HW 3

HW

Question 1

We are not assigning the output directly to the prediction because the statement `output = predict(weights, instance)` only indicates whether the output is 1 or 0. It does not provide information on how far off we are from making a correct prediction. Without knowing this distance, we cannot effectively adjust the weights of the parameters to improve our chances of getting the correct answer.

Question 2

Python

```python
instances_tr = read_data("train.dat")
instances_te = read_data("test.dat")
tr_percent = [5, 10, 25, 50, 75, 100] # percent of the training dataset to train with
num_epochs = [5, 10, 20, 50, 100]     # number of epochs
lr_array = [0.005, 0.01, 0.05]        # learning rate

result = []

for lr in lr_array:
  for tr_size in tr_percent:
    for epochs in num_epochs:
      size =  round(len(instances_tr)*tr_size/100)
      pre_instances = instances_tr[0:size]
      weights = train_perceptron(pre_instances, lr, epochs)
      accuracy = get_accuracy(weights, instances_te)
      print(f"#tr: {len(pre_instances):0}, epochs: {epochs:3}, learning rate: {lr:.3f}; "
          f"Accuracy (test, {len(instances_te)} instances): {accuracy:.1f}")
      temp = [len(pre_instances), epochs, lr, accuracy]
      result.append(temp)
```

Text

```
#tr: 20, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 85.7
#tr: 40, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 85.7
#tr: 400, epochs:   5, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  10, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  20, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  50, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs: 100, learning rate: 0.005; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 20, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 42.9
#tr: 40, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 100, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 28.6
#tr: 200, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 200, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
```

```
#tr: 300, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 85.7
#tr: 400, epochs:   5, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  10, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  20, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  50, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs: 100, learning rate: 0.010; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 20, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 20, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 42.9
#tr: 20, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 42.9
#tr: 20, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 42.9
#tr: 40, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 40, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 28.6
#tr: 40, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 42.9
#tr: 100, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 100, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 28.6
#tr: 100, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 28.6
#tr: 100, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 28.6
#tr: 200, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 200, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 300, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 300, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 85.7
#tr: 400, epochs:   5, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  10, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  20, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs:  50, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
#tr: 400, epochs: 100, learning rate: 0.050; Accuracy (test, 14 instances): 71.4
```

Question 3

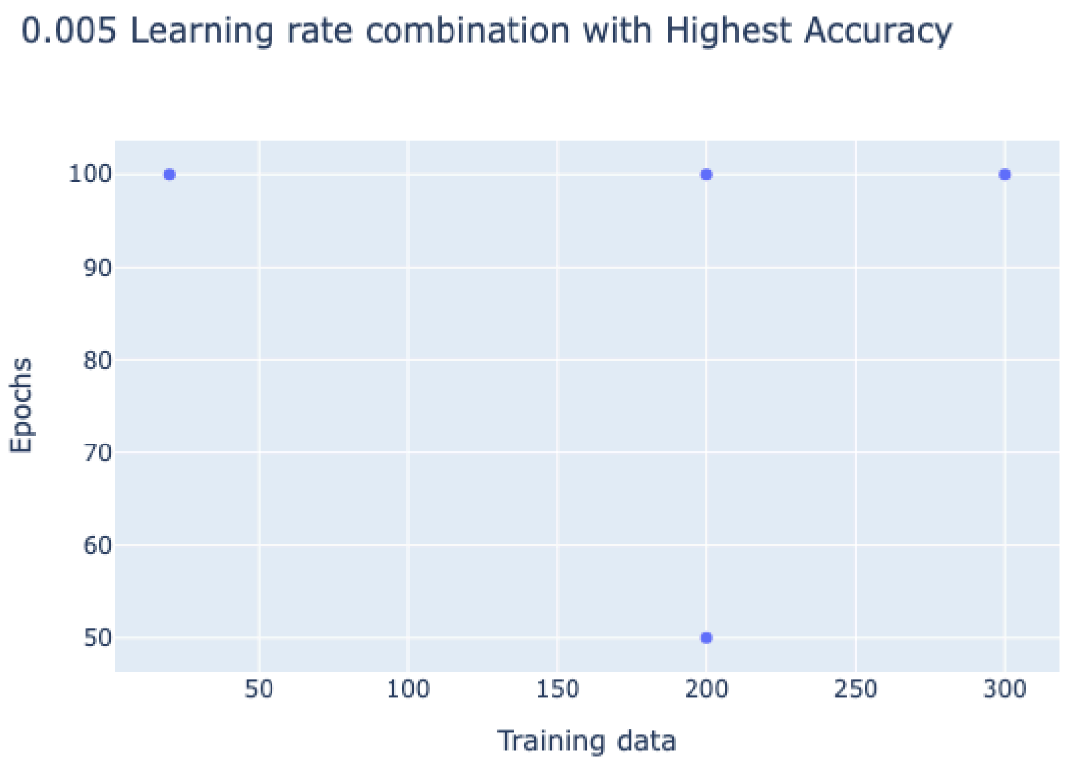
Code for Plotting Graph to better answer the questions

