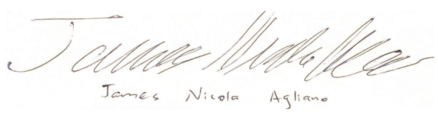
CMSC 409 Artificial Intelligence
Project 2
Due Thursday, October 3

Student certification

Team member 1:
Print Name: Nick Agliano
Date: 9/16/2019
I have contributed by doing the following:

- Coded perceptron class, plotting function, error calculation function, assisted in the written report, wrote summaries of results

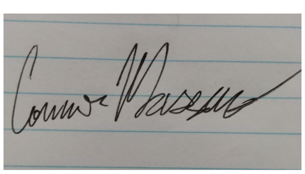
Signed:



Team member 2:
Print Name: Connor Massaro
Date: 9/16/2019
I have contributed by doing the following:

- Created test datasets, calculated error, assisted in coding perceptrons and activation functions, assisted in the development of the report

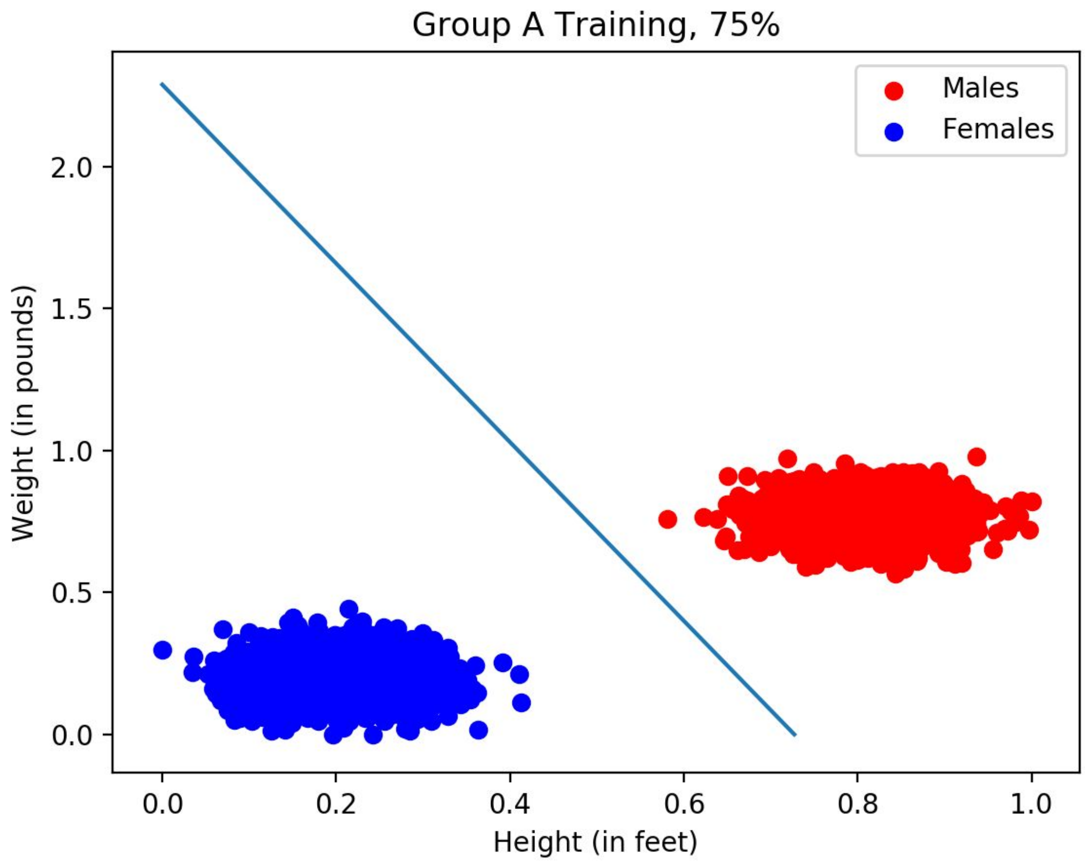
Signed: Connor Massaro



Scenario A - Hard Activation Function

1. Choose 75% of the data for training, and the rest for testing. Train and test your neuron. Plot the data and decision line for training and testing data (separately). Calculate errors for training and testing dataset.

Plot Group A:

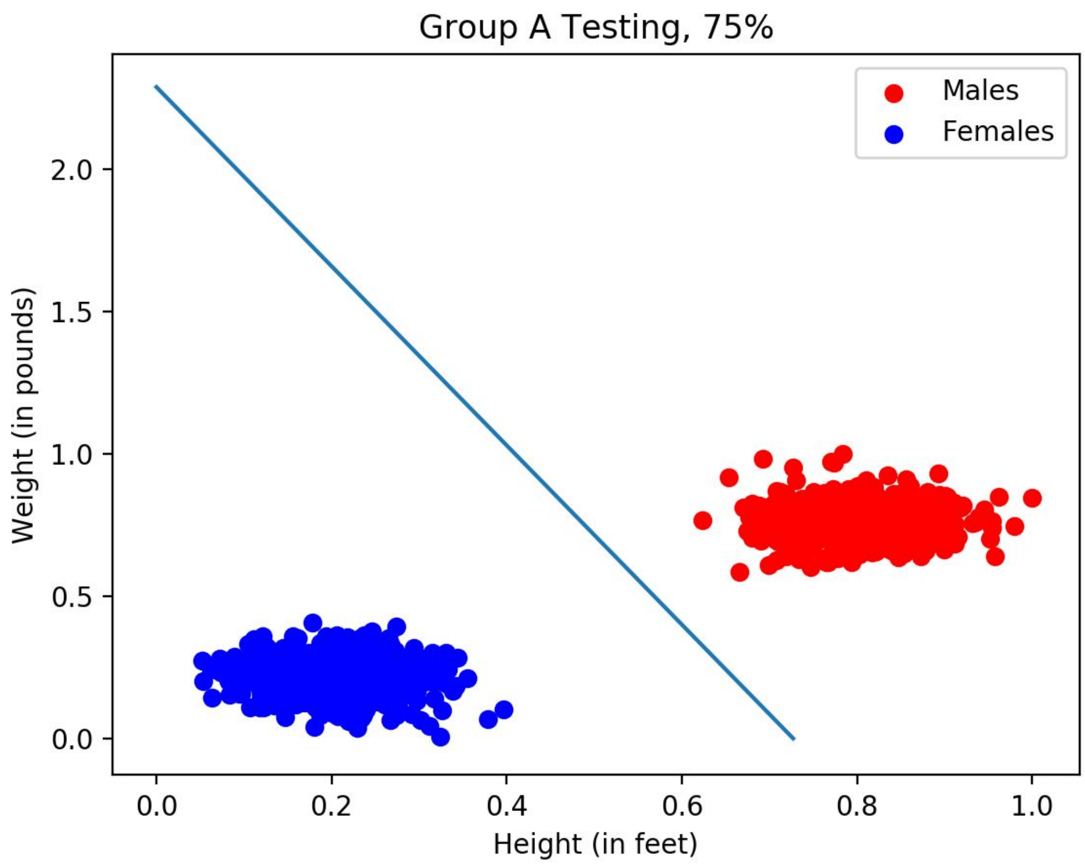


Confusion Matrix

TP = 1500	FN = 0
FP = 0	TN = 1500

Error: 0
Iterations: 0
weight of x: -0.3709149617793861
weight of y: -0.11784352799545883
weight of bias: 0.2696928077251003

Accuracy = 1.0
Error = 0.0
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.0
Precision = 1.0



Confusion Matrix

TP = 500	FN = 0
FP = 0	TN = 500

Accuracy = 1.0

Error = 0.0

Recall or True positive rate (TP) = 1.0

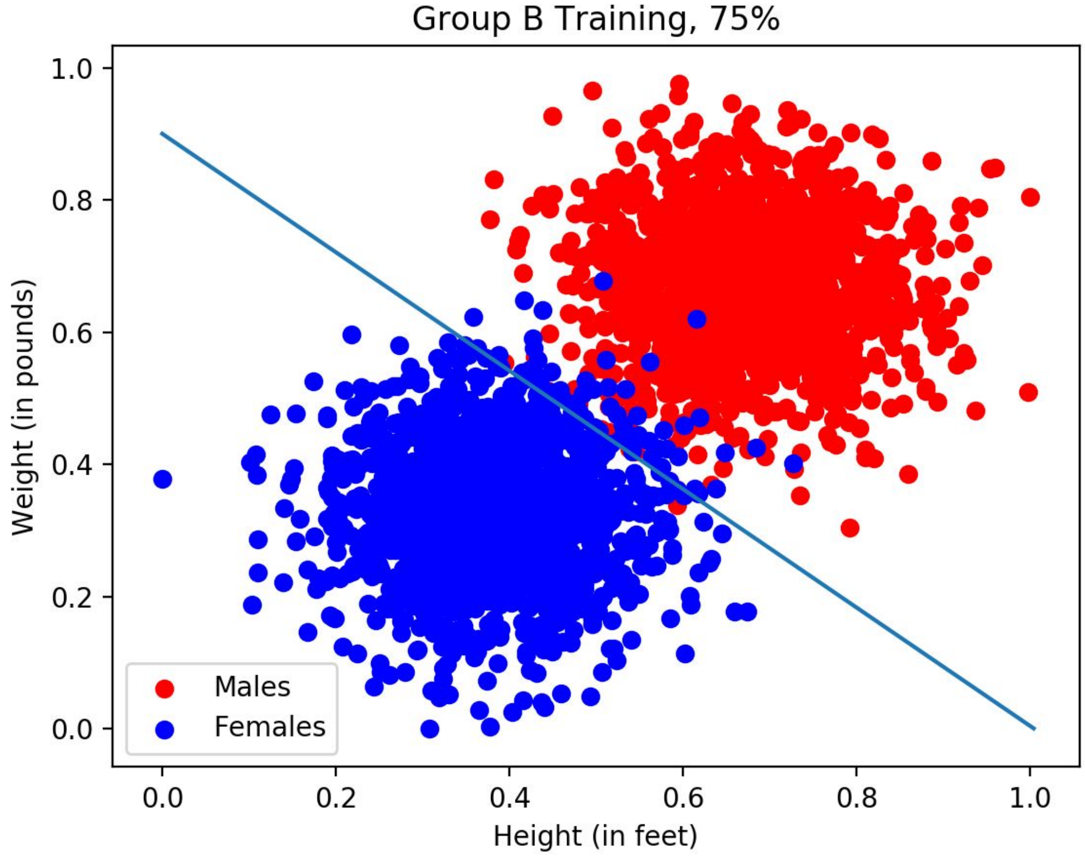
True negative rate (TN) = 1.0

False positive rate (FP) = 0.0

False negative rate (FN) = 0.0

Precision = 1.0

Plot Group B:

