Project2

October 8, 2017

Will Russell V00788739
Ayush Rohatgi

0.1 Analysis

Scenario A

The splits obtained for Scenario A were overall slightly less accurate than that of the models generated in Scenario B. This may be expected to some degree in that with Scenario A we only have a single feature from which we may try to delineate the decision function. When the models were generated multiple times, overall the accuracy of the models which were trained using the soft activation method were more accurate than those trained using hard activation.

Using hard activation, we found that the ultimate accuracy of the model depended more on the learning rate(alpha) and initial weight values than the ratio of the train/test split. These results are consistent with the expected behavior of a perceptron due to the overlapping nature of the data. The overlap causes a consistently high error level, which the model continuously tries to compensate for, but due to the high level of error correction(+/- 1) the resulting weights will need to be shifted ad infinitum. With no overlap, the perceptron using hard activation will consistently find whichever set of weights correspond to the first dividing function which is encountered. Since overlap always exists to such a degree that the error in the system never falls below epsilon, the training model continues to shift until the number of iterations has expired. The ultimate weights achieved using the hard activation function on this type of data are not necessarily the lowest encountered, they are simply where the modification stopped after the maximum number of iterations had been cycled.

In the case of soft activation, the results in overall accuracy found using soft activation is similar to that found using hard activation. The main difference lies in the error rate ultimately achieved when training the model on the original dataset. Similar to the hard activation method, the training done using soft activation will never achieve the level of epsilon in order to terminate training prior to the completion of all iterations. However, given enough time, the method will achieve a slightly higher accuracy with a low enough alpha that would generally be achieved using hard activation due to the relatively small modification of the error value and the corresponding correction of the weights.

By using the accompanying animation notebook, you can explore how the training is performed on various sizes of data. With few enough points, both hard and soft activation will find the gradient which can be used to classify the students. Generally, with a low enough epsilon, soft activation will converge to the more central point between the students however, whereas hard activation will lead to the stopping point whenever the first separating gradient is achieved.

Scenario B

The results obtained using both the hard and soft activation functions are very similar to those described above for Scenario A. Ultimately the same conclusions are reached regarding the behavior of the Perceptron, though there are some differences which are worth noting.

The overall accuracy of the same base models increases by +/- 8% when applied to Scenario B. The main reason for this is the increased dimensionality resulting from the addition of the weight feature. If the weight feature had a more significant overlap between male and female students, then it is not likely that we would see the gains.

0.1.1 Summary

Due to the large overlap, there was not a signifcant difference to be found in the behavior of either the hard or soft perceptron models on the train/test splits. The results as described in the graphs below and further corroborated using the attached animation notebook demonstrate a similar manner of behavior among both models. What is most notable about the models is the manner by which they approach their final weight measures. In the case of the sigmoid function we have a ratio which can be described as follows

$$out = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)}$$

This structure allows for smaller steps in the error correction than is available with the harder method described below

$$out = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i + w_{n+1} \geq 0 \\ 0 & if \sum_{i=1}^{n} w_i x_i + w_{n+1} < 0 \end{cases}$$

It is this smaller level of correction which enables the soft activation method to converge on a gradient more centrally located between two groups.