Python

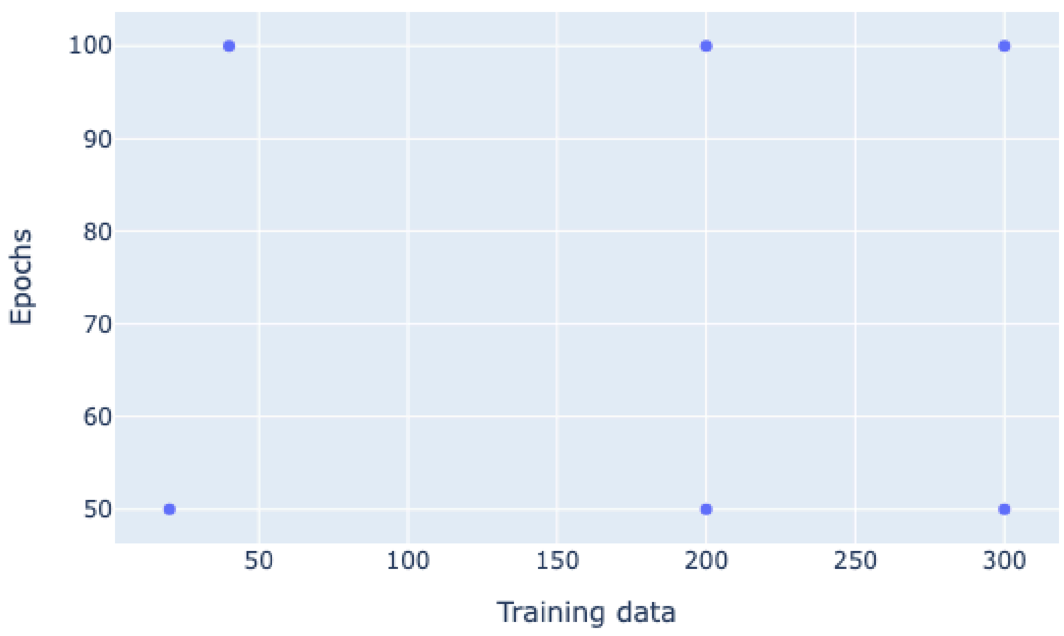
```python
highest_accuracy = []
high_acc = 0
i = 0
while i < len(result):
    if result[i][3] > high_acc:
        highest_accuracy.clear()
        high_acc = result[i][3]
        highest_accuracy.append(result[i])
    elif result[i][3] == high_acc:
        highest_accuracy.append(result[i])
    i += 1

lr_0005 = []
lr_001 = []
lr_005 = []

i = 0
while i < len(highest_accuracy):
    if highest_accuracy[i][2] == 0.005:
        lr_0005.append(highest_accuracy[i])
    elif highest_accuracy[i][2] == 0.01:
        lr_001.append(highest_accuracy[i])
    elif highest_accuracy[i][2] == 0.05:
        lr_005.append(highest_accuracy[i])
    i += 1
```

