EE655000 Machine learning HW3

Computer assignment

By

M Shri Harish 106061860

1. Neural Network

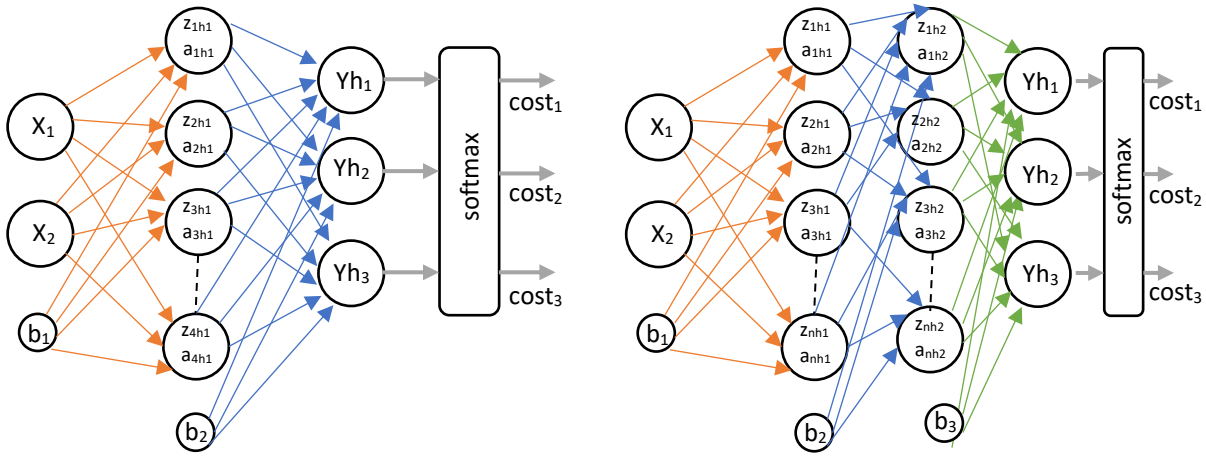


Figure 1: Neural network structure of 2 layer(left) and 3 layer (right) models, along with the layers and contents of network

Neural Networks (NN) are combinations of nodes connected with weights and governed by activation functions to provide probabilities of trained models. In this report, NN models are used for classification of fruits into Lychee, Carambula and Pear. Also, two layer and three layer NN models as in Figure 1 are used for classification with backpropagation and gradient descent to optimize the weights and bias. Later in this report the performance of each NN models are reported.

a. Working

The working of the NN model can be split into two phases [1], the forward phase and the backward phase that symbolises the backpropagation.

I. Initialization

The NN model is initialized with basic terms such as input x, output yh, actual output y, dimensions, number of samples and so on.

II. Pre-processing

The input images are converted into simple 2D data using principal component analysis. The result of PCA will be elements with two features of the image . with help of the scatter plot the spread of dataset can be visualized.

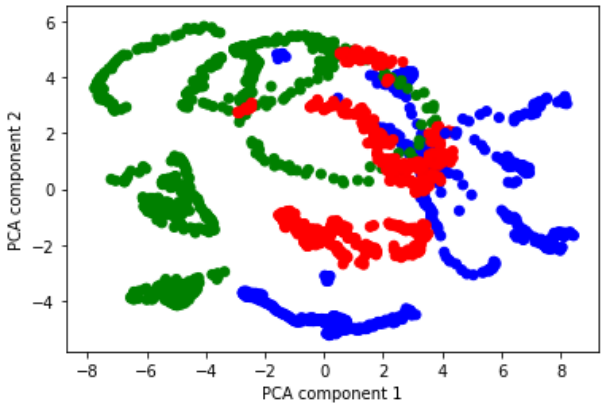


Figure 2: Scatter plot of the given three classes with PCA

III. Feed-Forward

In the forward pass, the input X is passed on to hidden layer and then to output layers. In between these passing, the information is multiplies with weights and added with bias. To keep it normalized an activation function is used. This activated result passes on to the next connected node, this happens for number of hidden layers available.

$$z_i = w_i . x + b_i$$

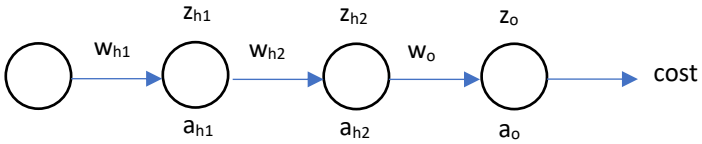
Where i denotes the number of layers, z denotes the bias and weight added term. This z is activated using sigmoid activation function.

$$a_i = \frac{1}{1 + e^{-z_i}}$$

a_i is the activated value, this value is transferred as input to the next hidden layer or output layer.