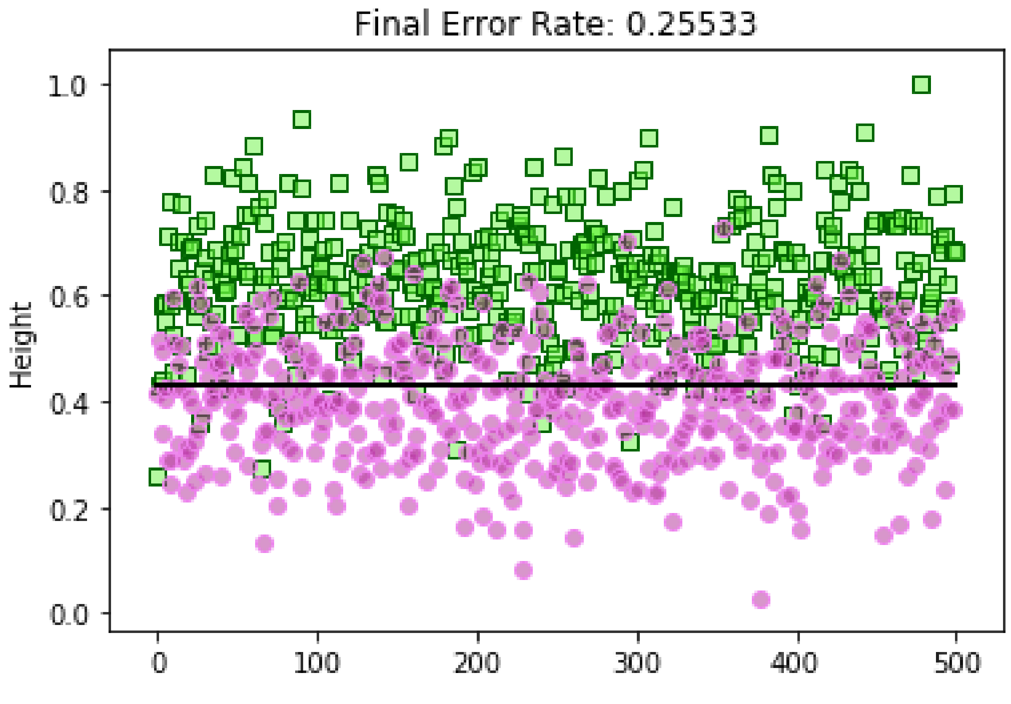
0.2 Graphing

- Below is the basic graphing module for the project. The code for the displayPerceptron method can be found in the **project_code/display.py** file. The source of the test data is the **data.txt** file located in the current directory.
- The displayPerceptron method takes 6 optional parameters:
  - ratio : Represents the Train/Test Ratio as a decimal(Default is .75 or 75% train/25% test)
  - activation : Represents the type of activation function to be used(Currently 2 options, "hard" or "soft")
  - alpha : Represents the learning rate of the Perceptron(Default is 0.1)
  - iterations : The number of iterations to train the model for
  - dimensions : The number of weights (excluding bias)
  - weights : An option to start with a preset weight, if not declared a set of random values between 0 and 1 will be used.
- The method then displays a graph based on the parameters used and shows the final error rate established during training.
  - *Note: This is not the test error rate*
- Animations corresponding to the training can be found in **Project2_Perceptron_Animated.ipynb**

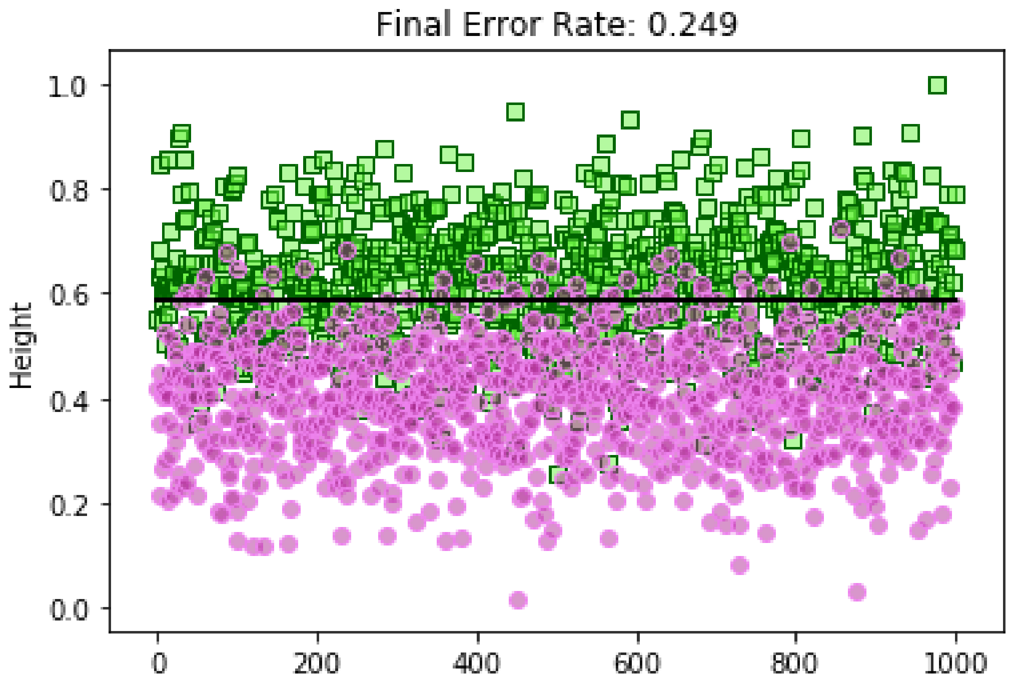
```
In [1]: from project_code.display import *
        %matplotlib inline
```

