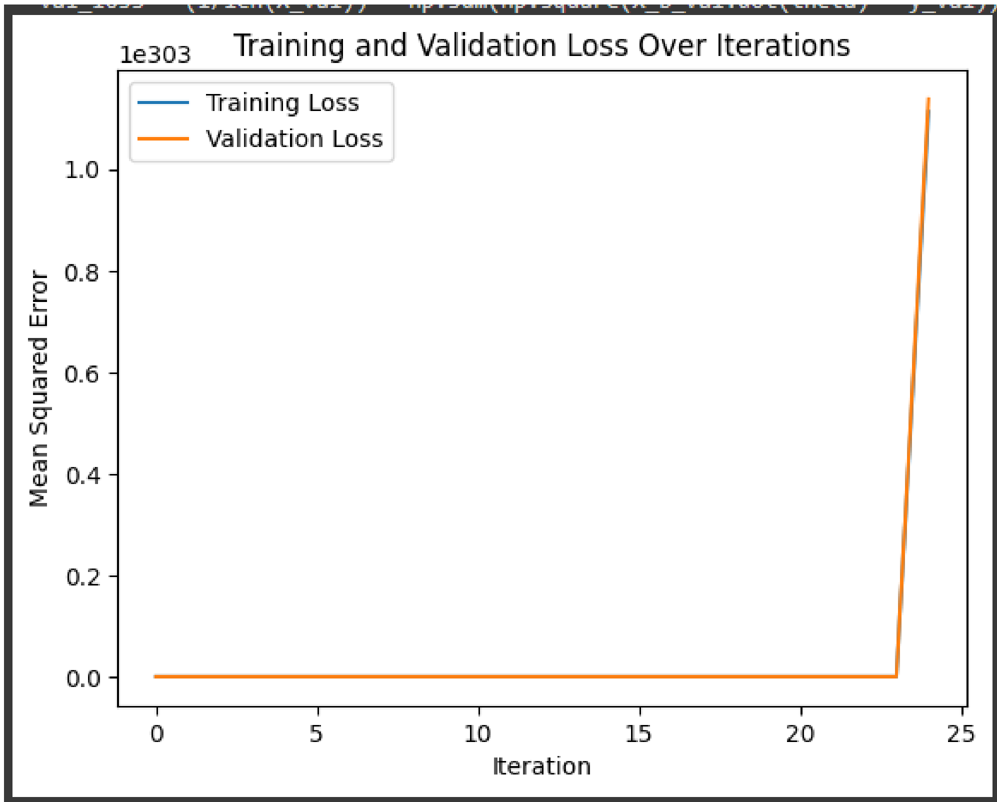
HW 2
Swayam Mehta - 801207470

Github: https://github.com/Swayyum/Intro-to-ML--4105/blob/main/HW2.ipynb

Problem 1 (30 points)

