3. Short Writeup

3.1 Estimate of two coins are: - ThetaA = 0.568 and ThetaB = 0.494

3.2 There were 4 ways to make multiple API calls to the given URL. 1- Use requests.get (URL) in a for loop, 2- Use asyncio to make asynchronous call to the given URL, 3- Use threads, 4- Use multiprocessing.

The first option is clearly a brute force approach with high runtime. I chose option 2 over 3 and 4 because the task in hand deals with doing many connections. Moreover, due to GIL usage of threads in python does not achieve true parallelism. The task is high I/O bound and hence using multiprocessing is not feasible.

After running asyncio to make 30 API calls I extract the "data" from the response json. These draws are then appended to a list. Since the coin tosses are un-labelled, this is a semi-supervised learning problem. I assumed the data is Gaussian Fit data. I used sklearn.mixture.gmm to fit the data as follows.

gmm = mixture.GaussianMixture (n_components=2, max_iter=10000, covariance_type='tied', init_params = "random").fit (coin_flips)

I had to choose the covariance type variable in the model, I chose "tied" instead of "full" because "full" gives good results if we have a large dataset, in smaller datasets like the one in hand tends to overfit the data.

I had to choose "random" for init_params, we don't know that our data is clustered via kmeans algorithm or not, assuming it is random I chose "random" for init_params. If we know the coin draws are kmeans clustered then choosing "kmeans" would be appropriate.

This gives us the model with the estimated gaussian parameters - mean, variances etc. The labels are then predicted for each coin draw using the fitted model. I get a dictionary with 0 and 1 as the keys which are the type of each coin, and coin draws as values. Now that I have labelled data, theta values are calculated.