0.3 Scenario A:

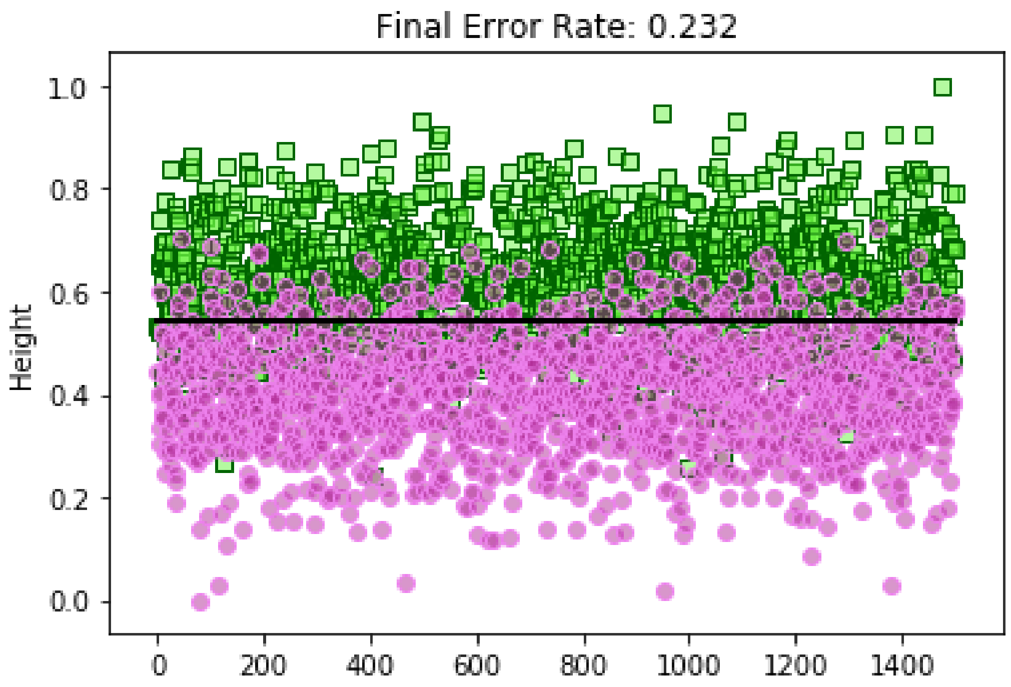
```
In [2]: displayPerceptron(ratio=0.75, activation="hard", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.50, activation="hard", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.25, activation="hard", alpha=0.1, iterations=1000
```



```
male correct, true positive : 0.956
male errors, false positive : 0.044
female errors, false negative : 0.396
female correct, true negative : 0.604
Accuracy : 0.78
Error: 0.22
```

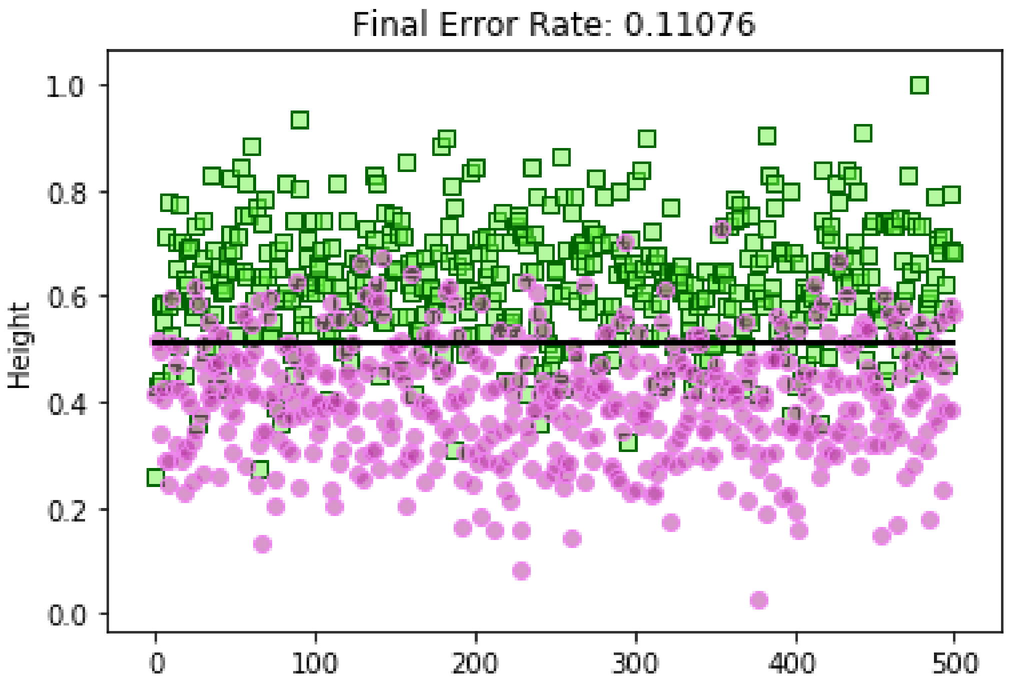


```
male correct, true positive : 0.64
male errors, false positive : 0.36
female errors, false negative : 0.053
female correct, true negative : 0.947
Accuracy : 0.7935
Error: 0.2065
```