IV. Back-propagation

In back propagation the resultant value from the output node is checked for cost value using cross entropy and the weights and bias are manipulated to minimize the cost value. For back propagation it is necessary to find the cost being affected by the weights at the beginning $\frac{\partial cost}{\partial w_1}$. This can be calculated for three layer network and two layer back propagation can be obtained during the process.



$$\frac{\partial cost}{\partial w_{h1}} = \frac{\partial cost}{\partial a_o} * \frac{\partial a_o}{\partial z_o} * \frac{\partial z_o}{\partial a_{h2}} * \frac{\partial a_{h2}}{\partial z_{h2}} * \frac{\partial z_{h2}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_{h1}}$$

The partial derivative of each can be found in [2]. The final result of back propagation for three layers is

$$\frac{\partial cost}{\partial w_{h1}} = (y_h - y) * \dot{a}_3 * w_3 * \dot{a}_2 * w_2 * \dot{a}_1 * x$$

Also for the bias

$$\frac{\partial cost}{\partial w_{h1}} = (y_h - y) * \dot{a}_3 * w_3 * \dot{a}_2 * w_2 * \dot{a}_1$$

Similarly back propagation of two layers is

$$\frac{\partial cost}{\partial w_{h1}} = (y_h - y) * \dot{a}_2 * w_2 * \dot{a}_1 * x$$

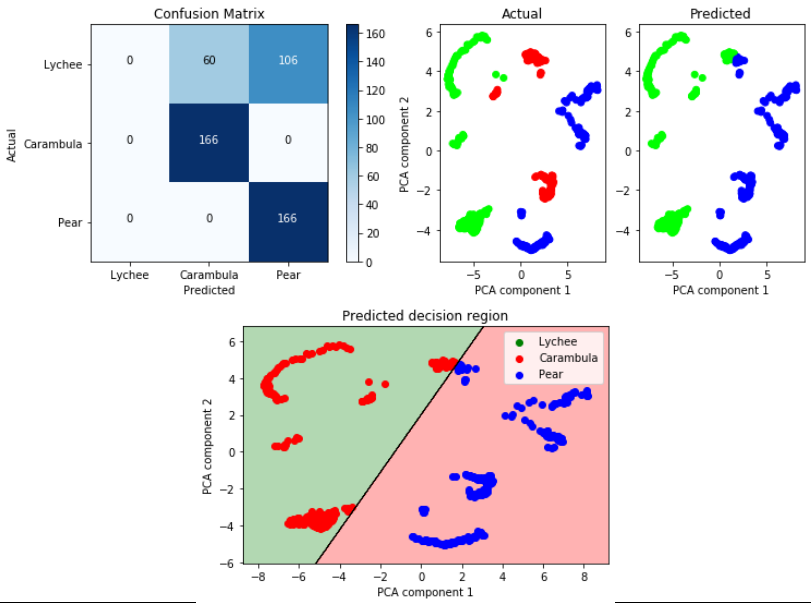
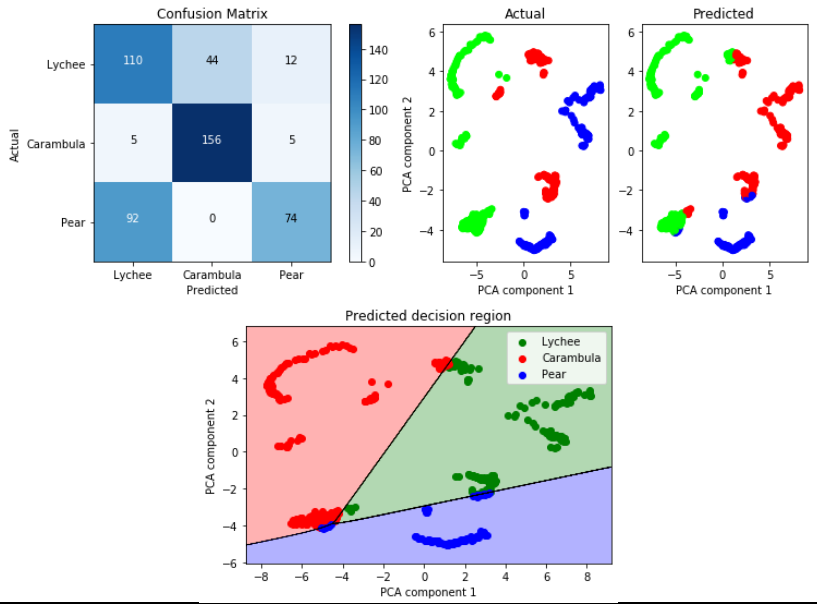
Also for the bias