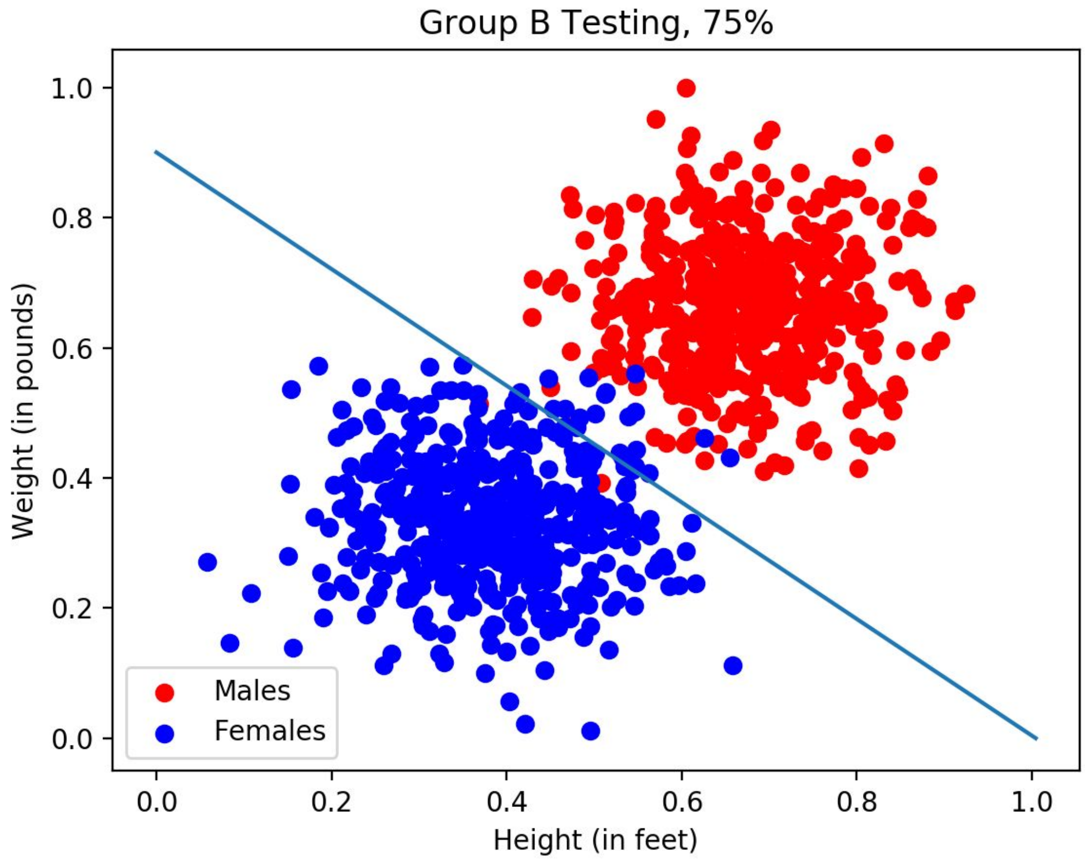


Confusion Matrix

TP = 1441	FN = 59
FP = 2	TN = 1498

Error: 83
Iterations: 1
weight of x: -14.184646922759491
weight of y: -15.815822695803359
weight of bias: 14.245927590737978

Accuracy = 0.980
Error = 0.020
Recall or True positive rate (TP) = 0.961
True negative rate (TN) = 0.999
False positive rate (FP) = 0.001
False negative rate (FN) = 0.039
Precision = 0.999

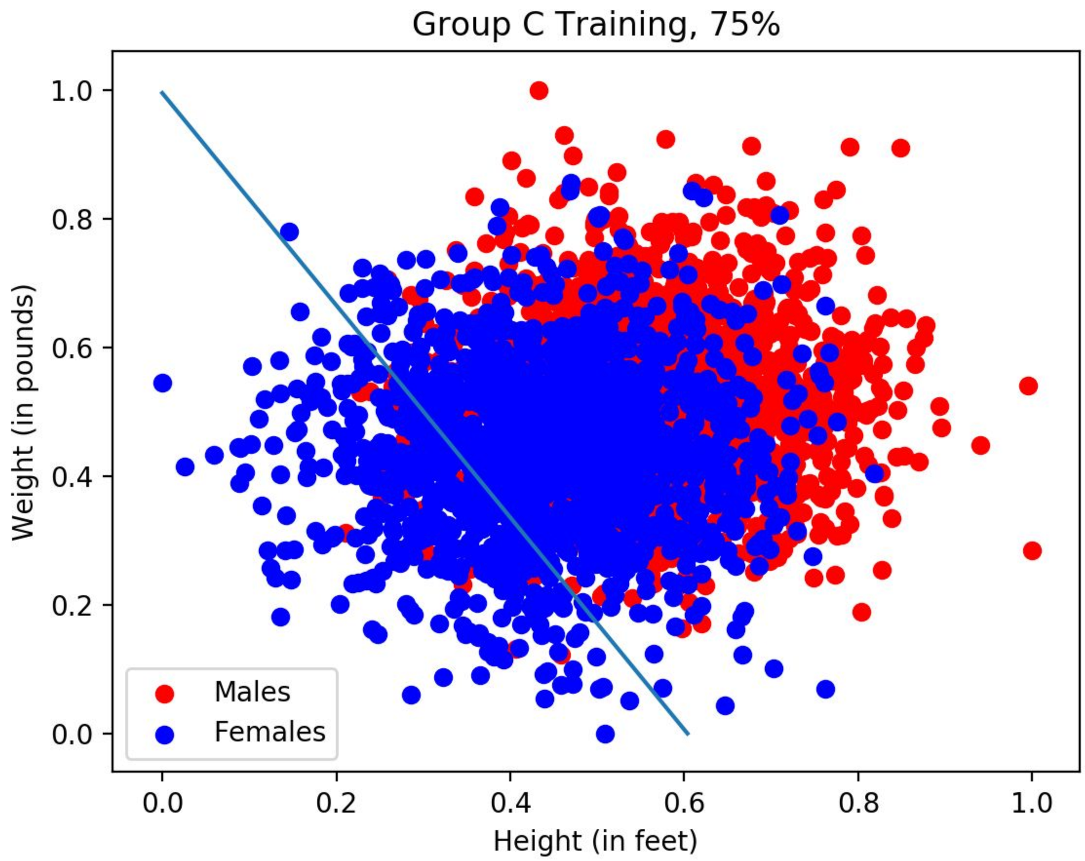


Confusion Matrix

TP = 480	FN = 20
FP = 3	TN = 497

Accuracy = 0.997
Error = 0.003
Recall or True positive rate (TP) = 0.960
True negative rate (TN) = 0.994
False positive rate (FP) = 0.006
False negative rate (FN) = 0.040
Precision = 0.994

Plot Group C:

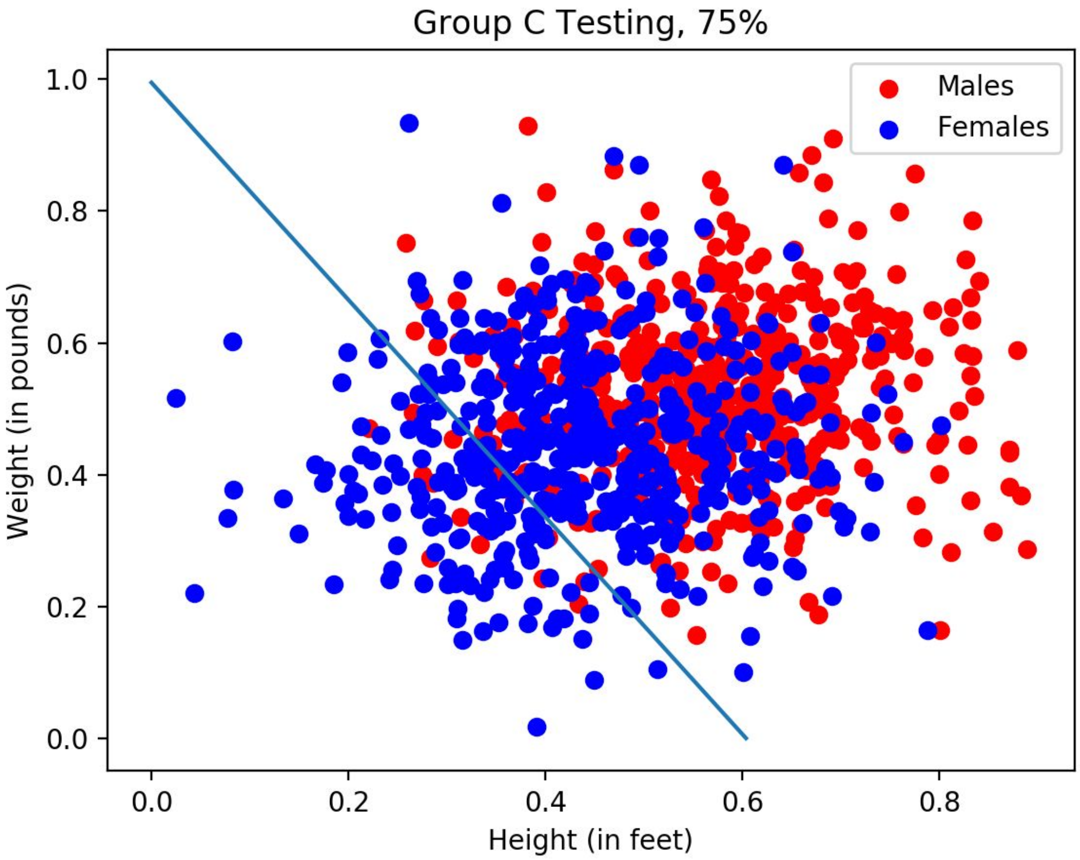


Confusion Matrix

TP = 363	FN = 1137
FP = 45	TN = 1455

Error: 1115
Iterations: 0
weight of x: -9.832730772118362
weight of y: -5.965693264988653
weight of bias: 5.935778372162098

Accuracy = 0.606
Error = 0.394
Recall or True positive rate (TP) = 0.242
True negative rate (TN) = 0.970
False positive rate (FP) = 0.030
False negative rate (FN) = 0.758
Precision = 0.890