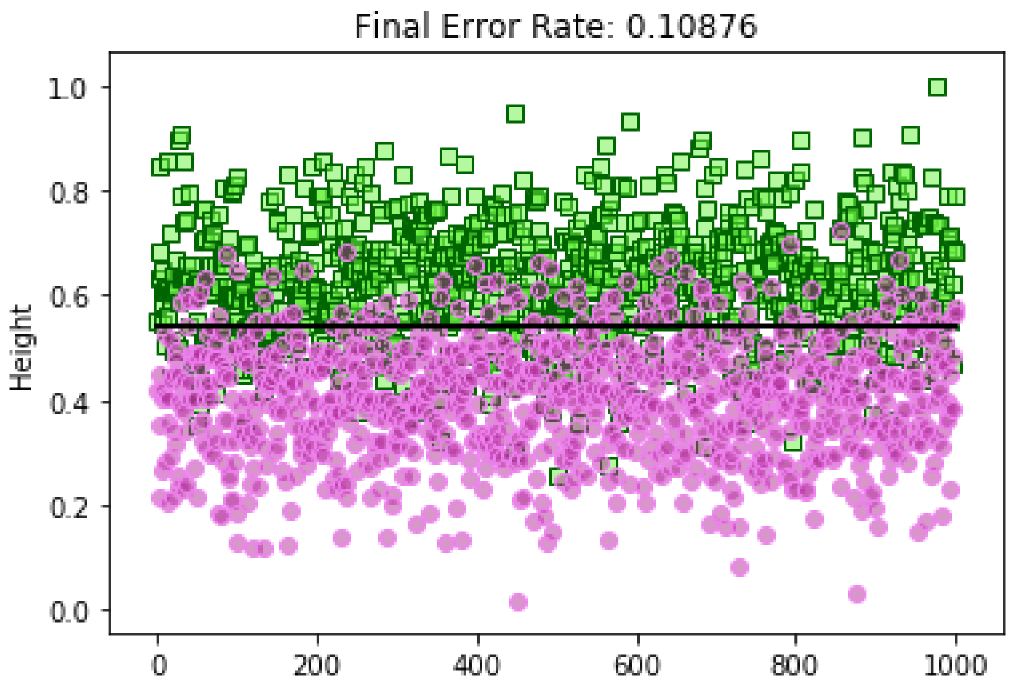


```
male correct, true positive : 0.76267
male errors, false positive : 0.23733
female errors, false negative : 0.10667
female correct, true negative : 0.89333
Accuracy : 0.828
Error: 0.172
```

