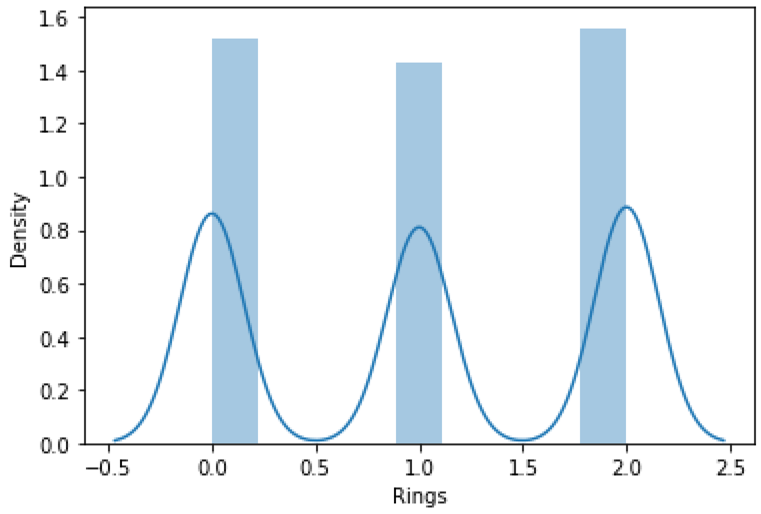
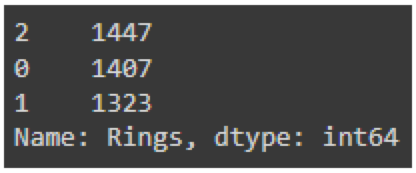
Pattern Recognition and Machine Learning Report for : Lab 7

Name : Khushi Parikh
Roll No : B20EE029

Question 1:

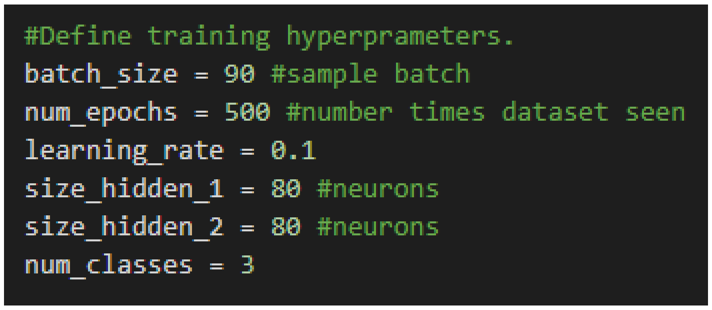
- The necessary libraries were imported and the data was analyzed. The output variable was divided into 3 classes based on the number of rings, as stated in the information document. (<9, 9-10, >10). The 'Sex' column was encoded. The output column values were analyzed :



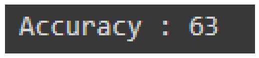


```
2    1447
0    1407
1    1323
Name: Rings, dtype: int64
```

- The data was split into two parts - training(70%) and testing(30%).
- An accuracy function was defined which checked the number of correct predictions.
- Hyperparameters were chosen as follows -



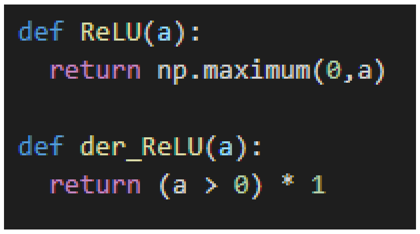
```
#Define training hyperprameters.
batch_size = 90 #sample batch
num_epochs = 500 #number times dataset seen
learning_rate = 0.1
size_hidden_1 = 80 #neurons
size_hidden_2 = 80 #neurons
num_classes = 3
```

- A class of neural networks called Net was implemented. It consisted of 2 hidden layers, each being trained on the input layer. The tanh activation function was applied to them.
- Forward propagation was defined as applying the activation function on the hidden layers and then getting the output by adding the two layers. A sigmoid activation function was then implemented on the output layer.
- An optimiser and loss function was defined from torch modules.
- In each epoch(number of times the whole data is seen), the data was shuffled. Mini-batch learning was implemented by choosing a subset of the original data. Inputs were defined as a Float tensor and labels were defined as a Long tensor. All gradients were set to 0. The neural network was trained on the training data. The accuracy on the training and testing set were around with the above hyperparameters. Final accuracy on testing set : 

```
Accuracy : 63
```

Question 2:

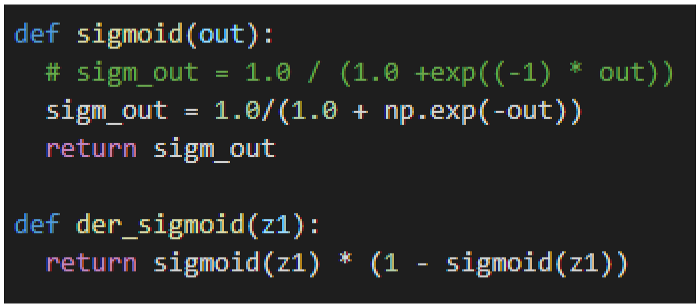
- After importing the data, unnecessary columns were dropped and the data was encoded. The data was split into training(60%), testing(20%) and validation(20%).
- Hidden layer size was taken as 50. A basic network was implemented by initializing the weights and biases to an array of required sizes.
- Three functions and their derivatives were initialized -
    1) ReLU -



```
def ReLU(a):
  return np.maximum(0,a)

def der_ReLU(a):
  return (a > 0) * 1
```

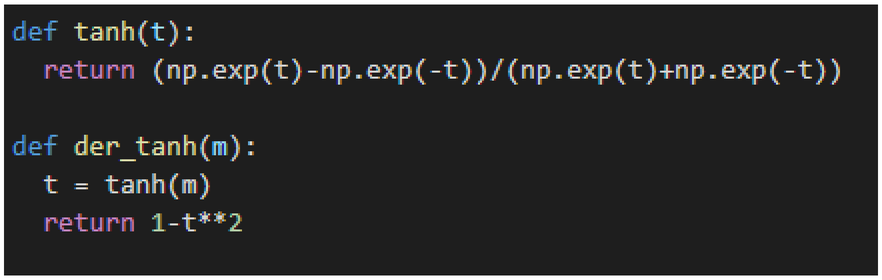
    2) Sigmoid -



```
def sigmoid(out):
  # sigm_out = 1.0 / (1.0 +exp((-1) * out))
  sigm_out = 1.0/(1.0 + np.exp(-out))
  return sigm_out

def der_sigmoid(z1):
  return sigmoid(z1) * (1 - sigmoid(z1))
```

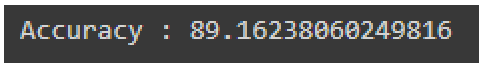
    3) Hyperbolic tan -



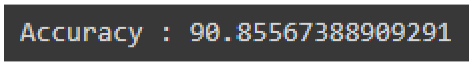
```
def tanh(t):
  return (np.exp(t)-np.exp(-t))/(np.exp(t)+np.exp(-t))

def der_tanh(m):
  t = tanh(m)
  return 1-t**2
```

- For forward propagation, the hidden layer was calculated by multiplying the weights with the neurons and adding the biases. The activation layer was then applied on both of them.

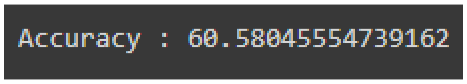
- For back propagation, the weights and biases were recalculated based on what the network had learned. The change in these parameters was returned.
- After this the function to update parameters was defined. This changed the weights and biases by multiplying the learning rate with the difference in weights obtained from the above learning.
- Now, the number of epochs was defined. For each epoch, there were 3 steps - forward propagation, backward propagation and updating the weights. For each output neuron we get the accuracy and the maximum of these is chosen as the final output.
- Accuracy on validation set : 

```
Accuracy : 89.16238060249816
```

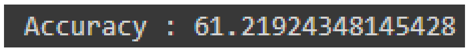
- Accuracy on testing set : 

```
Accuracy : 90.85567388909291
```

- Three functions for activation stated above were applied on the data.
    1) ReLU :

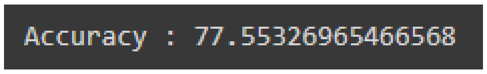
       Validation : 

```
Accuracy : 60.58045554739162
```

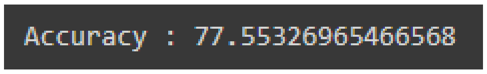
       Testing : 

```
Accuracy : 61.21924348145428
```

    2) Hyperbolic Tan :

       Validation : 

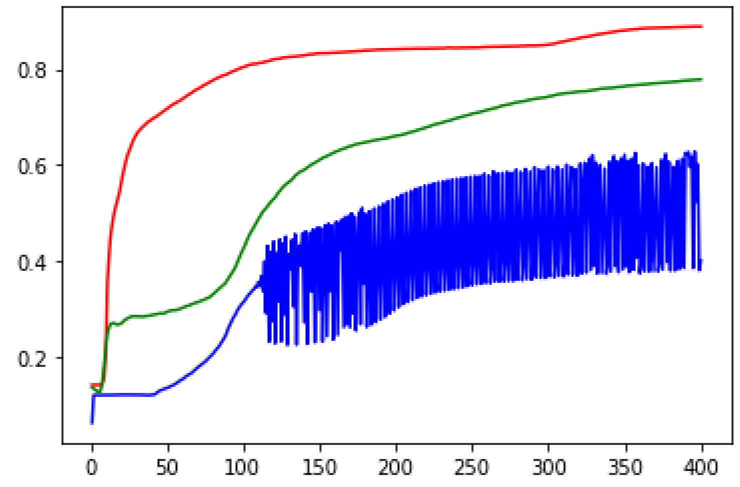
```
Accuracy : 77.55326965466568
```

       Testing : 

```
Accuracy : 77.55326965466568
```

    3) Sigmoid :
       Validation : As stated above.

