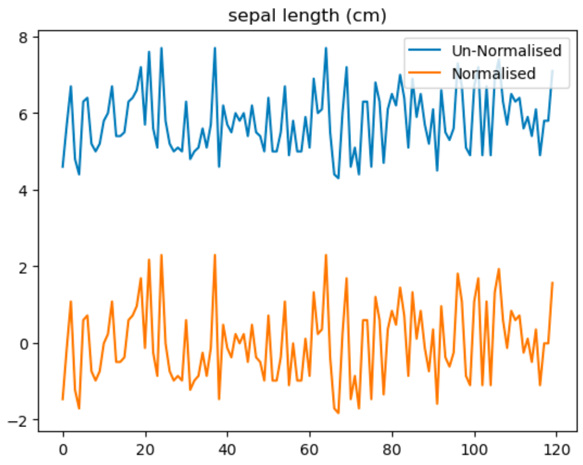
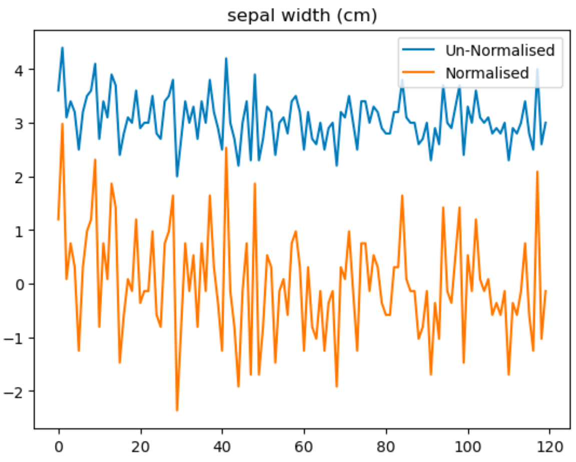
Machine Learning Assignment 2

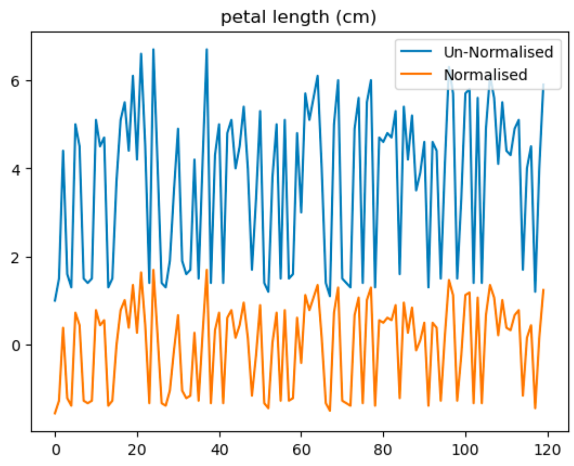
Week 4

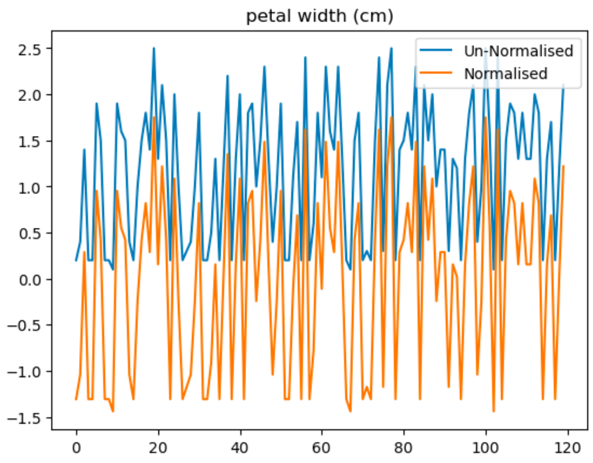
Q1. We again notice that the attributes are on different scales. Use the normalisation method from last lab, to standardize the scales of each attribute on both sets. Plot the normalized and raw training sets; what do you observe?

Below are the plots comparing each feature with its normalised values, we observe that data is now more confined to a certain interval range for example [-2, 2] for speal length. Features with larger scales can dominate the learning process, leading to biased influence and making it difficult for the algorithm to give equal consideration to all features. Normalizing features helps prevent this issue.



