Question 1.1:

Yes, it important and necessary to perform any coding on the class values for better performance of the project. This will prevent the model to not misinterpret the representation of classes in the data.

Question 1.2:

The type of encoding that needs to be done is a preprocessing technique called one hot encoding.

Question 1.3:

Since there are 10 classes for this hypothetical dataset, you will have split the column of these classes into 10 different columns, each one representing a class. So for example the class will that represents the value 0 will be called class\_0, and the class that represents the value 1 will be called class\_1, and so on. Next, the values of each new class will be either 1 or 0, 1 representing yes confirming the observation represents the class, and 0 means no, the observation does not represent the class. So, all the new class will be zeros except for one class.

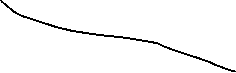
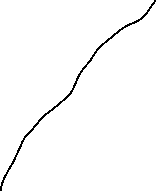
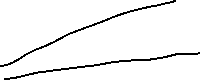
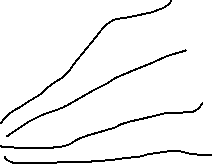
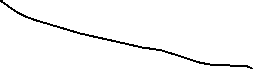
Question 1.4:

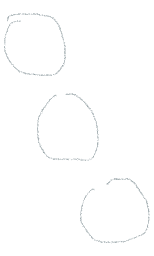
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class\_0 | Class\_1 | Class\_2 | Class\_3 | Class\_4 | Class\_5 | Class\_6 | Class\_7 | Class\_8 | Class\_9 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

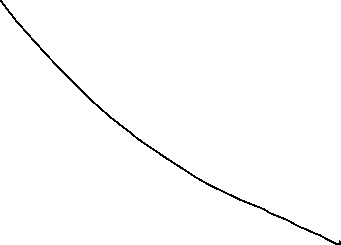
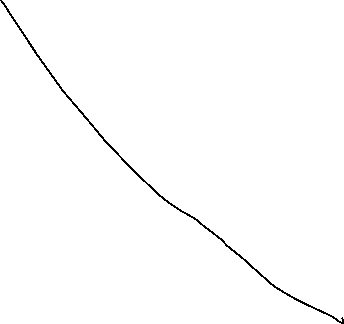
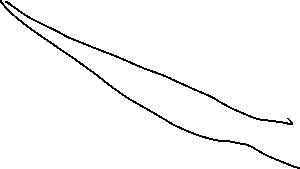
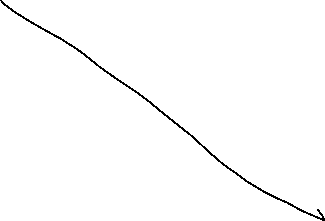
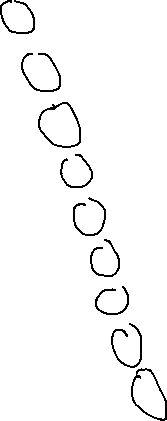
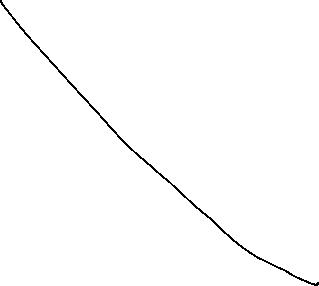
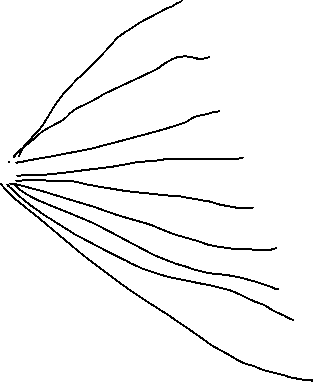
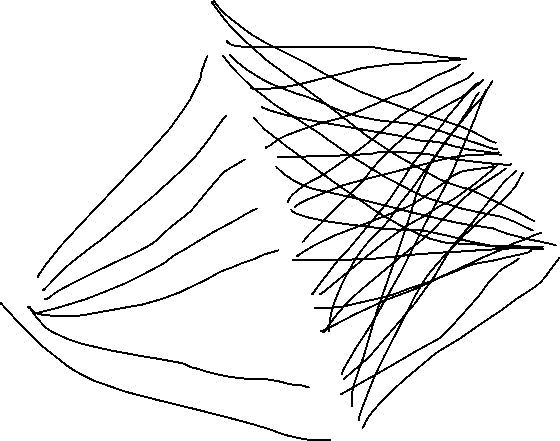
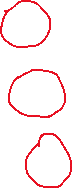
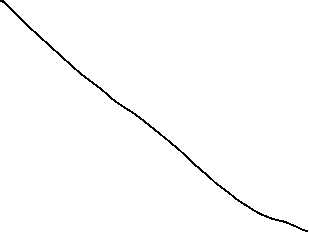
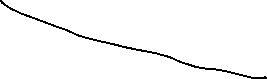
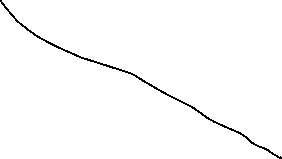
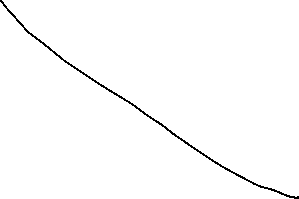
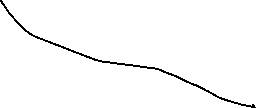
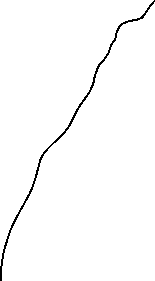
Part II:



In this one layer neural network, there are 4 neurons in the input layer, 3 neurons in the hidden layer and 3 inthe output layer. Each nueron in the input layer sends its value along with a set of weights to each neuron of the hidden layer. The weighted sum of all the input values for each neuron of the hidden layer is calculated. If the results of the weighted sum for each hidden neuron reaches a specific threshold, an activation function, in this case, ReLu, is computed. After that, the values of each hidden neuron is applied to a another set of weights and then transmitted to the final output layer. Since there are 3 possible outputs for this classification there will be 3 neurons in the output layer. The neuron that has the highest value in the output layer is the class that the neural network predicted. A softmax function will be used in the output layer to make sure one the neurons is close to 1 and the rest of the neurons close to zero, since this is a multiclassification problem.







So I ran an mlp consisting of 4 neurons in the input layer, 10 neurons in the hidden layer and 3 neurons in the output layer. A Relu activation function was used for the hidden layer and a softmax activation function for the output layer. After one hot encoding the species class, splitting the data into training, validation and testing subsets, and then running the model. I obtained an accuracy score of 97.78% for the training, 100% for the validation and a 91.66% for the testing. This shows that there is some overfitting in the model, due to the fact that the training and validation had a higher accuracy score than testing.

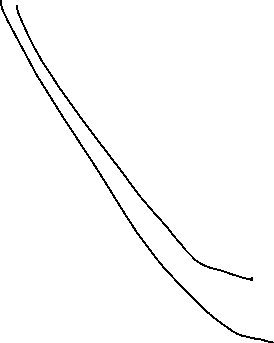
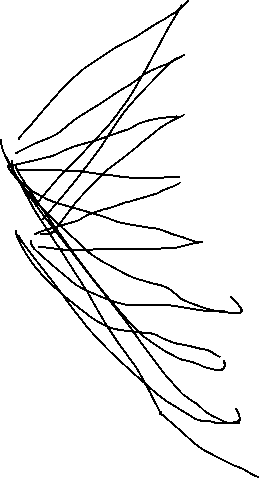
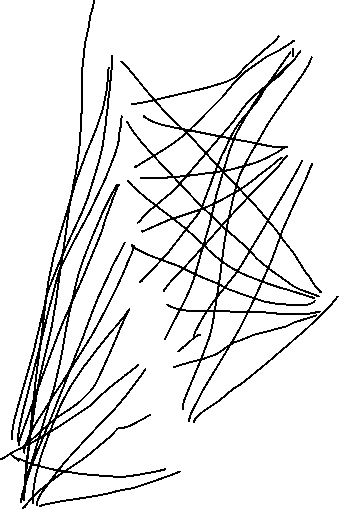
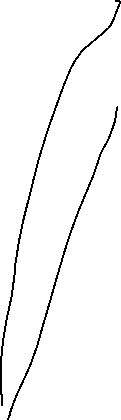
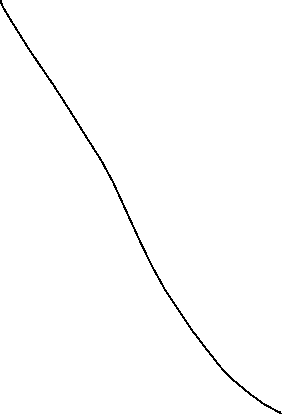
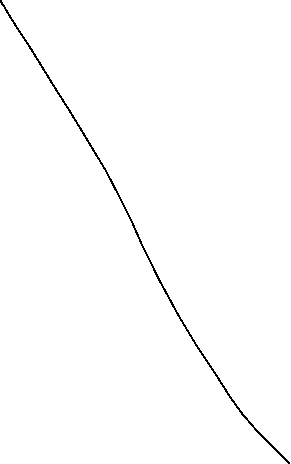
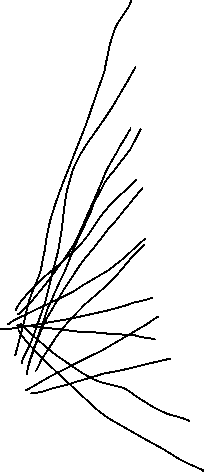
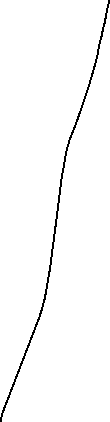
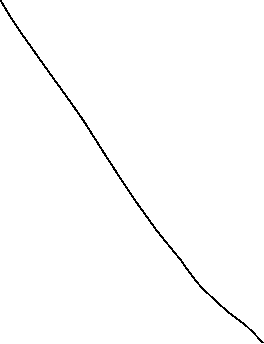
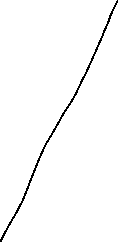
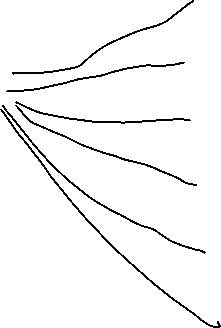
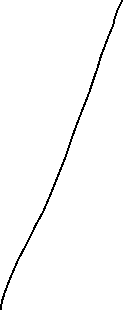
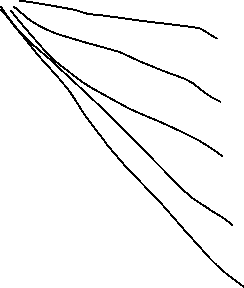
Part III:

Since my first model lacked complexity, I decided to add another hidden layer. So, now mlp will consist of 4 neurons in the input layer, 4 neurons in hidden layer 1, 10 neurons in hidden layer 2, and 9 neurons in the output layer. Each nueron in the input layer sends its value along with a set of weights to each neuron of hidden layer1. The weighted sum of all the input values for each neuron of hidden layer1 is calculated. If the results of the weighted sum for each hidden neuron reaches a specific threshold, an activation function, in this case, ReLu, is computed. After that, the values of each hidden neuron is applied to a another set of weights and then transmitted to hidden layer 2, the same process that happened to hidden layer 1 will happen to hidden layer 2. The activation for hidden layer 2 will be Relu as well. Since there are 3 possible outputs for this classification there will be 3 neurons in the output layer. The neuron that has the highest value in the output layer is the class that the neural network predicted. A softmax function will be used in the output layer to make sure one the neurons is close to 1 and the rest of the neurons close to zero, since this is a multiclassification problem.

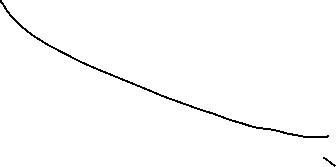
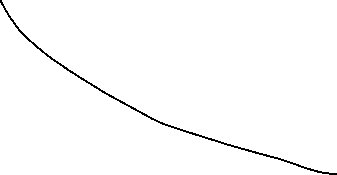
