Implementation Assignment1

CS534-Fall2018

Chi Wen
Yi-Jung Chiang
Han-Yu Wu

Part 0 (10 pts) : **Preprocessing and simple analysis.** Perform the following preprocessing of the your data.

(a) Remove the ID feature. Why do you think it is a bad idea to use this feature in learning?
Because ID has no impact on the price.

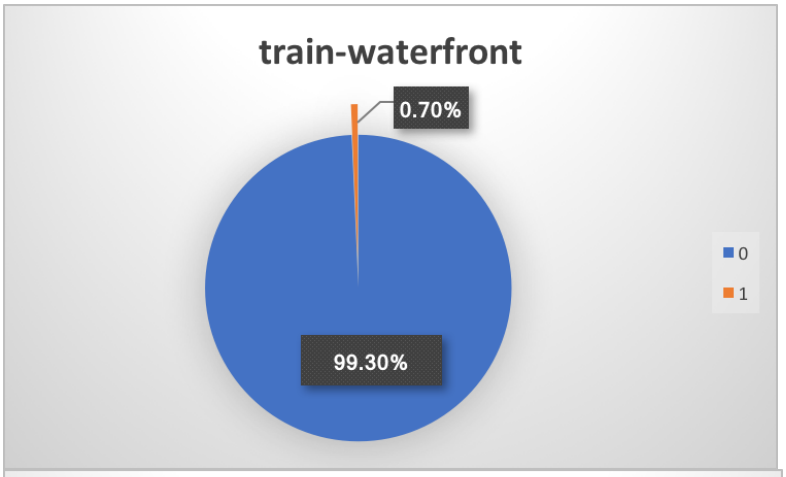
(b) Split the date feature into three separate numerical features: month, day, and year. Can you think of better ways of using this date feature?
Due to splitting date feature into three separate numerical features could increase the complexity of training the model, we could calculate date feature into a single unit feature that can present a more meaningful data for us to use. For example, we can calculate how many days between the date and the latest date we know from the dataset, and after normalizing, those data can be used as one feature.

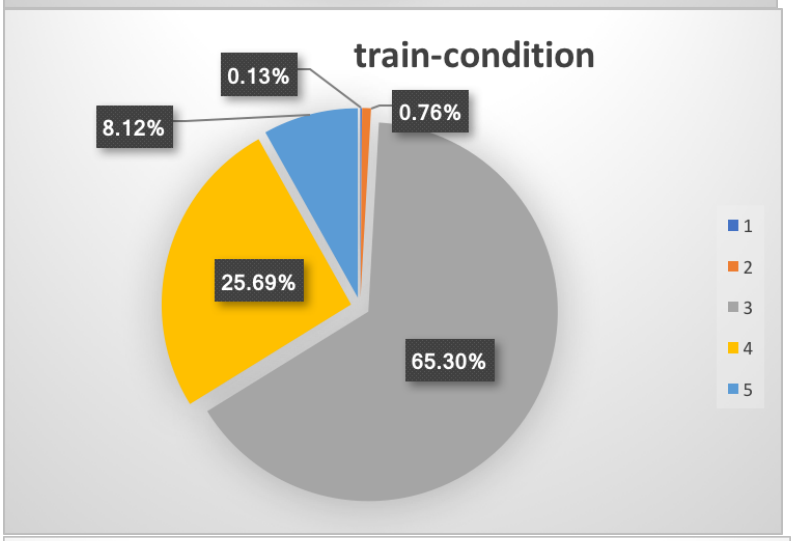
(c) Build a table that reports the statistics for each feature. For numerical features, please report the mean, the standard deviation, and the range. For categorical features such as waterfront, grade, condition (the later two are ordinal), please report the percentage of examples for each category.

