.0113988025 *
float(x[2]))) + 1.4681039), 0)
    o[0] = (0.6990876 * h_0)+ (-0.89847755 * h_1) + 2.2160573

    #visualization modification
    ax.scatter(h_0, h_1, marker=('o' if o[0] >= 0 else '^'), c=('r' if o[0] >= 0 else 'b
'))


    #Output Decision Rule
    if num_output_logits==1:
        return o[0]>=0
    else:
        return argmax(o)


def classify(arr):
    #init
    w_h = np.array([[-0.2341827750205993 7, 0.2055267989635467 5, -3.683856010437011 7], [3
.176229476928711, 0.2928542494773865, -0.011398802511394024]])
    b_h = np.array([-0.5669335126876831, 1.4681038856506348])
    w_o = np.array([[0.6990876197814941, -0.8984775543212891]])
    b_o = np.array(2.216057300567627)

    #Hidden Layer
    h = np.dot(arr, w_h.T) + b_h
```

```
    relu = np.maximum(h, np.zeros_like(h))


    #Output
    out = np.dot(relu, w_o.T) + b_o
    if num_output_logits == 1:
        return (out >= 0).astype('int').reshape(-1)
    else:
        return (np.argmax(out, axis=1)).reshape(-1)



def Predict(arr,headerless,csvfile, get_key, classmapping):

    with open(csvfile, 'r') as csvinput:
        #readers and writers
        writer = csv.writer(sys.stdout, lineterminator=os.linesep)
        reader = csv.reader(csvinput)

        #print original header
        if (not headerless):
            writer.writerow(','.join(next(reader, None) + ["Prediction"]))


        #visualization modification
        import matplotlib.pyplot as plt
        fig = plt.figure()
        ax = fig.add_subplot(111)

        for i, row in enumerate(reader):
            #use the transformed array as input to predictor
            pred = str(get_key(int(single_classify(arr[i],ax)), classmapping))
            #use original untransformed line to write out
            row.append(pred)
            writer.writerow(row)

        #visualization modification
        ax.set_xlabel('h0')
        ax.set_ylabel('h1')
        plt.savefig('titanic_visual.png')
```

3. Run -validate

You can make this modification in your own python script most easily by replacing lines 359-422 with the code above. Below I'm using an already modified version of the script called 'vis_titanic_mod.py', but you would use your edited 'vis_titanic.py'. We simply have to run the following line of code and it will output our desired png.

In [18]:

```
! python3 vis_titanic_mod.py titanic_test.csv
```

```
S,e,x,",",P,a,r,c,h,",",F,a,r,e,",",P,r,e,d,i,c,t,i,o,n
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
male,0,24.15,0
male,0,7.8958,0
male,0,26,0
female,0,82.2667,1
male,0,26,0
female,0,61.175,1
female,0,27.7208,1
male,0,12.35,0
```

```
male,0,7.225,0
female,0,7.925,1
female,0,7.225,1
male,0,59.4,0
male,1,3.1708,0
female,0,31.6833,1
male,1,61.3792,0
female,3,262.375,1
male,0,14.5,0
female,1,61.9792,1
male,0,7.225,0
male,0,30.5,0
male,0,21.6792,0
male,0,26,0
male,0,31.5,0
female,2,20.575,1
female,2,23.45,1
male,0,57.75,0
male,0,7.2292,0
female,0,8.05,1
female,0,8.6625,1
male,0,9.5,0
male,0,56.4958,0
male,1,13.4167,0
male,0,26.55,0
male,0,7.85,0
female,0,13,1
female,0,52.5542,1
male,0,7.925,0
male,0,29.7,0
male,0,7.75,0
female,0,76.2917,1
female,2,15.9,1
male,0,60,0
male,0,15.0333,0
female,1,23,1
female,2,263,1
male,0,15.5792,0
male,1,29.125,0
male,0,7.8958,0
male,0,7.65,0
male,0,16.1,0
female,0,262.375,1
male,0,7.8958,0
male,0,13.5,0
male,0,7.75,0
female,0,7.725,1
male,2,262.375,1
female,0,21,1
female,0,7.8792,1
male,0,42.4,0
male,0,28.5375,0
female,4,263,1
female,0,7.75,1
male,0,7.8958,0
female,0,7.925,1
male,0,27.7208,0
female,0,211.5,1
male,0,211.5,1
male,0,8.05,0
female,0,25.7,1
male,0,13,0
female,0,7.75,1
male,1,15.2458,0
male,0,221.7792,1
male,0,26,0
male,0,7.8958,0
male,0,10.7083,0
male,0,14.4542,0
female,0,7.8792,1
female,0,8.05,1
female,0,7.75,1
```

