Question 3: Conduct experiments with different values for learning rate of Adam optimizer (low values, just right values, and large values).

In the training loop, we used Adam optimizer to optimize our data and according to the most researches, we can declare learning rate with different values but it causes some problems :

CASE 01:

Learning Rate (lr) = 1

```
[37] # w1 = w0 - lr * w0.grad # w0 = 3, w1 = 70

     epochs = 10
     device = "cuda" # "cuda"
     model.to(device) # model is automatically on cpu at first. so we need to switch to gpu
     loss_fn = torch.nn.CrossEntropyLoss()
     optimizer = torch.optim.Adam(params = model.parameters(), lr = 1)
```

In the case of training, we can observe that the learning rate of 1.0 certainly is not the best. If lower down the learning rate is from 1.0 to 0.1, the AUC score jumps from 0.75 to 0.90, but the problem with this learning rate is that with an increase in epochs it becomes almost constant, which is not the best.

Output :

```
Epoch 1 train is finished!
Epoch 1 train loss -> 2.3565942501004837!
Epoch 1 train acc -> 0.10435185185185185!
Epoch 1 validation is finished!
Epoch 1 validation loss -> 2.3567388412800243!
Epoch 1 validation acc -> 0.1045!
```

.

.

```
.
Epoch 10 train is finished!
Epoch 10 train loss -> 2.356740022157606!
Epoch 10 train acc -> 0.10440740740740741!
Epoch 10 validation is finished!
Epoch 10 validation loss -> 2.3567388412800243!
Epoch 10 validation acc -> 0.1045!
```

Here, we can see the accuracy around 10 % during the whole epochs which is pretty bad.

CASE 02:

Learning Rate (lr) = 0.000001

```
epochs = 10
device = "cuda" # "cuda"
model.to(device) # model is automatically on cpu at first. so we need to switch to gpu
loss_fn = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(params = model.parameters(), lr = 0.000001)
```

```
Train boshlandi
422it [00:16, 25.36it/s]Epoch 1 train is finished!
Epoch 1 train loss -> 2.3011858197750072!
Epoch 1 train acc -> 0.12553703703703703!
Epoch 1 validation is finished!
Epoch 1 validation loss -> 2.299337442885054!
Epoch 1 validation acc -> 0.186!
.
.
.
Epoch 10 train is finished!
Epoch 10 train loss -> 2.2184234405580856!
Epoch 10 train acc -> 0.33916666666666667!
Epoch 10 validation is finished!
```

```
Epoch 10 validation loss -> 2.211006981261233!
Epoch 10 validation acc -> 0.3406666666666667!
```

The accuracy is increasing slightly in each epoch. But it is not a good choice.

CASE 03:

**Best learning rate for Adam Optimizer 3e-4 is the best learning rate for Adam, hands down. (*https://www.jeremyjordan.me/nn-learning-rate/)*

Learning Rate (lr) = 0.0003

```
# w1 = w0 - lr * w0.grad # w0 = 3, w1 = 70

epochs = 10
device = "cuda" # "cuda"
model.to(device) # model is automatically on cpu at first. so we need to switch to gpu
loss_fn = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(params = model.parameters(), lr = 0.0003)
```

```
Epoch 1 train is finished!
Epoch 1 train loss -> 1.7445425097411278!
Epoch 1 train acc -> 0.7372407407407408!
Epoch 1 validation is finished!
Epoch 1 validation loss -> 1.633535699641451!
Epoch 1 validation acc -> 0.834!

.
.
.

Epoch 10 train is finished!
Epoch 10 train loss -> 1.5152802780906172!
Epoch 10 train acc -> 0.9476111111111111!
Epoch 10 validation is finished!
```

```
Epoch 10 validation loss -> 1.5157500023537493!
Epoch 10 validation acc -> 0.9473333333333334!
```

I can see the model is performing well enough when my learning rate is 0.0003 from 83% to 94%.