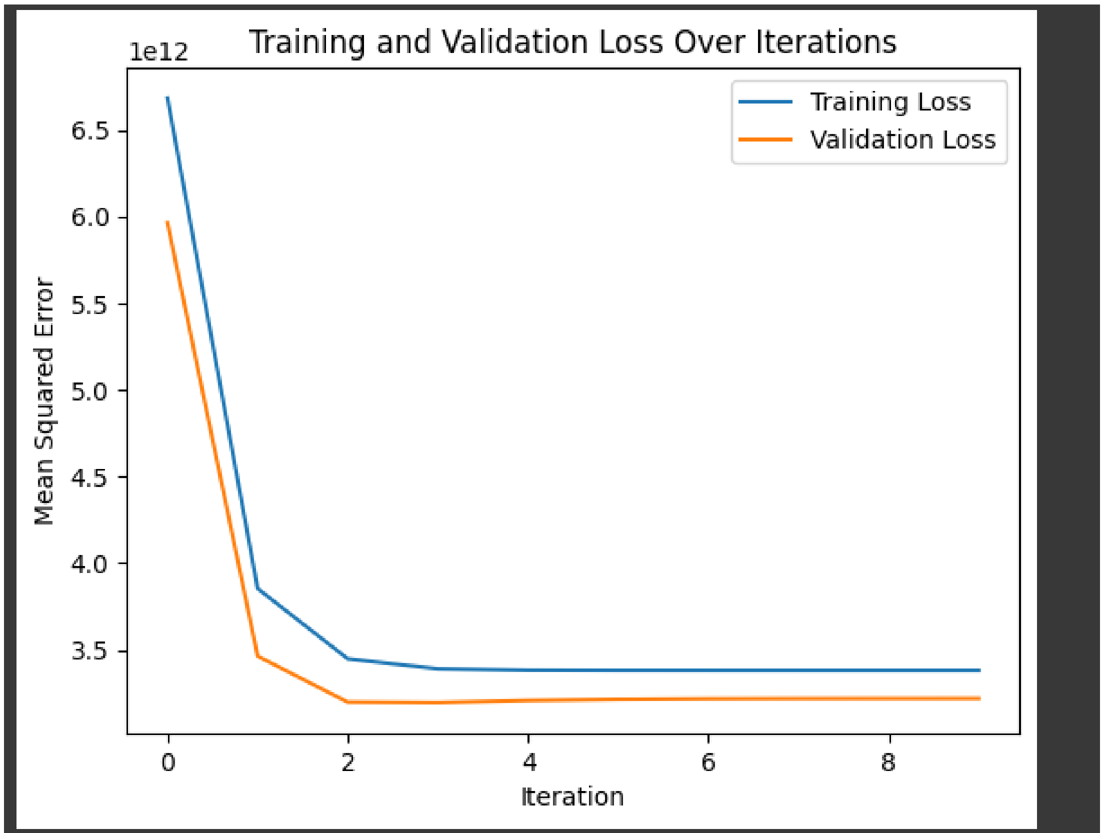
1. a) Identify the best parameters for your linear regression model, based on the above input variables.
   - **Based on the input values, I found the learning rate of 0.01 and 10 iterations as the best parameters for the linear regression model**
   - **The graph goes to infinity for a higher learning rate because the data is not preprocessed**

Plot the training and validation losses (in a single graph, but two different lines). For the learning rate, explore different values