```
male,1,23,0
female,0,13.9,1
male,0,7.775,0
female,2,52,1
male,0,8.05,0
male,0,26,0
male,0,7.7958,0
female,0,78.85,1
male,0,7.925,0
female,0,7.8542,1
male,0,8.05,0
female,0,55.4417,1
male,0,26,0
male,0,7.75,0
male,0,7.775,0
female,1,8.5167,1
male,0,22.525,0
male,0,7.8208,0
male,0,7.75,0
male,0,8.7125,0
male,0,13,0
male,0,15.0458,0
female,0,7.7792,1
female,0,31.6792,1
female,0,7.2833,1
female,0,221.7792,1
male,0,14.4542,0
male,0,6.4375,0
female,1,16.7,1
male,0,75.2417,0
female,0,26,1
female,0,15.75,1
male,0,7.75,0
female,0,57.75,1
male,0,7.25,0
male,0,7.75,0
female,1,16.1,1
male,0,7.7958,0
female,0,23.25,1
male,0,13,0
male,0,8.05,0
male,0,8.05,0
male,0,28.5,0
female,4,25.4667,1
male,0,6.4375,0
male,0,7.8958,0
male,0,7.8542,0
male,0,7.225,0
male,0,13,0
female,0,8.05,1
male,6,46.9,0
female,2,46.9,1
female,0,151.55,1
male,3,262.375,0
male,0,26,0
male,0,26.55,0
male,0,18,0
male,0,51.8625,0
male,0,8.05,0
male,0,26.55,0
male,1,26,0
female,1,83.1583,1
male,0,7.8958,0
male,0,,0
female,2,12.1833,1
male,2,31.3875,0
male,0,7.55,0
female,0,221.7792,1
female,0,7.8542,1
male,0,26.55,0
female,2,13.775,1
female,0,7.7333,1
```

male,1,15.2458,0
female,0,13.5,1
male,0,7,0
male,0,13,0
female,1,22.025,1
male,0,50.4958,0
male,2,34.375,0
female,0,27.7208,1
female,0,8.9625,1
male,0,7.55,0
male,0,7.225,0
male,0,13.9,0
male,0,7.2292,0
male,5,31.3875,0
female,2,39,1
female,0,36.75,1
male,0,55.4417,0
female,3,39,1
female,2,83.1583,1
male,0,13,0
male,1,83.1583,0
female,0,53.1,1
male,0,7.75,0
female,1,247.5208,1
male,0,16,0
female,1,21,1
male,0,8.05,0
female,2,69.55,1
male,0,13,0
male,0,26,0
male,0,26,0
male,1,14.5,0
male,0,12.35,0
male,2,32.5,0
male,0,7.8542,0
male,2,134.5,0
female,0,7.775,1
male,0,10.5,0
female,0,8.1125,1
female,0,15.5,1
male,2,14.4,0
male,0,227.525,1
female,1,26,1
male,0,10.5,0
male,0,25.7417,0
female,0,7.75,1
male,0,10.5,0
female,0,27.7208,1
male,0,7.8958,0
male,0,22.525,0
male,0,7.05,0
male,0,73.5,0
female,0,26,1
female,2,7.775,1
male,0,42.5,0
female,0,7.8792,1
male,1,164.8667,0
female,1,211.5,1
male,0,8.05,0
female,0,13.8583,1
male,0,8.05,0
female,0,10.5,1
male,0,7.7958,0
female,0,27.4458,1
female,2,15.2458,1
male,0,7.7958,0
female,0,7.75,1
male,0,15.1,0
male,0,13,0
male,0,65,0
female,0,26.55,1
male,0,6.4958,0

male,0,7.8792,0
male,0,71.2833,0
male,0,7.8542,0
male,0,75.25,0
male,0,7.225,0
female,1,13,1
female,0,106.425,1
female,0,27.7208,1
female,2,30,1
male,1,134.5,0
male,0,7.8875,0
male,2,23.45,0
male,0,51.8625,0
female,0,21,1
male,1,32.5,0
female,0,26,1
female,0,14.4542,1
female,2,27.75,1
male,0,7.925,0
male,0,136.7792,0
male,0,9.325,0
male,0,9.5,0
male,0,7.55,0
male,0,7.75,0
male,0,8.05,0
female,0,13,1
male,0,7.775,0
male,0,17.4,0
male,0,7.8542,0
female,2,23,1
female,1,12.1833,1
male,0,12.7375,0
male,0,7.8958,0
male,0,0,0
male,0,7.55,0
female,0,8.05,1
male,0,8.6625,0
male,0,75.2417,0
male,0,7.75,0
female,0,136.7792,1
female,0,15.5,1
male,0,7.225,0
female,0,26,1
male,0,10.5,0
male,0,26,0
male,0,21,0
male,0,10.5,0
female,0,8.6625,1
male,1,13.775,0
female,0,7.75,1
female,1,15.2458,1
female,1,20.2125,1
male,0,7.25,0
male,0,7.25,0
male,0,82.2667,0
male,0,7.2292,0
male,0,8.05,0
male,0,39.6,0
female,0,6.95,1
male,0,7.2292,0
male,1,81.8583,0
male,0,9.5,0
male,0,7.8958,0
female,2,41.5792,1
male,0,21.6792,0
male,0,45.5,0
male,0,7.8542,0
male,0,7.775,0
male,0,15.0458,0
male,1,21,0
male,0,8.6625,0
female,0,7.75,1