- Parameters for each function were the same : Learning Rate is 0.01 and epochs were taken as 400.
  Blue is ReLU function - could not perform well due to high learning rate for it.
  Red is Sigmoid - performed the best.
  Green is tanh and it performed moderately well.



- The weights were initialized in 3 different ways :
  1) Randomly using np.uniform which generates numbers between 0 and 1.

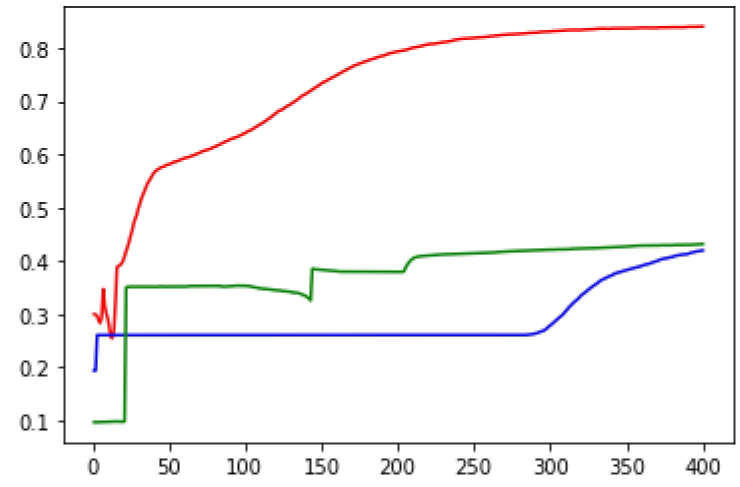
     Validation :
     ```
     Accuracy : 84.97428361498898
     ```
     Testing :
     ```
     Accuracy : 85.01652589056188
     ```
  2) Zero -

     Validation :
     ```
     Accuracy : 42.57898603967671
     ```
     Testing :
     ```
     Accuracy : 43.55490268086669
     ```

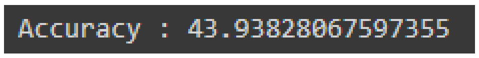
  3) Constant - The weights were initialized to a constant value of 5.

     Validation :
     ```
     Accuracy : 43.16678912564291
     ```
     Testing :
     ```
     Accuracy : 43.77524788835843
     ```

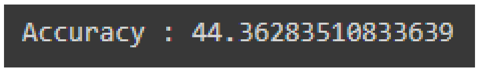
- The blue line is weights as 0, red line is weights random value and green line is weights as a constant number.The model learns best when the weights are chosen randomly.



- The hidden layer size was changed to 100. Accuracy -

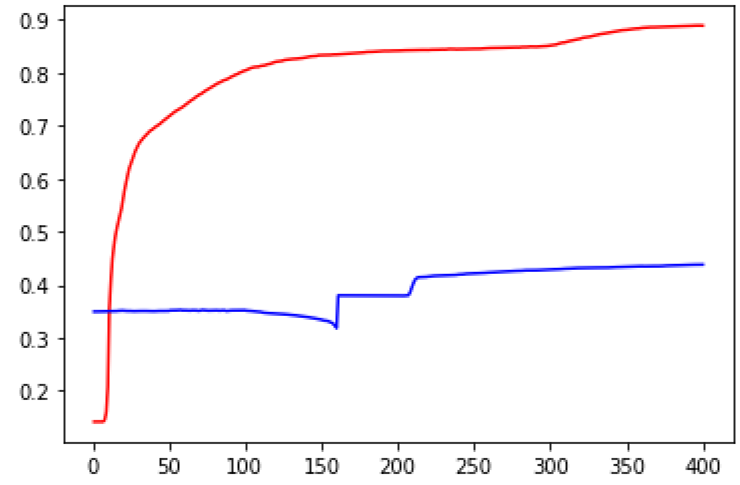
  Validation : 

```
Accuracy : 43.93828067597355
```

  Testing : 

```
Accuracy : 44.36283510833639
```

- The blue line is hidden layer size = 100 and red line is hidden layer size = 50. The variation is not deductible and does not vary directly or inversely.



- Neural networks can be trained by changing the hyperparameters but there is no completely correct way to choose the parameters with the best accuracy as each parameter does not depend only on the model.