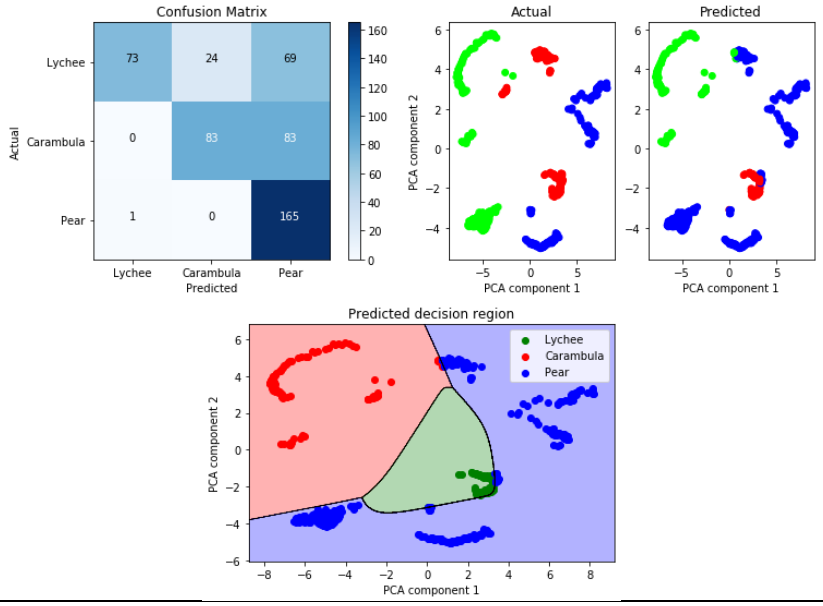
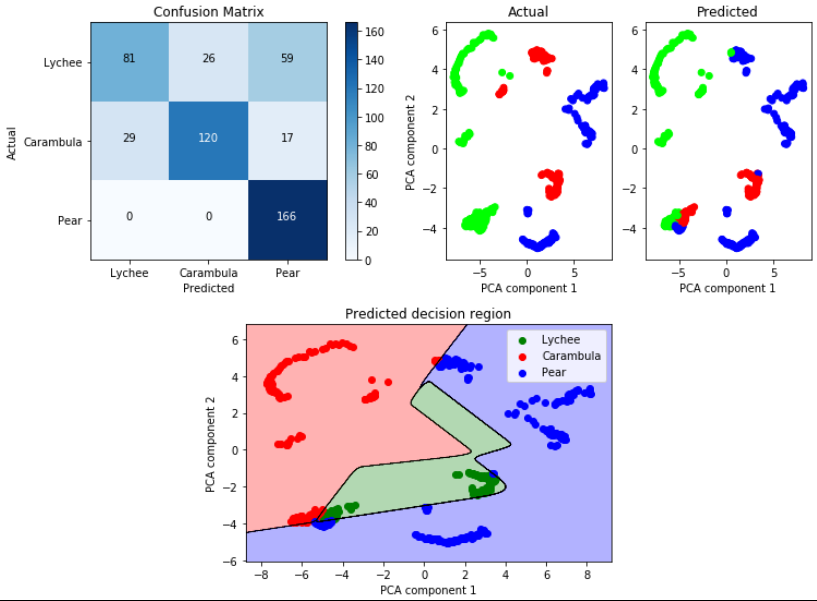
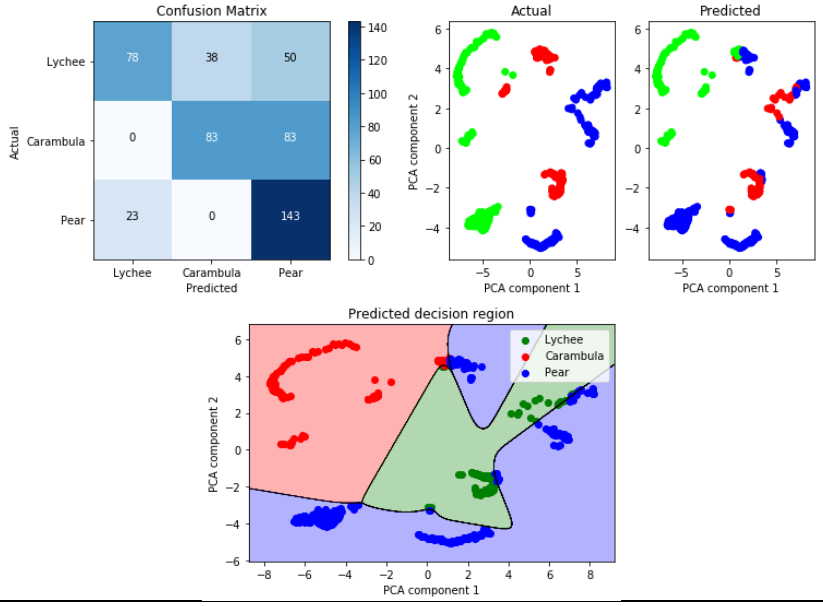
$$\frac{\partial cost}{\partial w_{h1}} = (y_h - y) * \dot{a}_2 * w_2 * \dot{a}_1$$