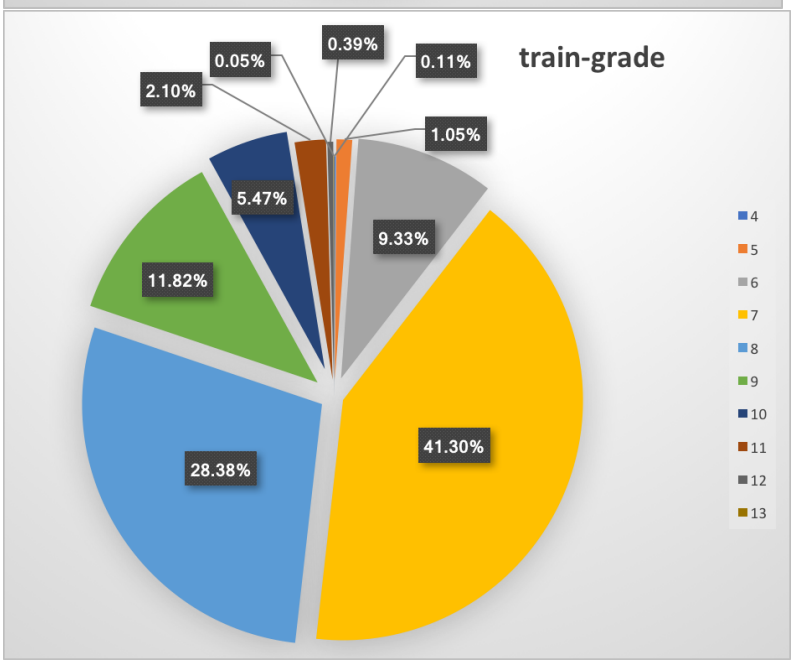
We take data from PA1_train.csv for example.

feature	max_value	min_value	mean_value	std_value
['dummy']	1	1	1	0
['id']	9.9E+09	1000102	4.587E+09	2.884E+09
['bedrooms']	33	1	3.3752	0.9431993
['bathrooms'	7.75	0.5	2.118875	0.7650899
['sqft_living']	9890	370	2080.2232	911.28879
['sqft_lot']	1651359	572	15089.201	41201.835
['floors']	3.5	1	1.5037	0.5426199
['view']	4	0	0.2294	0.7558939
['sqft_above'	8860	370	1793.0993	830.82389
['sqft_basem	2720	0	287.1239	434.98351
['yr_built']	2015	1900	1971.1249	29.47912
['yr_renovate	2015	0	81.2267	394.36008
['zipcode']	98199	98001	98078.293	53.515715
['lat']	47.7776	47.1559	47.559814	0.1386437
['long']	-121.319	-122.514	-122.21329	0.1413976
['sqft_living1	6110	460	1994.3261	691.86571
['sqft_lot15']	871200	660	12746.323	28239.831
['price']	68.9	0.82	5.3852968	3.5737224

For categorical features such as waterfront, grade, condition.







(d) Based on the meaning of the features as well as the statistics, which set of features do you expect to be useful for this task? Why?

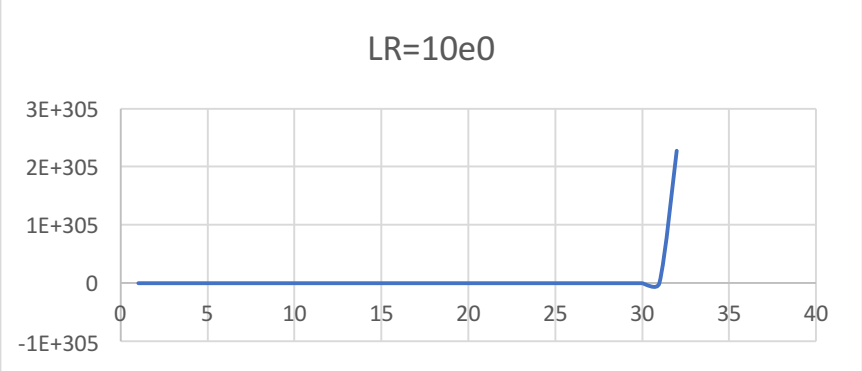
For the graph of waterfront, grade and condition, waterfront is the one that is unlikely to be helpful for this task. It is because there are only 0.76% of houses that has the waterfront feature. As for grade and condition, since they are both features that consider all aspects of a house, they should be more useful for us to predict the price.

