Exercise 1

Exercise 2

Exercise 3

Exercise 4

See Referencesappendix for code.

(a)

```
Final probabilities: [0.53950429 0.26975214 0.19074357]

Tracked weight vectors (without initial one vector):

Round:  1 Weights:  [0.5 0.5 1. ]
Round:  2 Weights:  [0.5  0.25 0.5 ]
Round:  3 Weights:  [0.25 0.25 0.25]
Round:  4 Weights:  [0.25  0.125 0.125]
Round:  5 Weights:  [0.125  0.125  0.0625]
Round:  6 Weights:  [0.125      0.0625     0.04419417]
```

Exercise 5

Exercise 6

Appendix

Code for Exercise 4

```python
import numpy as np


def mwu_algorithm(loss_matrix, events, rounds, alpha):
    # initial weight vector of 1s
    weights = np.ones((loss_matrix.shape[0]))
    weights_tracking = {}
    # more convenient to loop through rounds and events
    rounds_arr = [i for i in range(rounds)]
    for round, event in zip(rounds_arr, events):
        # need to use event-1 as events start at 1 but indexing at 0
        weights = np.power((1 - alpha), loss_matrix[:, event-1]) * weights
        # getting the current probabilities, not really needed here
        p = probabilities(weights)
        # loss isn't really needed
        loss = calculate_loss(loss_matrix, p, event-1)
        weights_tracking[round] = weights

    return p, weights_tracking

def probabilities(weights):
    return weights / np.sum(weights)

```

```
24 def calculate_loss(loss_matrix, probabilities, event):
25     return np.sum(probabilities * loss_matrix[:, event])
26
27
28 loss_matrix = np.array([[0,1,1,0],
29                         [1,0,1,1],
30                         [1,1,0,0.5]])
31
32 observed_events = [3,1,2,1,2,4]
33
34 p_6, weights_tracking = mwu_algorithm(loss_matrix, observed_events, 6, alpha
       =0.5)
35
36 print(f'Final probabilities: {p_6}\n')
37 print(f'Tracked weight vectors (without initial one vector): \n')
38 for key, val in weights_tracking.items():
39     print(f'Round:\t{key + 1}\tWeights:\t{val}')
```