Name: *Souham Biswas* ***CWID***: *A20365242* ***Email***: *sbiswas7@hawk.iit.edu*

CS 584 – Machine Learning, HW 3

1. Problem Statement-

Our objective in this assignment is to implement various algorithms for discriminative learning.

2. Proposed Solution-

The various algorithms utilized in this assignment include mainly, 2 Class Logistic Regression, K Class Logistic Regression and Artificial Neural Networks.

3. Implementation Details-

We utilize the scikit-learn machine learning python library for the purposes of calculating various metrics like precision, accuracy etc in this assignment along with the matplotlib library for plotting and visualization of various observations and metrics we encounter throughout the experiments. Also, the library idx2numpy has been used to extract the MNIST dataset to numpy arrays from their original .idx format.

Multiple challenges were faced in performing the experiments. A few of them are highlighted below-

- Figuring out a proper stop condition was a challenge as both overfitting and underfitting were to be avoided
- Smart list comprehension techniques had to be developed and implemented as traditional python loops are slow in execution.
- Parsing the raw data was an interesting challenge. The data had to be analyzed for structure so as to devise the most efficient techniques and data structures to handle them.
- While implementing the neural network algorithms, smart techniques to vectorize the data to speed up execution using mostly matrix multiplications had to be devised which resulted in quick execution of the training code.

Instructions for using the code-

Simply double click the code file and everything should run smoothly. Do note that the code needs an active internet connection to fetch the datasets from the UCI repository. Also, during the execution of the code, some overflow/infinite etc warnings might be thrown. It has been ensured that they do not affect the results hence, there is no matter of concern for the same.

Only the MNIST data files have been included as the others are directly fetched from the internet.

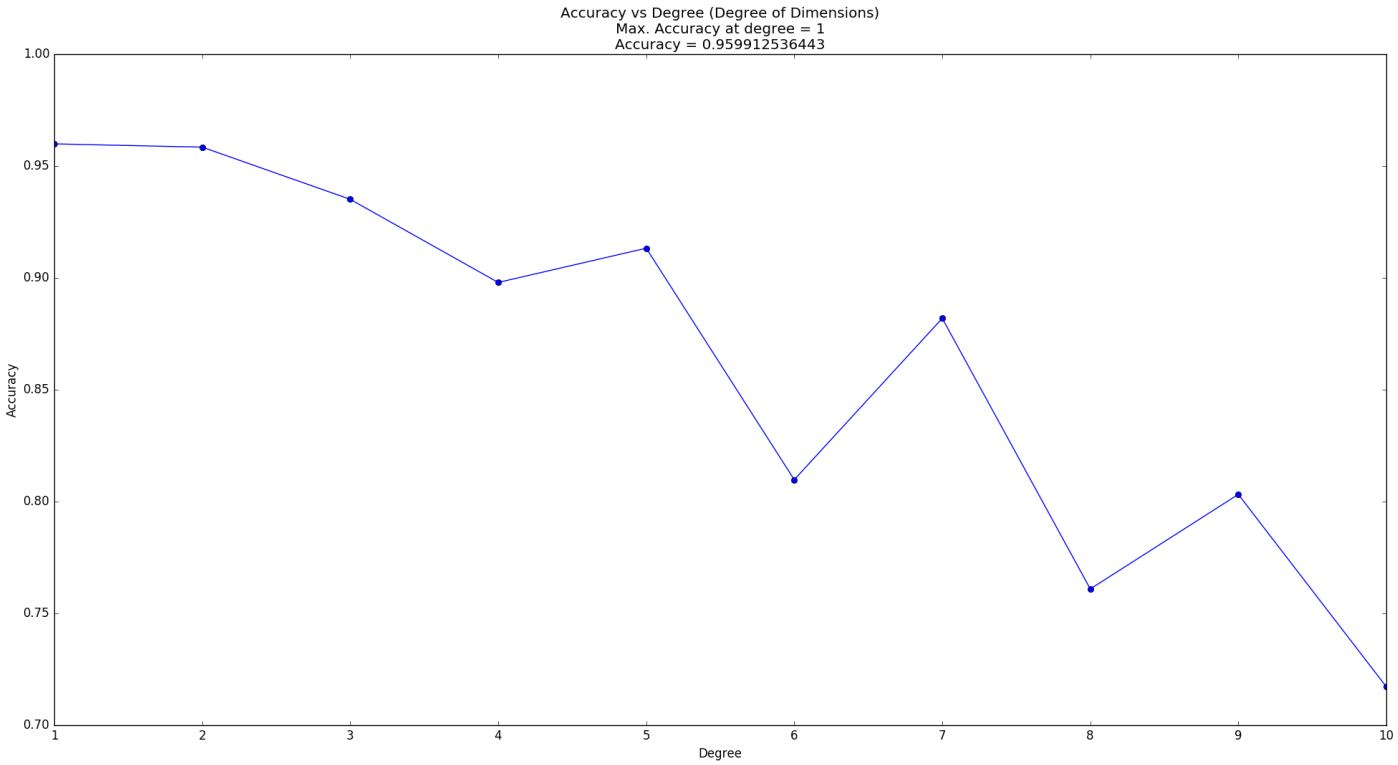
4. Results and Discussion-

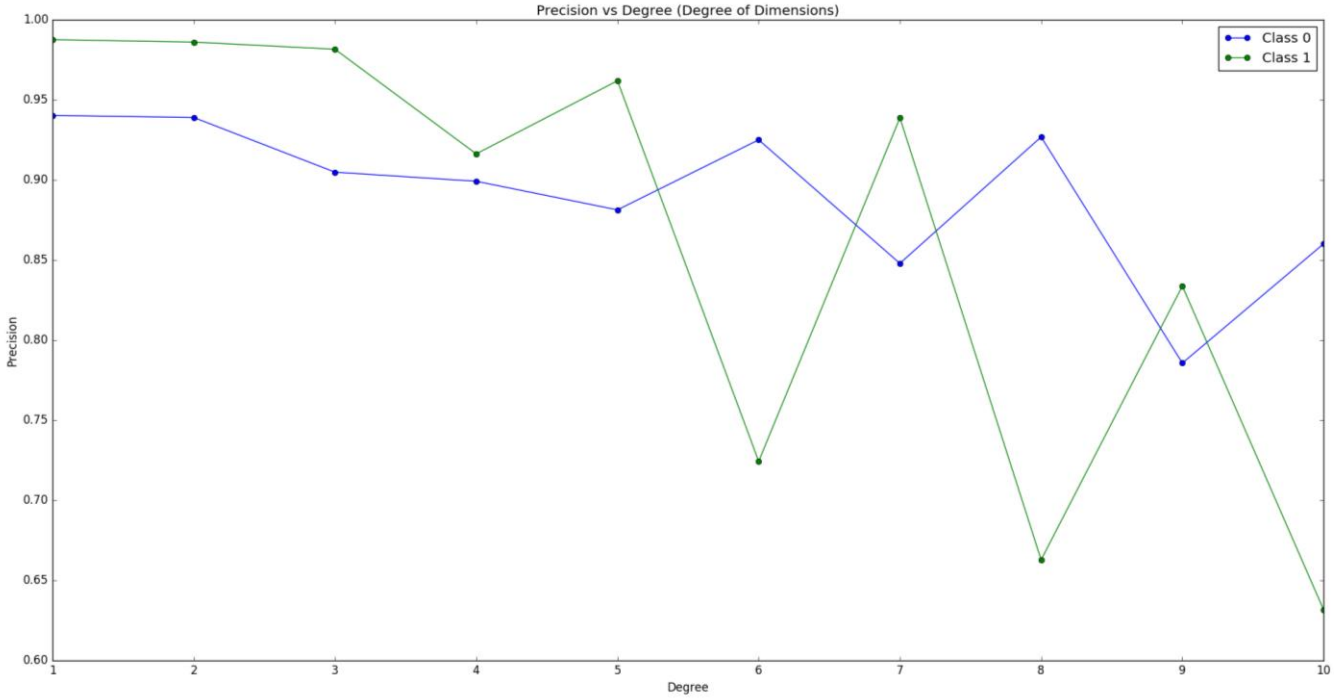
All the experiments have been performed with 4 fold cross validation. The results shown below are the means of the metrics obtained from each of the cross validation iterations.

