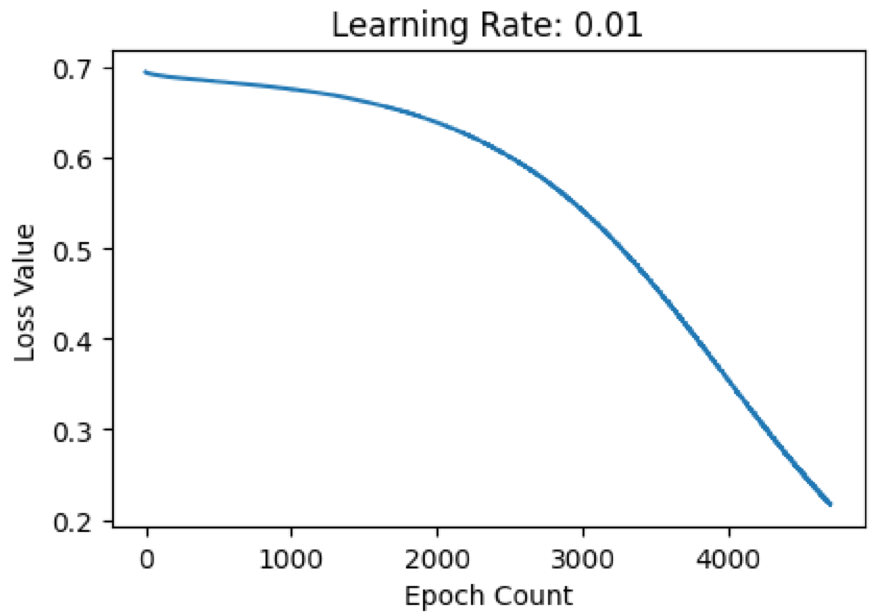
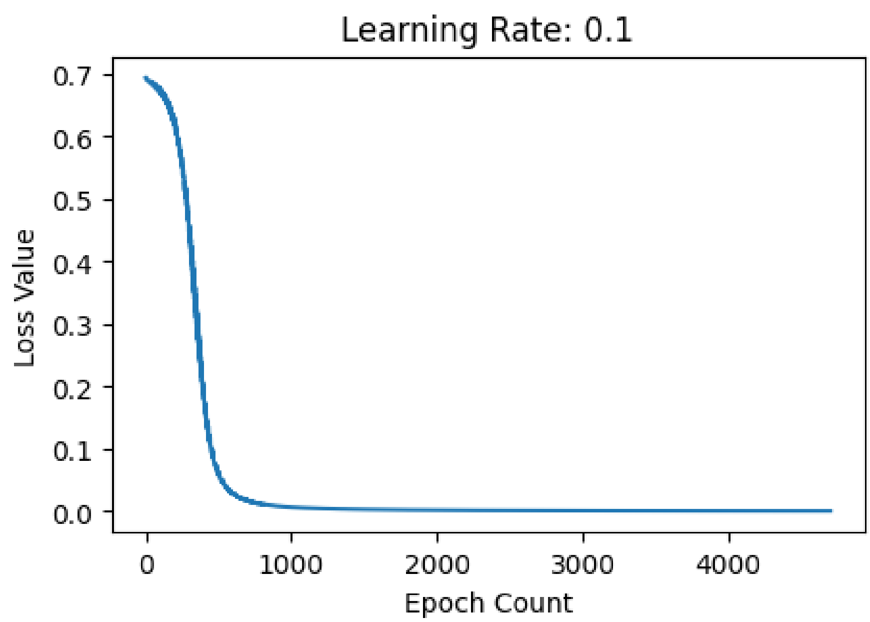
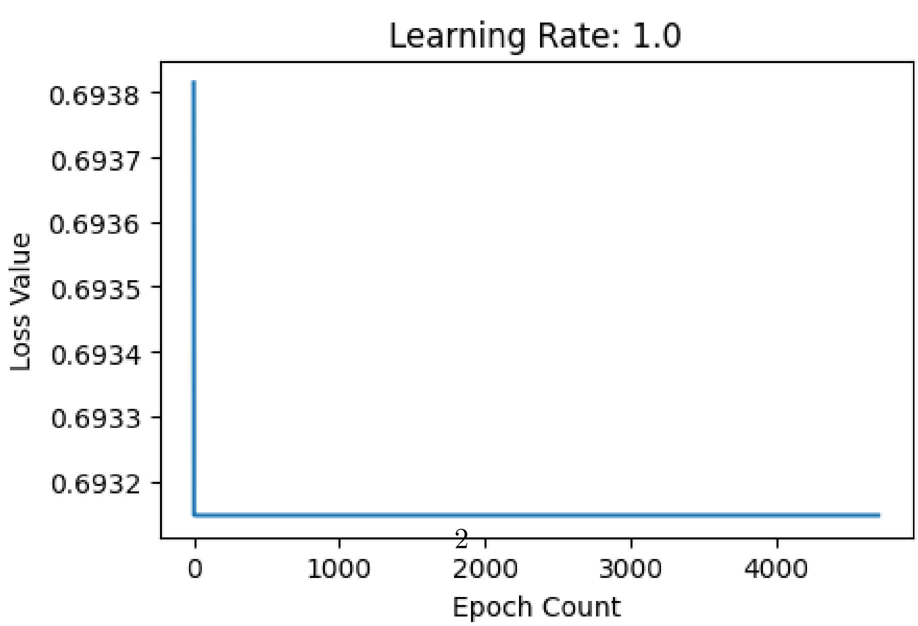
Analysis of the impact of the learning rate on the reduction of the loss value

The following code trains and tests models on the XOR dataset with varying learning rates, and then plots graphs of Loss Value against Epoch Count.

[17]:
```python
import os

import matplotlib.pyplot as plt
import numpy as np

from school_project.models.cpu.xor import XORModel as Model

# Change to root directory of project
os.chdir(os.getcwd())

# Set width and height of figure
plt.rcParams["figure.figsize"] = [5, 10]

learning_rates = [0.01, 0.1, 1.0]

figure, axis = plt.subplots(nrows=len(learning_rates), ncols=1)

for count, learning_rate in enumerate(learning_rates):
    model = Model(hidden_layers_shape=[100, 100],
                  train_dataset_size=4,
                  learning_rate=learning_rate,
                  use_relu=True)
    model.create_model_values()
    model.train(epoch_count=4_700)
    model.test()

    axis[count].set_title(f"Learning Rate: {model.learning_rate}")
    axis[count].set_xlabel("Epoch Count")
    axis[count].set_ylabel("Loss Value")
    axis[count].plot(np.squeeze(model.train_losses))

plt.tight_layout()
plt.show()
```


Learning Rate: 0.01
Loss Value
Epoch Count
0.2
0.3
0.4
0.5
0.6
0.7
0
1000
2000
3000
4000


Learning Rate: 0.1
Loss Value
Epoch Count
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1000
2000
3000
4000


Learning Rate: 1.0
Loss Value
Epoch Count
0.6932
0.6933
0.6934
0.6935
0.6936
0.6937
0.6938
0
1000
2000
3000
4000

As shown above, if the learning rate is set to too low of a value (0.01 in this case) the model will take more epochs to reduce the loss value, and may even get stuck in unwanted local minimums. If the learning rate is set to an optimal value (0.1 in this case) the model reduces the loss value efficiently and to a small enough value for predictions. On the other hand, if the learning rate is set to too high of a value (1.0 in this case) the model may learn too quickly and even ‘jump over’ minima, causing the loss value to stop reducing.