between 0.1 and 0.01 (your choice). Initialize your parameters (thetas to zero). For the training iteration, choose what you believe fits the best.



```
Best parameters: [-6.87732743e+288 -4.15567186e+292 -2.05067054e+289 -9.15106451e+288
 -1.26723506e+289 -5.86967994e+288]
```

- **Lowering the learning rate to 0.00000001 shows the loss more accurately**

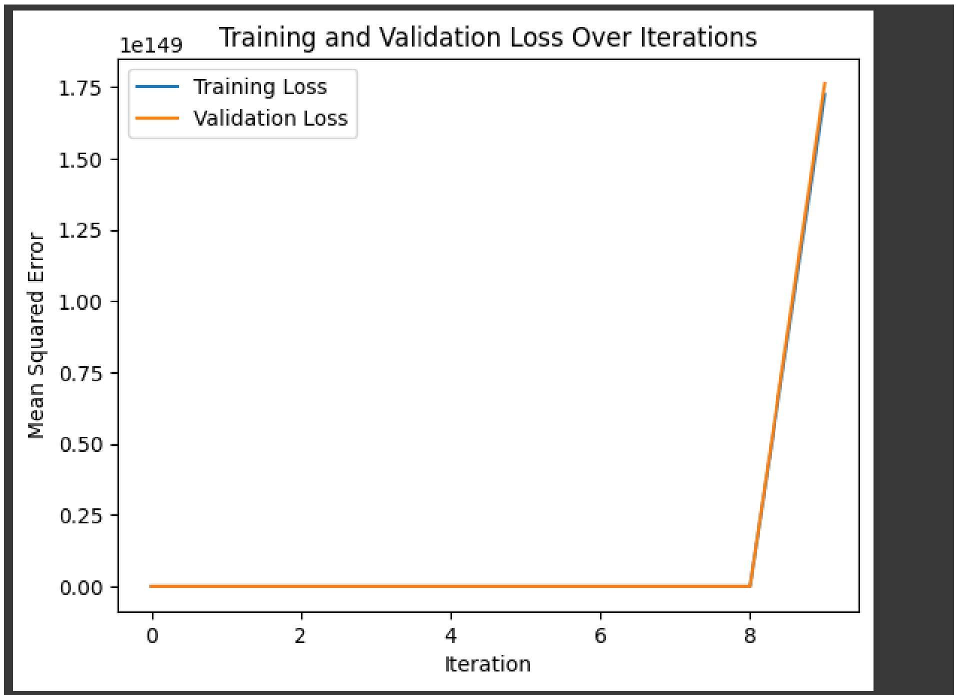


```
Best parameters: [2.14811956e-01 8.61862601e+02 6.97796647e-01 3.48937270e-01
 5.07559681e-01 1.78137854e-01]
```