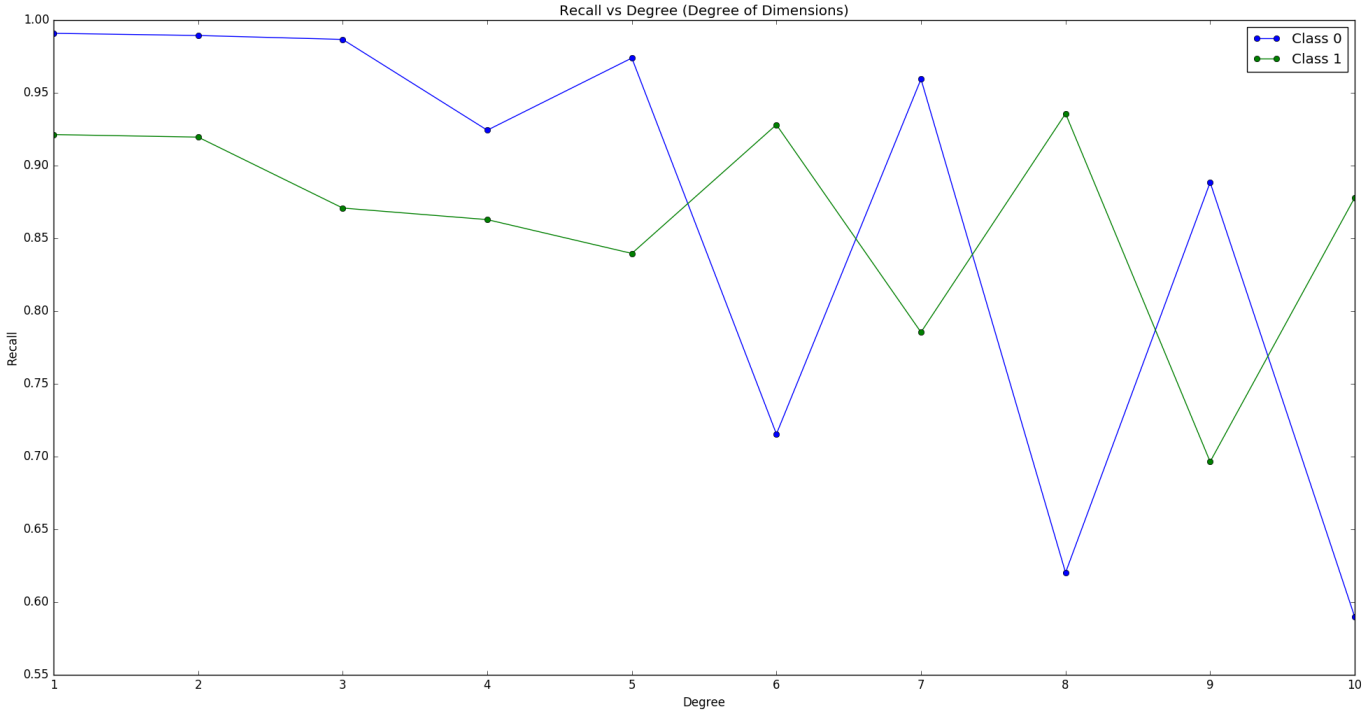
Answer 1-

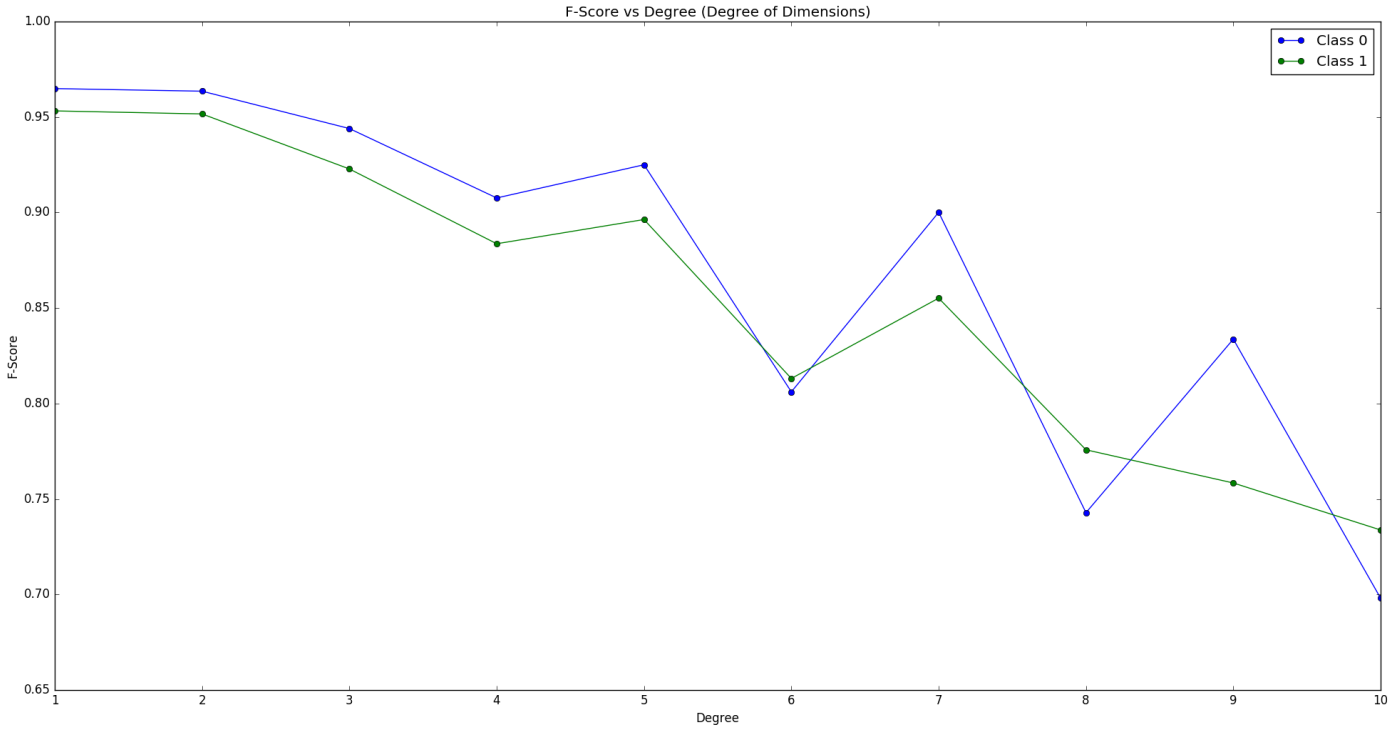
2 Class-

The following results were obtained-




Precision vs Degree (Degree of Dimensions)
Class 0
Class 1
Precision
Degree
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
1
2
3
4
5
6
7
8
9
10


Recall vs Degree (Degree of Dimensions)
Class 0
Class 1
Recall
Degree
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
1
2
3
4
5
6
7
8
9
10



The dataset used here is the Bank Note dataset in which specific image features are mapped onto 2 labels denoting a fake currency and an original currency note.

As we can see, as the degree increases, all the metrics (accuracy, precision, recall & f score) tend to decrease. The reason can be attributed to possible overfitting as the reduced performance is being caused by an increase in the number of features being used for the classification.

