For this project, I try four different types of codes. I found BO\_UCB\_2 most successful in finding maximum observation. Following are details of the code files in this folder.

**Initial codebase**

BO\_UCB\_2. ipynb Develop using code from MP-PCMLAI-M12-BayesianOptimNotebook from Required activity 12.2: Jupyter Notebook on Bayesian optimisation. It uses RBF kernel and GaussianProcessRegressor from sklearn.gaussian\_process

Botorch\_UCB\_2.ipynb Develop using code from CapstonUsingBotorchNB2.ipynb from week 18 office hours. It uses UpperConfidenceBound from botorch.acquisition

Monte\_carlo\_EI\_2.ipynb Develop using code from CapstonUsingBotorchNB2.ipynb from week 18 office hours. It uses qExpectedImprovement from botorch.acquisition.monte\_carlo

BO\_EI\_2.ipynb Develop using code from <https://machinelearningmastery.com/what-is-bayesian-optimization/>. It uses estimated improvement using GaussianProcessRegressor from sklearn.gaussian\_process and minimize function from scipy

**Code Modifications**

BO\_UCB\_2.ipynb   
Modify it to accept the number of input values according to the function and also product values between 0 and 1. Modify it to run code in a loop so I can easily experiment with different values of beta, rbf\_lengthscale and noise\_assumption. Later modify it to use grid values around the existing highest observed values.

Botorch\_UCB\_2.ipynb  
Modify it to use two inputs and produce output in the range of 0 to 1. Modify it to run code in a loop for different values of beta and noise-level, also manually fine-tune different values of num\_restarts and raw\_samples. It uses UpperConfidenceBound from botorch.acquisition  
  
Monte\_carlo\_EI\_2.ipynb  
Modify it to use two inputs and produce output in the range of 0 to 1. Modify it to run code in a loop for different values of beta and noise level, also manually fine-tune different values of num\_restarts and raw\_samples  
   
BO\_EI\_2. ipynb

Modify it to work with multiple dimensions, limiting the output range between 0 and 1. I experiment with different values of noise, length scale, number of iterations etc. It uses estimated improvement using GaussianProcessRegressor from sklearn.gaussian\_process and minimize function from scipy.optimize. I tried different values but the output is not useful I produce random values very different or away from existing higher values.

**Results**  
After fine-tuning the values of beta = 0.001 and rbf\_lengthscale = 0.9 in code file BO\_UCB\_2.ipynb. I am able to push up the observation values from 0.26215 and 0.3079 to 0.69483 during the final weeks of the project. With the passage of time, my results are getting better and better if I have more time, I may be able to get the maximum value of this function. Also, I will try look into the other code files again to find out why some functions are working well and some are not. Although in theory, all of them should work.