Confusion Matrix

TP = 123	FN = 377
FP = 19	TN = 481

Accuracy = 0.604

Error = 0.396

Recall or True positive rate (TP) = 0.246

True negative rate (TN) = 0.962

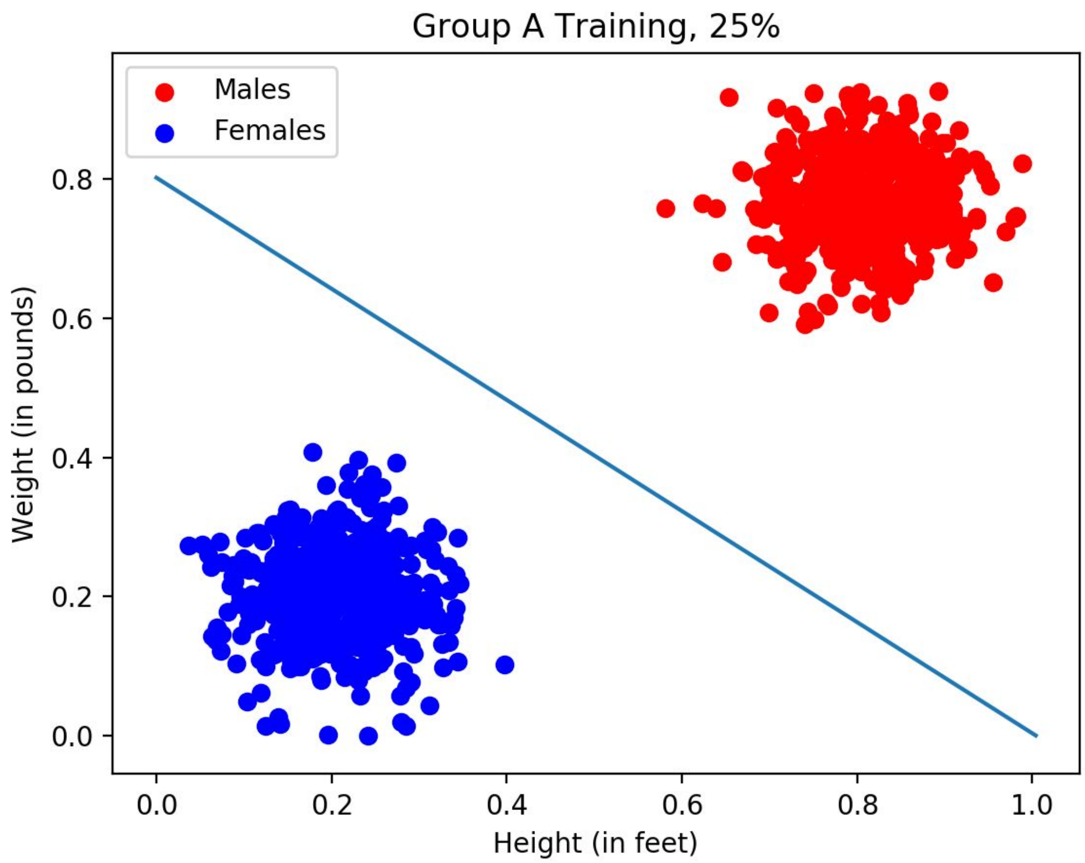
False positive rate (FP) = 0.038

False negative rate (FN) = 0.754

Precision = 0.866

2. Choose 25% of the data for training, and the rest for testing. Train and test your neuron. Plot the data and decision line for training and testing data (separately). Calculate errors for training and testing dataset.

Plot Group A:

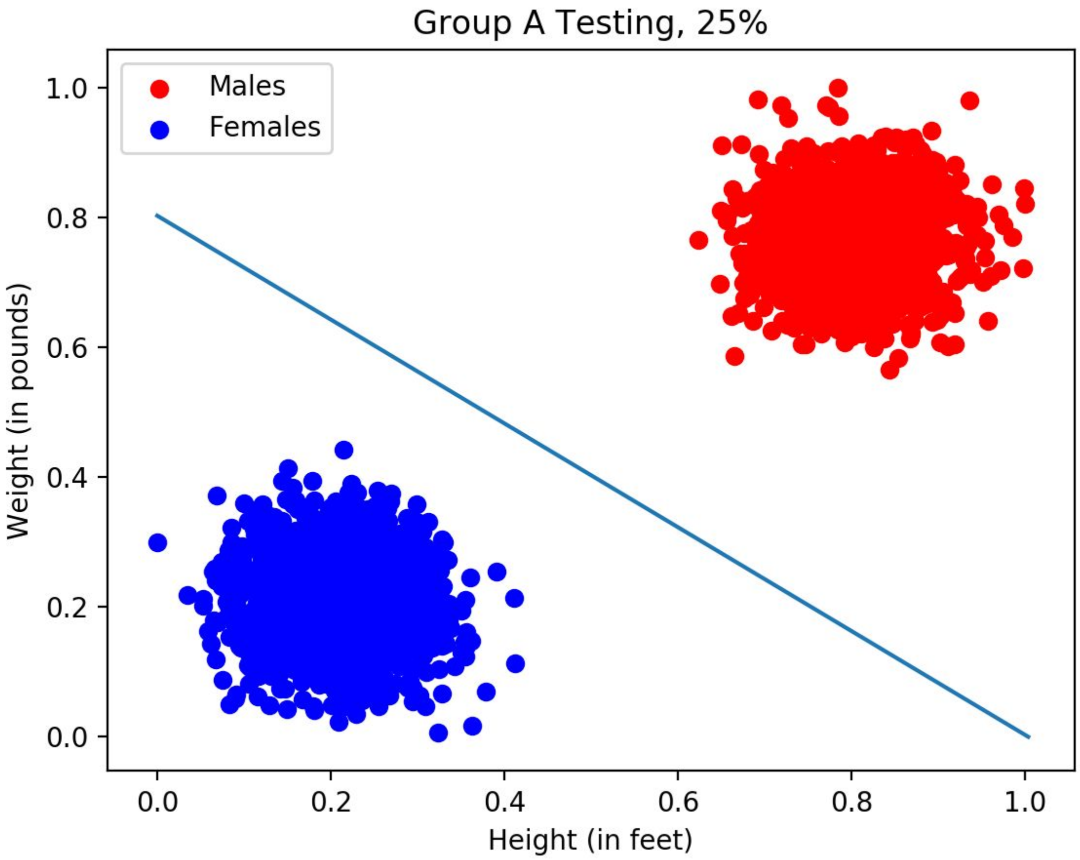


Confusion Matrix

TP = 500	FN = 0
FP = 0	TN = 500

Error: 0
Iterations: 1
weight of x: -1.9395221305586587
weight of y: -2.4268653335515533
weight of bias: 1.9469582425331136

Accuracy = 1.0
Error = 0.0
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.0
Precision = 1.0



Confusion Matrix

TP = 1500	FN = 0
FP = 0	TN = 1500

Accuracy = 1.0

Error = 0.0

Recall or True positive rate (TP) = 1.0

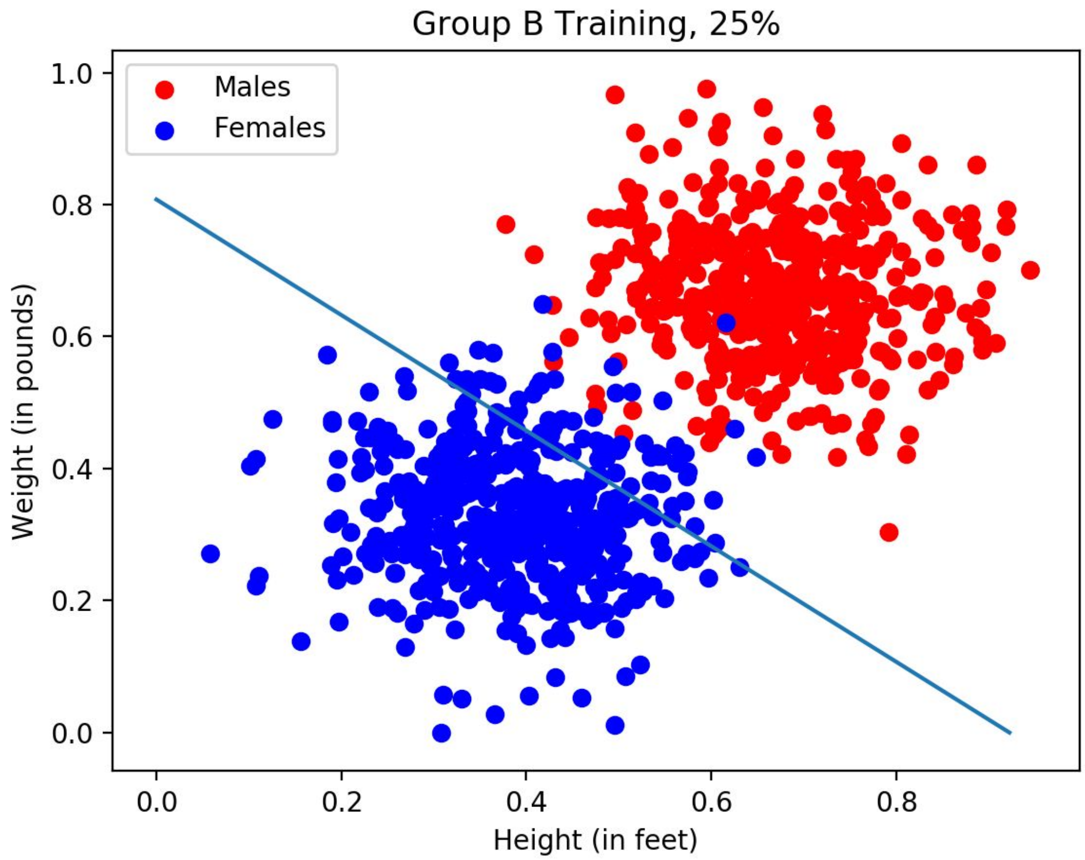
True negative rate (TN) = 1.0

False positive rate (FP) = 0.0

False negative rate (FN) = 0.0

Precision = 1.0

Plot Group B:

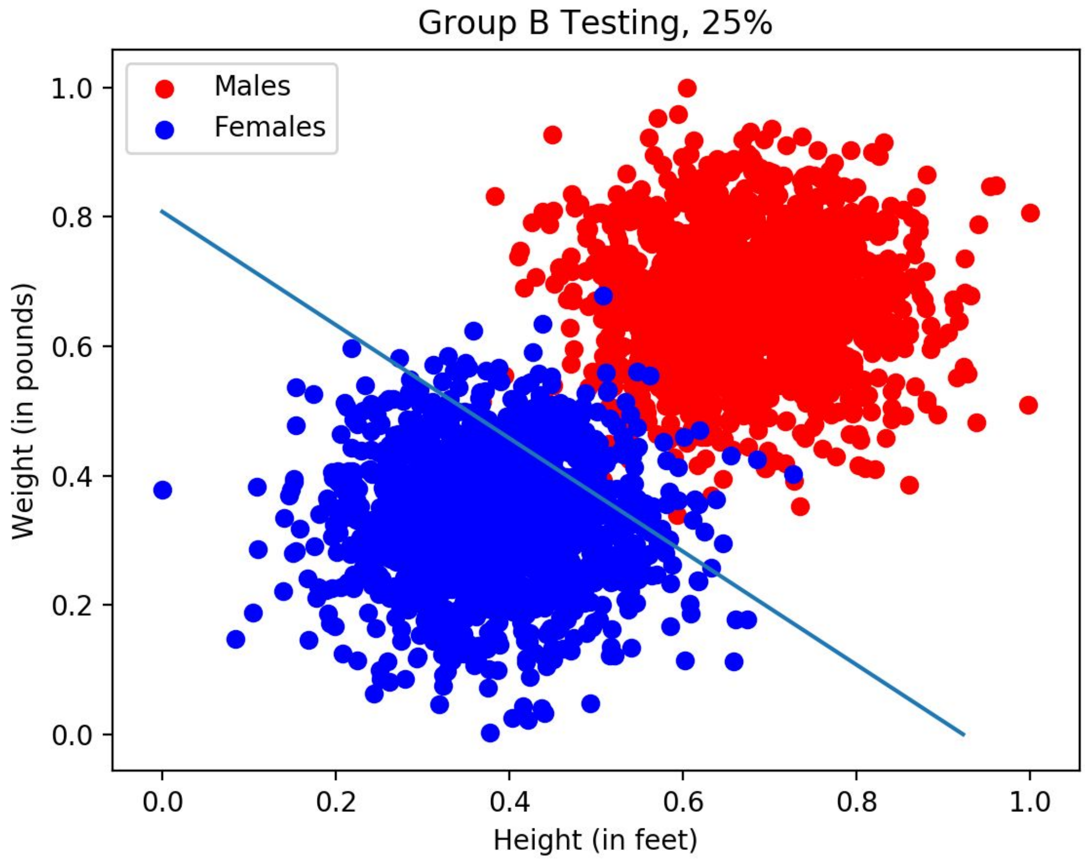


Confusion Matrix

TP = 426	FN = 74
FP = 0	TN = 500

Error: 74
Iterations: 0
weight of x: -8.567811174385698
weight of y: -9.785042978809246
weight of bias: 7.9033452181410375