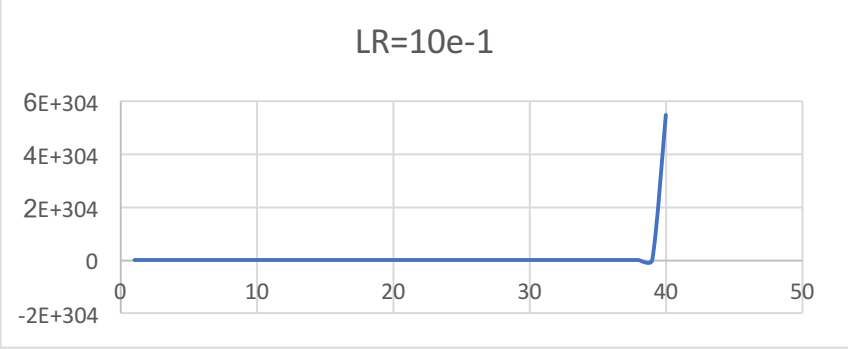
(e) Normalize all features to the range between 0 and 1 using the training data. Note that when you apply the learned model from the normalized data to test data, you should make sure that you are using the same normalizing procedure as used in training.

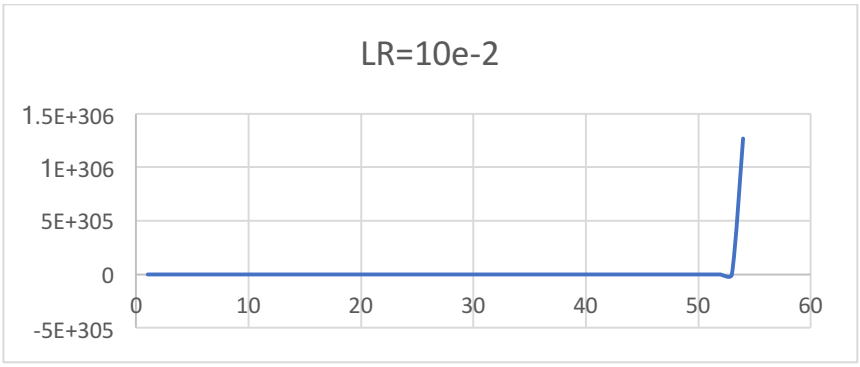
Part 1 (30 pts). Explore different learning rate for batch gradient descent. For this part, you will work with the preprocessed and normalized data and fix λ to 0 and consider at least the following values for the learning rate: 10^{0}; 10^{-1}; 10^{-2}; 10^{-3}; 10^{-4}; 10^{-5}; 10^{-6}; 10^{-7}.

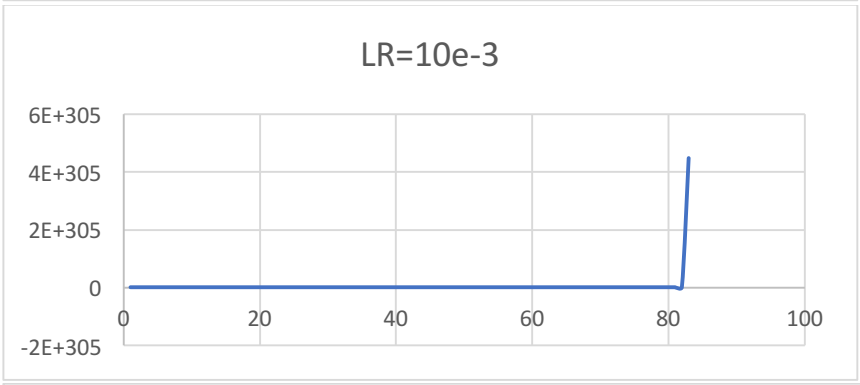
(a) Which learning rate or learning rates did you observe to be good for this particular dataset? What learning rates make the gradient decent explode? Report your observations together with some example curves showing the training SSE as a function of training iterations and its convergence or non-convergence behaviors.