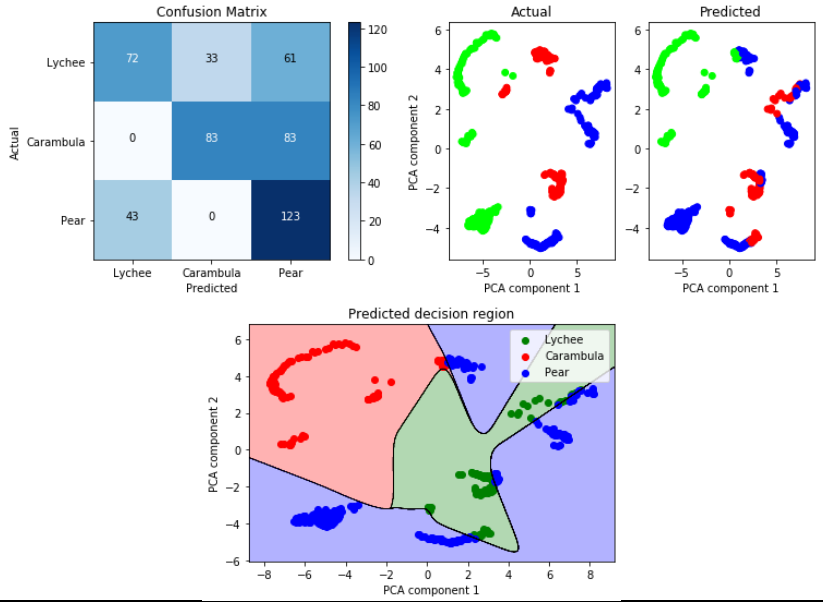
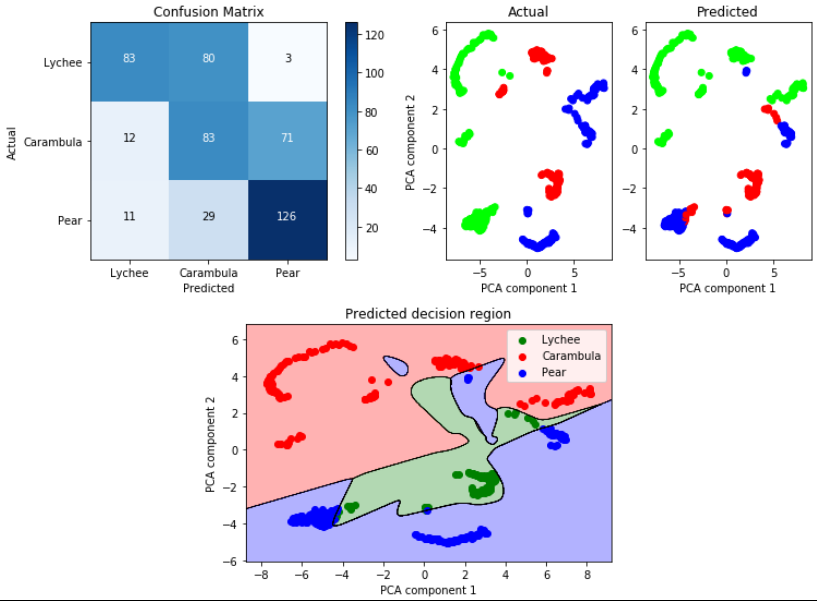
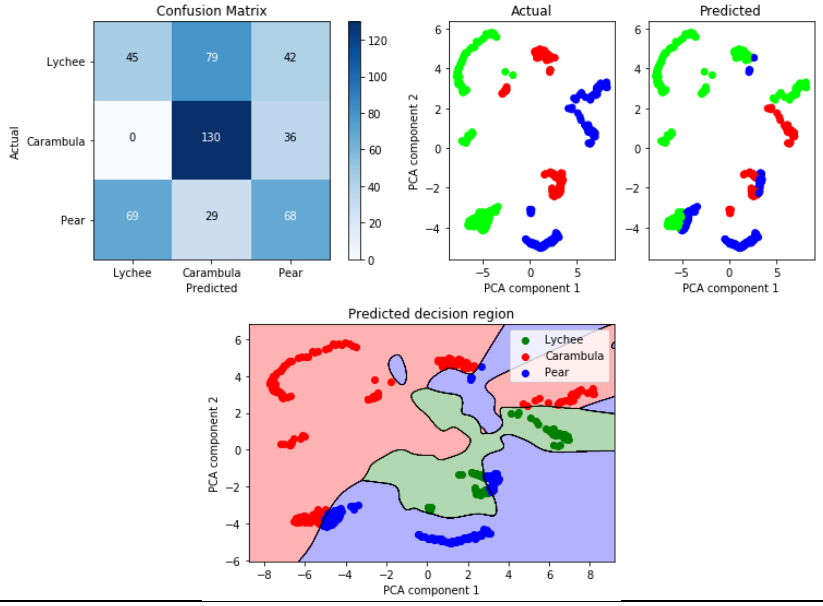
Where $\dot{a}_i = a_i * (1 - a_i)$