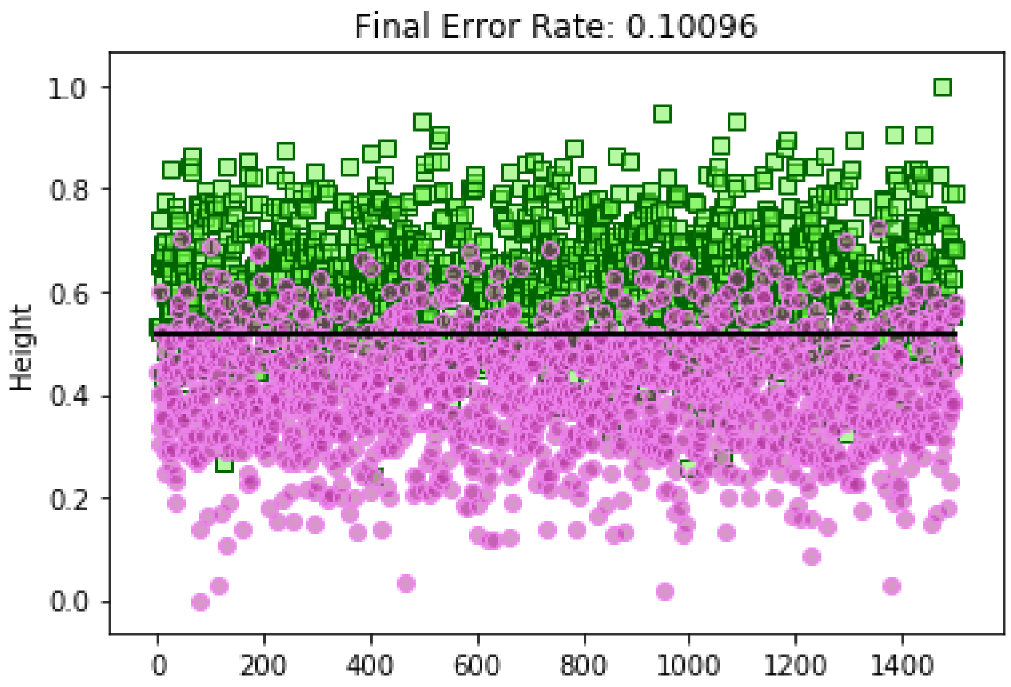
```
In [3]: displayPerceptron(ratio=0.75, activation="soft", alpha=0.1, iterations=100(
        displayPerceptron(ratio=0.50, activation="soft", alpha=0.1, iterations=100(
        displayPerceptron(ratio=0.25, activation="soft", alpha=0.1, iterations=100(
```



```
male correct, true positive : 0.846
male errors, false positive : 0.154
female errors, false negative : 0.17
female correct, true negative : 0.83
Accuracy : 0.838
Error: 0.162
```



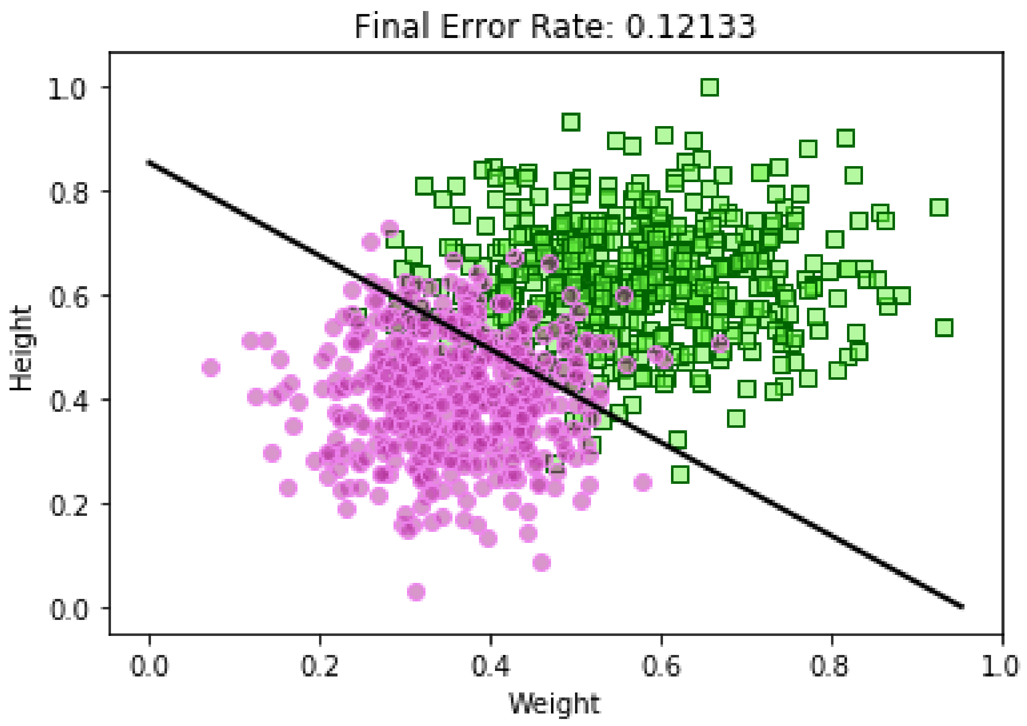
```
male correct, true positive : 0.767
male errors, false positive : 0.233
female errors, false negative : 0.105
female correct, true negative : 0.895
Accuracy : 0.831
Error: 0.169
```