1. b) Develop a gradient decent training and evaluation code, from scratch, that predicts housing price based on the following input variables: Area, bedrooms, bathrooms, stories, main road, guestroom, basement, hot water heating, air conditioning, parking, and prefarea. Identify the best parameters for your linear regression model, based on the above input variables.

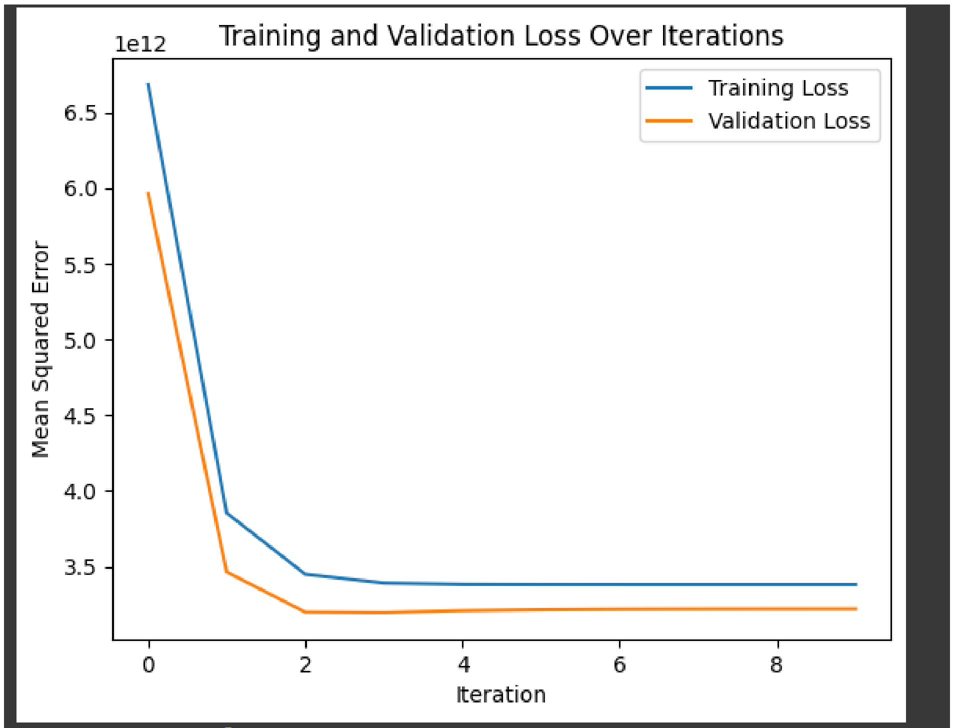
- **For this data set, I found the learning rate of 0.01 and the 10 iterations to be the best parameters**
- **Trying different iteration values from 5 - 50 did not affect the model**

Plot the training and validation losses (in a single graph, but two different lines) over your training iteration. Compare your linear regression model against problem 1 a. For the learning rate, explore values between 0.1 and 0.01 (your choice). Initialize your parameters (thetas to zero). For the training iteration, choose what you believe fits the best.



```
Best parameters: [-1.23277678e+67 -7.44913749e+70 -3.67587417e+67 -1.64034939e+67
 -2.27154802e+67 -1.11757432e+67 -2.42530096e+66 -4.54614458e+66
 -6.84331373e+65 -4.19005402e+66 -1.05215365e+67 -3.32907853e+66]
```