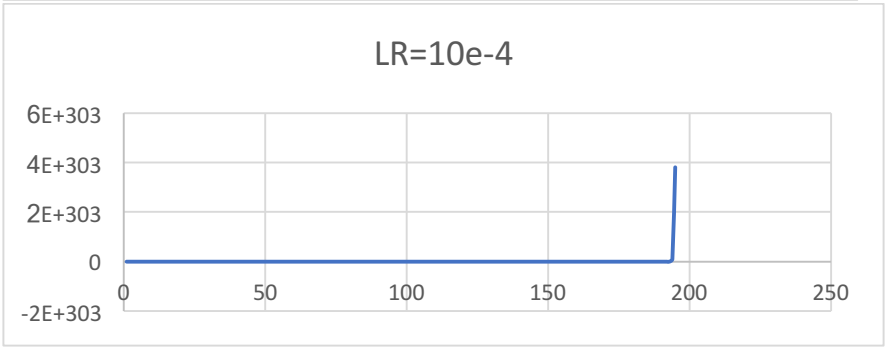
According to our observation, we think the learning rate= 10^{-5} is the best value for this particular dataset. Because the learning rate larger than this value (10^{-5}) would cause the gradient descent to explode. However, smaller than this the training process would take too long to process. We also compared the three weight values (w) we got from 10e-5, 10e-6 and 10e-7. When we applied these to dev.csv data, we found that the sse are no big differences. For the no convergence part, we can see that the larger learning rate more quickly explode. For convergence part, the smaller learning rate needs more iterations to the limit value. In following pictures x-axis is number of iterations and y-axis is sse value.