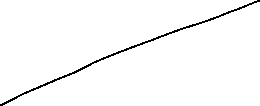
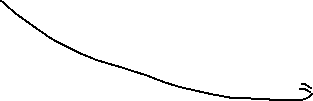
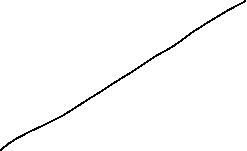
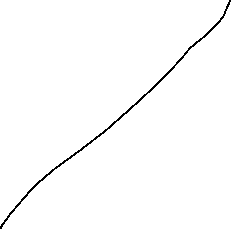
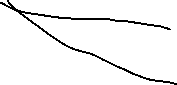
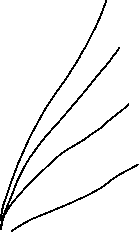
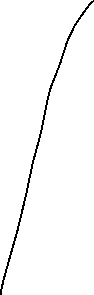
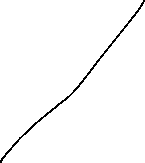
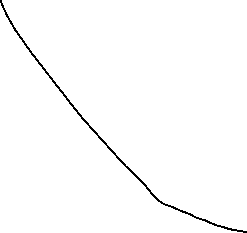
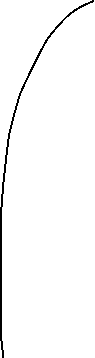
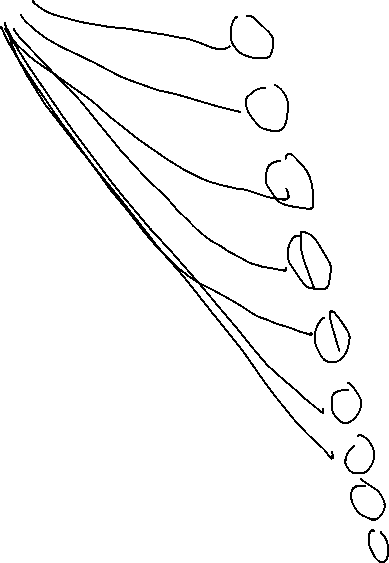
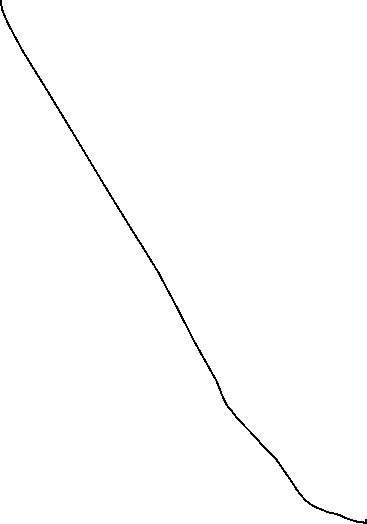
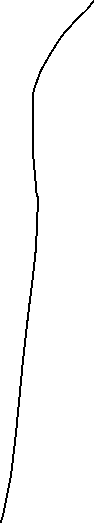
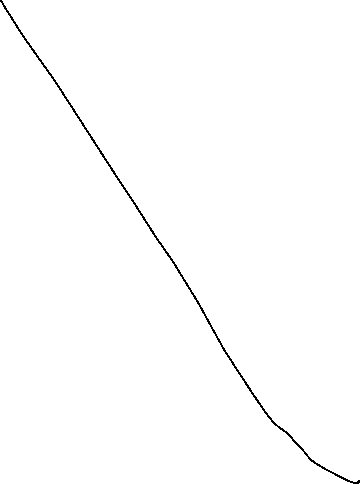
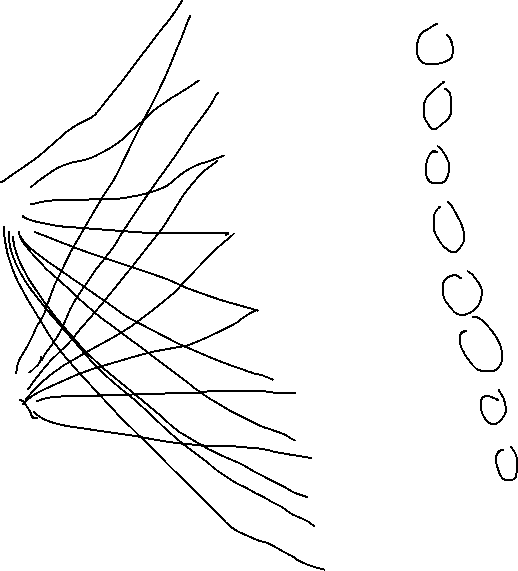
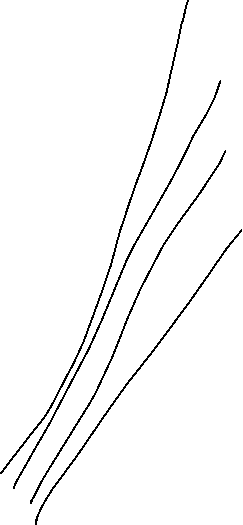
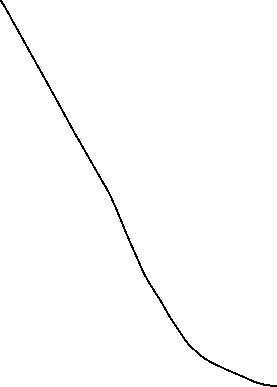
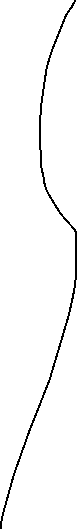
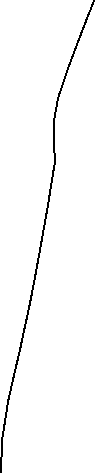
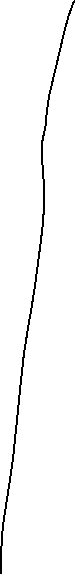
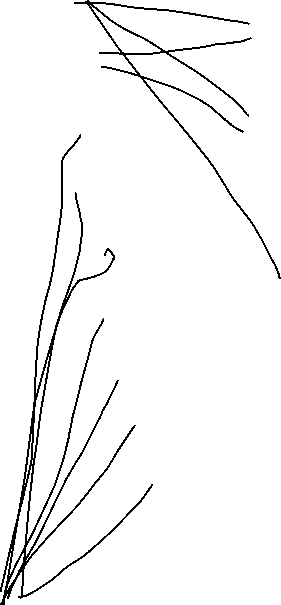












After adding an extra hidden layer that consists of 9 neurons, I ended up receiving an accuracy score of 95.44% for the training, 100% for the validation, and 100% for the testing. my accuracy for testing in this model was significantly higher than the other model. This model lacks overfitting and indicates that it will generalize well over unseen data. So, adding a hidden layer increased the generalization of the model.