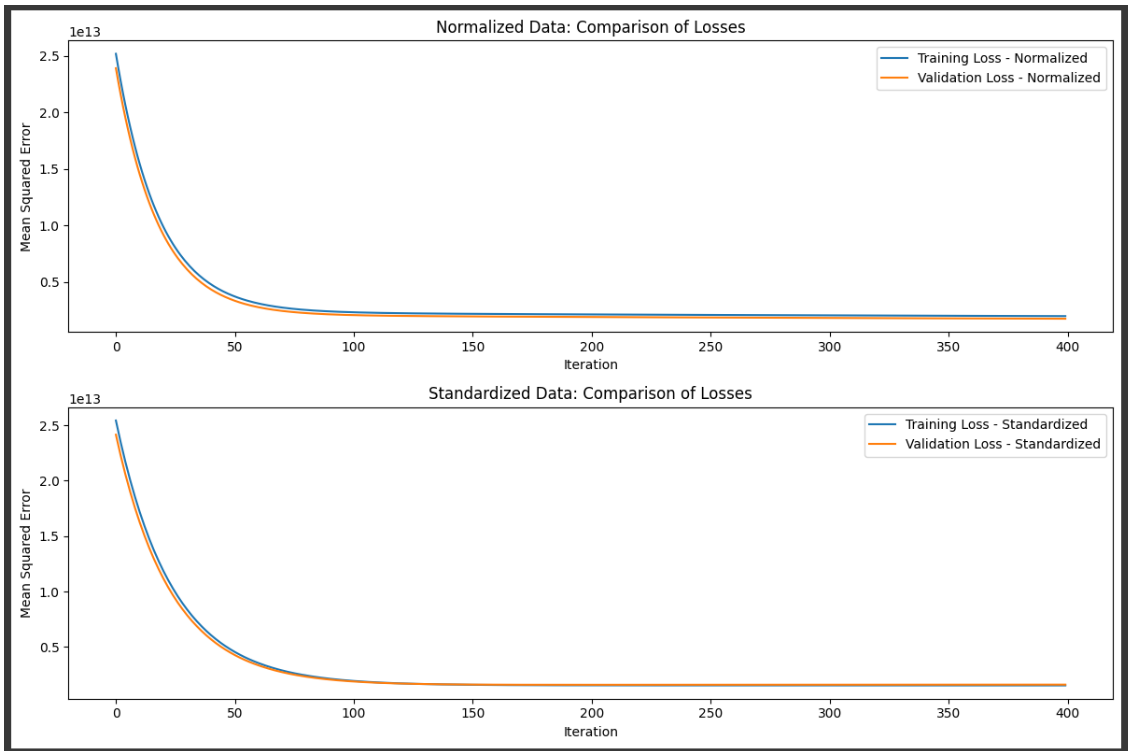
- **Using 0.00000001 as the learning rate to show loss more accurately**

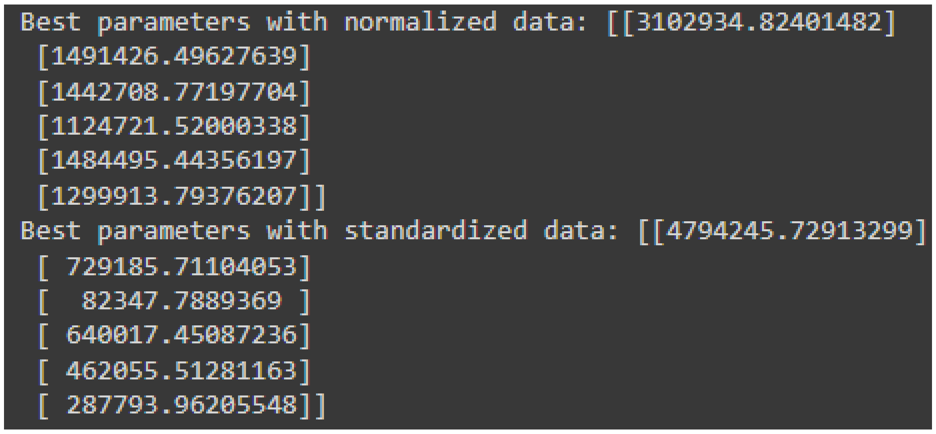


```
Best parameters: [2.14811946e-01 8.61862558e+02 6.97796616e-01 3.48937257e-01
 5.07559661e-01 1.91187916e-01 5.59824265e-02 9.57666100e-02
 2.14144857e-02 1.09964782e-01 1.78137847e-01 6.78880374e-02]
```