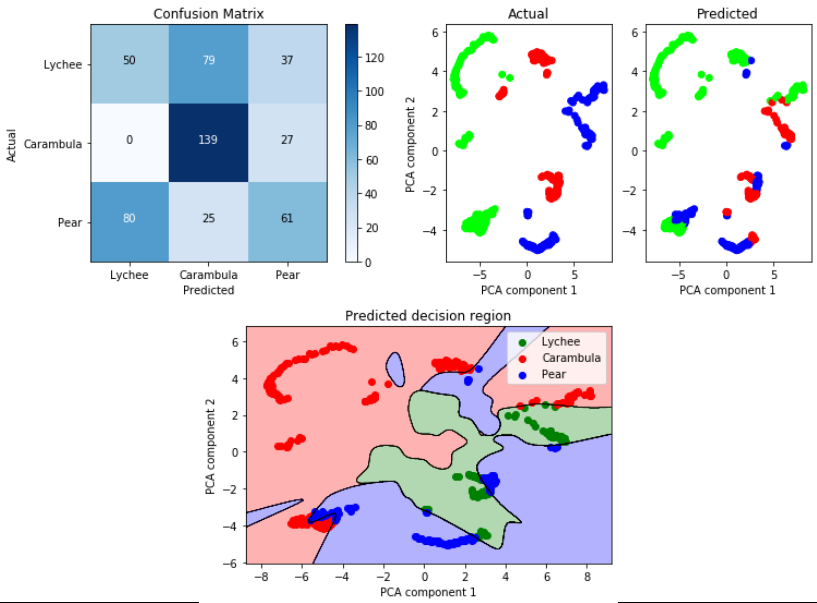
b. Results

V. 2 layer NN model

Hidden nodes : 1	Accuracy : 66.7% Iteration: 45500	
Hidden nodes : 2	Accuracy : 68.3% Iteration: 100000	

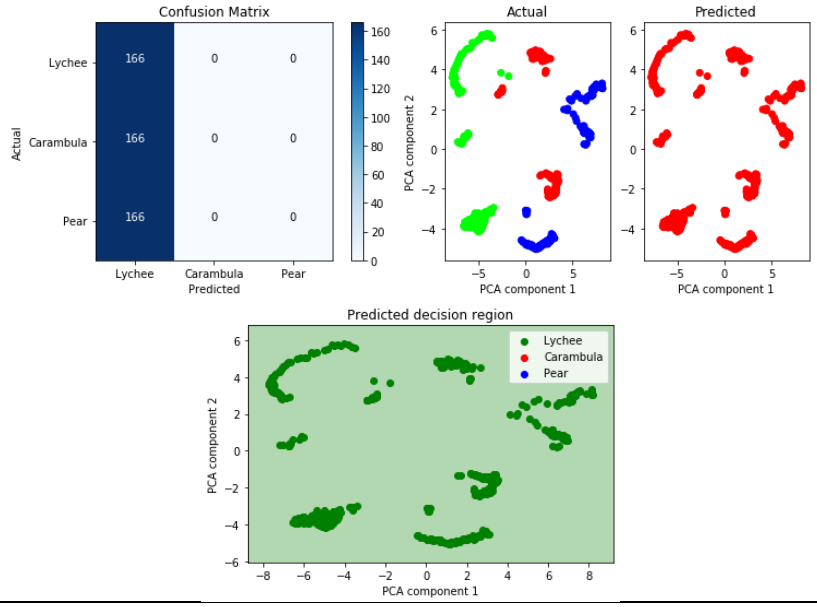
Hidden nodes : 3	Accuracy : 64.5% iteration : 74500	
Hidden nodes: 4	Accuracy : 73.7% Iteration: 115000	
Hidden nodes: 5	Accuracy : 61.0% iteration 65000	