```
female,1,26.55,1
male,2,151.55,0
male,1,9.35,0
male,1,93.5,0
female,0,14.1083,1
male,0,8.6625,0
male,0,7.225,0
male,0,7.575,0
female,0,7.75,1
female,0,135.6333,1
female,0,7.7333,1
male,0,146.5208,0
male,0,10.5,0
male,0,7.8542,0
male,0,31.5,0
male,0,7.775,0
male,0,7.2292,0
male,0,13,0
male,0,26.55,0
female,0,211.3375,1
male,0,7.05,0
female,1,39,1
male,0,79.2,0
male,0,26,0
male,0,13,0
female,2,36.75,1
male,0,29.7,0
male,0,7.225,0
female,1,15.7417,1
male,0,7.8958,0
male,0,26,0
male,0,13,0
male,2,7.2292,0
male,0,31.5,0
male,0,7.2292,0
male,0,10.5,0
male,0,7.5792,0
male,9,69.55,0
female,1,512.3292,1
male,1,14.5,0
female,0,7.65,1
male,0,13,0
female,0,7.2292,1
male,0,13.5,0
female,0,21,1
female,1,63.3583,1
male,0,10.5,0
male,0,73.5,0
male,2,65,0
female,2,20.575,1
male,0,26,0
female,0,51.4792,1
male,0,7.8792,0
male,0,7.75,0
female,0,15.55,1
male,2,69.55,0
female,1,37.0042,1
female,0,21,1
male,0,8.6625,0
female,0,55.4417,1
female,9,69.55,0
male,0,14.4583,0
female,0,39.6875,1
female,1,59.4,1
male,0,13.8583,0
male,0,11.5,0
female,0,134.5,1
male,0,0,0
male,0,13,0
female,1,81.8583,1
female,0,262.375,1
female,0,8.6625,1
```

```
male,0,11.5,0
male,0,50,0
male,2,31.3875,0
male,0,7.75,0
male,0,7.8792,0
female,0,14.5,1
female,0,16.1,1
male,0,12.875,0
female,2,65,1
male,0,7.775,0
male,0,13,0
male,0,7.75,0
male,1,21.075,0
male,0,93.5,0
female,1,39.4,1
male,2,20.25,0
male,0,10.5,0
male,1,22.025,0
female,0,60,1
male,0,7.25,0
female,1,79.2,1
male,0,7.775,0
male,0,7.7333,0
female,0,164.8667,1
male,0,21,0
female,1,59.4,1
male,0,47.1,0
male,0,27.7208,0
male,0,13.8625,0
male,0,10.5,0
male,1,211.5,0
female,0,7.7208,1
female,1,13.775,1
female,0,7.75,1
female,0,90,1
female,0,7.775,1
male,0,8.05,0
female,0,108.9,1
male,0,7.25,0
male,0,8.05,0
male,1,22.3583,0
```

Your current directory should now have a png called 'titanic_visual.png' in it that looks like the following:

Here we see the 2D hidden space that corresponds to the two neurons the model uses. We can see a clear plane separating the two classifications. This is the plane that the predictor uses to distinguish between the two classes.