Python

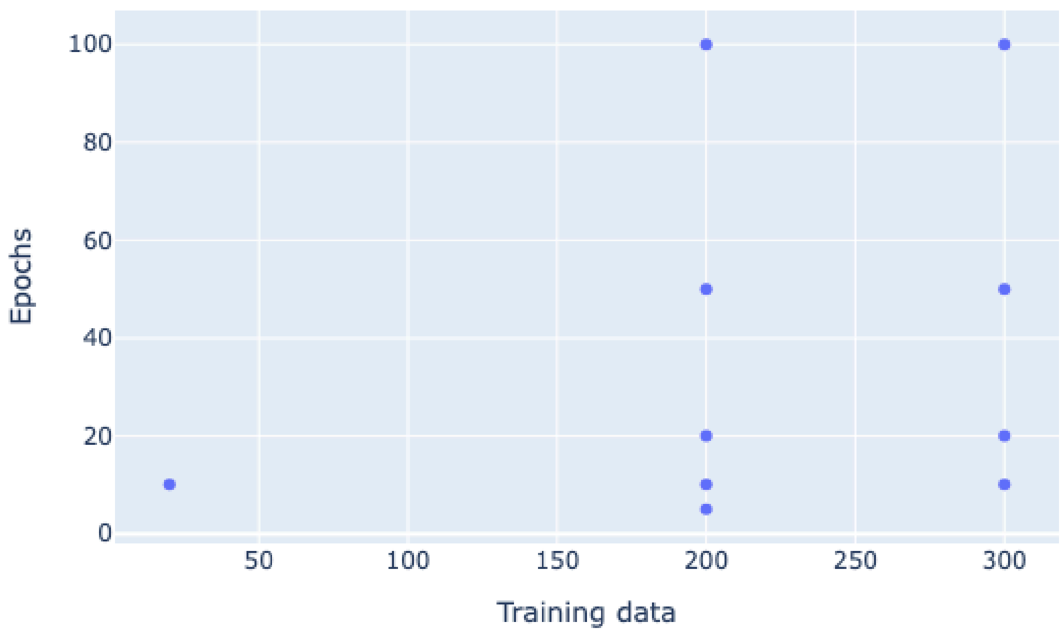
```python
import plotly.graph_objects as go

def plotting(lis):
    i = 0
    x = []
    y = []
    while i < len(lis):
        x.append(lis[i][0])
        y.append(lis[i][1])
        i += 1

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=x, y=y, mode='markers'))

    fig.update_layout(
        title=f'{lis[0][2]} Learning rate combination with Highest Accuracy',
        xaxis_title='Training data',
        yaxis_title='Epochs'
    )

    fig.show()

print(f'Highest accuracy: {lr_0005[0][-1]:.2f}')
plotting(lr_0005)
plotting(lr_001)
plotting(lr_005)
```



0.01 Learning rate combination with Highest Accuracy



0.05 Learning rate combination with Highest Accuracy



A

No, you don't need to train with all the data to achieve the highest accuracy. In fact, using more data can lead to lower accuracy. Based on the graphs provided, it appears that 200 samples which is half of the training data seems to be the ideal size for achieving the highest possible accuracy from the model.

B

Adding more data can lead to overfitting or underfitting, especially if the learning rate is too high or the number of epochs is too large. Instead of learning, the model will start to memorize the data, decreasing accuracy.

C

Getting higher accuracy is possible by adding additional hyperparameters like beta decay.

D

No, it isn't worth training for more epochs. As shown in the graphs above, you can achieve the same level of accuracy more consistently with fewer epochs rather than more.