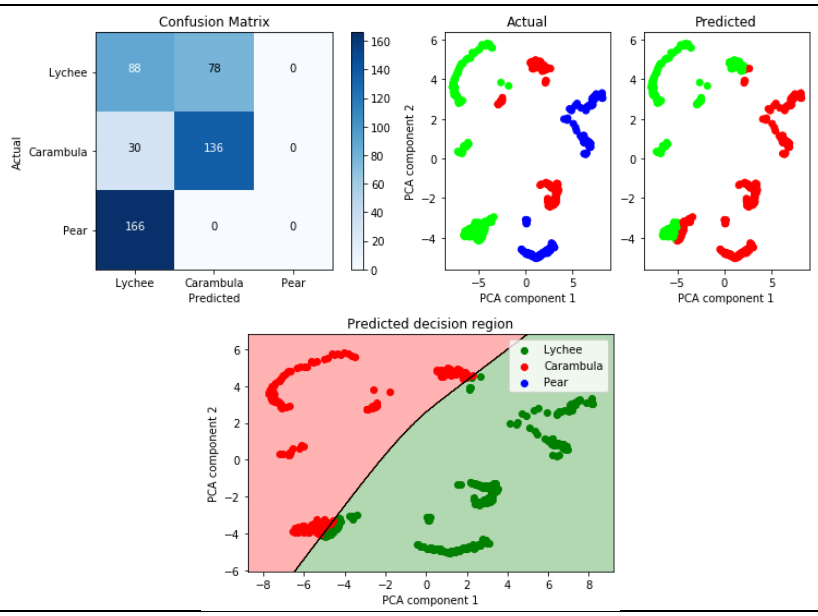
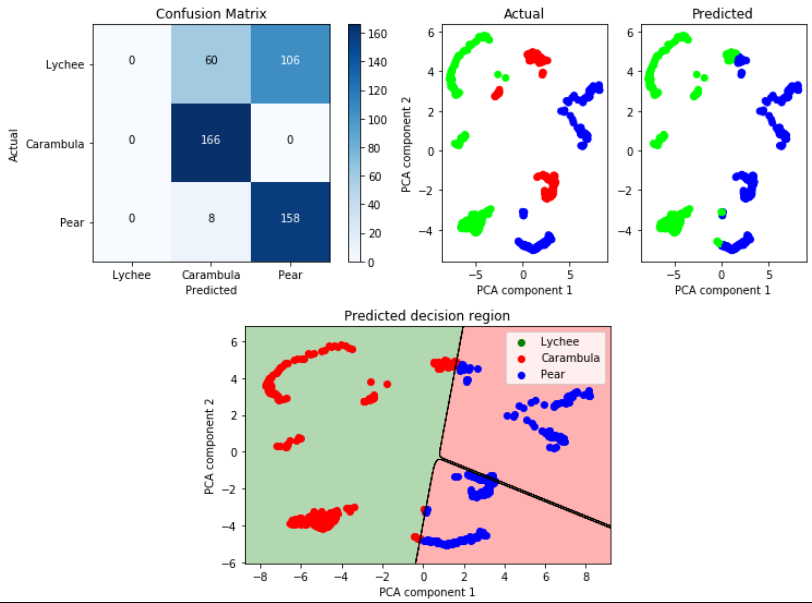
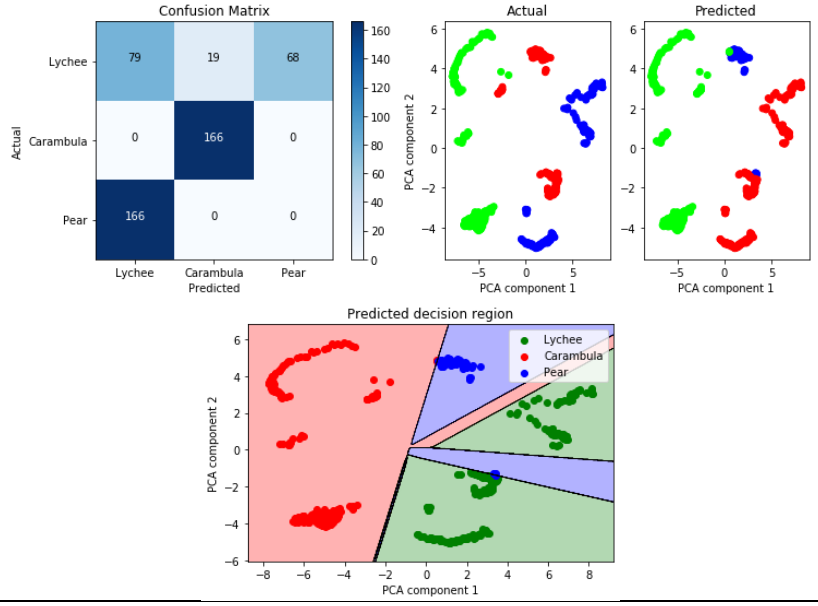
Hidden nodes: 6	Accuracy : 55.8% iteration 136000	
Hidden nodes: 10	Accuracy : 58.6% iteration 52500	
Hidden nodes: 50	Accuracy : 48.8%	