2. a) Repeat problem 1 a, this time with input normalization and input standardization as part of your pre-processing logic. You need to perform two separate trainings for standardization and normalization. **In both cases, you do not need to normalize the output!**

Plot the training and validation losses for both training and validation sets based on input standardization and input normalization. Compare your training accuracy between both scaling approaches as well as the baseline training in problem 1 a. Which input scaling achieves the best training? Explain your results.



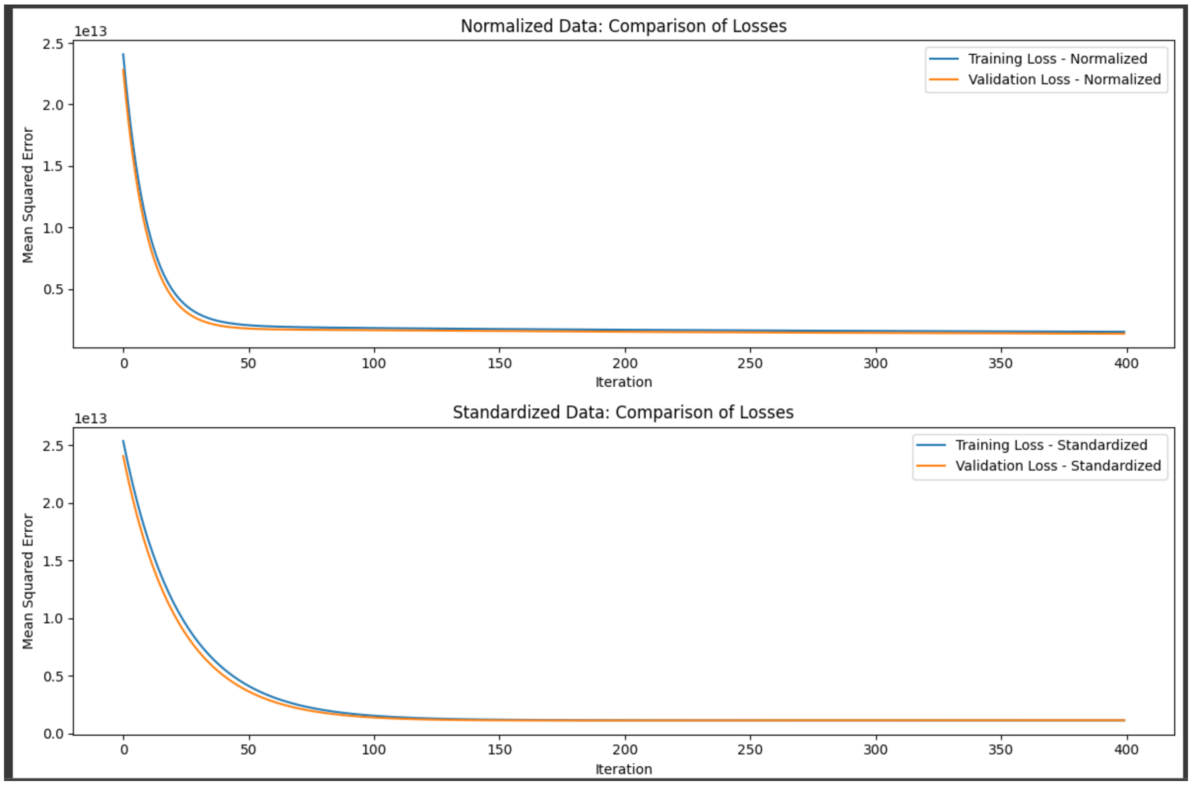


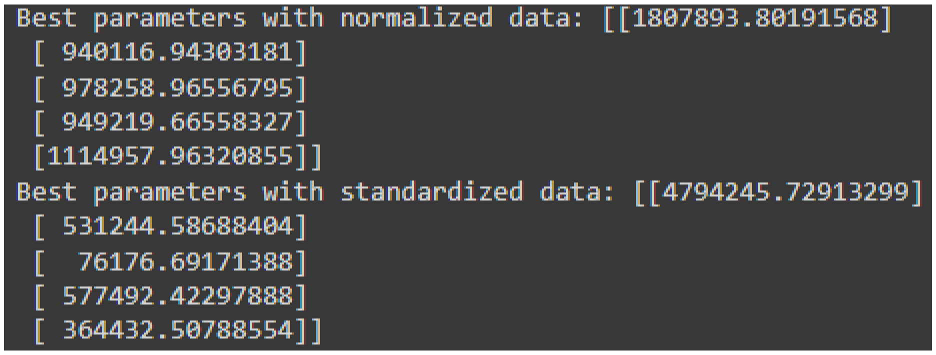
```
Best parameters with normalized data: [[3102934.82401482]
 [1491426.49627639]
 [1442708.77197704]
 [1124721.52000338]
 [1484495.44356197]
 [1299913.79376207]]
Best parameters with standardized data: [[4794245.72913299]
 [ 729185.71104053]
 [  82347.7889369 ]
 [ 640017.45087236]
 [ 462055.51281163]
 [ 287793.96205548]]
```

- **Based on the graphs and the output data, the standardization input scaling achieves the best results.**
- **The standardization graph converges vs. the normalization graph doesn't converge**
- **Using a learning rate of 0.01 and 400 iterations showed a better accuracy plot with the standardization converging around 200 iterations**

2. b) Repeat problem 1 b, this time with input normalization and input standardization as part of your pre-processing logic. You need to perform two separate trainings for standardization and normalization. **Experiment between normalizing and not normalizing the output!**

Plot the training and validation losses for both training and validation sets based on input standardization and input normalization. Compare your training accuracy between both scaling approaches and the baseline training in problem 1 b. Which input scaling achieves the best training? Explain your results. **Experiment between normalizing and not normalizing the output!**