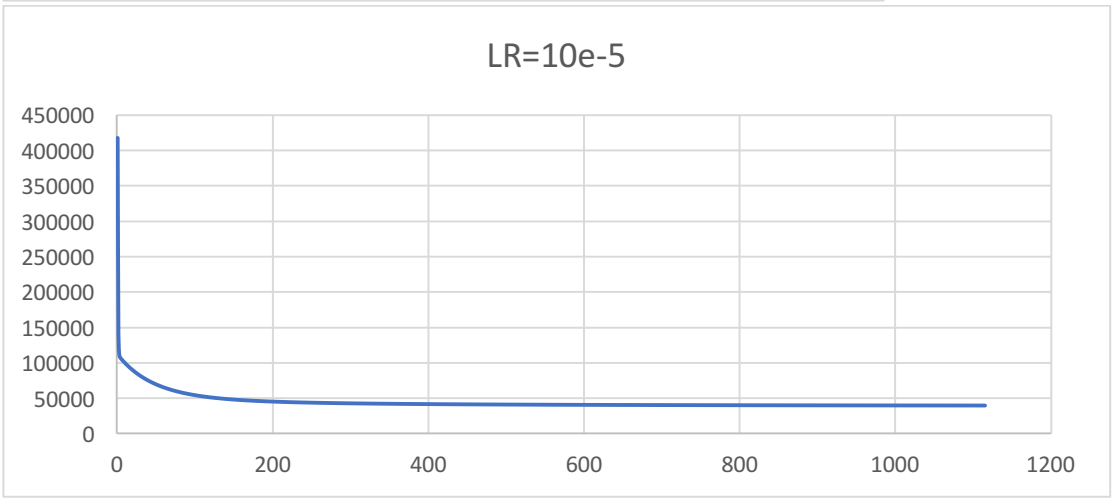


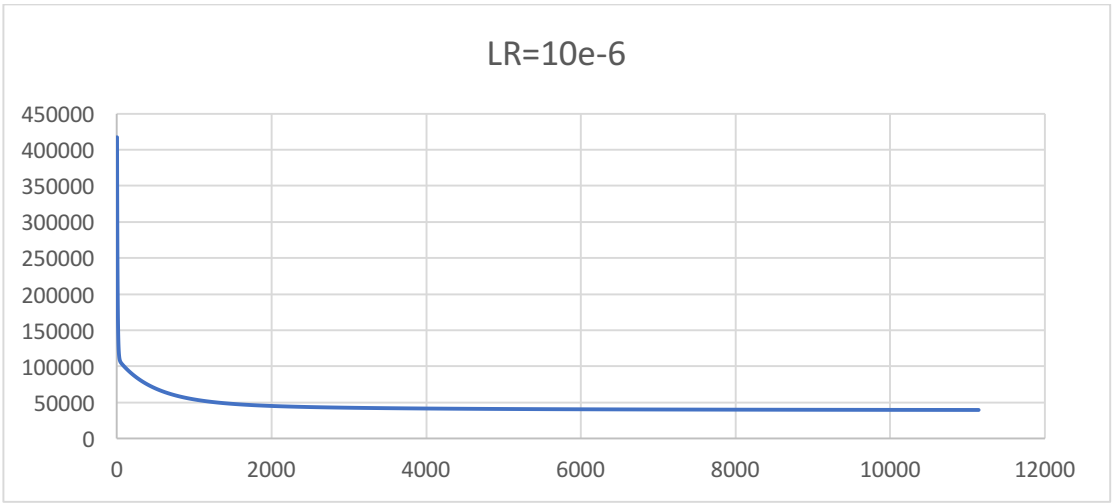


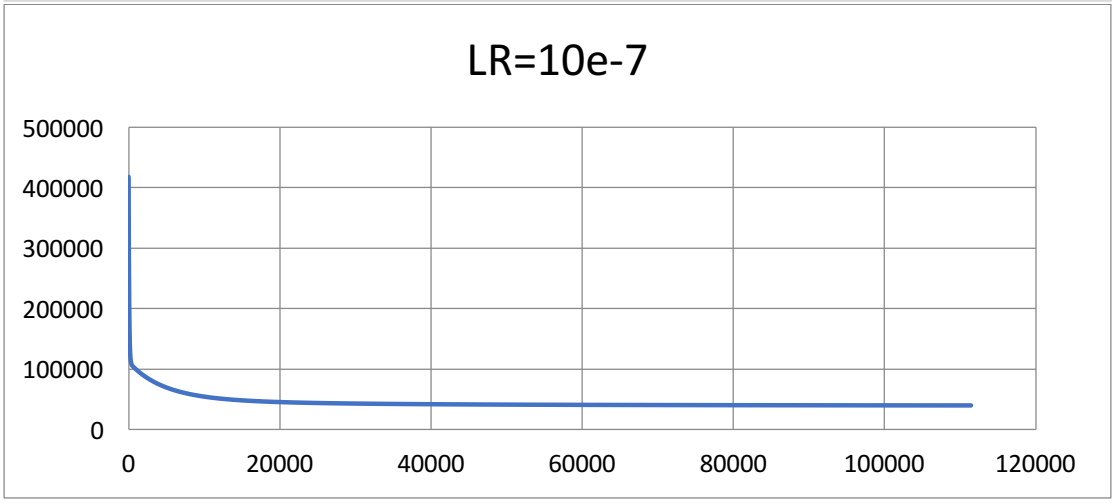

LR=10e-2
1.5E+306
1E+306
5E+305
0
-5E+305
0
10
20
30
40
50
60


LR=10e-3
6E+305
4E+305
2E+305
0
-2E+305
0
20
40
60
80
100


LR=10e-4
6E+303
4E+303
2E+303
0
-2E+303
0
50
100
150
200
250

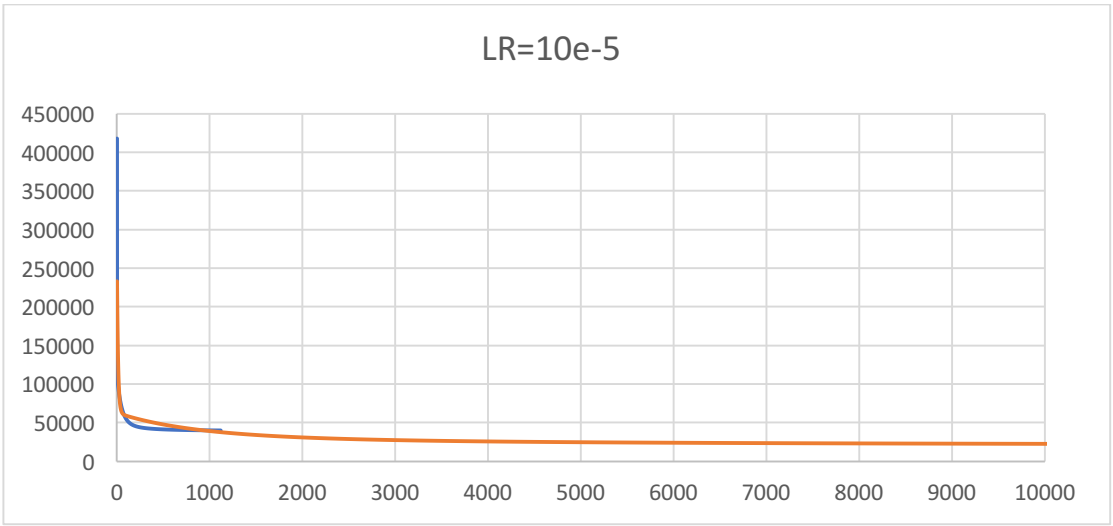

LR=10e-5
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
200
400
600
800
1000
1200