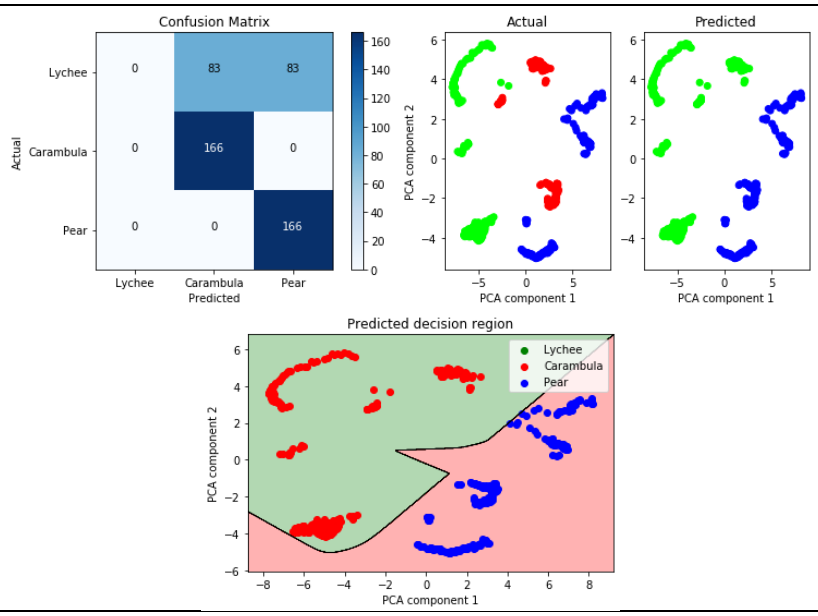
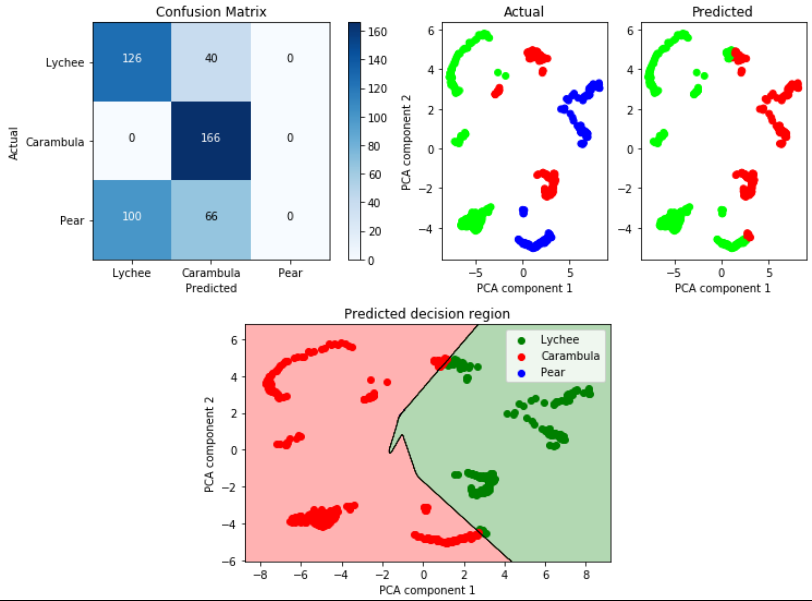
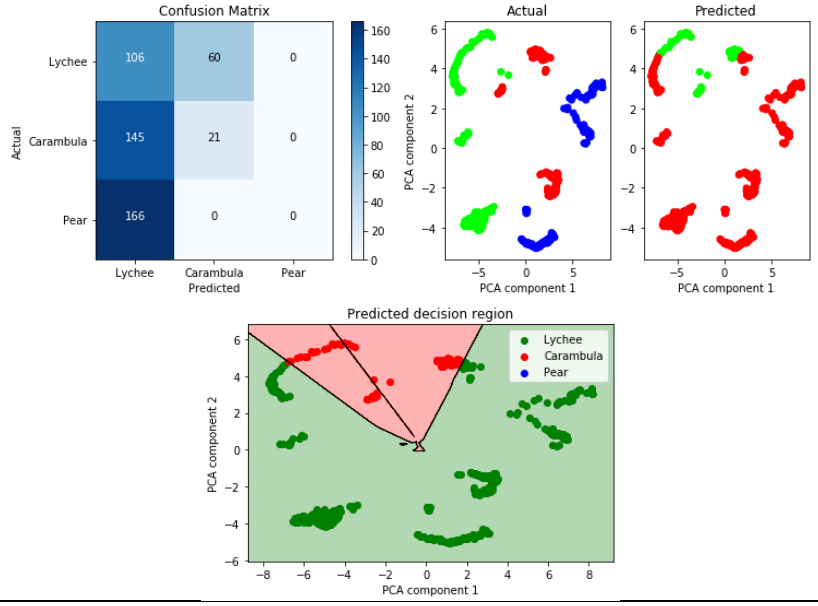
Hidden nodes: 100	Accuracy : 50.2%	