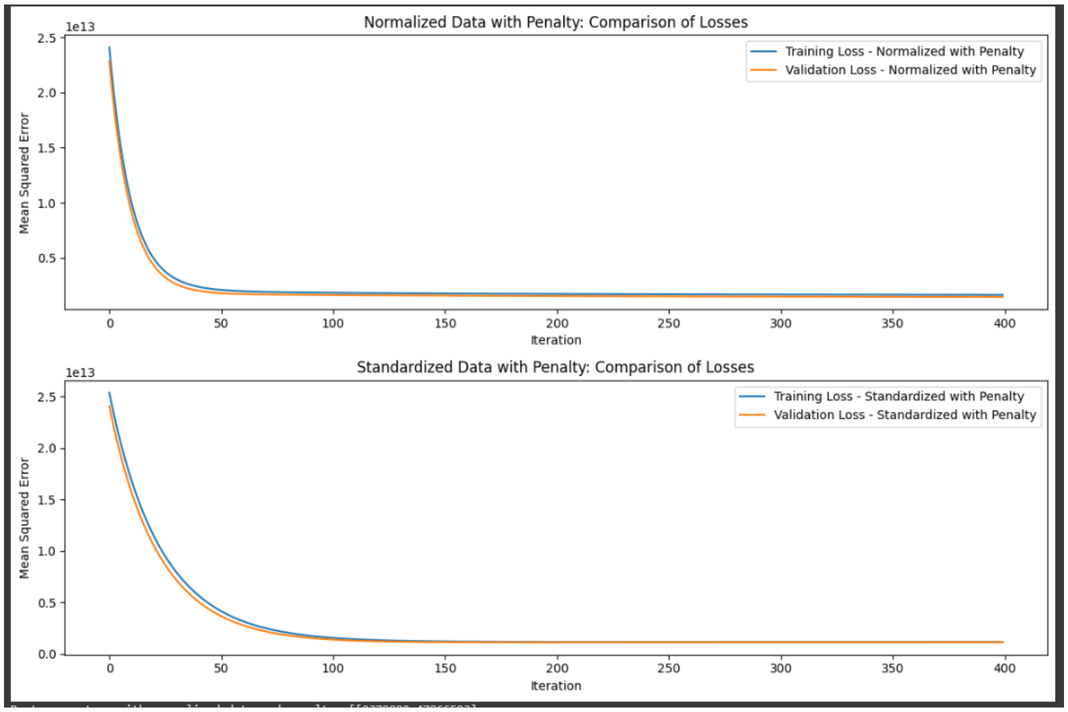


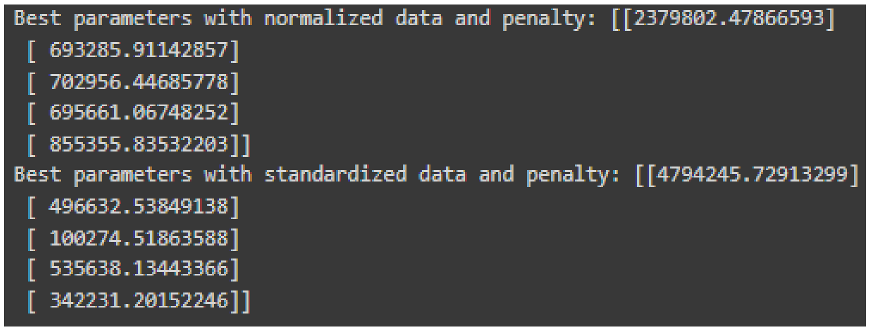
```
Best parameters with normalized data: [[1807893.80191568]
 [ 940116.94303181]
 [ 978258.96556795]
 [ 949219.66558327]
 [1114957.96320855]]
Best parameters with standardized data: [[4794245.72913299]
 [ 531244.58688404]
 [  76176.69171388]
 [ 577492.42297888]
 [ 364432.50788554]]
```

- **Based on the graphs and the output data, the standardization input scaling achieves the best results.**
- **The standardization graph converges vs. the normalization graph doesn't converge**
- **Using a learning rate of 0.01 and 400 iterations showed a better accuracy plot with the standardization converging around 200 iterations**

3. a) Repeat problem 2 a, this time by adding a parameters penalty to your loss function. **Note that in this case, you need to modify the gradient decent logic for your training set, but you don't need to change your loss for the evaluation set.**

Plot your results (both training and evaluation losses) for the best input scaling approach (standardization or normalization). Explain your results and compare them against problem 2 a.