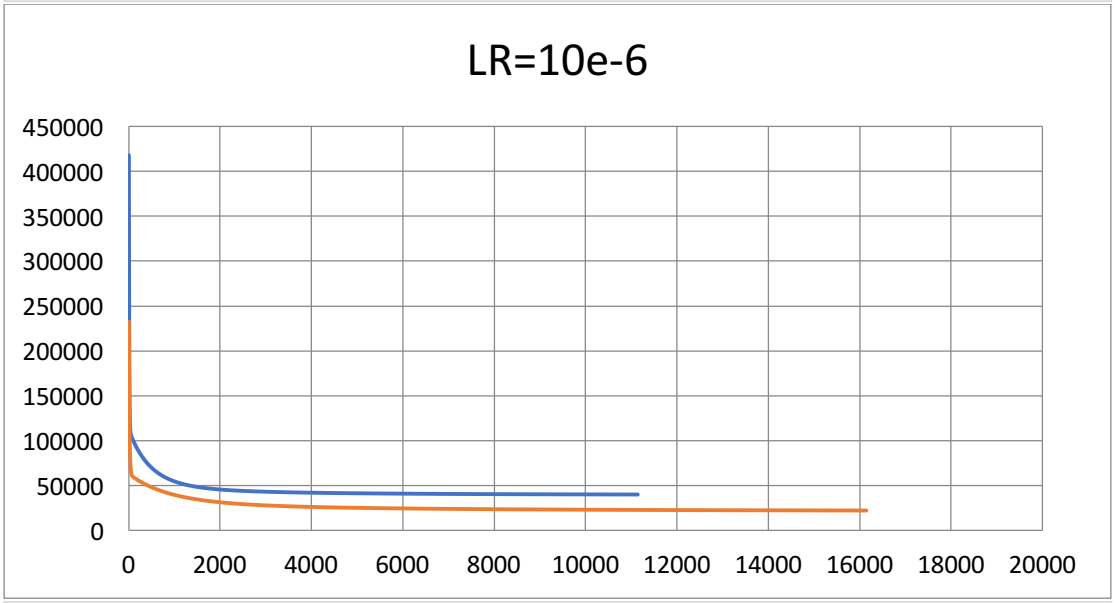


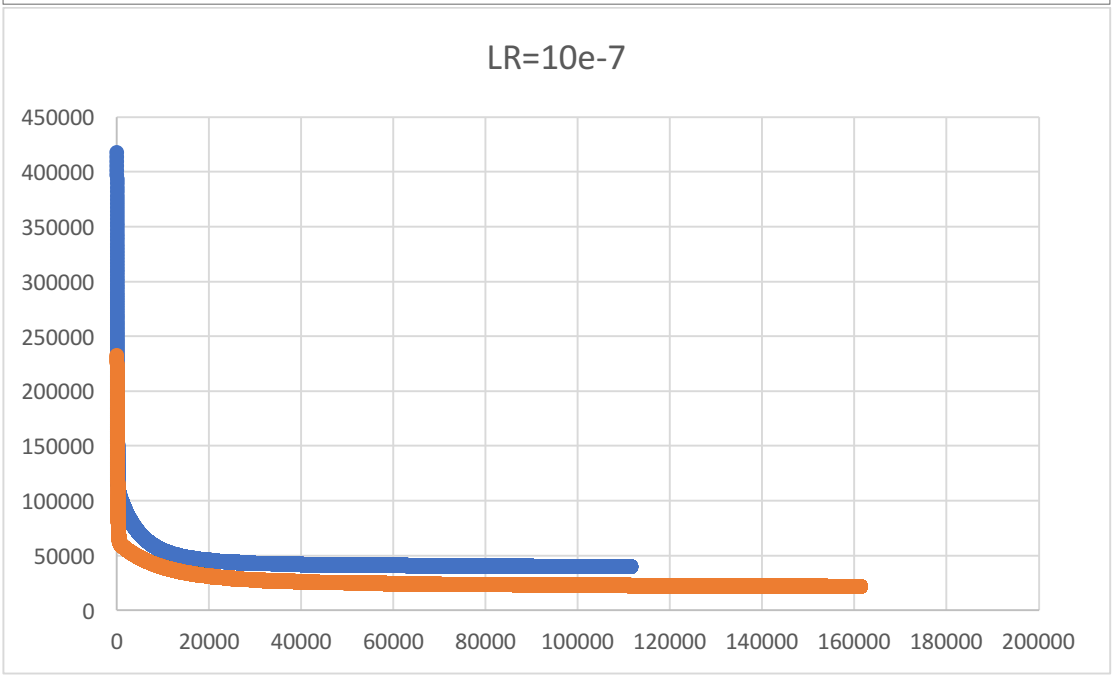


(b) For each learning rate worked for you, Report the SSE on the training data and the validation data respectively and the number of iterations needed to achieve the convergence condition for training. What do you observe?

In comparison of training dataset and validation dataset, we observed the obvious phenomenon that during learning process the validation dataset needs more iterations to hit the convergence value. Moreover, the smaller learning rate this phenomenon is more conspicuous. We think the reason why is mainly about the size of dataset. In each iteration, the training included more data and that means we have better gradient to update the next weight value. The larger learning rate also enlarge this learning process. In following pictures x-axis is number of iterations and y-axis is sse value. The blue lines represent train data and orange lines represent validation data.


LR=10e-5