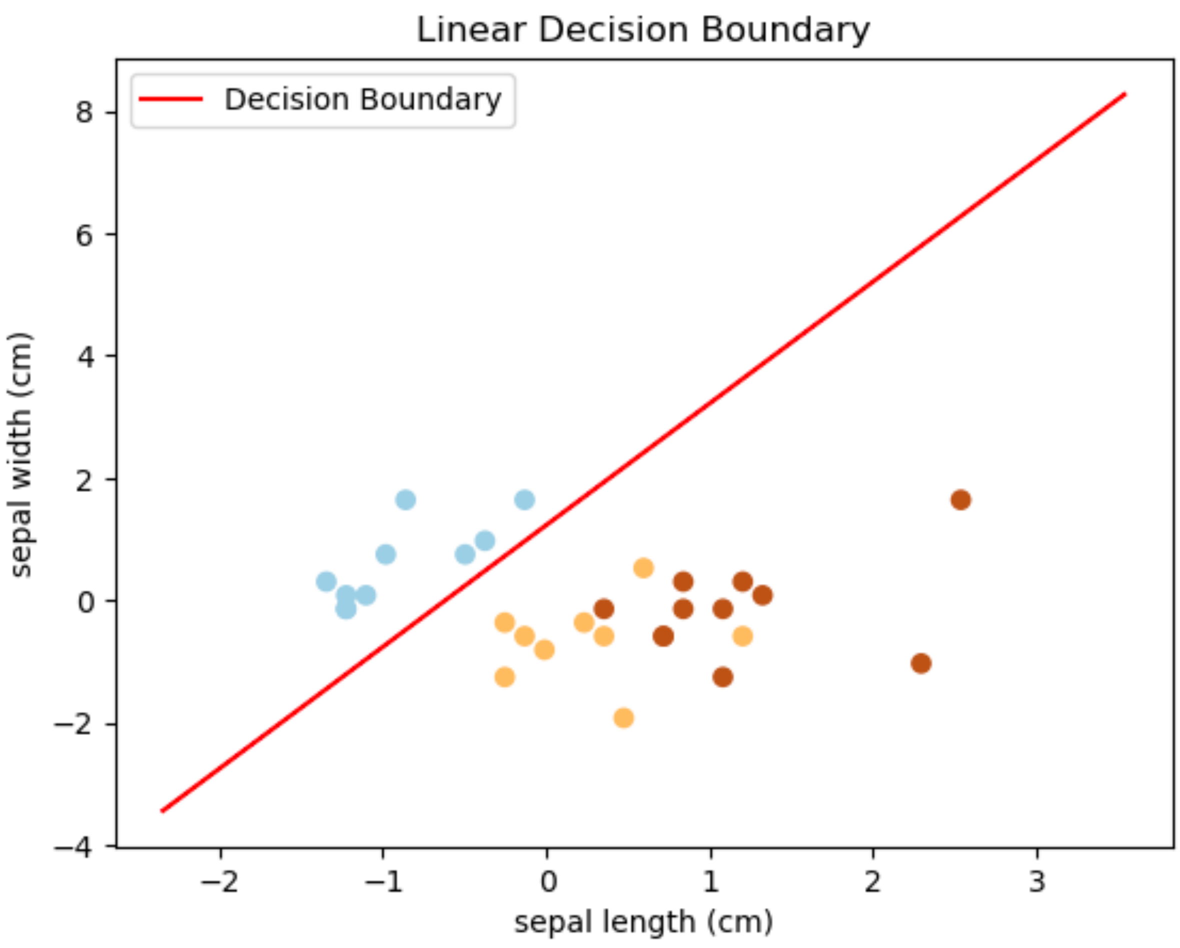






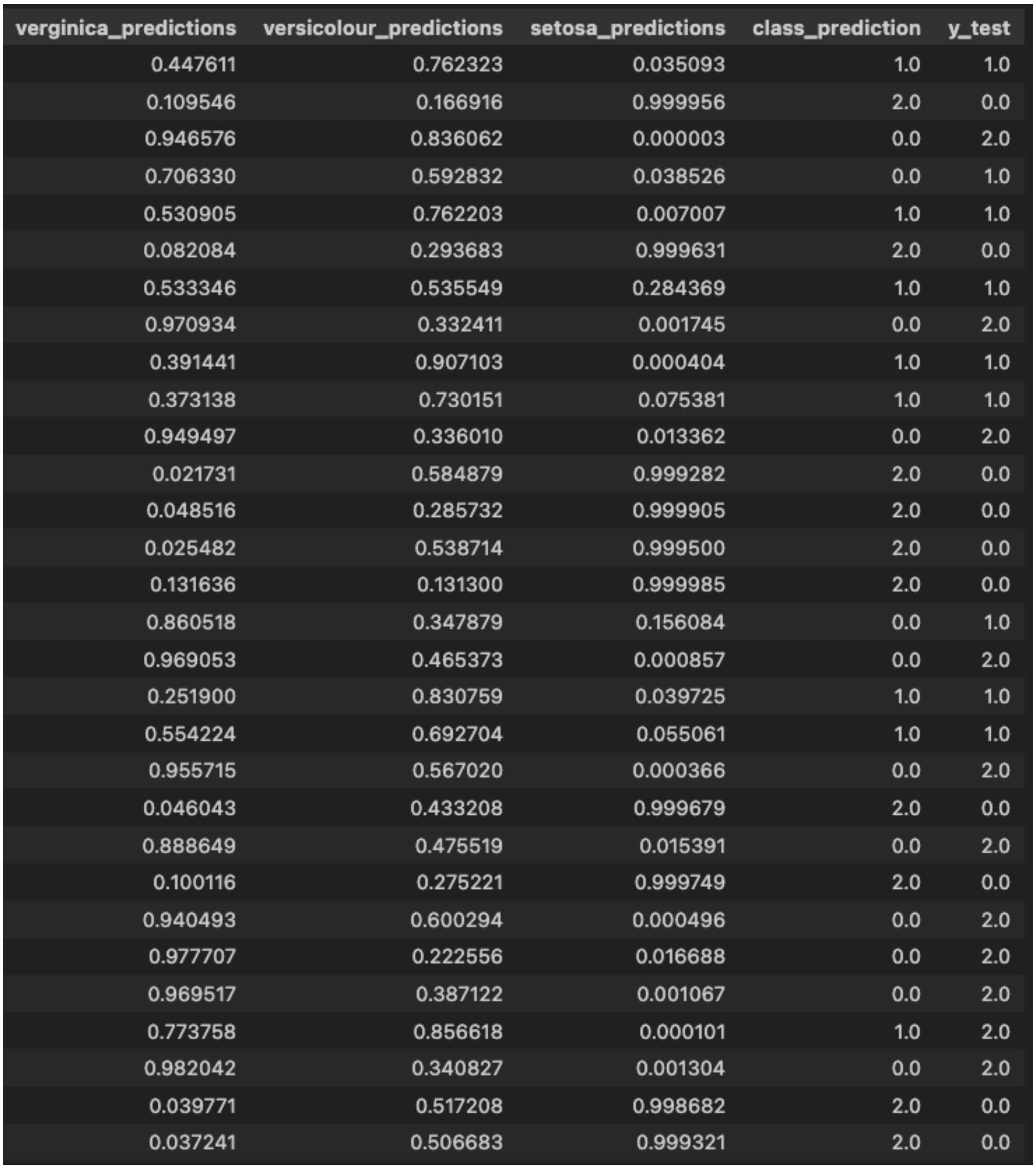
Q5. Draw the decision boundary on the test set using the learned parameters. Is this decision boundary separating the classes? Does this match our expectations?

The decision boundary plot is able sperate Setosa vs other species based on the sepal length (cm) and sepal width (cm). Data points with a light blue shade and points in yellow and red shades are separated by the decision boundary which is expected as we've used a one vs all technique of classification.



Q6. Using the 3 classifiers, predict the classes of the samples in the test set and show the predictions in a table. Do you observe anything interesting?

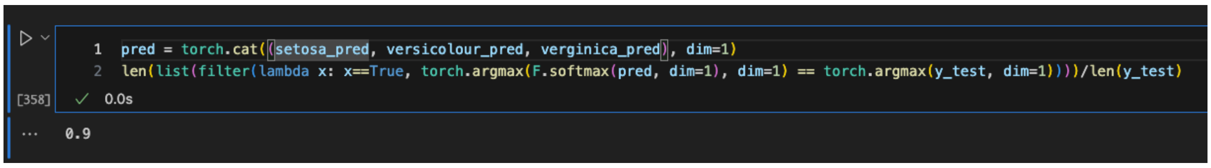
We observe that some of the class predictions are not matching the ground truth (y_test), which is expected as the model operates at 90% accuracy.