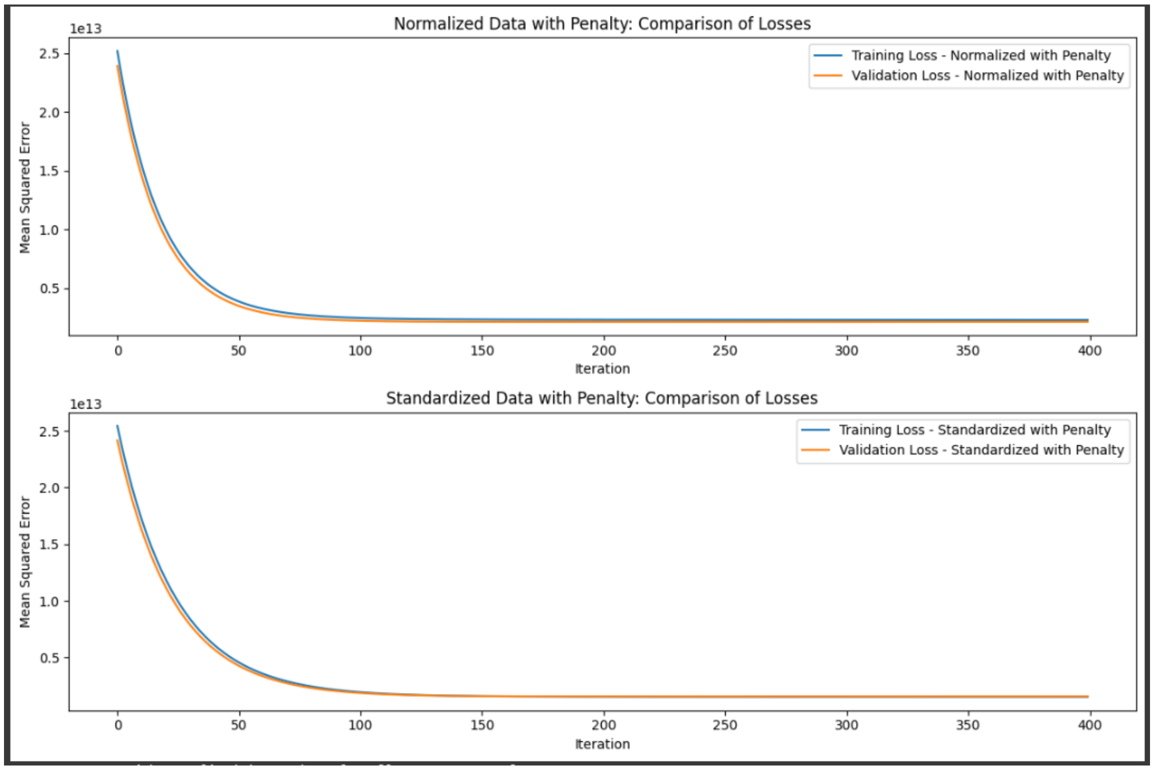


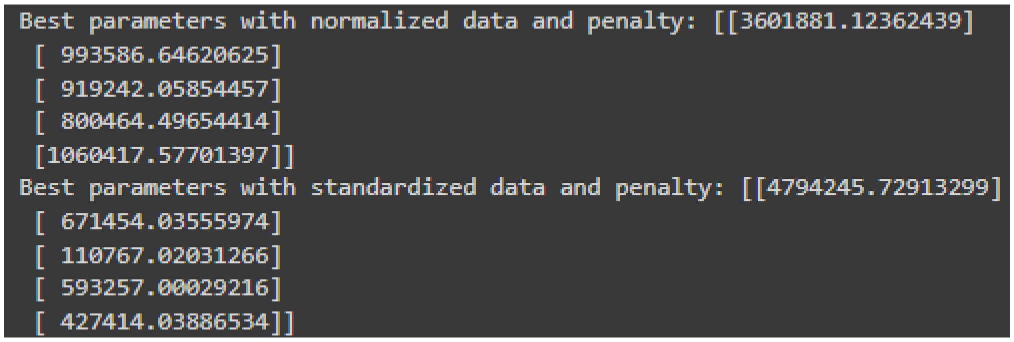

```
Best parameters with normalized data and penalty: [[2379802.47866593]
 [ 693285.91142857]
 [ 702956.44685778]
 [ 695661.06748252]
 [ 855355.83532203]]
Best parameters with standardized data and penalty: [[4794245.72913299]
 [ 496632.53849138]
 [ 100274.51863588]
 [ 535638.13443366]
 [ 342231.20152246]]
```

- **Based on the graphs and the output data, the standardization input scaling achieves the best results.**
- **The standardization graph converges vs. the normalization graph doesn't converge**
- **Using a learning rate of 0.01,400 iterations and a loss penalty of 0.1 showed a better accuracy plot with the standardization converging around 200 iterations**

3 b) Repeat problem 2 b, this time by adding a parameters penalty to your loss function. **Note that in this case, you need to modify the gradient decent logic for your training set, but you don't need to change your loss for the evaluation set.**

Plot your results (both training and evaluation losses) for the best input scaling approach (standardization or normalization). Explain your results and compare them against problem 2 b.





```
Best parameters with normalized data and penalty: [[3601881.12362439]
 [ 993586.64620625]
 [ 919242.05854457]
 [ 800464.49654414]
 [1060417.57701397]]
Best parameters with standardized data and penalty: [[4794245.72913299]
 [ 671454.03555974]
 [ 110767.02031266]
 [ 593257.00029216]
 [ 427414.03886534]]
```

- **Based on the graphs and the output data, the standardization input scaling achieves the best results.**
- **The standardization graph converges vs. the normalization graph doesn't converge**
- **Using a learning rate of 0.01, 400 iterations and a loss penalty of 0.1 showed a better accuracy plot with the standardization converging around 200 iterations**