K-Class-

The following results were obtained-

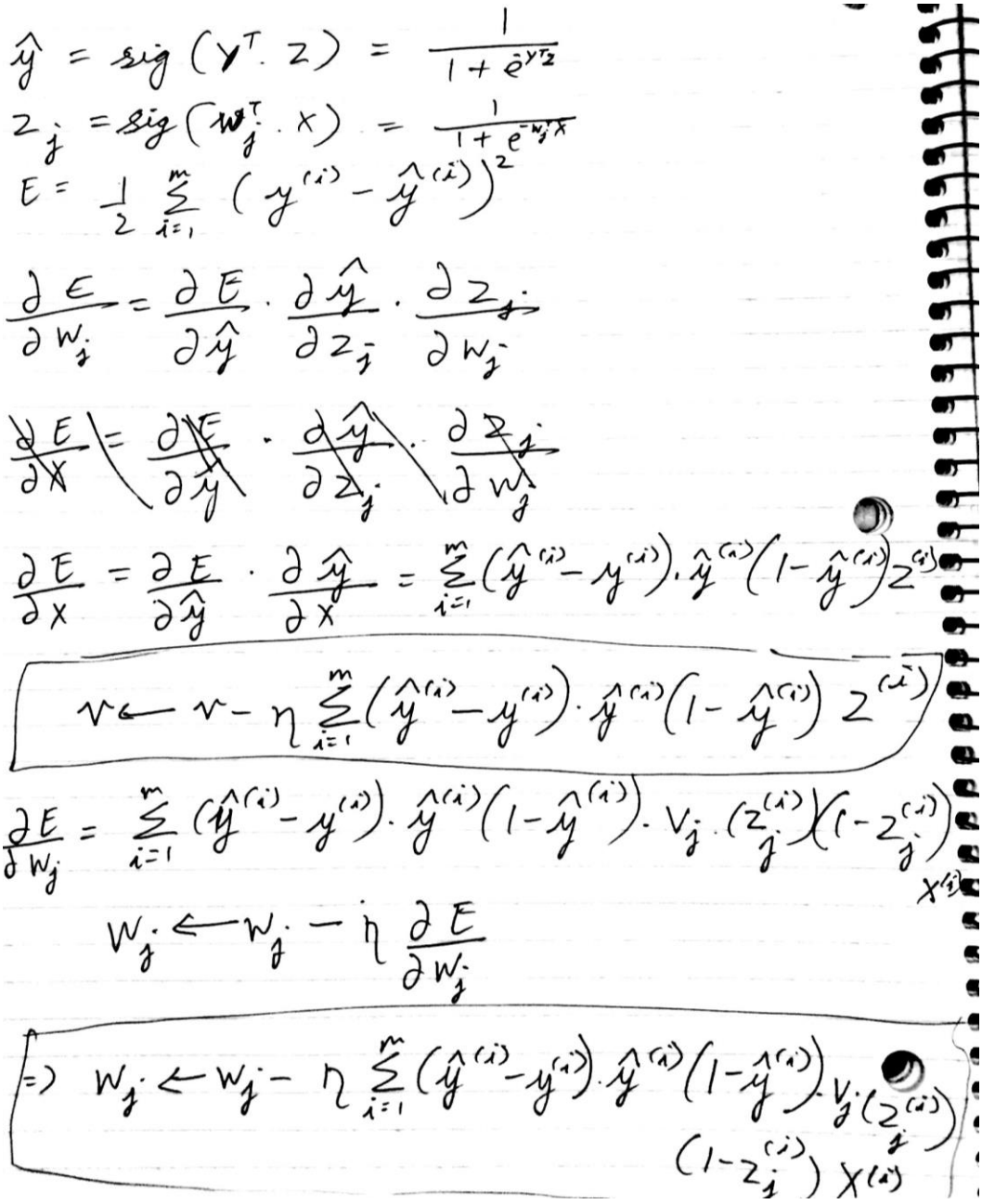
```
Performing experiments on k class classifier on Iris Dataset
Best Degree = 2
Best Accuracy = 0.666607396871
Best Precisions = [ 1.          0.2          0.49810606]
Best Recalls = [ 1.  0.3  1.]
Best F-Scores = [ 1.         0.45         0.6617894]
```

The dataset used here is the Iris dataset which maps characteristics of various Iris plants to their respective class label. The labels are namely "Iris Setosa", "Iris Versicolour" & "Iris Virginica".

The performance, as we can see is not very impressive. This is mainly because the classifier is not performing well when it comes to identifying class 1. An approximate guess behind this behavior would be that this is being caused by the very nature of the data.

Also, it can be observed that the best performance is obtained at degree 2. The classifier tends to throw many overflow warnings at degrees greater than 2 and the performance decreases.

Answer 2(a)-



Answer 2(b)-

The following results were obtained for a Neural Network trained & tested on the MNIST dataset of handwritten digits with `momentum=.0001, learn_rate=.0005`

Here, the classifier tries to map 28 x 28 resolution grayscale images of handwritten digits between 0 and 2 (3 class) to their respective numeric labels. The experiments were conducted with various sizes of the hidden layer. The sizes used were 50, 125 and 200. The results are as follows-

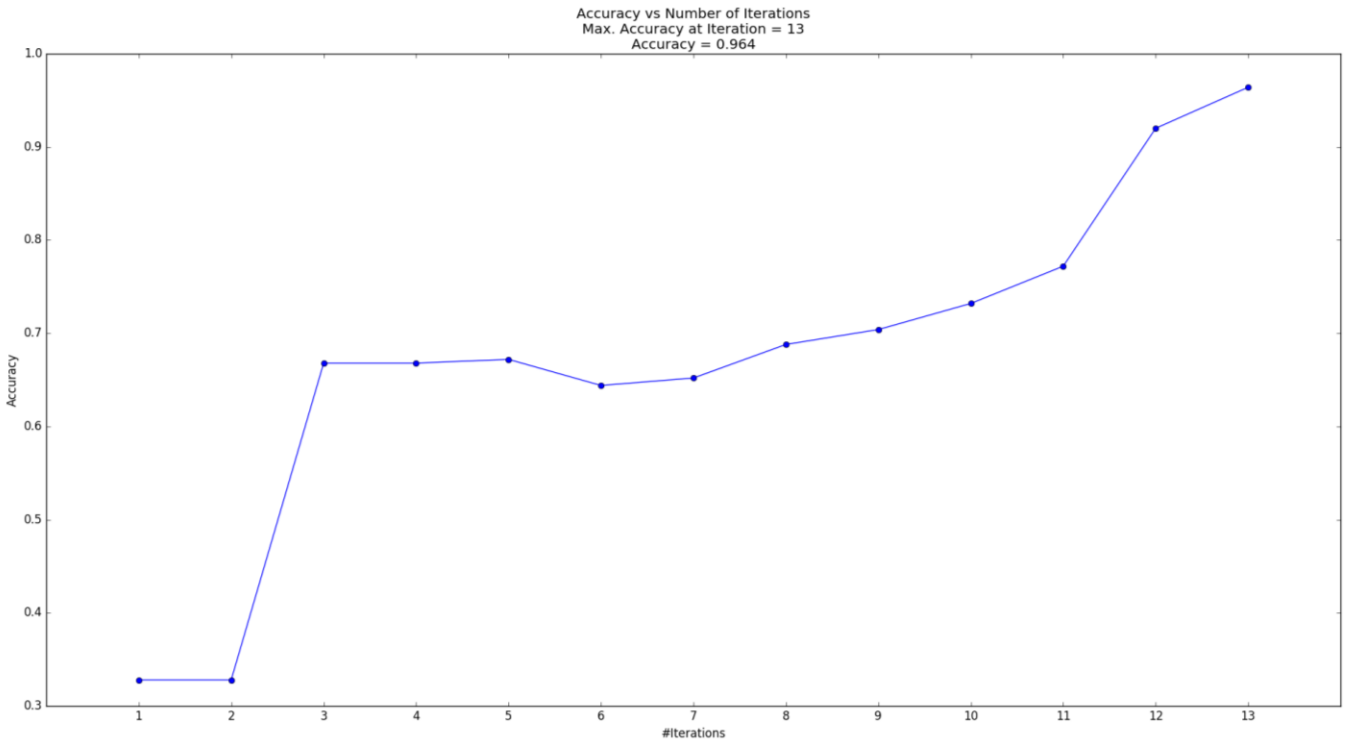
```
Performing Neural Network experiments on MNIST dataset

Hidden Units = 200
Accuracy = 0.924
Precision (3 Classes) = [ 0.86363636  0.99029126  0.89830508]
Recall (3 Classes) = [ 0.98701299  0.95327103  0.8030303 ]
F-Score (3 Classes) = [ 0.92121212  0.97142857  0.848     ]
Confusion Matrix (3 Classes) =
[[ 76   0   1]
 [  0 102   5]
 [ 12   1  53]]

Hidden Units = 125
Accuracy = 0.928
Precision (3 Classes) = [ 0.92391304  0.9875      0.87179487]
Recall (3 Classes) = [ 0.90425532  0.9875      0.89473684]
F-Score (3 Classes) = [ 0.91397849  0.9875      0.88311688]
Confusion Matrix (3 Classes) =
[[85  0  9]
 [ 0 79  1]
 [ 7  1 68]]


Hidden Units = 50
Accuracy = 0.964
Precision (3 Classes) = [ 0.98666667  0.98734177  0.92708333]
Recall (3 Classes) = [ 0.96103896  0.95121951  0.97802198]
F-Score (3 Classes) = [ 0.97368421  0.9689441   0.95187166]
Confusion Matrix (3 Classes) =
[[74  0  3]
```

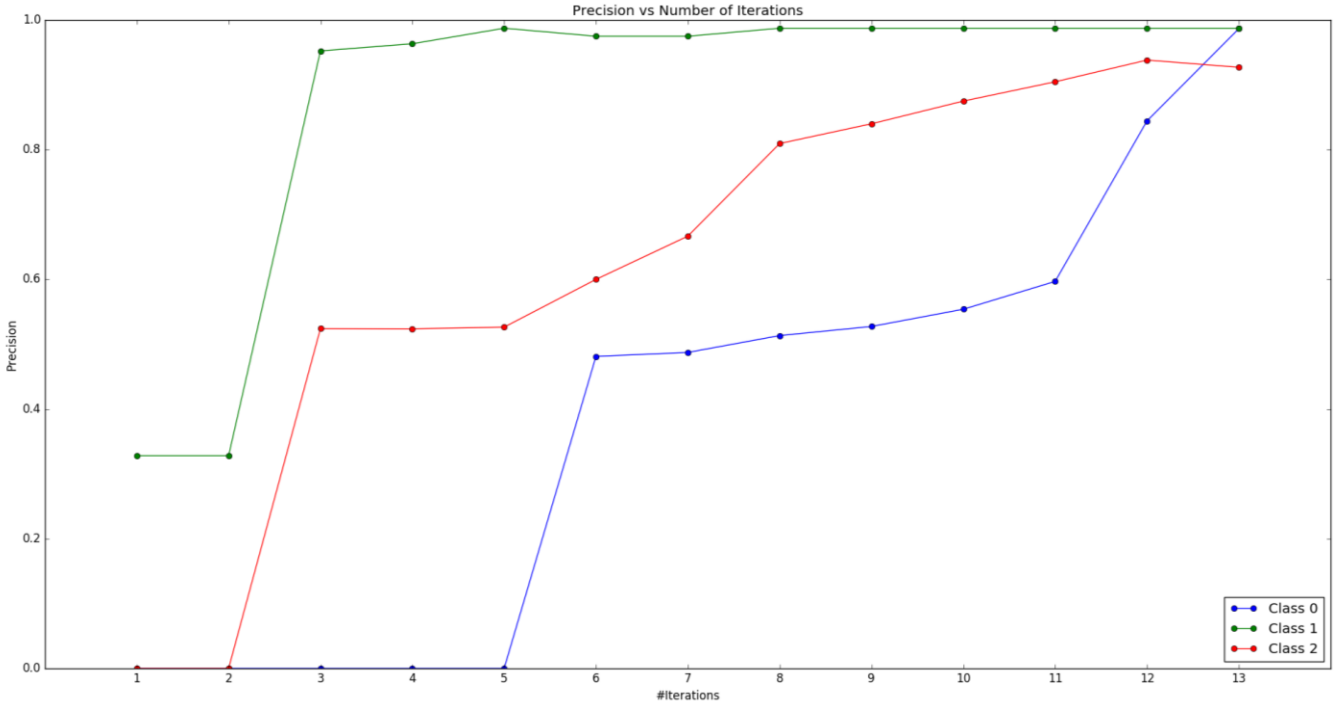
```
[ 0 78  4]
[ 1  1 89]]
```

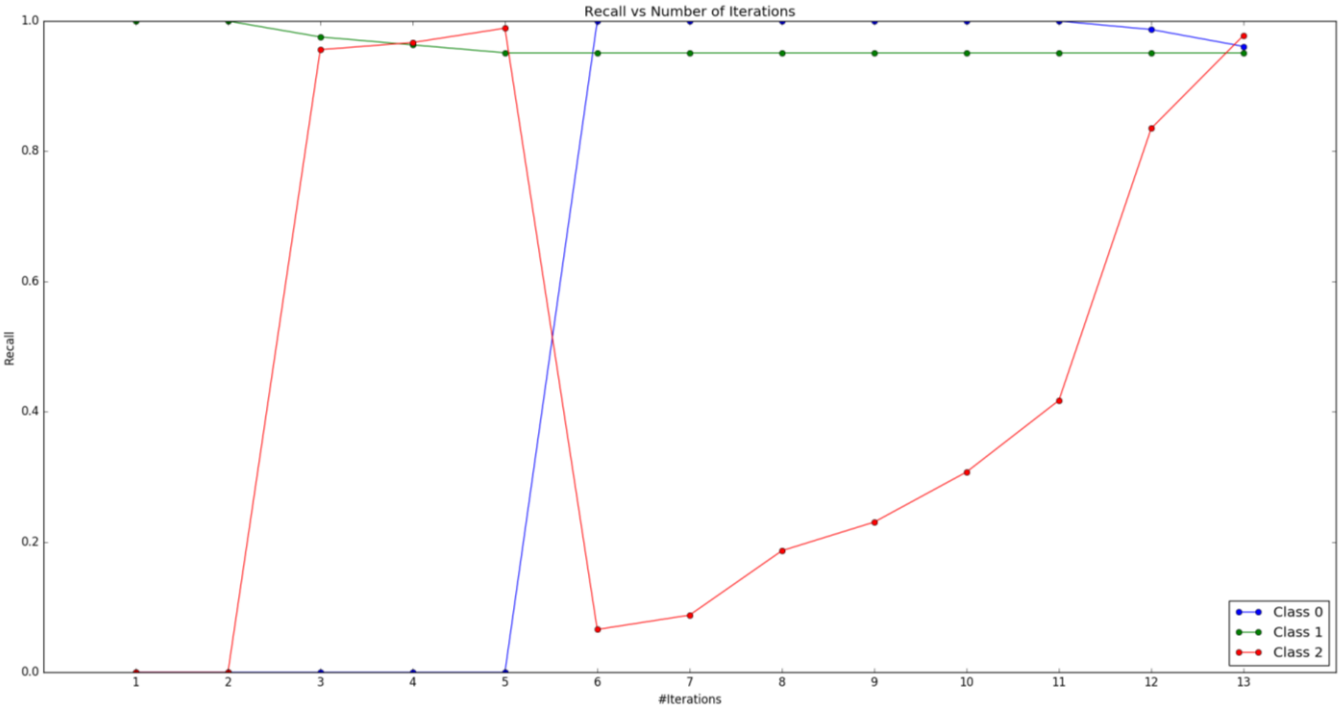
As we can observe from the results above, the best performance was obtained with a hidden layer of 50 neurons. The accuracy and other metrics reduce with the increase in the hidden layer size. A possible reason behind this could be overfitting. For the purposes of this experiment, only 1000 observations have been considered.

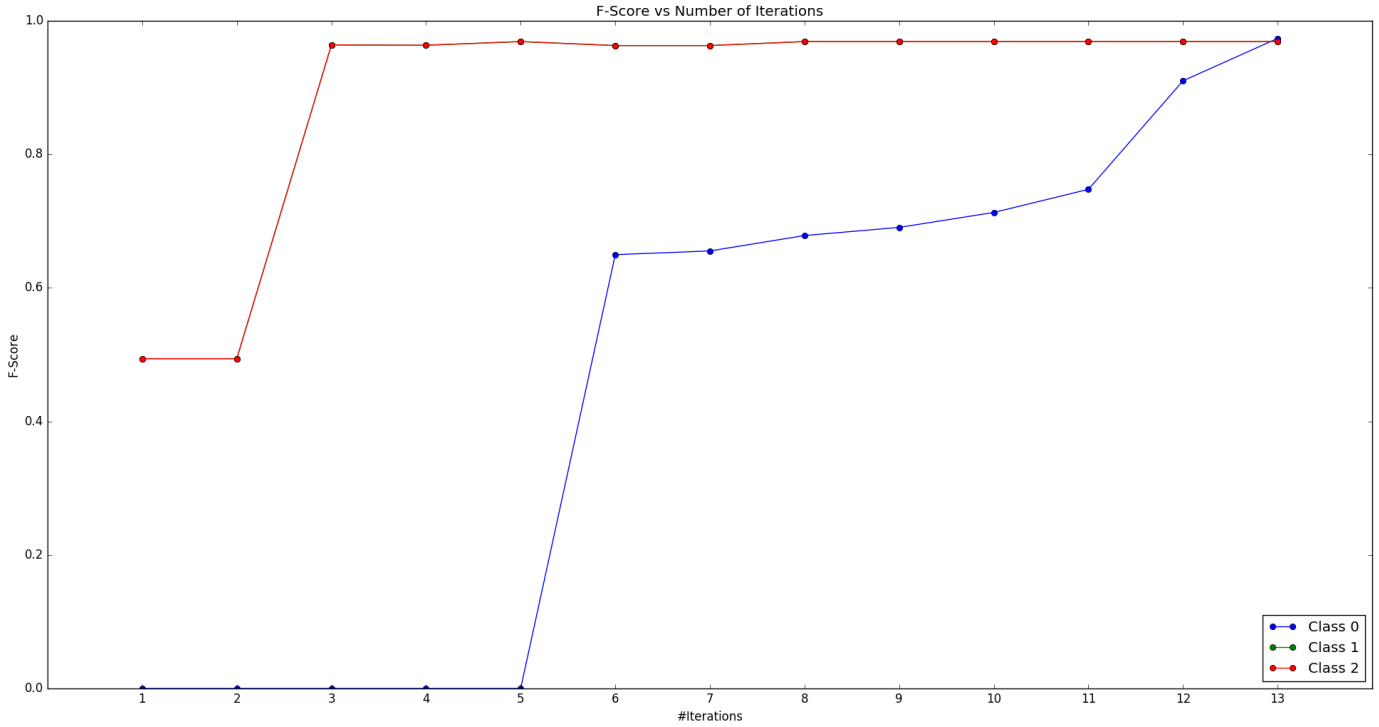
The various graphs pertaining to different sizes of the hidden layers are presented below-

#Hidden Units = 50 -

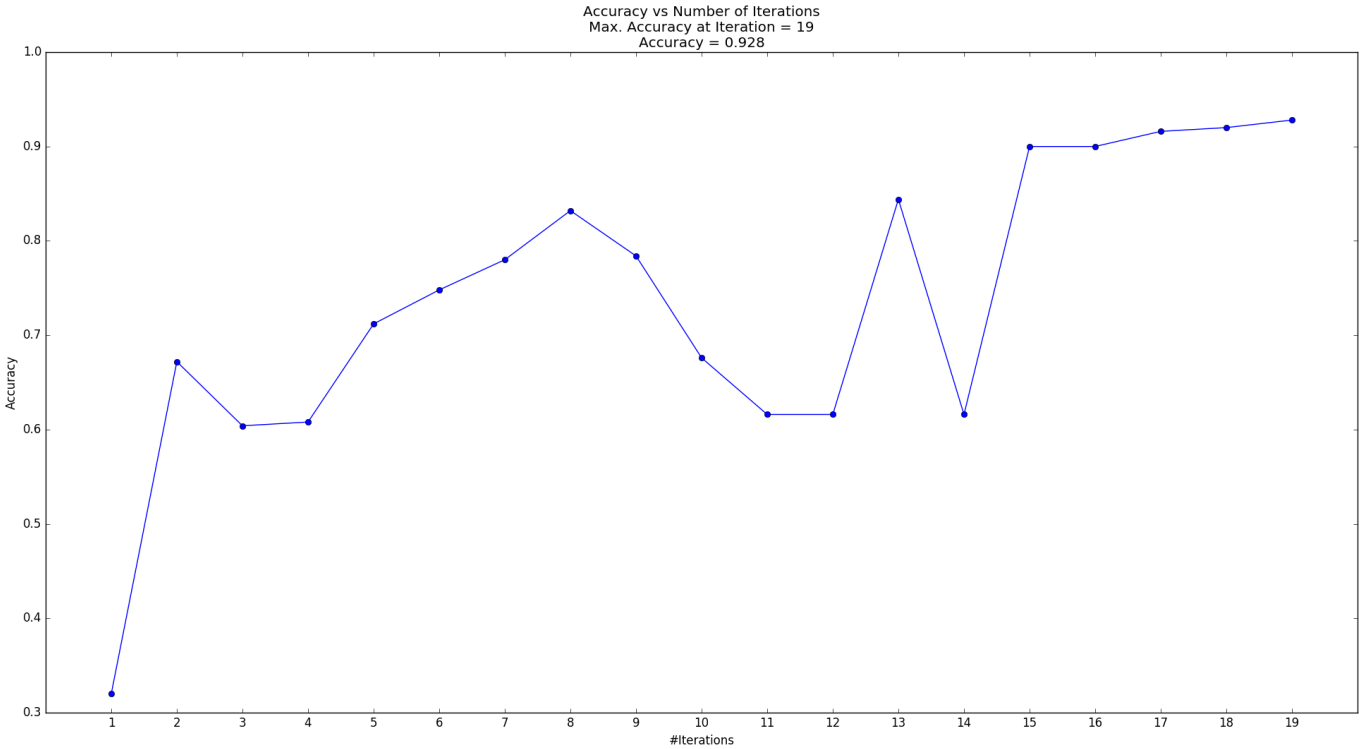


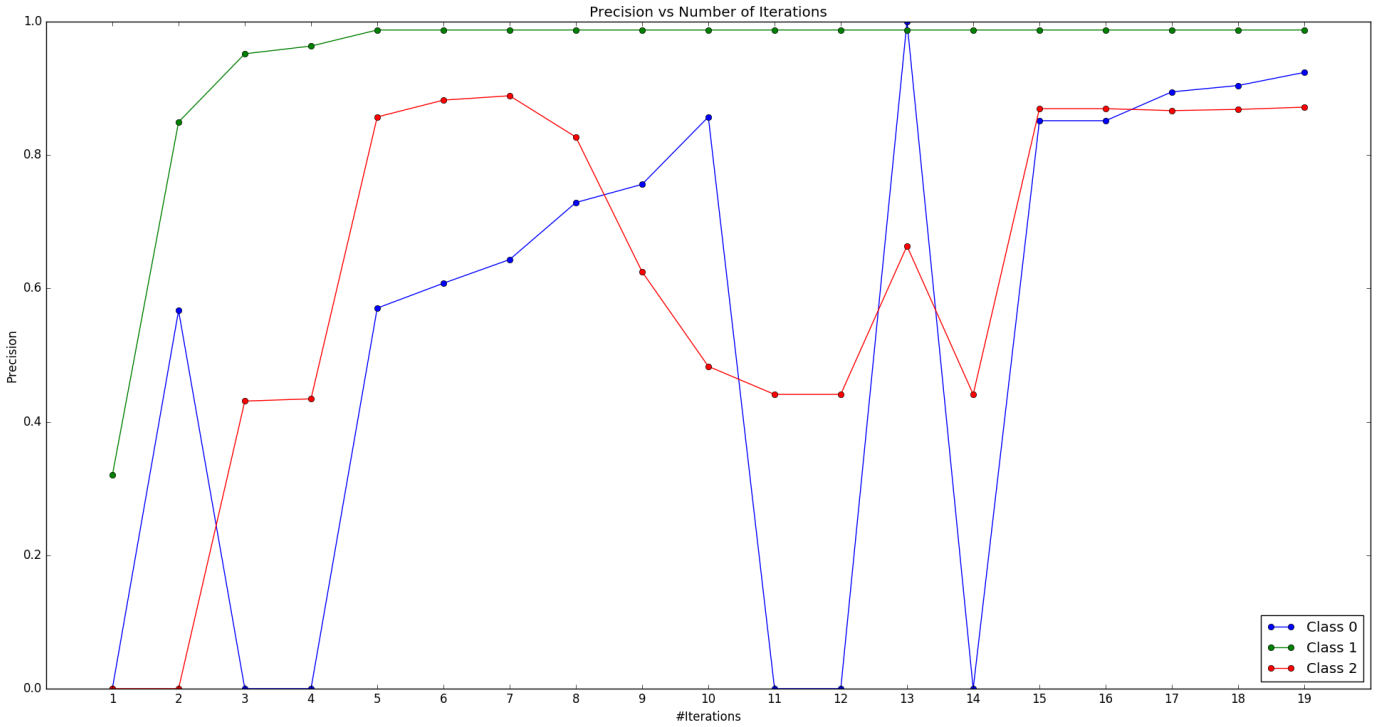

Precision vs Number of Iterations
Precision
#Iterations
Class 0
Class 1
Class 2

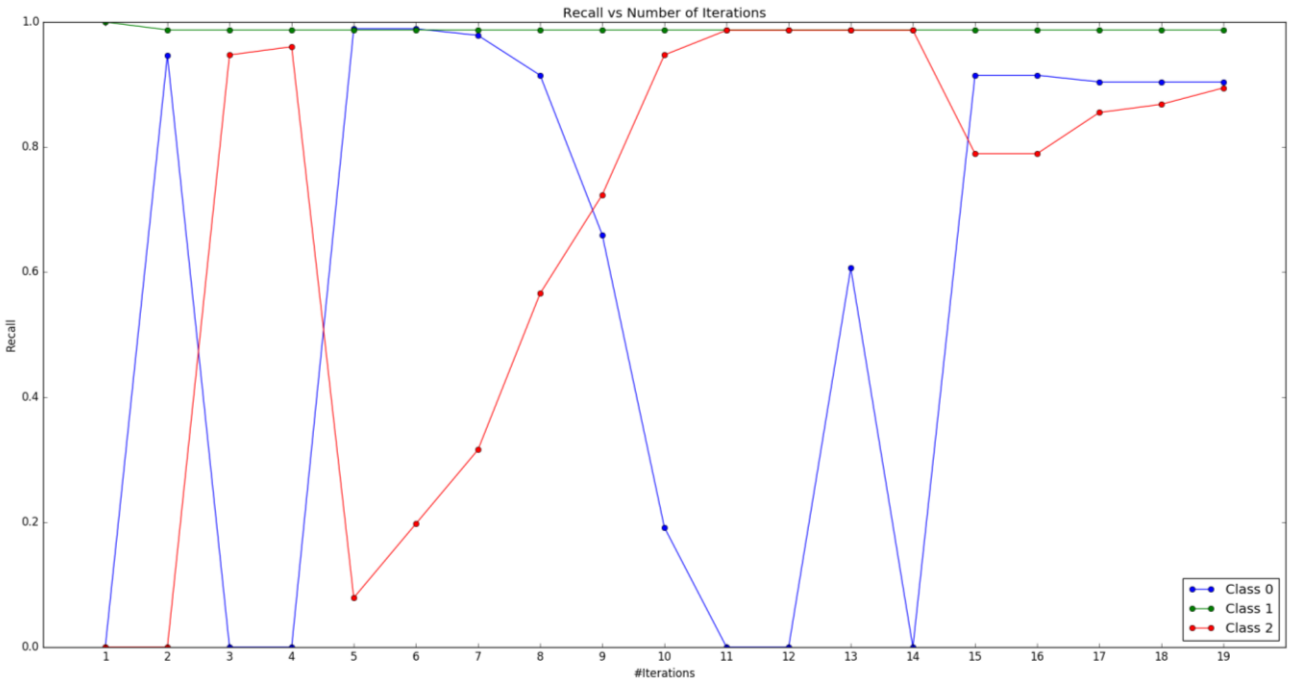

Recall vs Number of Iterations
Recall
#Iterations
Class 0
Class 1
Class 2

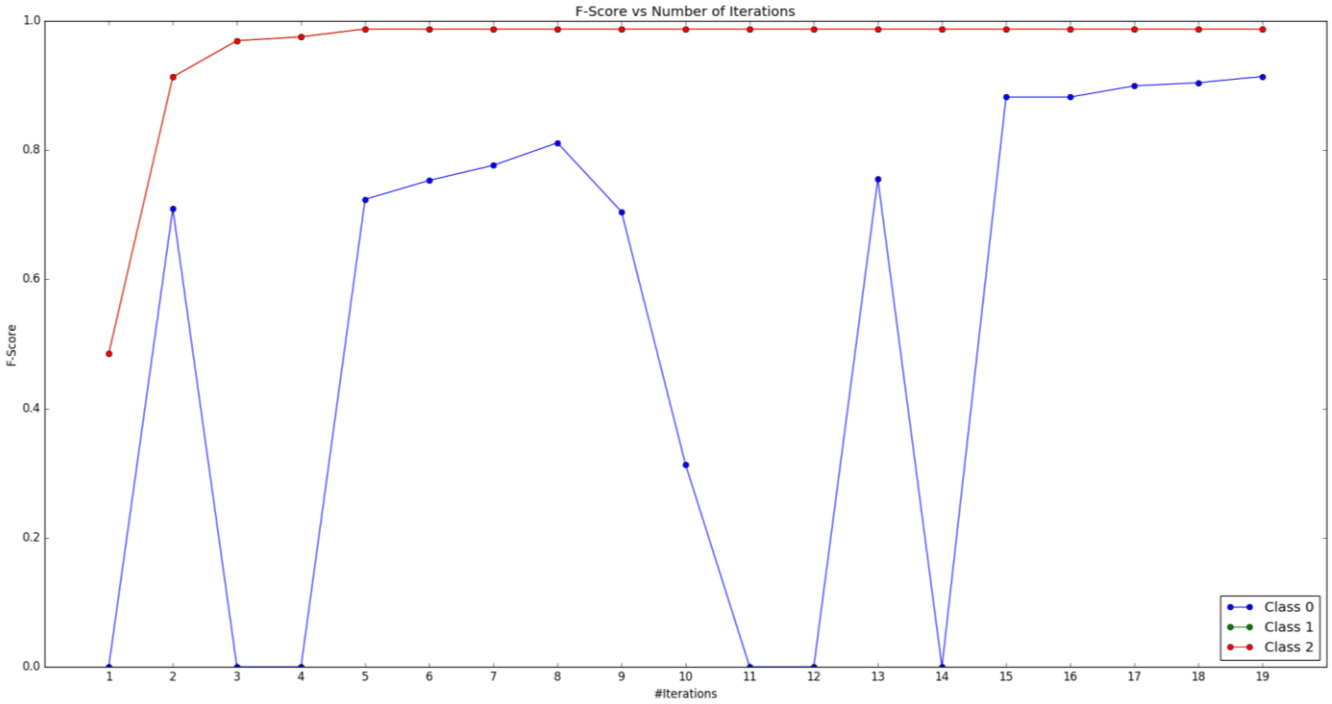


#Hidden Units = 125 -

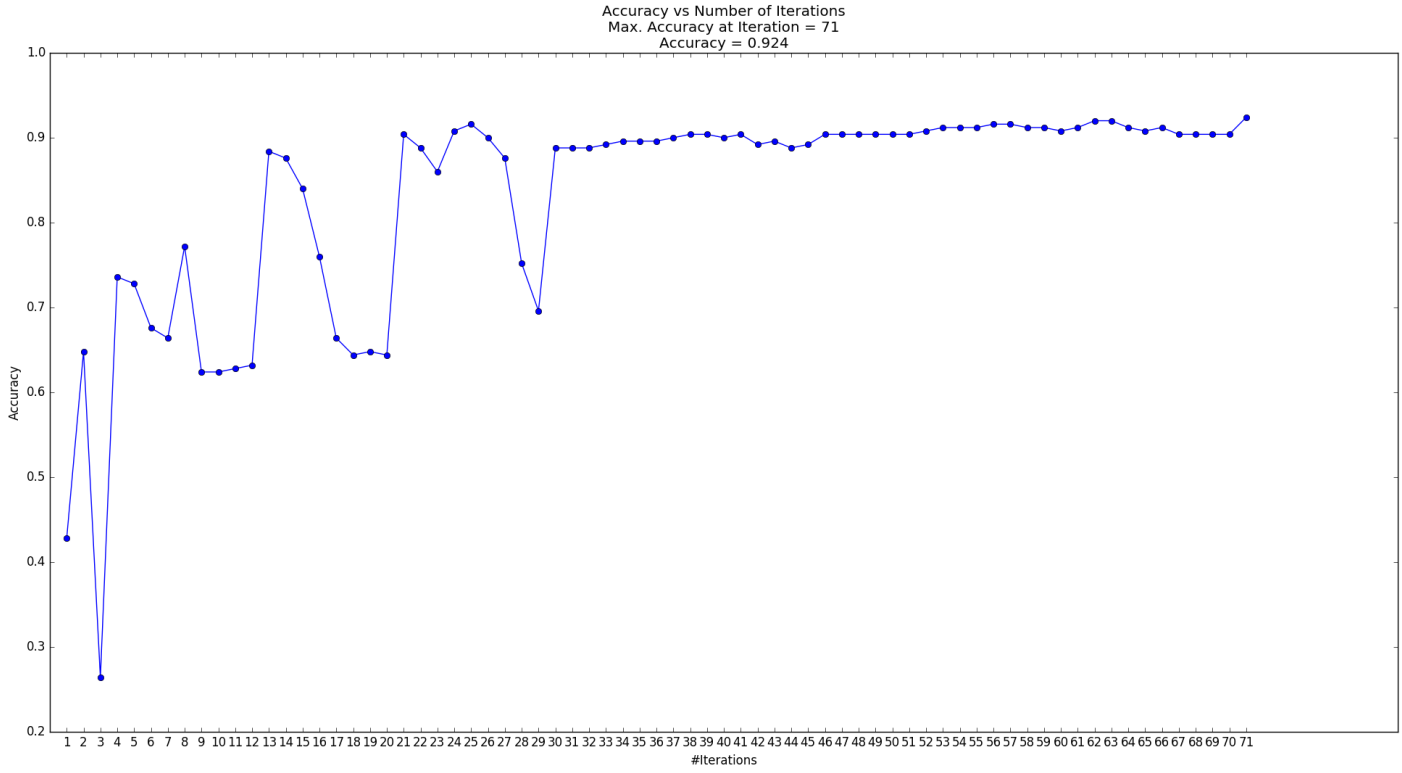


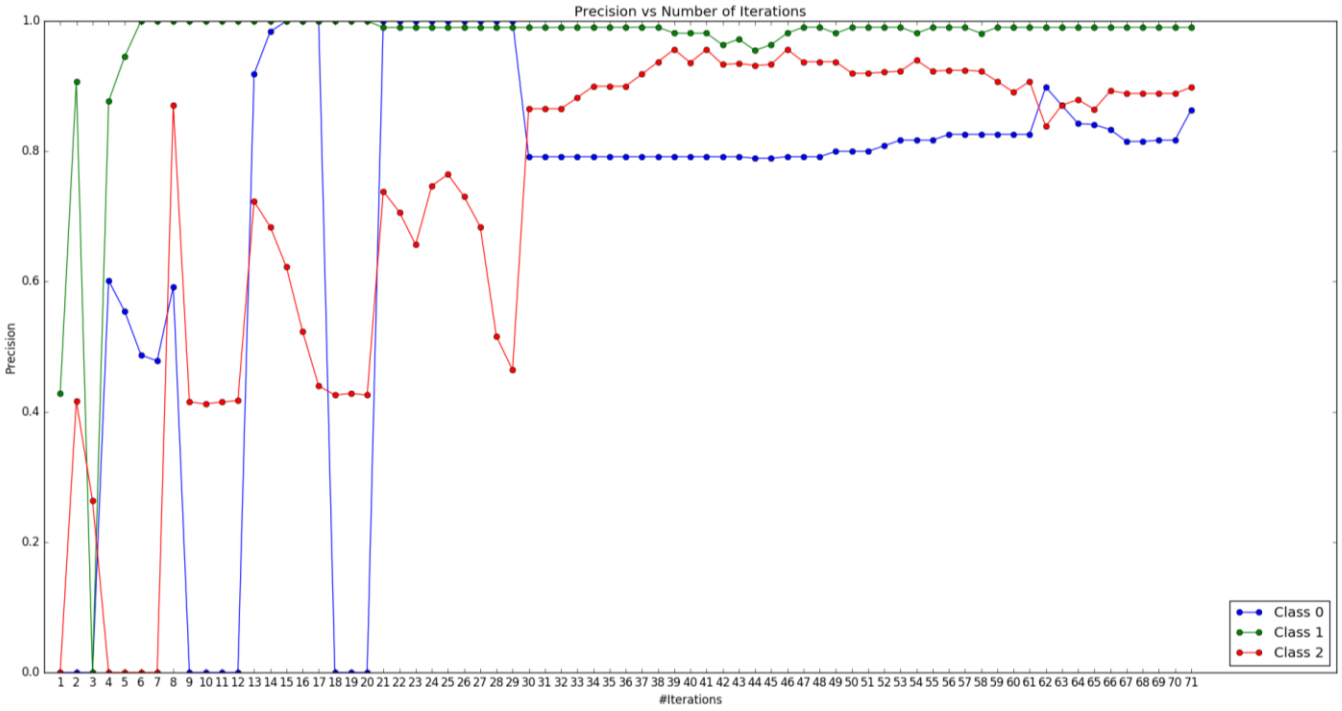

Precision vs Number of Iterations
Precision
#Iterations
Class 0
Class 1
Class 2

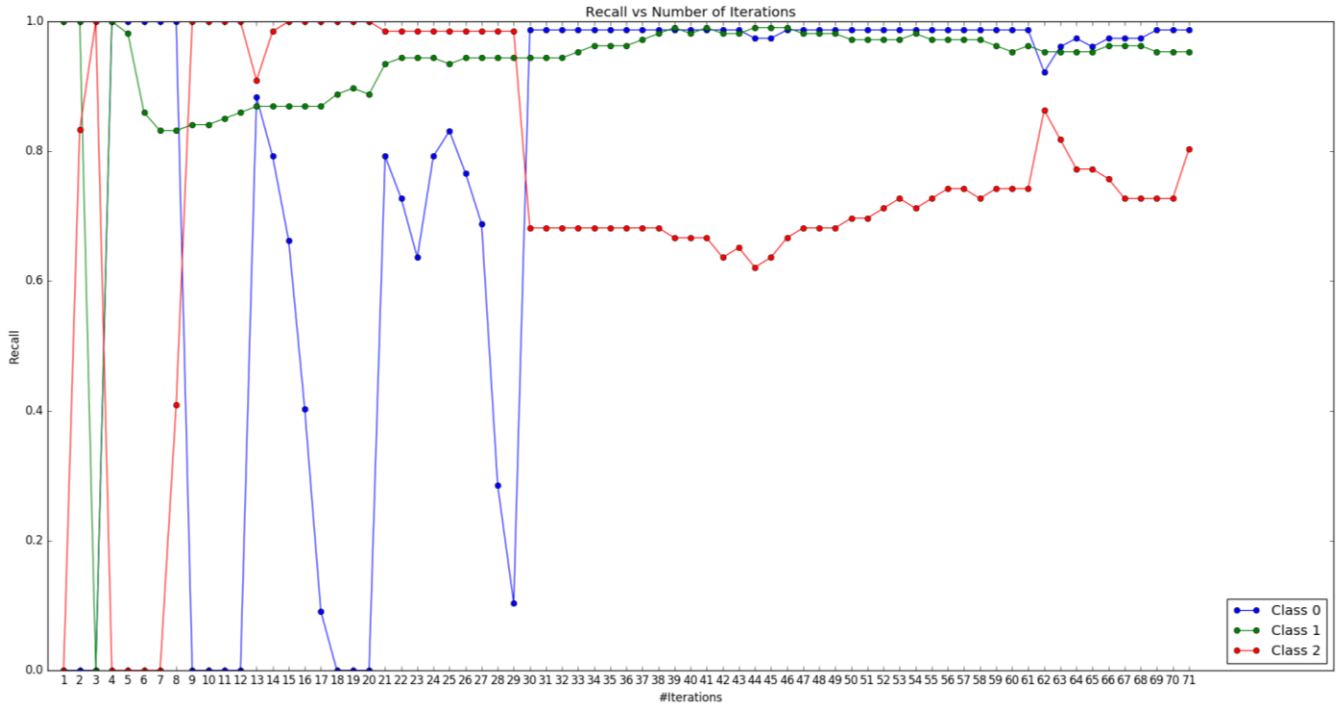

Recall vs Number of Iterations
Recall
#Iterations
Class 0
Class 1
Class 2

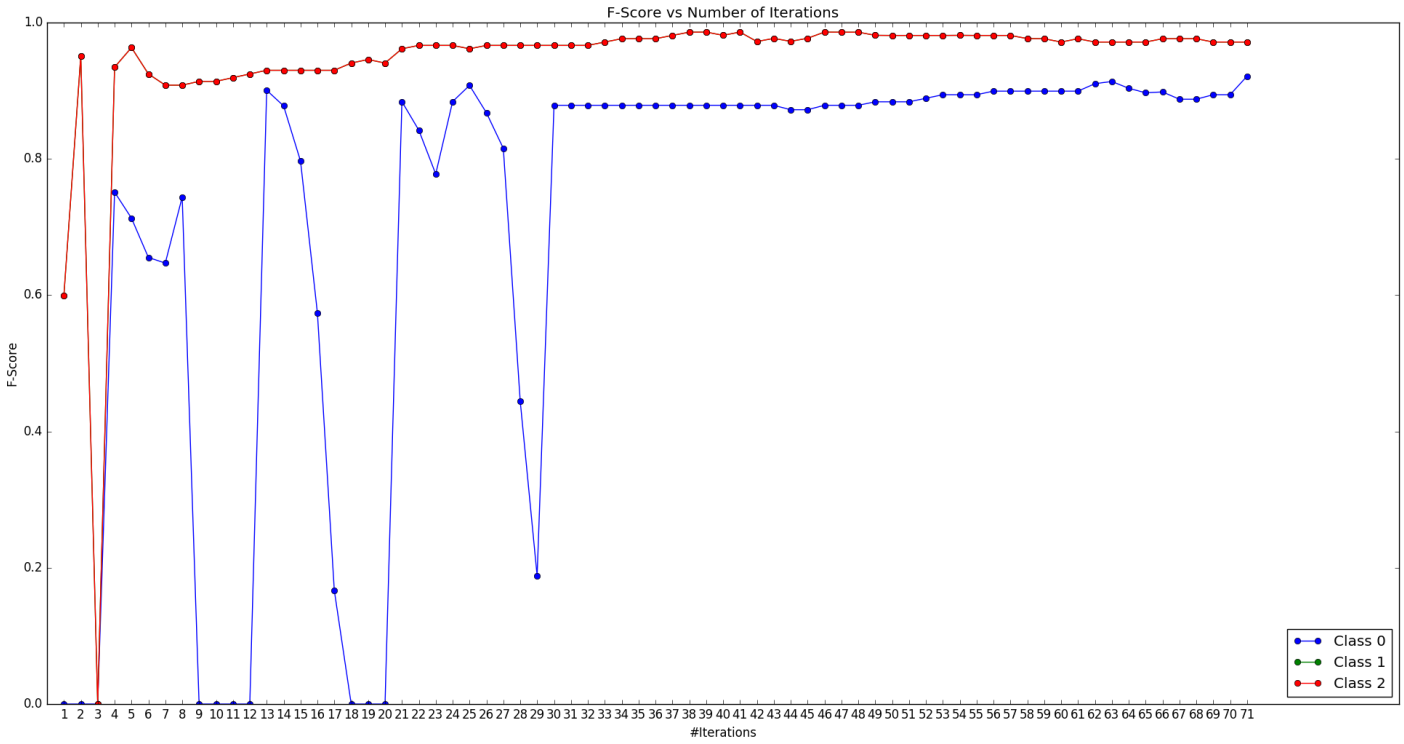


#Hidden Units = 200-




Precision vs Number of Iterations
Precision
#Iterations
Class 0
Class 1
Class 2


Recall vs Number of Iterations
Recall
#Iterations
Class 0
Class 1
Class 2



As we can see from the graphs above, it can also be seen that the model with the least number of hidden units (50) converges to the solution in the least number of iterations (13). The scikit learn implementation performs similar to the self-implemented model, both accuracy-wise and performance-wise. The model with 125 hidden units converges in 19 iterations and the one with 200 neurons converges in 71 iterations.

The training stops when either the change in likelihood is lesser than 0.1 or the number of iterations has reached 100 or the testing accuracy is above 92%; whichever occurs earlier.