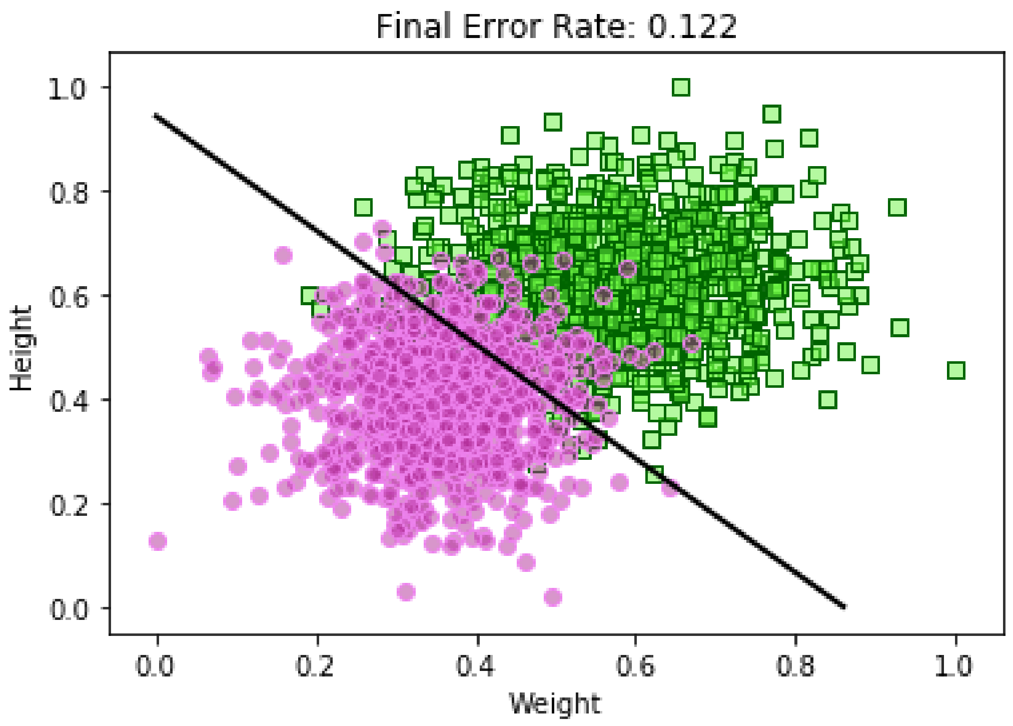
```
male correct, true positive : 0.826
male errors, false positive : 0.174
female errors, false negative : 0.142
female correct, true negative : 0.858
Accuracy : 0.842
Error: 0.158
```

0.4 Scenario B

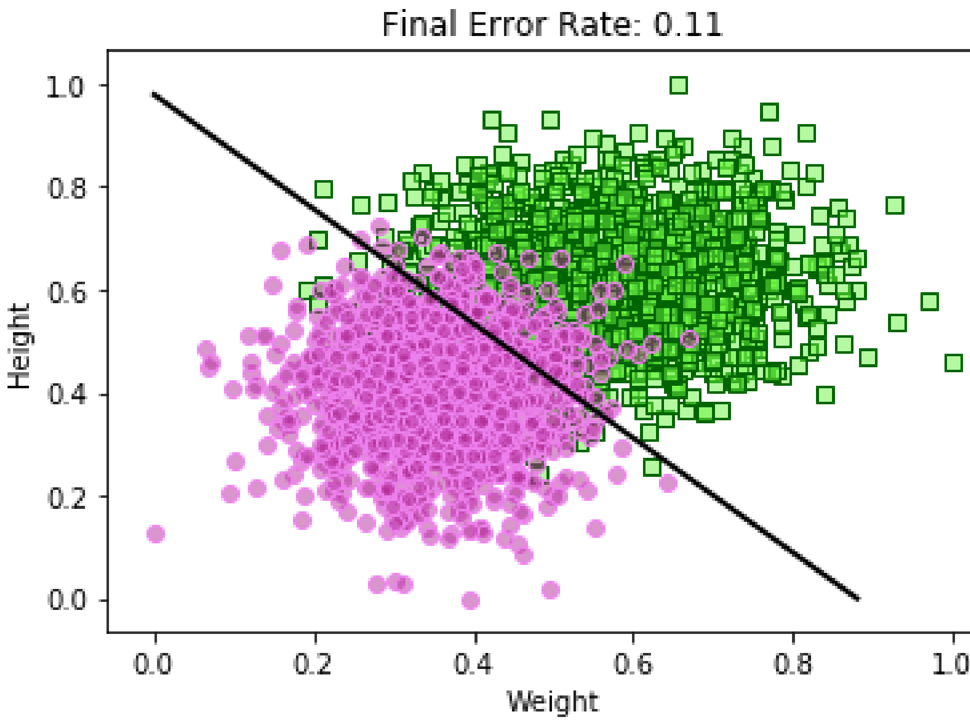
```
In [6]: displayPerceptron(ratio=0.75, activation="hard", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.50, activation="hard", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.25, activation="hard", alpha=0.1, iterations=1000
```



```
male correct, true positive : 0.99
male errors, false positive : 0.01
female errors, false negative : 0.376
female correct, true negative : 0.624
Accuracy : 0.807
Error: 0.193
```