verginica_predictions	versicolour_predictions	setosa_predictions	class_prediction	y_test
0.447611	0.762323	0.035093	1.0	1.0
0.109546	0.166916	0.999956	2.0	0.0
0.946576	0.836062	0.000003	0.0	2.0
0.706330	0.592832	0.038526	0.0	1.0
0.530905	0.762203	0.007007	1.0	1.0
0.082084	0.293683	0.999631	2.0	0.0
0.533346	0.535549	0.284369	1.0	1.0
0.970934	0.332411	0.001745	0.0	2.0
0.391441	0.907103	0.000404	1.0	1.0
0.373138	0.730151	0.075381	1.0	1.0
0.949497	0.336010	0.013362	0.0	2.0
0.021731	0.584879	0.999282	2.0	0.0
0.048516	0.285732	0.999905	2.0	0.0
0.025482	0.538714	0.999500	2.0	0.0
0.131636	0.131300	0.999985	2.0	0.0
0.860518	0.347879	0.156084	0.0	1.0
0.969053	0.465373	0.000857	0.0	2.0
0.251900	0.830759	0.039725	1.0	1.0
0.554224	0.692704	0.055061	1.0	1.0
0.955715	0.567020	0.000366	0.0	2.0
0.046043	0.433208	0.999679	2.0	0.0
0.888649	0.475519	0.015391	0.0	2.0
0.100116	0.275221	0.999749	2.0	0.0
0.940493	0.600294	0.000496	0.0	2.0
0.977707	0.222556	0.016688	0.0	2.0
0.969517	0.387122	0.001067	0.0	2.0
0.773758	0.856618	0.000101	1.0	2.0
0.982042	0.340827	0.001304	0.0	2.0
0.039771	0.517208	0.998682	2.0	0.0
0.037241	0.506683	0.999321	2.0	0.0

Q7. Calculate the accuracy of the classifier on the test set, by comparing the predicted values against the ground truth. Use a SoftMax for the classifier outputs.

Achieved an accuracy of 90% on the testing set.



```
pred = torch.cat((setosa_pred, versicolour_pred, verginica_pred), dim=1)
len(list(filter(lambda x: x==True, torch.argmax(F.softmax(pred, dim=1), dim=1) == torch.argmax(y_test, dim=1))))/len(y_test)
```

[358] ✓ 0.0s

```
0.9
```

Q8. Looking at the datapoints below, can we draw a decision boundary using Logistic Regression? Why? What are the specific issues or logistic regression with regards to XOR?

A non-linear decision boundary must be drawn to separate the classes. This cannot be directly achieved by Logistic regression, which is known to classify linearly separable data. However, this can be achieved by using a feature space that includes not only the original input features but also their interactions.

For example, if your original features are denoted as x1 and x2, you can create additional features for example x3 = x1 * x2 and x4 = x1 + x2. Now, you can use logistic regression with these new features:

Output = sigma (w0 + w1 * x1 + w2 * x2 + w3 * x3 + w4 * x4)

Here, sigma is the logistic sigmoid function, and w0, w1, w2, w3 and w4 are the parameters of the logistic regression model.

By adjusting the parameters during training, the logistic regression model should be able to learn a decision boundary in this higher-dimensional space that effectively captures the XOR function.

Week 6

Q1. Why is it important to use a random set of initial weights rather than initializing all weights as zero in a Neural Network?

When all weights are initialized to the same value, the neurons in each layer will produce the same output during forward propagation, and during backpropagation, the gradients for each weight will be the same and will be updated equally. This results in the network failing to learn meaningful representations. On the other hand, using random initialization breaks this symmetry. Each neuron will have a slightly different initial set of weights, causing them to learn different features from the input data. This asymmetry is crucial for the network to learn diverse representations and avoid redundant neurons.

Q2. How does a NN solve the XOR problem?

XOR problem is not linearly separable, meaning a single straight line cannot be drawn to separate the two classes (0 and 1). To handle this, the neural network introduces non-linear activation functions. Common choices include the sigmoid function or the hyperbolic tangent function. These activation functions introduce non-linearities, allowing the network to learn and represent more complex relationships between inputs and outputs.

The hidden layer in the neural network acts as a feature extractor. It learns to represent the input data in a way that makes it easier for the final output layer to perform the desired classification. The non-linear activation function ensures that the hidden layer can capture complex patterns in the input data.

In the case of XOR, a single hidden layer with enough neurons can learn to represent the XOR function. The network learns to activate certain neurons for specific input combinations that result in a XOR output of 1 and deactivate for others that result in a XOR output of 0.

Thus, the combination of non-linear activation functions, hidden layers, and weight learning during training allows a neural network to learn and represent complex functions like XOR.

Q3. Explain the performance of the different networks on the training and test sets. How does it compare to the logistic regression example? Make sure that the data you are referring to is clearly presented and appropriately labelled in the report.

As the number of neurons in the hidden layer increases model performs well on the testing set, given by the accuracy measure. Neural network (IrisNN) is capable of multiclass classification unlike Logistic regression which could only handle one vs all classification. Neural networks are capable of modelling non-linearity very well in comparison to logistic regression. As the complexity of the neural network increases i.e. more neuron or more layers it'll be able to better capture a representation of the data. Logistic regression peaked at 90% accuracy while neural networks are able to achieve 100%.

Number of neurons in hidden layer	**Accuracy**	**Loss**
1	70.0	0.8919
2	83.33	0.8212
4	90.0	0.7666
8	93.33	0.7133
16	93.33	0.6993
32	100.00	0.6784