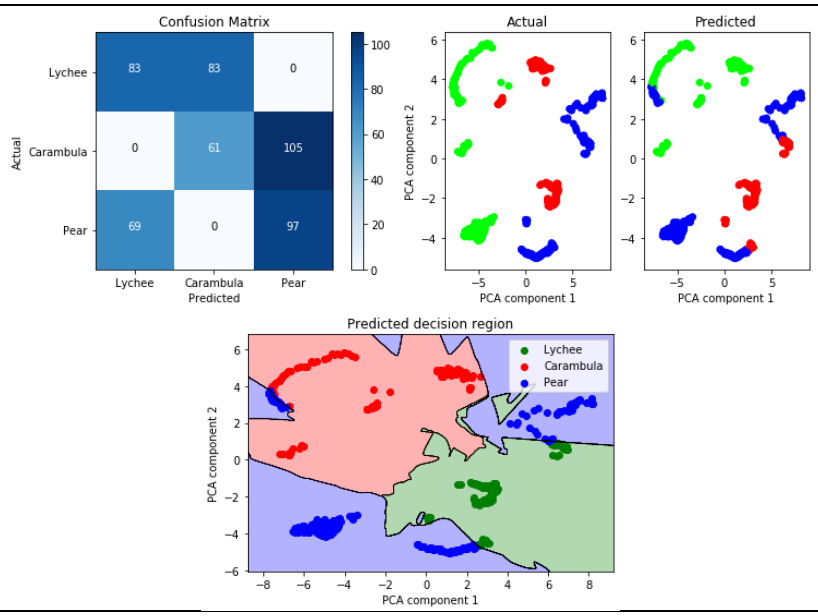
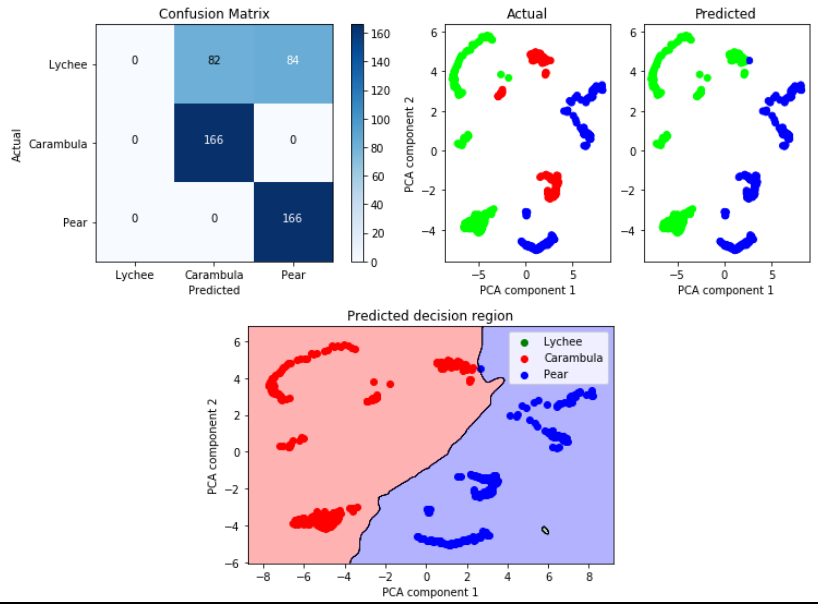
VI. 3 layers NN model

Both the hidden layers are given same numbers of nodes for simplicity.

Hidden nodes : 1	Accuracy : 33.3%	

Hidden nodes : 2	Accuracy : 45.0%	
Hidden nodes : 3	Accuracy : 65.1%	
Hidden nodes: 4	Accuracy : 49.2%	

Hidden nodes: 5	Accuracy : 66.7%	
Hidden nodes: 6	Accuracy : 58.6%	
Hidden nodes: 10	Accuracy : 25.5%	

Hidden nodes: 50	Accuracy : 48.4%	
Hidden nodes: 100	Accuracy : 66.7%	

c. Inference

From the above simulations in 2 layer NN model the hidden nodes from 2 till 4 gives a good performance. While hidden node 1 provides underfitting and all the above hidden nodes provide overfitting.

In the 3 layer NN model the hidden node 3 and 6 gives an acceptable result, thought the accuracy was down the graph decision region and scatter plot provides a reasonable solution. But 1 and 2 hidden nodes give underfitting and the all the rest gives heavy overfitting.

As the number of layers increases the model is able to find dimensionality compared to 2 layer model. The 3 layer model is able to look at possibilities of solution but its highly prone to noises and overfitting. The dimensionality is achieved in 2 layer node while the nodes increase to 100 but in 3 layer its seen early.

The sweet spot or the best configuration for the given dataset will be 2 layer NN model with nodes varying between 2 till 4.

2. Reference

[1]. https://towardsdatascience.com/coding-a-2-layer-neural-network-from-scratch-in-python-4dd022d19fd2

[2]. https://stackabuse.com/creating-a-neural-network-from-scratch-in-python-multi-class-classification/