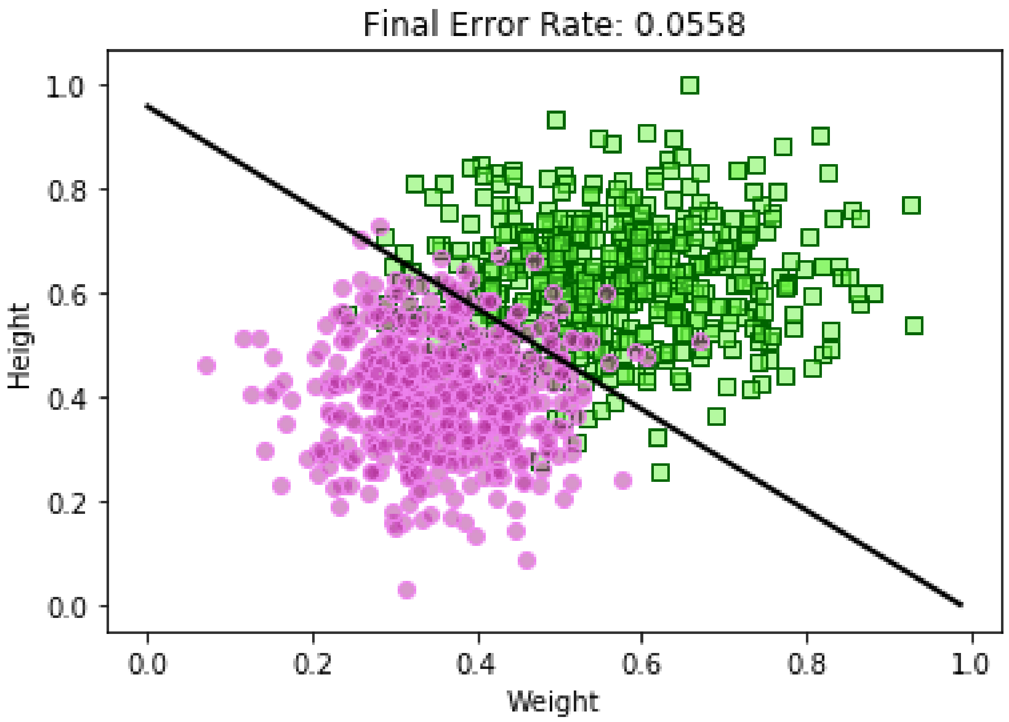


```
male correct, true positive : 0.897
male errors, false positive : 0.103
female errors, false negative : 0.053
female correct, true negative : 0.947
Accuracy : 0.922
Error: 0.078
```

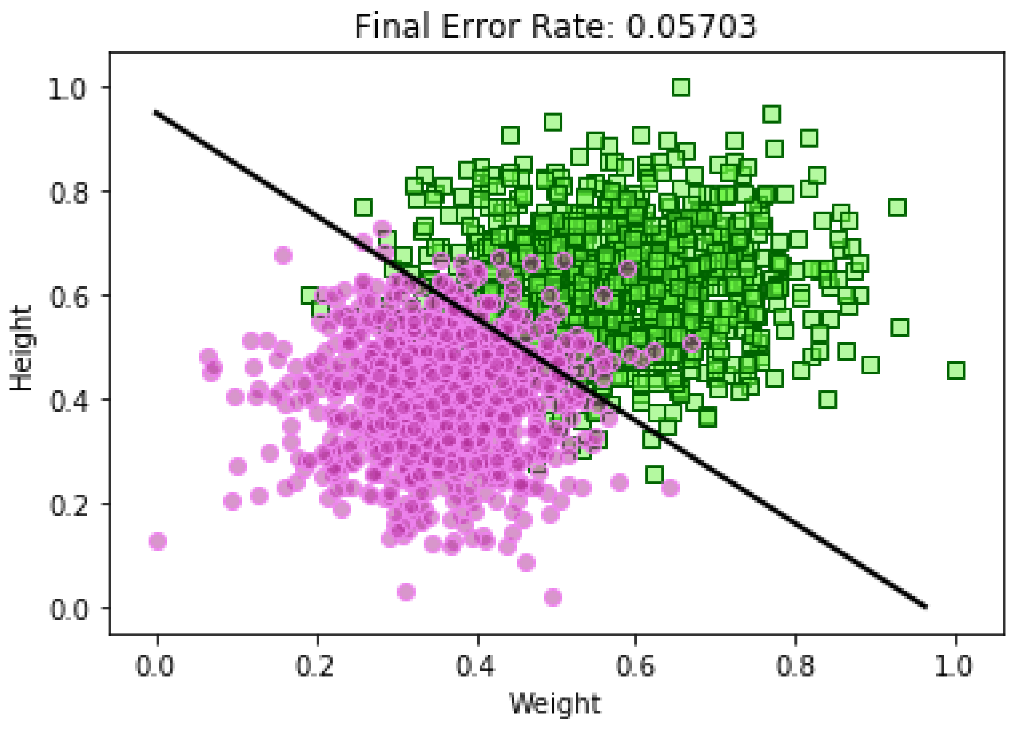


```
male correct, true positive : 0.81267
male errors, false positive : 0.18733
female errors, false negative : 0.02133
female correct, true negative : 0.97867
Accuracy : 0.8957
Error: 0.1043
```