Accuracy = 0.926
Error = 0.074
Recall or True positive rate (TP) = 0.852
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.148
Precision = 1.0

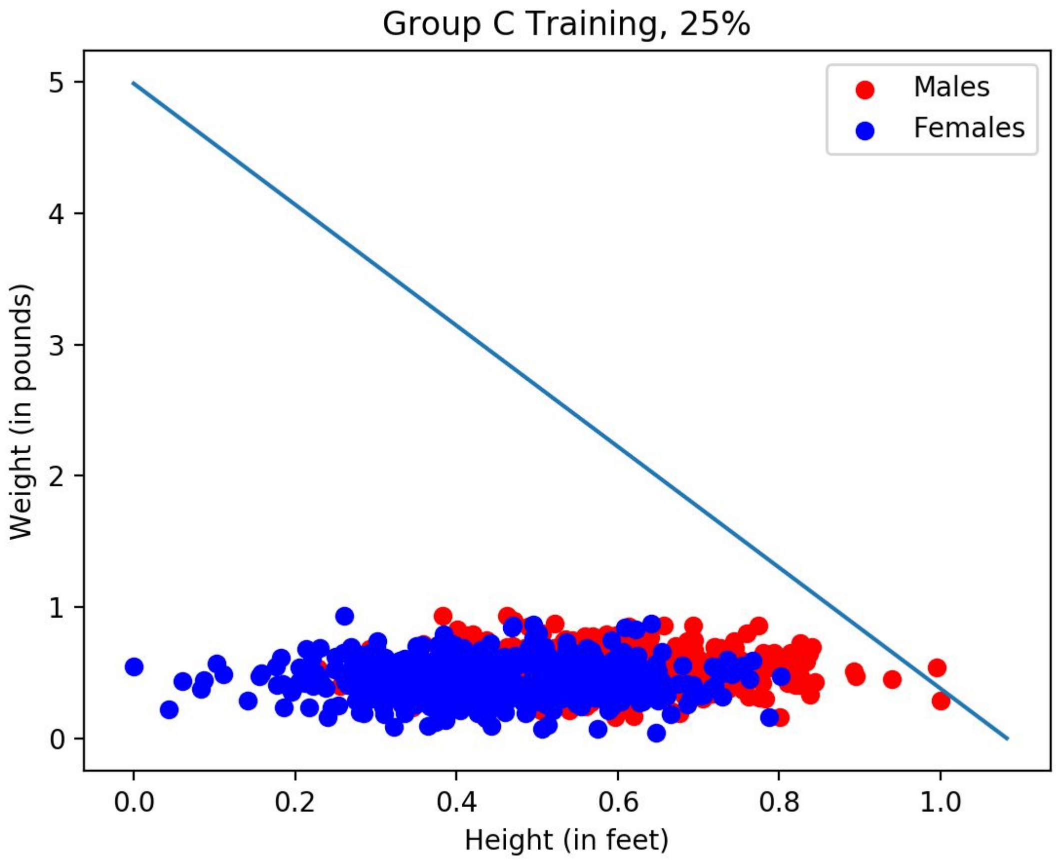


Confusion Matrix

TP = 1322	FN = 178
FP = 0	TN = 1500

Accuracy = 0.941
Error = 0.059
Recall or True positive rate (TP) = 0.881
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.119
Precision = 1.0

Plot Group C:

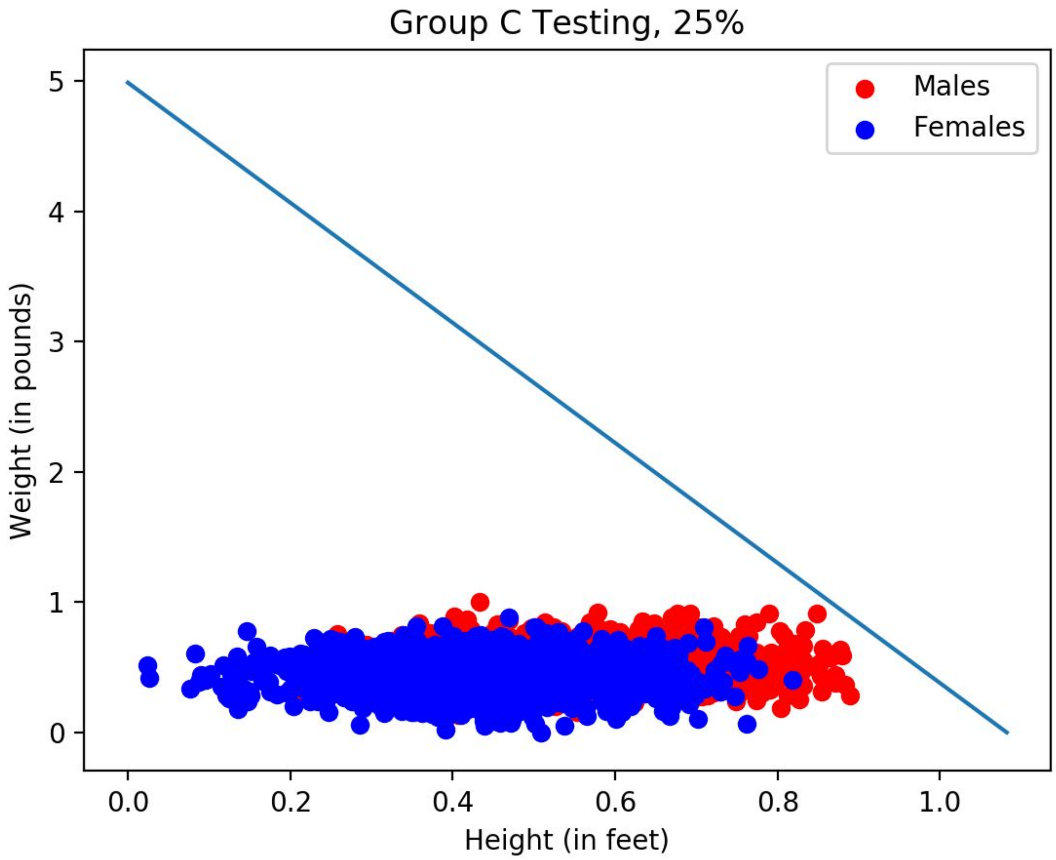


Confusion Matrix

TP = 500	FN = 0
FP = 499	TN = 1

Error: 412
Iterations: 0
weight of x: -7.486961536684076
weight of y: -1.6246717570722569
weight of bias: 8.10219520429627

Accuracy = 0.501
Error = 0.499
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 0.002
False positive rate (FP) = 0.998
False negative rate (FN) = 0.0
Precision = 0.500



Confusion Matrix

TP = 1500	FN = 0
FP = 1500	TN = 0

Accuracy = 0.500
Error = 0.500
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 0.0
False positive rate (FP) = 1.0
False negative rate (FN) = 0.0
Precision = 0.500

3. Compare 1. and 2. Are errors different and if so, why? What is the effect of different data set and effect of different training/testing distributions? When would you use option 1 and when option 2 above? Comment and discuss.

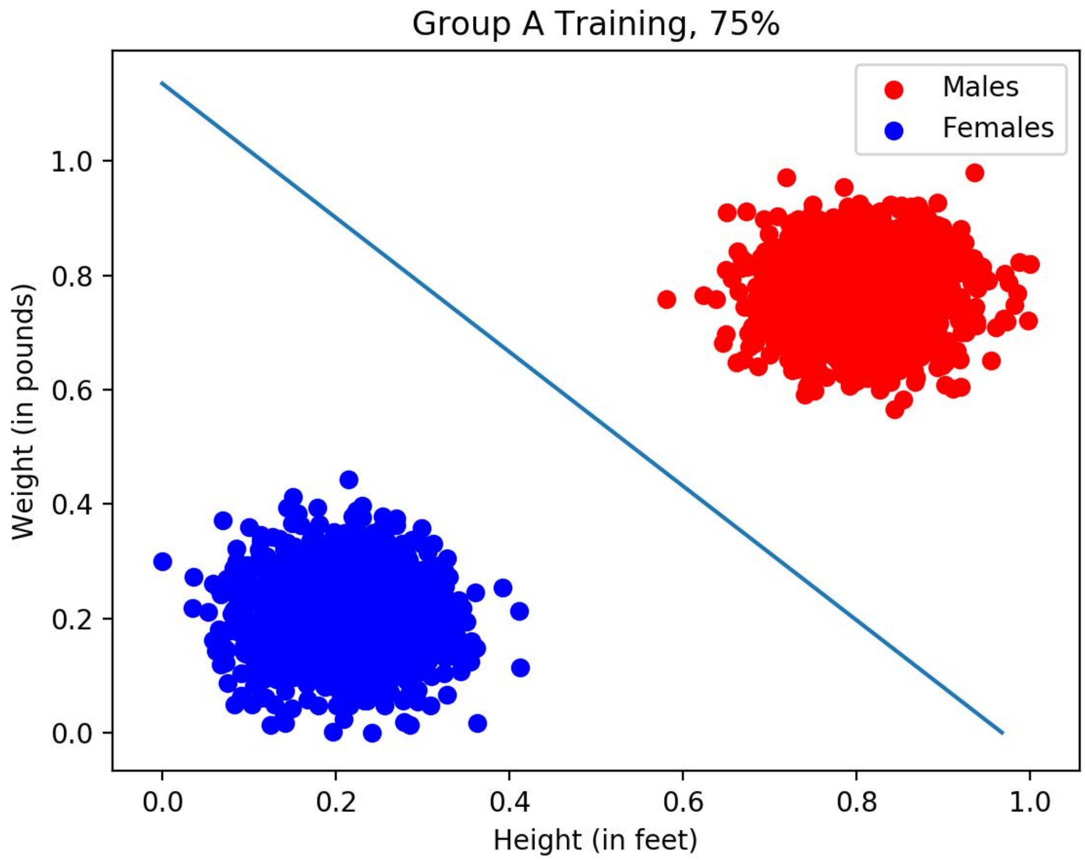
- For **Group A**:
    - Since Group A is so perfectly clustered, it was very easy to classify.
    - The perceptron classified every data point correctly, both when training on 75% of the dataset and when training on 25% of the dataset.
    - It can also be thought of in this way -- the 25% of the dataset that was used for training was well representative of the 75% 'unknown' data points because of how clustered the two classes are.
- For **Group B**:
    - When training on 75% of the dataset, the perceptron classified the training data with 98.0% accuracy and the testing data with 99.7% accuracy -- which is unexpected for the testing accuracy to be higher than the training accuracy. This was a stroke of luck from the perceptron -- it got lucky with where the testing data points were plotted.
        - This accuracy measurement is from just one run of the perceptron (as are all the other accuracies), but it would be a more accurate measure of accuracy if it was an average over multiple runs on a random sampled data set.
    - When training on 25% of the dataset, again the perceptron beat the odds by classifying with higher accuracy on the testing than the training, with 94.1% and 92.6% respective accuracies.
    - **The accuracy for the classification for Group B is higher when it learns from 75% of the dataset than when it learns from 25%.**
        - This is because Group B is still highly clustered data. When it trains on 75% of the data, it is not at a high risk of overfitting the data, but instead it learns more about the data set and is able to better predict the test dataset. However, when it trains on 25% of the data, it is not able to learn as much, so it cannot predict as accurately on the test data set.
- For **Group C**:
    - When trained on 75% of data, the perceptron classified the training data 60.6% accurately, and when trained on 25% of the data, it classified the training data 50.1% accurately and the testing data 50.0% accurately.
    - The perceptron isn't able to learn enough from 25% of the dataset to be able to make a classification better than randomly guessing.
        - (part of this reason is that the perceptron has to potential to learn more, but it stops because the error reaches the stopping criterion)
    - On 75% of the data, the accuracy is just barely better than guessing, at 60.4%

- **Comments/discussion/takeaways**:
    - In these case studies above, the perceptron performed better when it had more data to train on (the 75% groups) -- this could be because it's getting better at classifying, but it could also mean it's 'memorizing' the data
        - It could be overfitting to a certain degree. In the case that we get a million data points appended to our dataset, the perceptron that was trained on 25% could be more accurate because it is more generalized.
    - In the case that you have reason to believe that you have a good understanding of the dataset as a whole, training on 75% of the data would be better -- however, if you only have access to a small chunk of data out of massive amounts of unknown data, it would be better to train on 25% of the data so that you don't overfit your small chunk of data and fail to generalize to the dataset as a whole.

Scenario B - Soft Activation Function

1. Choose 75% of the data for training, and the rest for testing. Train and test your neuron. Plot the data and decision line for training and testing data (separately). Calculate errors for training and testing dataset.

Plot Group A: alpha = 1, k = 1

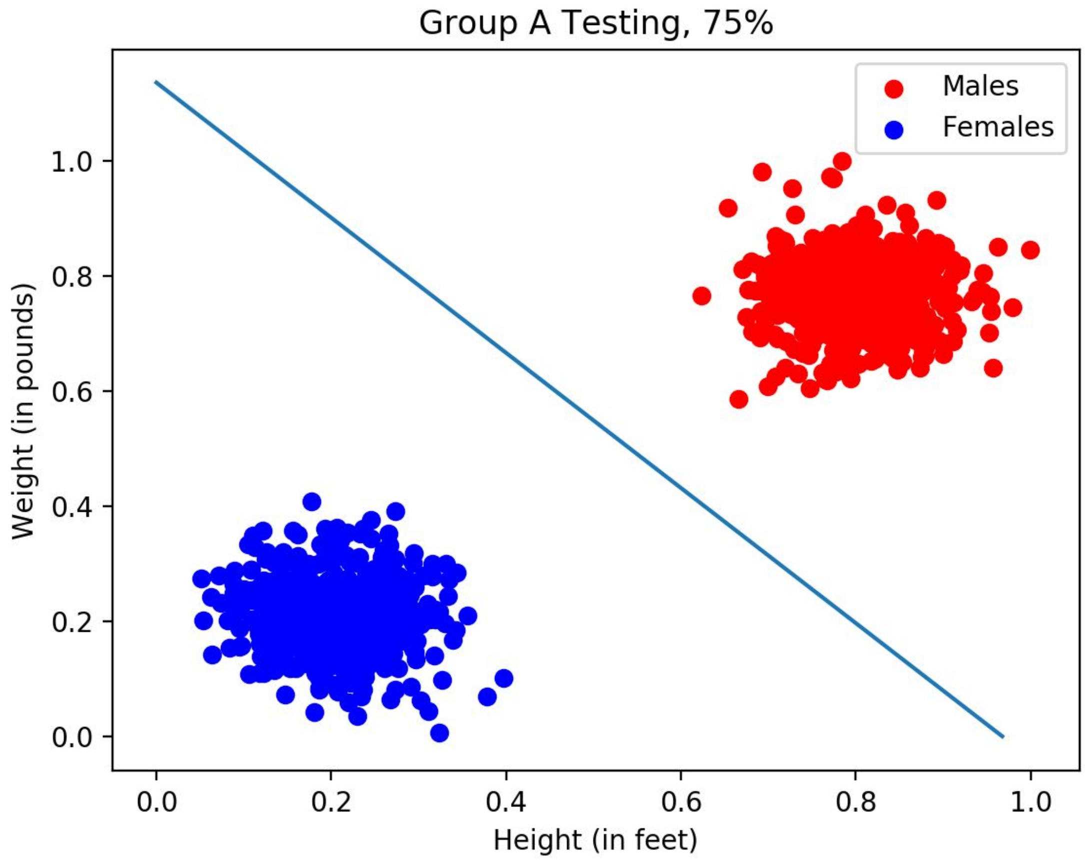


Confusion Matrix

TP = 1500	FN = 0
FP = 0	TN = 1500

Perceptron Stopping Error (**ε**): 0.0
Iterations: 1
weight of x: -2.15052490451637
weight of y: -1.8320065703336914
weight of bias: 2.080592438881662

Accuracy = 1.0
Error = 0.0
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.0
Precision = 1.0



Confusion Matrix

TP = 500	FN = 0
FP = 0	TN = 500

Accuracy = 1.0

Error = 0.0

Recall or True positive rate (TP) = 1.0

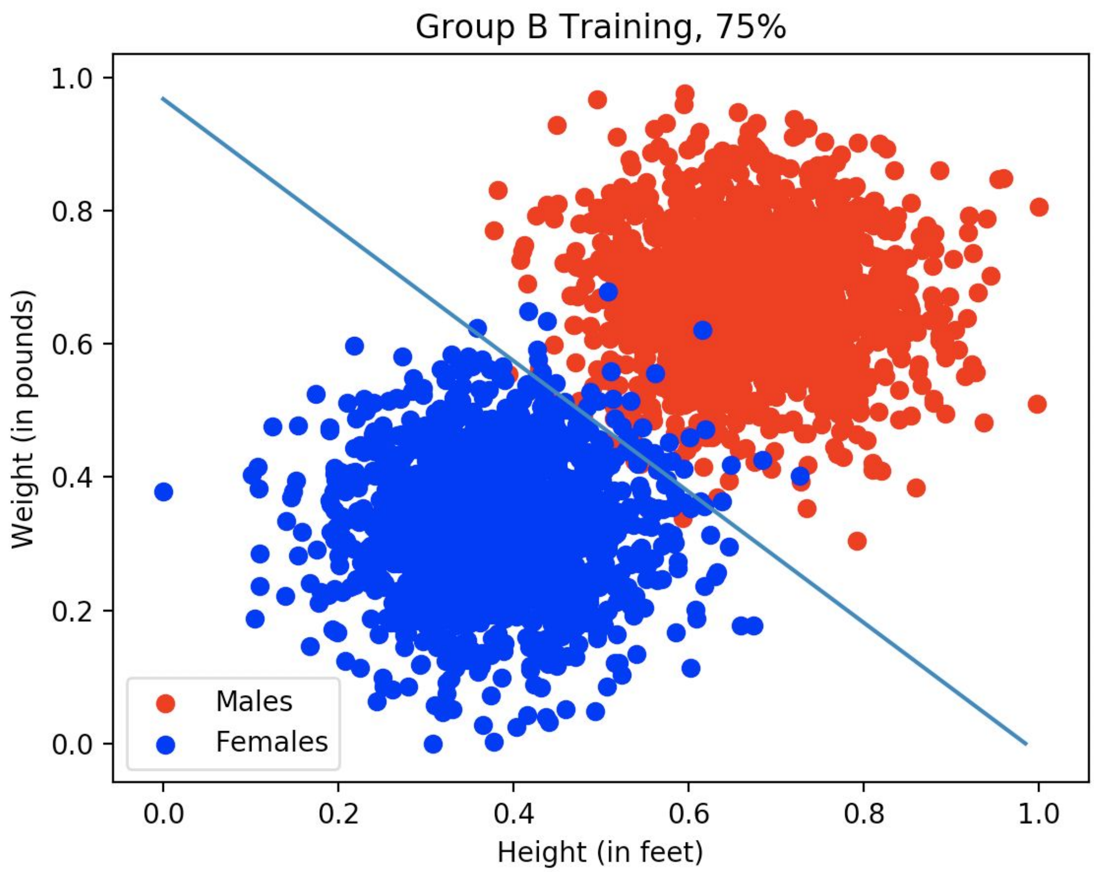
True negative rate (TN) = 1.0

False positive rate (FP) = 0.0

False negative rate (FN) = 0.0

Precision = 1.0

Plot Group B: alpha = 1, k = 3, k_range = (0.2-0.8)

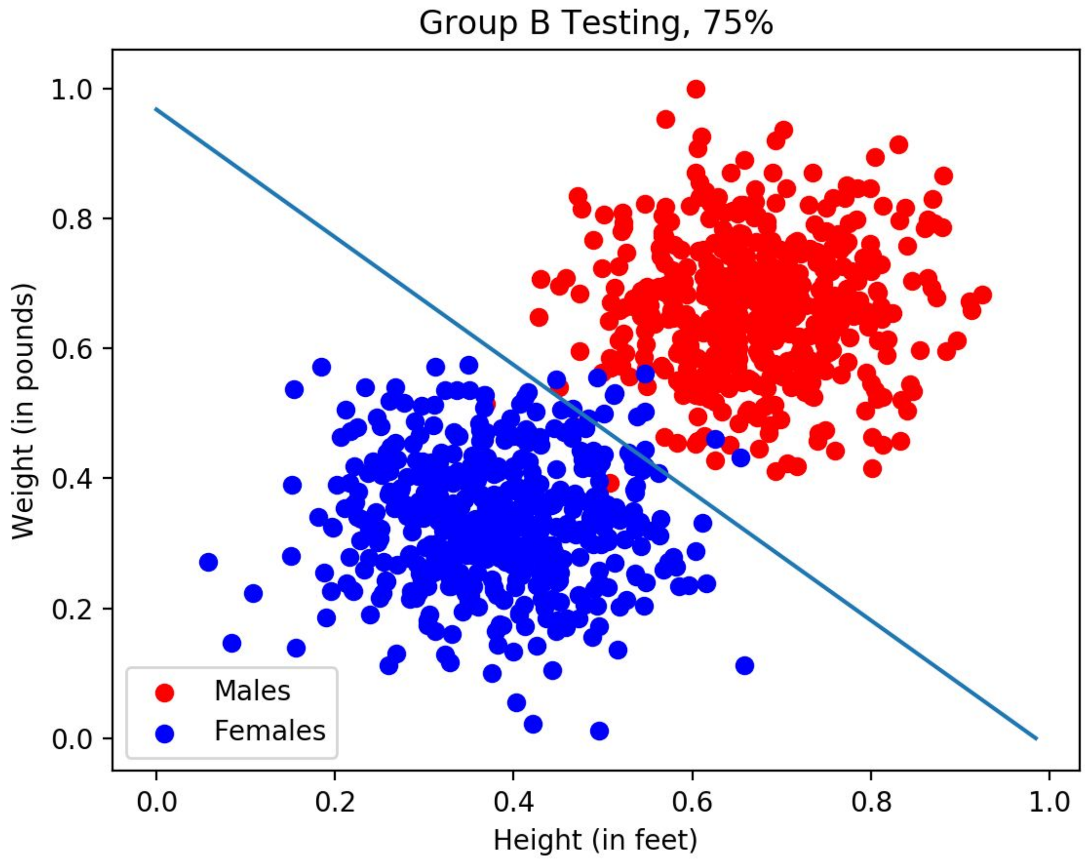


Confusion Matrix

TP = 1467	FN = 33
FP = 11	TN = 1489

Iterations: 19
weight of x: -11.415175029318068
weight of y: -11.621415417069407
weight of bias: 11.240684542216435

Accuracy = 0.985
Error = 0.015
Recall or True positive rate (TP) = 0.978
True negative rate (TN) = 0.993
False positive rate (FP) = 0.007
False negative rate (FN) = 0.022
Precision = 0.993



Confusion Matrix

TP = 489	FN = 11
FP = 3	TN = 497

Accuracy = 0.986

Error = 0.014

Recall or True positive rate (TP) = 0.978

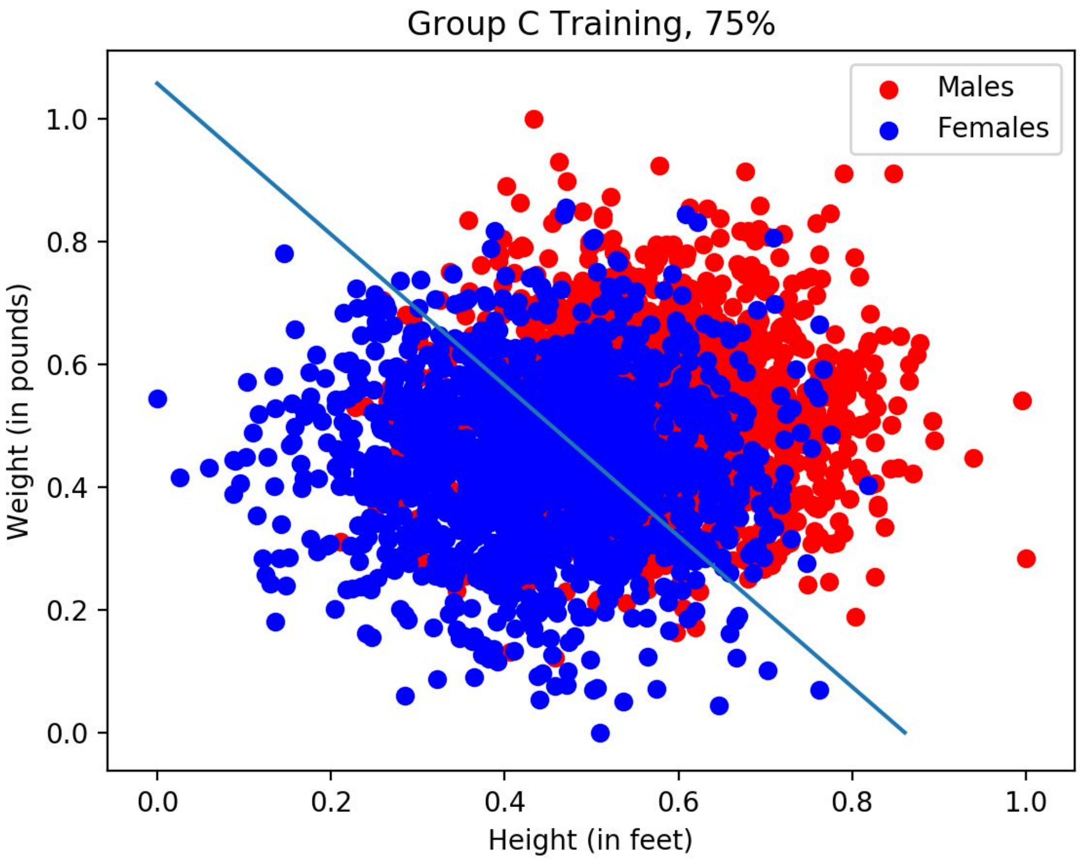
True negative rate (TN) = 0.994

False positive rate (FP) = 0.006

False negative rate (FN) = 0.022

Precision = 0.994

Plot Group C: alpha = 2, k = 3, k_range = (.2-.8)

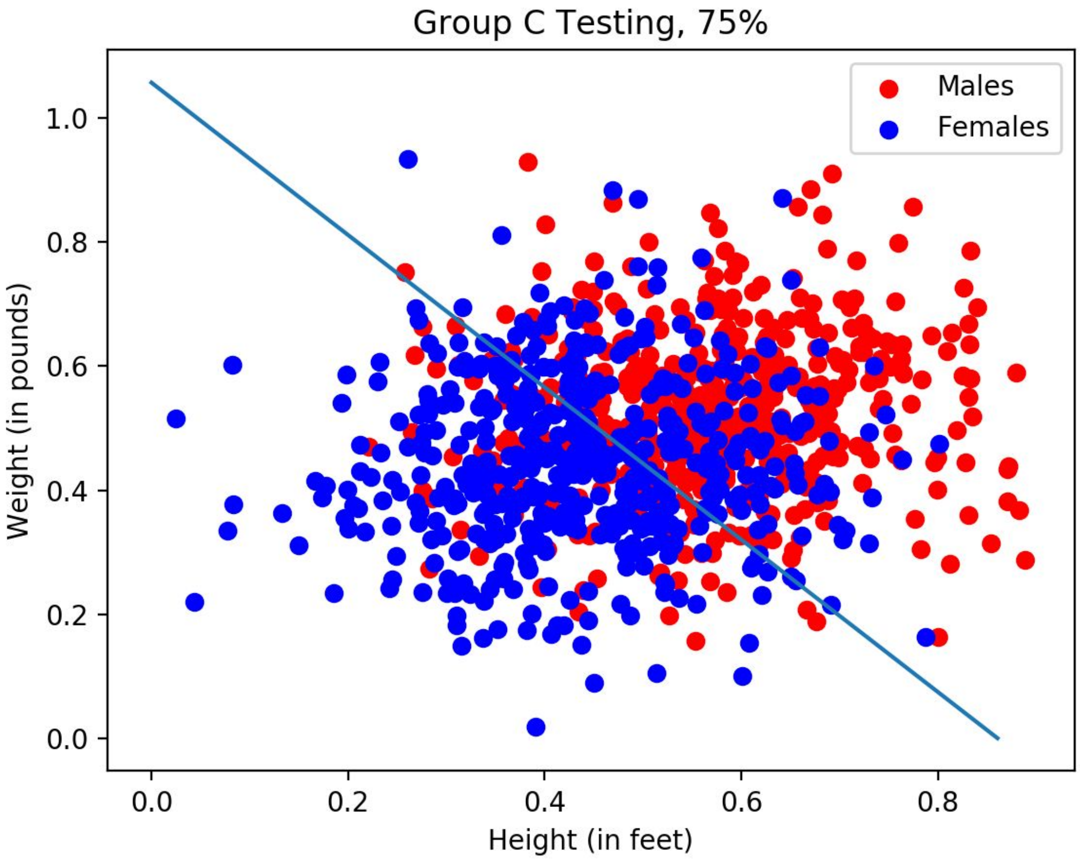


Confusion Matrix

TP = 967	FN = 533
FP = 299	TN = 1201

Perceptron Stopping Error (**ε**): 1344.0
Iterations: 0
weight of x: -8.107466119266336
weight of y: -6.600269405770846
weight of bias: 6.977830627130431

Accuracy = 0.723
Error = 0.277
Recall or True positive rate (TP) = 0.645
True negative rate (TN) = 0.801
False positive rate (FP) = 0.199
False negative rate (FN) = 0.355
Precision = 0.764



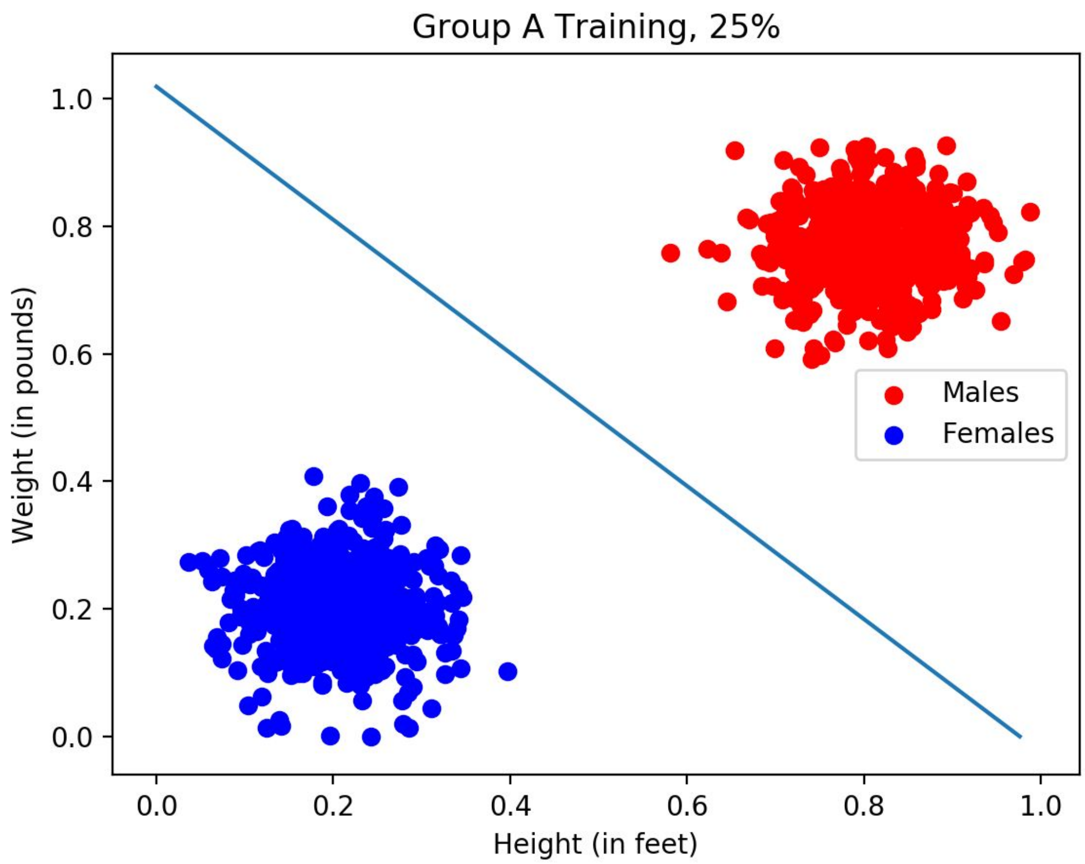
Confusion Matrix

TP = 321	FN = 179
FP = 103	TN = 397

Accuracy = 0.718
Error = 0.282
Recall or True positive rate (TP) = 0.642
True negative rate (TN) = 0.794
False positive rate (FP) = 0.206
False negative rate (FN) = 0.358
Precision = 0.757

2. Choose 25% of the data for training, and the rest for testing. Train and test your neuron. Plot the data and decision line for training and testing data (separately). Calculate errors for training and testing dataset.

Plot Group A:

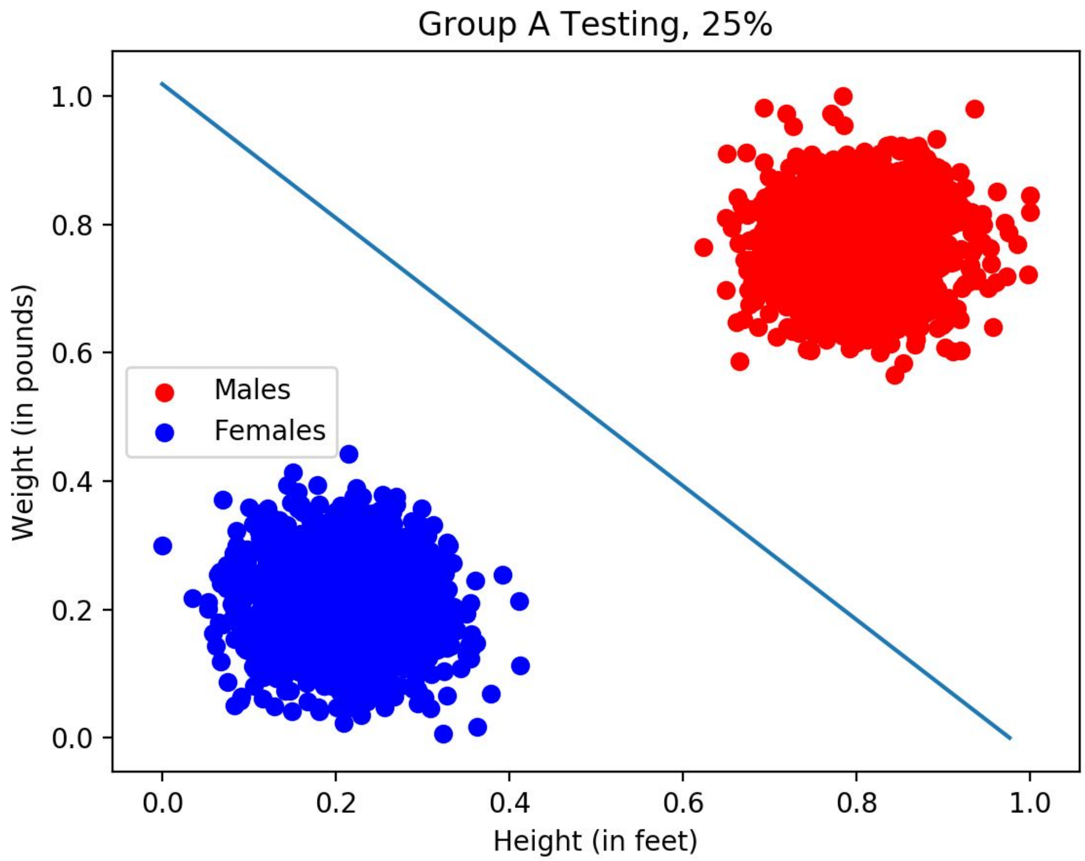


Confusion Matrix

TP = 500	FN = 0
FP = 0	TN = 500

Perceptron Stopping Error (**ε**): 0.0
Iterations: 1
weight of x: -2.239613681166906
weight of y: -2.1464439678154394
weight of bias: 2.186667983967667

Accuracy = 1.0
Error = 0.0
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.0
Precision = 1.0

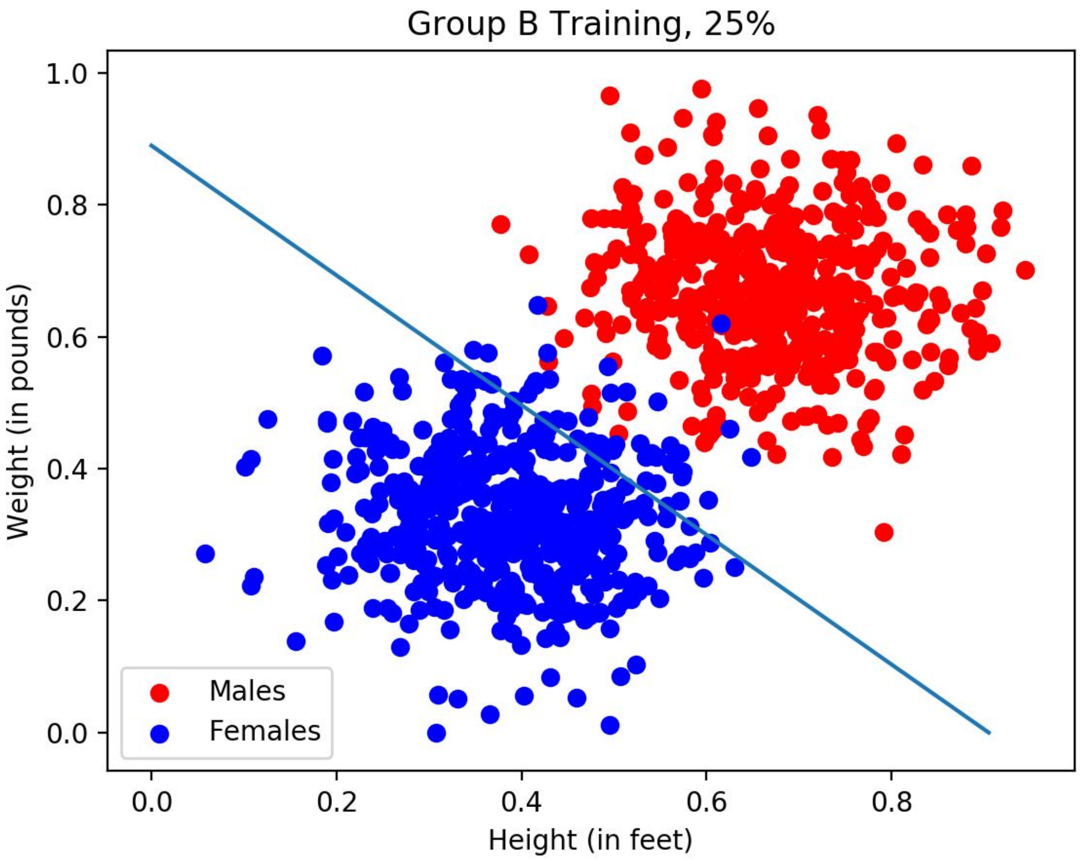


Confusion Matrix

TP = 1500	FN = 0
FP = 0	TN = 1500

Accuracy = 1.0
Error = 0.0
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.0
Precision = 1.0

Plot Group B:

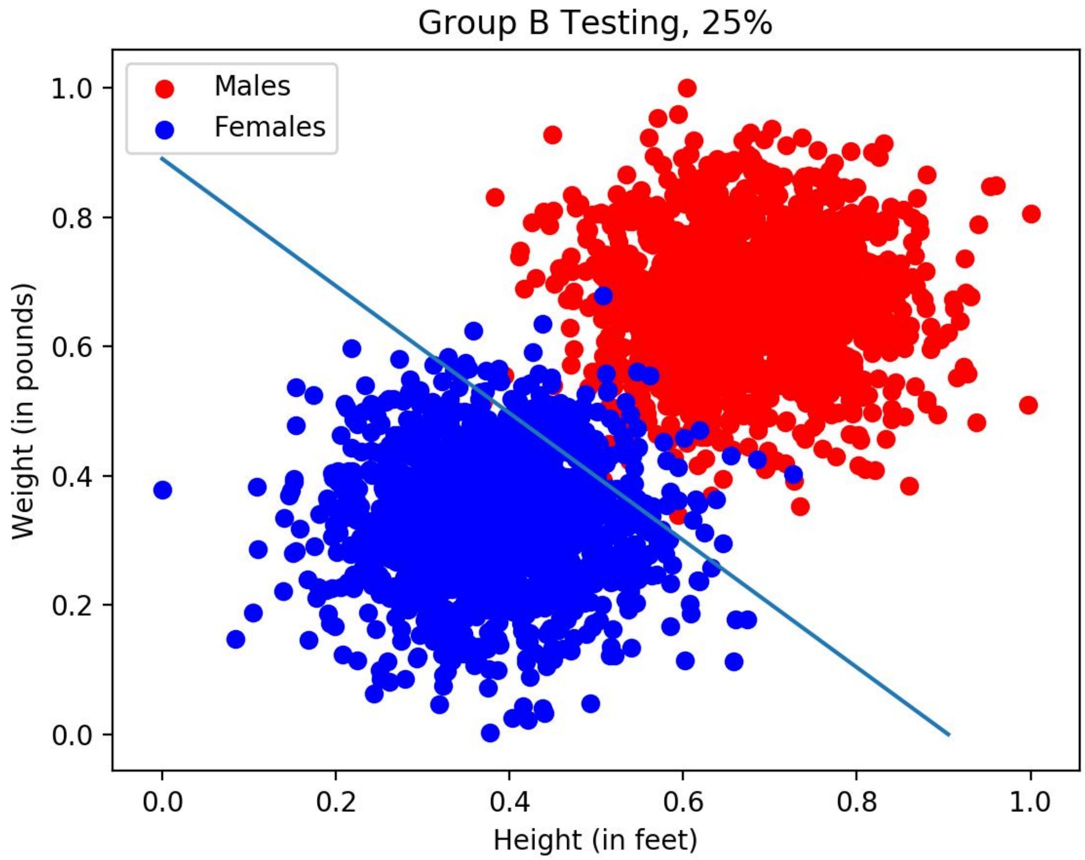


Confusion Matrix

TP = 457	FN = 43
FP = 0	TN = 500

Perceptron Stopping Error (**ε**): 80.0
Iterations: 1
weight of x: -5.397522839636957
weight of y: -5.489656190848991
weight of bias: 4.886096224967319

Accuracy = 0.957
Error = 0.043
Recall or True positive rate (TP) = 0.914
True negative rate (TN) = 1.0
False positive rate (FP) = 0.0
False negative rate (FN) = 0.086
Precision = 1.0



Confusion Matrix

TP = 1376	FN = 124
FP = 1	TN = 1499

Accuracy = 0.958

Error = 0.042

Recall or True positive rate (TP) = 0.917

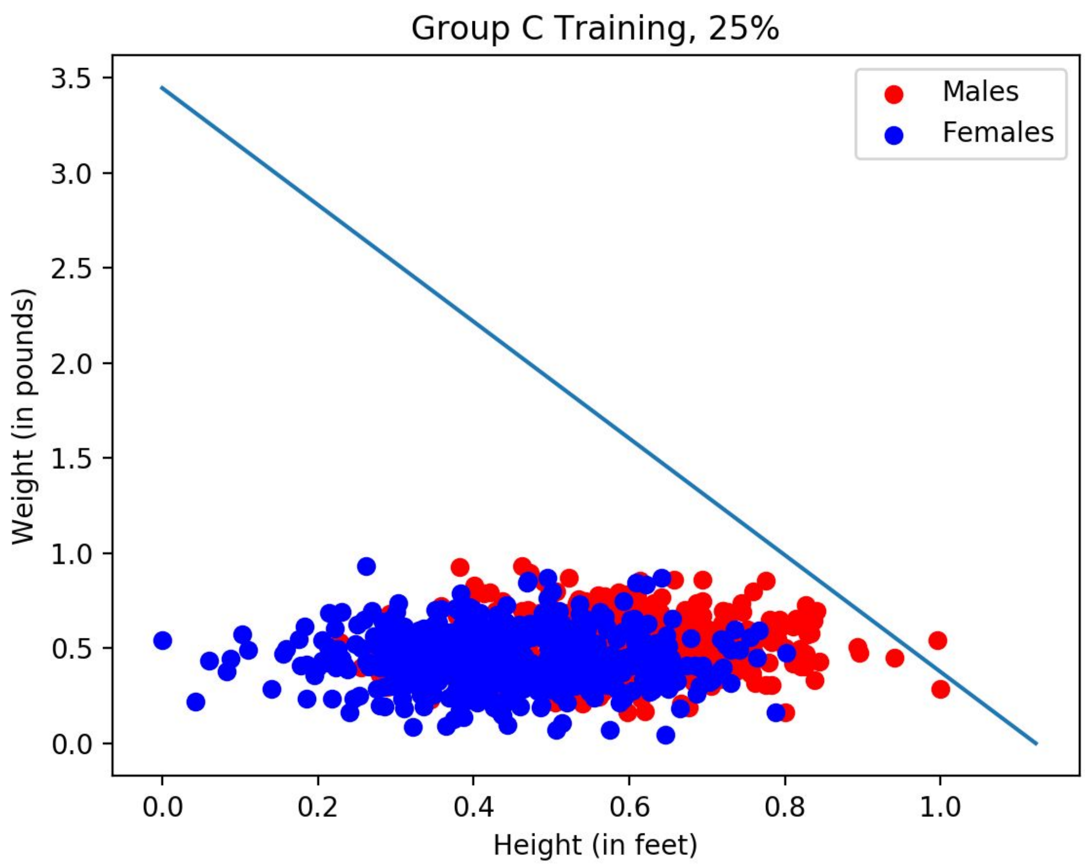
True negative rate (TN) = 0.999

False positive rate (FP) = 0.001

False negative rate (FN) = 0.083

Precision = 0.999

Plot Group C:

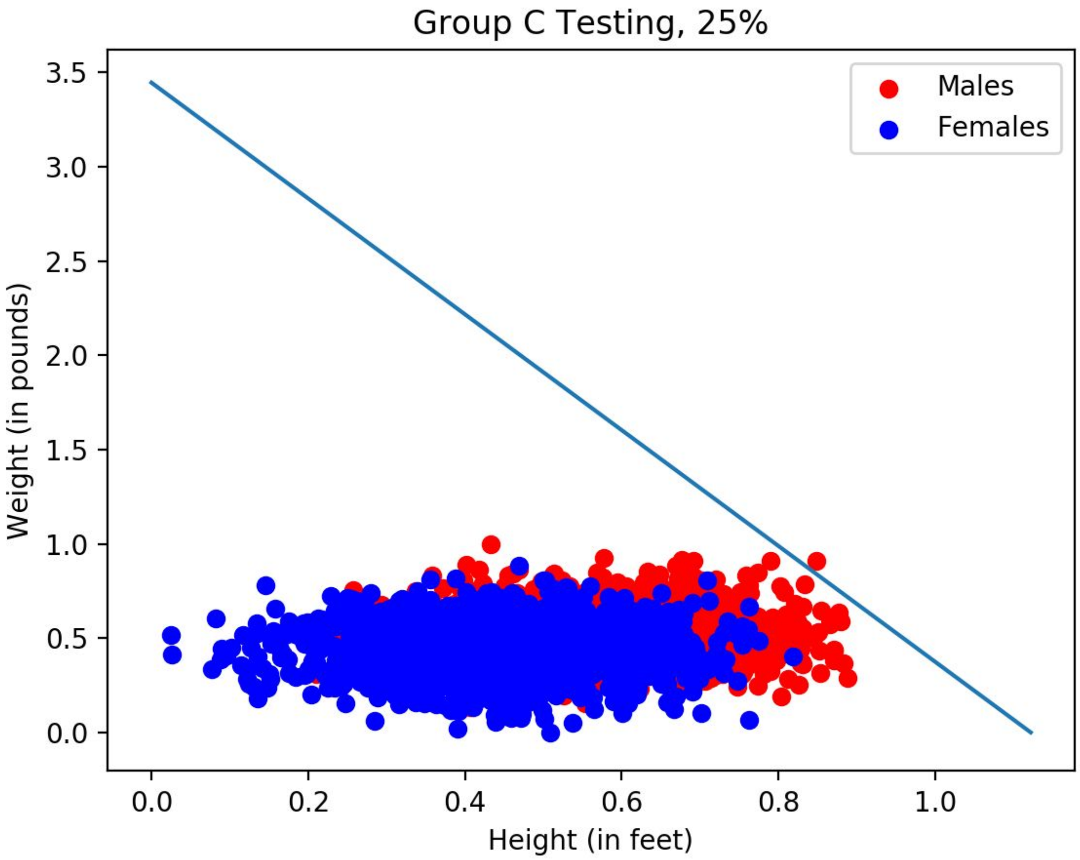


Confusion Matrix

TP = 500	FN = 0
FP = 499	TN = 1

Perceptron Stopping Error (ε): 488.0
Iterations: 0
weight of x: -6.46735315487476
weight of y: -2.1066934841158274
weight of bias: 7.2576757150504125

Accuracy = 0.501
Error = 0.499
Recall or True positive rate (TP) = 1.0
True negative rate (TN) = 0.002
False positive rate (FP) = 0.998
False negative rate (FN) = 0.0
Precision = 0.501



Confusion Matrix

TP = 1500	FN = 0
FP = 1499	TN = 1

Accuracy = 0.5

Error = 0.5

Recall or True positive rate (TP) = 1.0

True negative rate (TN) = 0.001

False positive rate (FP) = 0.999

False negative rate (FN) = 0.0

Precision = 0.5

3. Compare 1. and 2. Are errors different and if so, why? What is the effect of different data set and effect of different training/testing distributions? When would you use option 1 and when option 2 above? Comment and discuss.

- For **Group A**:
    - Group A was able to be classified with 100% accuracy whether it trained on 25% or 75% of the data.
- For **Group B**:
    - When trained on 75% of the dataset, the perceptron classified the training group 98.5% correctly and the testing group 98.6% correctly.
    - When trained on 25% of the dataset, the perceptron classified the training group 95.7% correctly and the testing group 95.8% correctly.
    - There was virtually no difference between the accuracy of the training and testing datasets. One reason for this could be that the training datasets well represents the testing datasets. It also implies that the dataset is fairly evenly distributed.
    - See comments section for why accuracy was higher when trained on more data.
- For **Group C**:
    - When trained on 75% of the dataset, the perceptron classified the training group 72.3% correctly and the testing group 71.8% correctly.
    - When trained on 25% of the dataset, the perceptron classified the training group 50.1% correctly and the testing group 50.0% correctly.
    - See comments section for why accuracy was higher when trained on more data.
- **Comments/discussion/takeaways**
    - The perceptron was more accurate when trained on 75% of the dataset, which is for the same reason the hard activation functions were more accurate when they had more data to train on. 25% of the data just wasn't representative enough of the dataset as a whole
        - (but remembering that using 25% of the dataset could be useful in some scenarios -- i.e., more unknown data than known data)
    - This relationship between training size vs testing sizes is very well represented in the differences in accuracy for group C

Pr. 2.2 Soft vs. hard activation function
Compare and discuss results when hard activation was used vs. when soft activation was used. Comment for each training/testing distribution 1, 2, and 3.

Dataset	Hard	Soft	Percent Training	Training Accuracy	Testing Accuracy
Group A	x		75%	1.0	1.0
Group A	x		25%	1.0	1.0
Group A		x	75%	1.0	1.0
Group A		x	25%	1.0	1.0
Group B	x		75%	0.980	0.997
Group B	x		25%	0.926	0.941
Group B		x	75%	0.985	0.986
Group B		x	25%	0.957	0.958
Group C	x		75%	0.606	0.604
Group C	x		25%	0.501	0.500
Group C		x	75%	0.723	0.718
Group C		x	25%	0.501	0.5

Group A:
- The activation function did not affect Group A. For data that is so clearly clustered, using a hard activation function will suffice and it will reduce computational complexity and keep things simple.

Group B:
- The soft vs hard activation function only marginally affected the classification accuracy for Group B. The soft activation function function increased accuracy just barely enough

to be noticeable. Still, since the data is in fairly distinct clusters, the hard activation function is able to get the job done virtually just as well as the soft activation function.

Group C:

- With Group C we are able to see where the soft activation function is able to play a big role in increasing classification accuracy. The testing accuracy for the hard activation function trained on 75% of data was 60.4%, but the soft activation function was able to predict with 71.8% accuracy.
- This increase in accuracy is the most significant out of the entire project.
- The perceptron with the hard activation function isn't able to learn from all of the intricacies, but the perceptron with the soft activation function can.
- When learning from data that isn't in obvious clusters, the soft activation function is very useful.
    - Data in Group C is more representative of real world data that needs to be classified by perceptrons. It's not very common for data to be so clustered like in Groups A and B. This heightens the importance of the soft activation function.