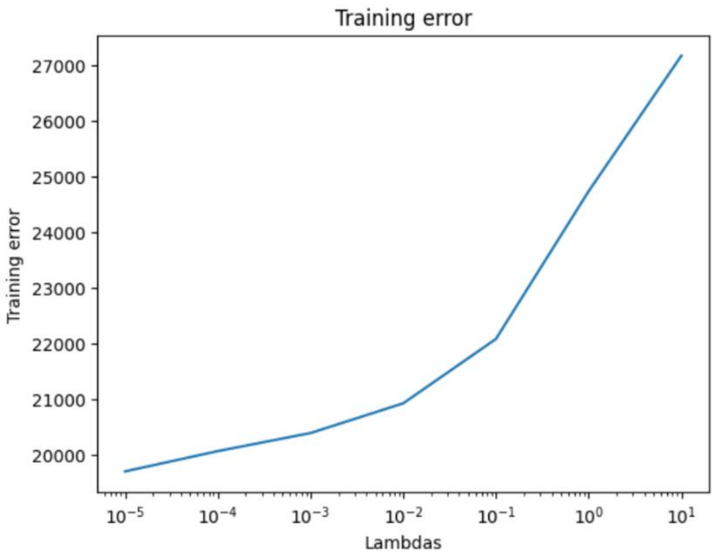

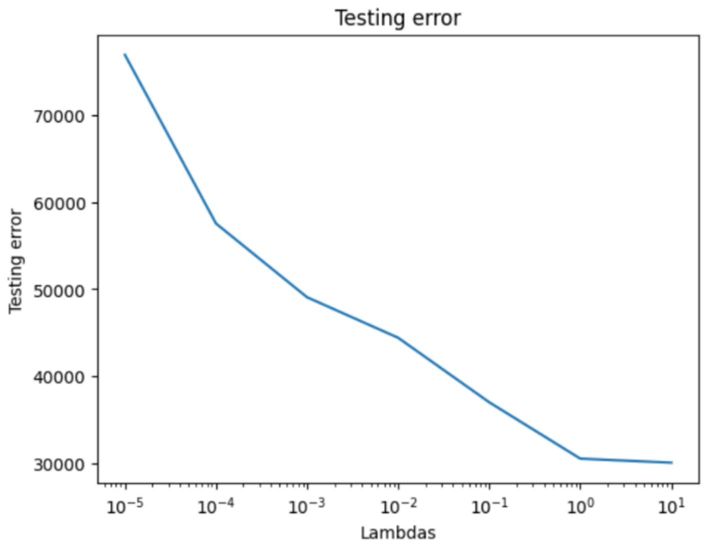

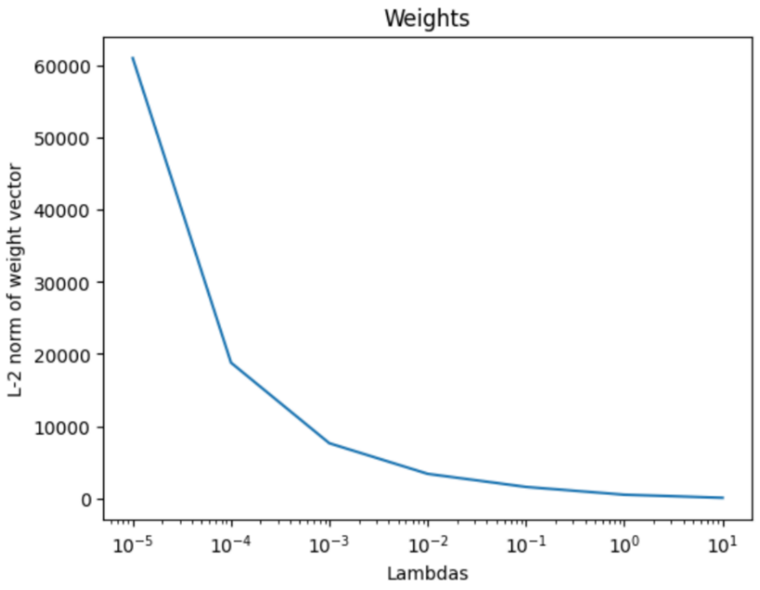

1-4)

Training error: [19695.354823963666, 20060.116138737925, 20384.664250733877, 20918.86518165216, 22075.715668745594, 24730.739328369953, 27165.12612819299]

Testing error: [76911.07208682552, 57531.76027004541, 49060.59804050351, 44405.139402623994, 36994.6967266464, 30517.773939737766, 30057.30349989966]

L2 norms: [60957.42731559241, 18794.92102474763, 7644.179449987964, 3408.8665326749483, 1597.6750186712156, 516.3293421748878, 92.22727252676054]

Weight vectors: See code output for full list of weight vectors

5) Analyze: How does regularization affect model weights, training performance, and generalization? Use your plots/results as evidence

From the regularization done in my code, it would appear that regularization increases training error but improves generalization (decrease in testing error). As for model weights, it heavily decreases them and their impact. These can be seen in the above plots -- as lambda values increase, the effects of regularization increase.

Use of AI:

Used ChatGPT to create a code skeleton for me and to explain what needs to be done in python to match assignment instructions (e.g. tells me that here I need to compute X^T X) and debugging. Also asked it some questions to gain understanding of concepts and the variables being used.

Google Gemini (google search) was used to tell me how to calculate certain things such as L2 norm, as well as for numpy functions (e.g. told me how to use plt.xscale('log'))

From the code skeleton built by ChatGPT, I filled out each section accordingly based on the listed requirements, sometimes adding, removing, or combining steps that it gave me to better fit the assignment instructions. I made sure to include code comments that explained what I was calculating at different steps.