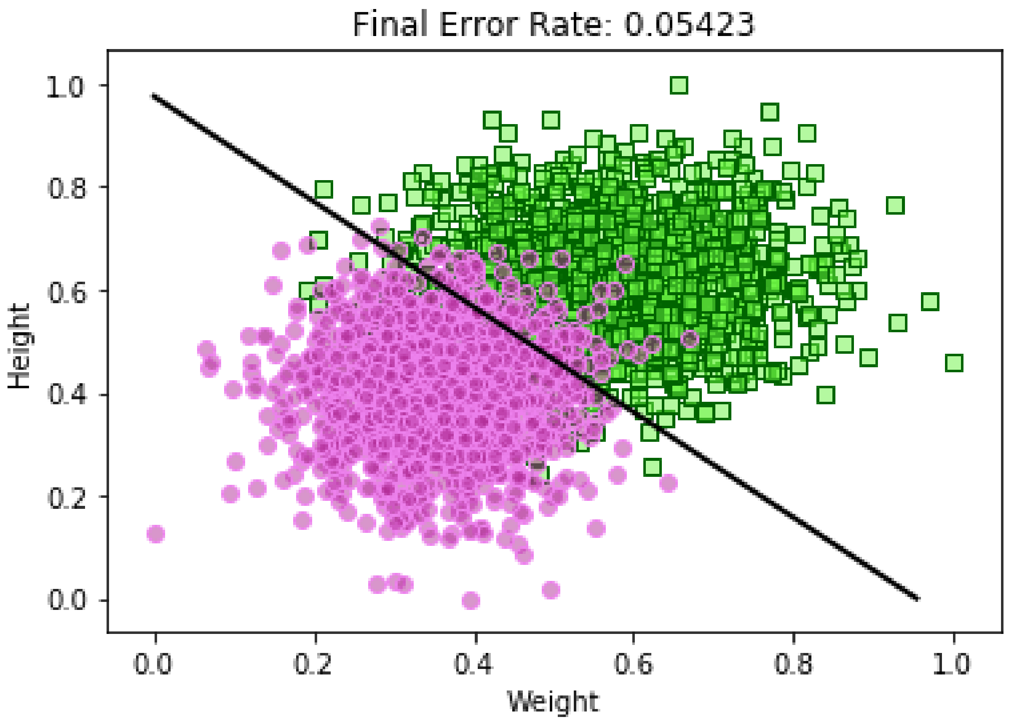
```
In [5]: displayPerceptron(ratio=0.75, activation="soft", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.50, activation="soft", alpha=0.1, iterations=1000
        displayPerceptron(ratio=0.25, activation="soft", alpha=0.1, iterations=1000
```



```
male correct, true positive : 0.924
male errors, false positive : 0.076
female errors, false negative : 0.106
female correct, true negative : 0.894
Accuracy : 0.909
Error: 0.091
```



```
male correct, true positive : 0.936
male errors, false positive : 0.064
female errors, false negative : 0.102
female correct, true negative : 0.898
Accuracy : 0.917
Error: 0.083
```



```
male correct, true positive : 0.896
male errors, false positive : 0.104
female errors, false negative : 0.05067
female correct, true negative : 0.94933
Accuracy : 0.9227
Error: 0.0773
```

In [ ]: