GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105

Part 1a)

Develop a gradient decent training and evaluation code that predicts housing price based on the following input variables:

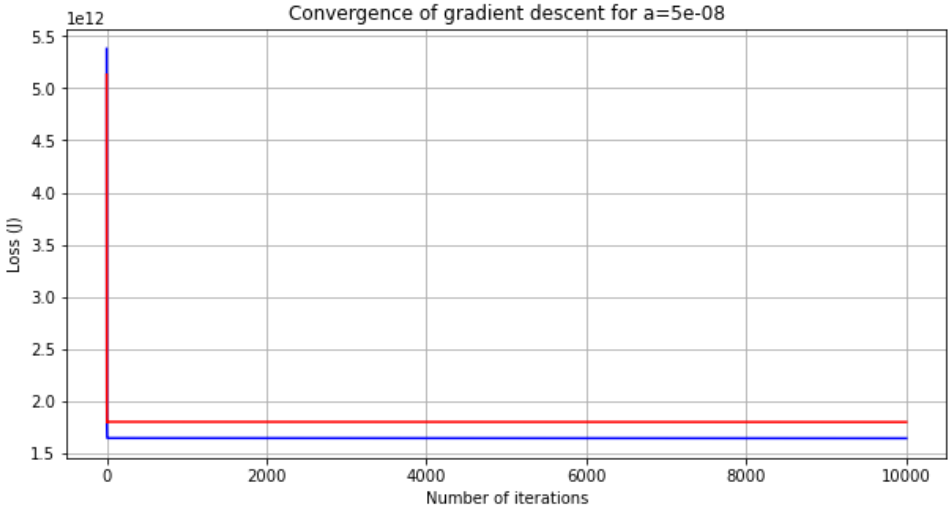
[area, bedrooms, bathrooms, stories, parking]

Identify the best parameters for your linear regression model, based on the above input variables. 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.

No value of the learning rate in the range of 0.1 to 0.01 produced a converging output for the gradient descent algorithm. The learning rate had a maximum value in the order of 10^{-8} in order to achieve a non-divergent loss during learning.

Using a learning rate of $5 \cdot 10^{-8}$, The following results were achieved after 10k iterations of the gradient descent:

```
Converged Training Loss J = 1640371583531.789
Final Validation Loss J = 1796490856056.696
```



These results are not good. The loss is enormous. Worth noting is that it may still be converging, as the graph is so highly scaled in the Y-axis that it is hard to tell, however it was reaching the maximum number of iterations that my computer could safely handle.

The final model parameters produced can be seen below:

```
Theta = [[177.1673769 ]
 [851.43845112]
 [656.80845063]
 [369.30754091]
 [595.56440216]
 [133.30470252]]
```

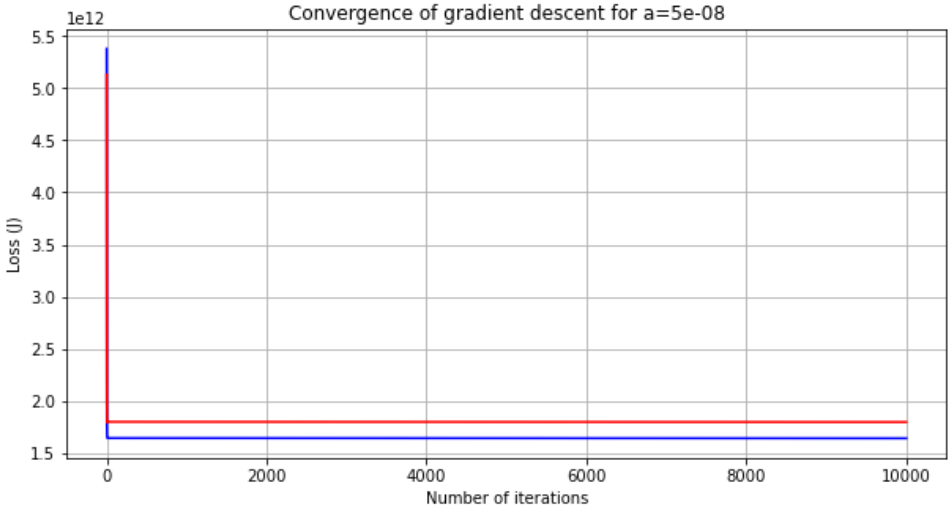
Part 1b)

GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105

Develop a gradient decent training and evaluation code that predicts housing price based on the following input variables:
[Area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hotwaterheating, airconditioning, parking, prefarea]
Identify the best parameters for your linear regression model, based on the above input variables. 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 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.

Similarly to Part1a, no value of the learning rate in the range of 0.1 to 0.01 produced a converging output for the gradient descent algorithm. The learning rate had a maximum value in the order of 10^{-8} in order to achieve a non-divergent loss during learning.
Using a learning rate of $5 \cdot 10^{-8}$, The following results were achieved after 10k iterations of the gradient descent:

```
Converged Training Loss J = 1640219523234.121
Final Validation Loss J = 1796331915056.757
```



These results are not good. The loss is enormous. Worth noting is that it may still be converging, as the graph is so highly scaled in the Y-axis that it is hard to tell, however it was reaching the maximum number of iterations that my computer could safely handle.
The final model parameters produced can be seen below:

GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105

```
Theta = [[177.16196901]
 [851.39254663]
 [656.79149008]
 [369.29942061]
 [595.55147017]
 [133.30172726]
 [157.29239896]
 [ 74.84861923]
 [112.37594565]
 [ 17.07433362]
 [152.94483886]
 [ 97.26423902]]
```

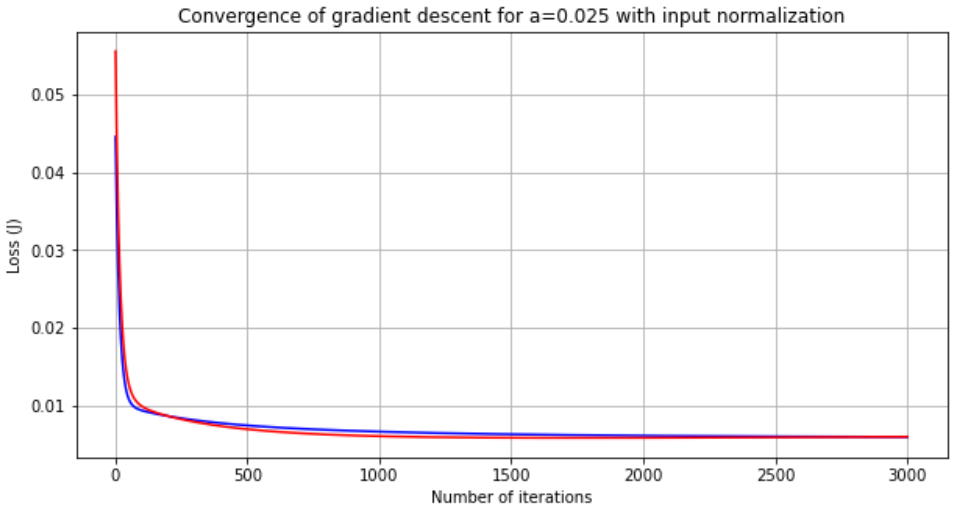
Part 2a)

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.

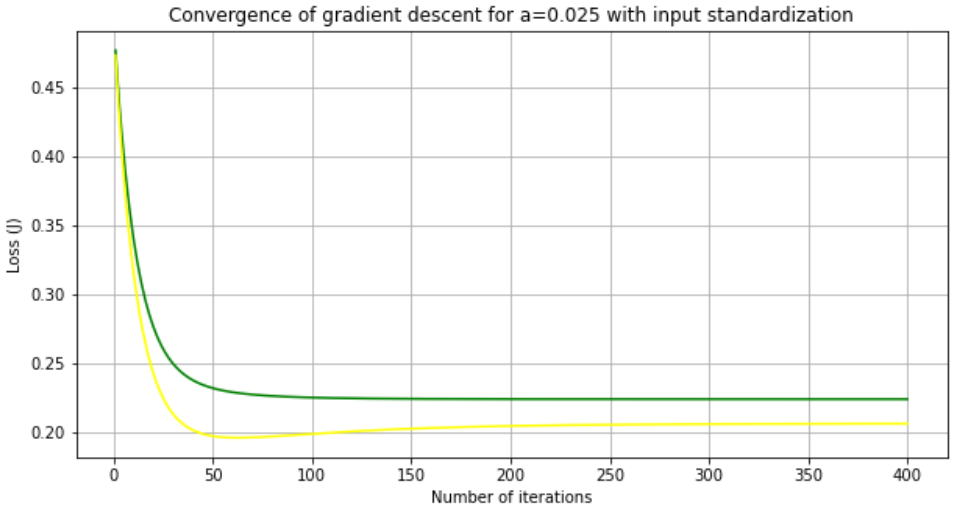
Plot the training and validation losses for both training and validation set based on input standardization and input normalization. Compare your training accuracy between both scaling approaches as well as the baseline training in problem 1 a. Which input scaling achieves the best training? Explain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

Adding the input scaling produced wonderful results. While the results without scaling were basically unusable, these results are quite good. Especially for the min/max normalized set. The loss is very low, and the validation set loss is roughly identical to the training set loss. The standardized set had an odd quirk of the validation set having less loss than the training set: a fortunate coincidence.

```
Converged Training Loss J for normalized set = 0.006
Final Validation Loss J for normalized set = 0.006
Converged Training Loss J for standardized set = 0.224
Final Validation Loss J for standardized set = 0.206
```



GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105



From this data, the Min/Max normalization scaling is the best approach for this dataset because it both has the lowest loss, and the closest match between training and validation loss.
The final model parameters produced by both scaling methods can be seen below:

```
Normalized Theta = [[0.05832319]
 [0.35839946]
 [0.06342379]
 [0.25440021]
 [0.15871885]
 [0.11216531]]
Standardized Theta = [[4.78974658e-17]
 [4.14055189e-01]
 [3.33386959e-02]
 [2.90975856e-01]
 [2.75804041e-01]
 [1.57998917e-01]]
```

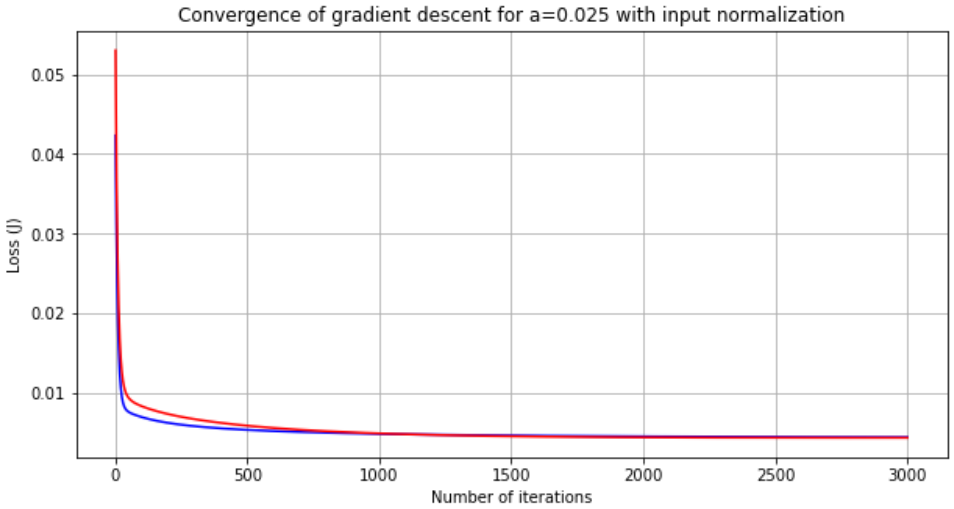
Part 2b)

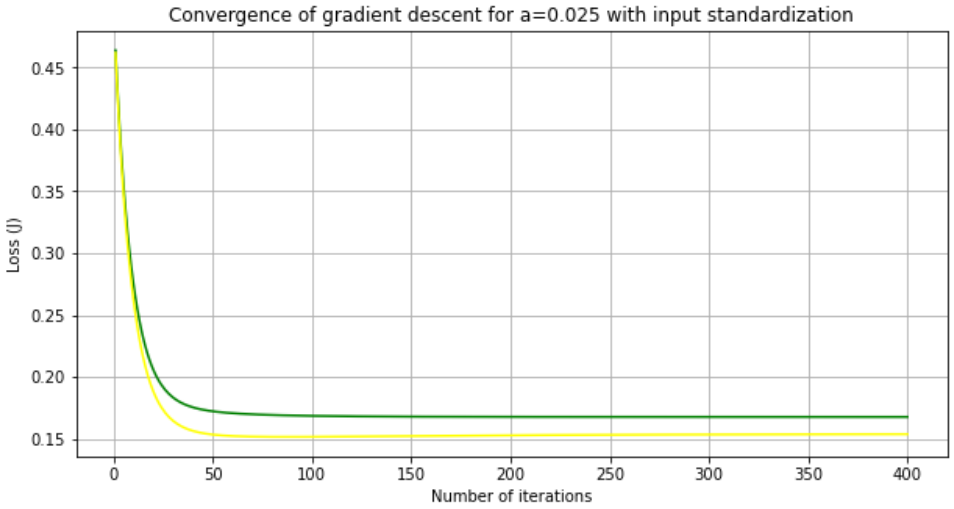
Repeat problem 1 b, this time with input normalization and input standardization as part of your pre-processing logic. You need to perform two separate trainings for standardization and normalization.
Plot the training and validation losses for both training and validation set based on input standardization and input normalization. Compare your training accuracy between both scaling approaches as well as the baseline training in problem 1 b. Which input scaling achieves the best training? Explain your results.

Like Part 2a, the input scaling had a tremendous effect on the efficacy of the training. The results of the training can be seen below.

```
Converged Training Loss J for normalized set = 0.004
Final Validation Loss J for normalized set = 0.005
Converged Training Loss J for standardized set = 0.168
Final Validation Loss J for standardized set = 0.154
```


GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105





The data shows that, once again, the min/max normalization scaling produces the best loss characteristics for this dataset and produces the lowest difference between training loss and validation loss.

The final model parameters produced by both scaling methods can be seen below:

GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105

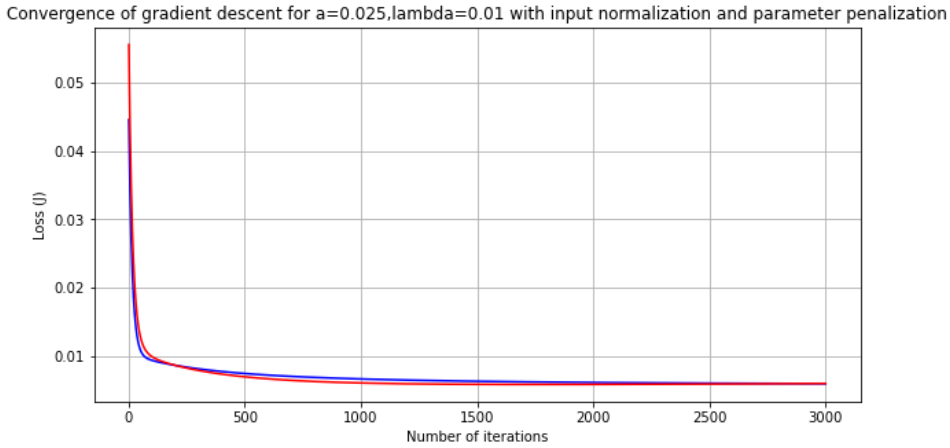
```
Normalized Theta = [[0.00571338]
 [0.25459249]
 [0.04973833]
 [0.21951471]
 [0.12779848]
 [0.08839834]
 [0.0458586 ]
 [0.02674523]
 [0.03364802]
 [0.05127891]
 [0.08031061]
 [0.06858081]]
Standardized Theta = [[5.95344365e-17]
 [3.07148779e-01]
 [2.59633038e-02]
 [2.55753688e-01]
 [2.25697778e-01]
 [1.26377042e-01]
 [8.98721270e-02]
 [5.39972666e-02]
 [9.99097249e-02]
 [6.72282523e-02]
 [2.19594016e-01]
 [1.67027160e-01]]
```

Part 3a)

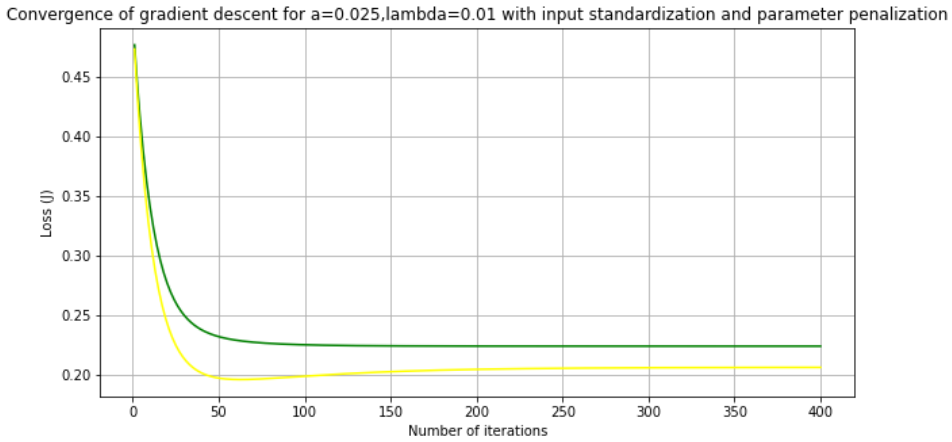
Repeat problem 2 a, this time by adding parameters penalty to your loss function. Note that in this case, you need to modify the gradient decent logic for your training set, but you don't need to change your loss for the evaluation set.

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.

```
Converged Training Loss J for normalized set = 0.006
Final Validation Loss J for normalized set = 0.006
Converged Training Loss J for standardized set = 0.224
Final Validation Loss J for standardized set = 0.206
```



GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105



Once again, it seems that min/max normalization produces the best training results, and interestingly, the parameter penalization has resulted in an equivalent final loss value. This implies that the size of the parameters was not contributing to any training-validation error issues in Part 2a.

The final model parameters produced by both scaling methods can be seen below:

```
Normalized Theta = [[0.05837351]
 [0.35820404]
 [0.063445  ]
 [0.25428011]
 [0.158702  ]
 [0.11217477]]
Standardized Theta = [[5.03801906e-17]
 [4.14046633e-01]
 [3.33433655e-02]
 [2.90971291e-01]
 [2.75797770e-01]
 [1.57998359e-01]]
```

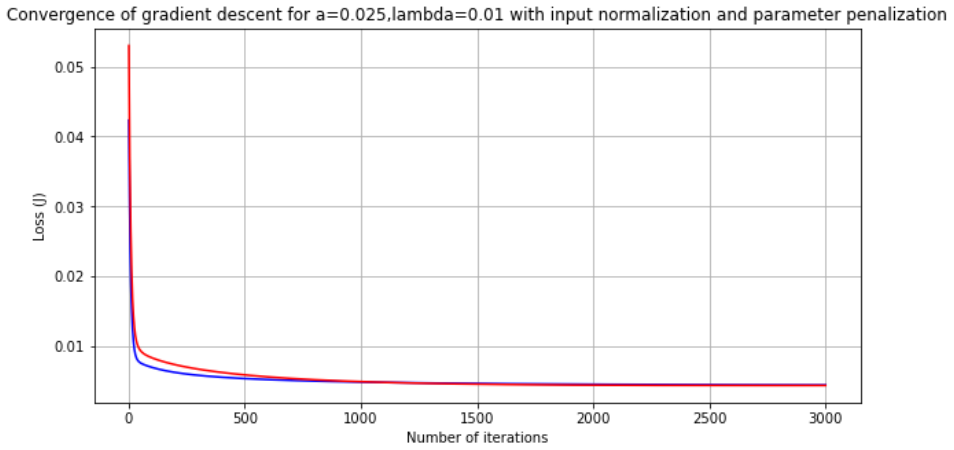
Part 3b)

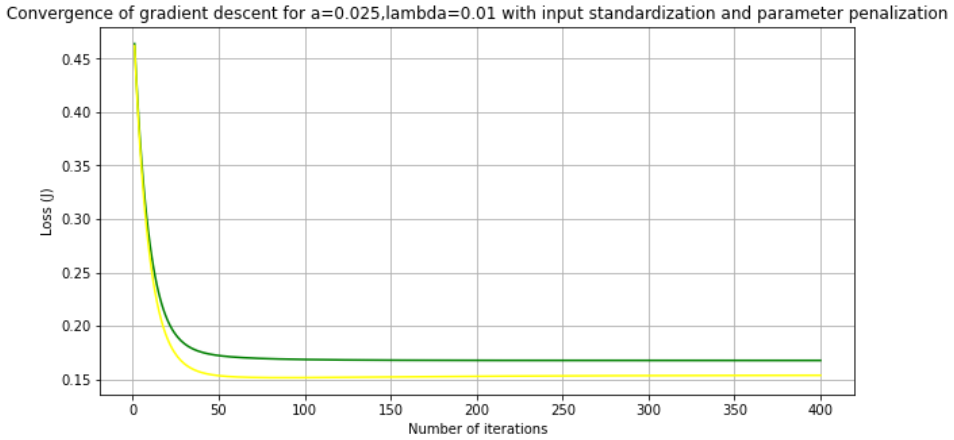
Repeat problem 2 b, this time by adding parameters penalty to your loss function. Note that in this case, you need to modify the gradient decent logic for your training set, but you don't need to change your loss for the evaluation set.

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.

```
Converged Training Loss J for normalized set = 0.004
Final Validation Loss J for normalized set = 0.004
Converged Training Loss J for standardized set = 0.168
Final Validation Loss J for standardized set = 0.154
```


GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105





Here we see that the min/max normalization once again produces better loss and training-validation error characteristics than the standardization. Additionally, the parameter penalization again seems not to have produced a significant effect on the end-loss of the training as compared to the non-penalized counterpart.
The final model parameters produced by both scaling methods can be seen below:

GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105

```
Normalized Theta = [[0.00574689]
 [0.25444651]
 [0.04974596]
 [0.21940653]
 [0.1277795 ]
 [0.0883993 ]
 [0.04586924]
 [0.02675469]
 [0.03364668]
 [0.05125839]
 [0.08031928]
 [0.06858441]]
Standardized Theta = [[6.31488829e-17]
 [3.07143161e-01]
 [2.59676880e-02]
 [2.55749705e-01]
 [2.25692689e-01]
 [1.26376641e-01]
 [8.98731313e-02]
 [5.39988944e-02]
 [9.99070723e-02]
 [6.72264200e-02]
 [2.19591420e-01]
 [1.67025372e-01]]
```


GitHub Code Repository: https://github.com/pharmon0/Harmon_ECGR5105