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


LR=10e-6
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000


LR=10e-7
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000

(c) Use the validation data to pick the best converged solution, and report the learned weights for each feature. Which feature are the most important in deciding the house prices according to the learned weights? Compare them to your pre-analysis results (Part 0 (d)).

The table shows the weight we picked up.

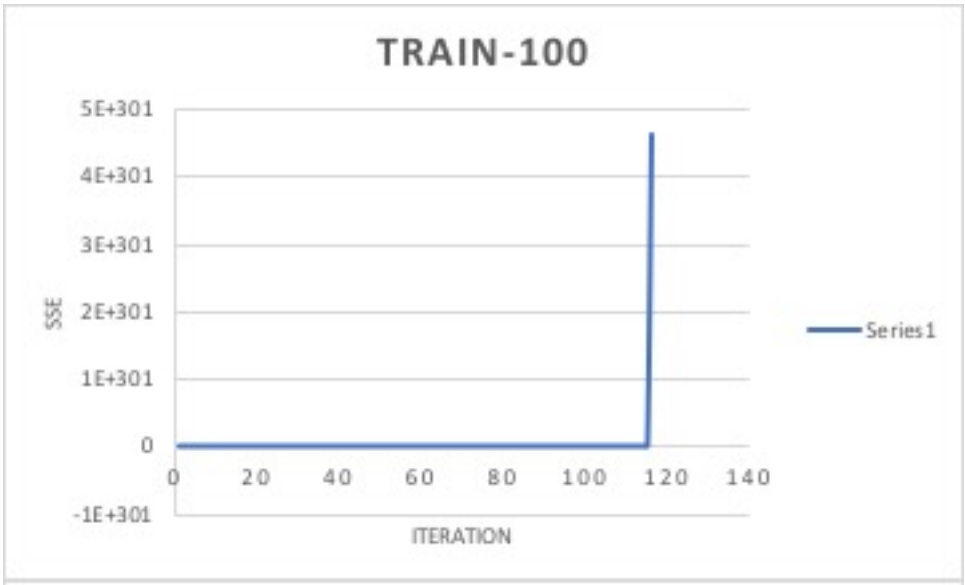
features	weight
dummy	−2.24365372
Year	0.33132115
month	0.10492302
day	−0.19633755
Bedrooms	−0.40596083
Bathrooms	3.206994
Sqft_living	5.45934202
Sqft_lot	0.20121181
floors	0.63545153
watrerfront	3.57623811
view	2.66348025
condition	1.10335433
grade	8.02629767
Sqft_above	5.72299752
Sqft_basement	1.24437024
Yr_built	−2.96677281
Yr_renovated	0.31218811
Zipcode	−0.92216952
Lat	3.73918201
Long	−2.45400468
Sqft_living15	3.44147831
Sqft_lot15	−0.01564212

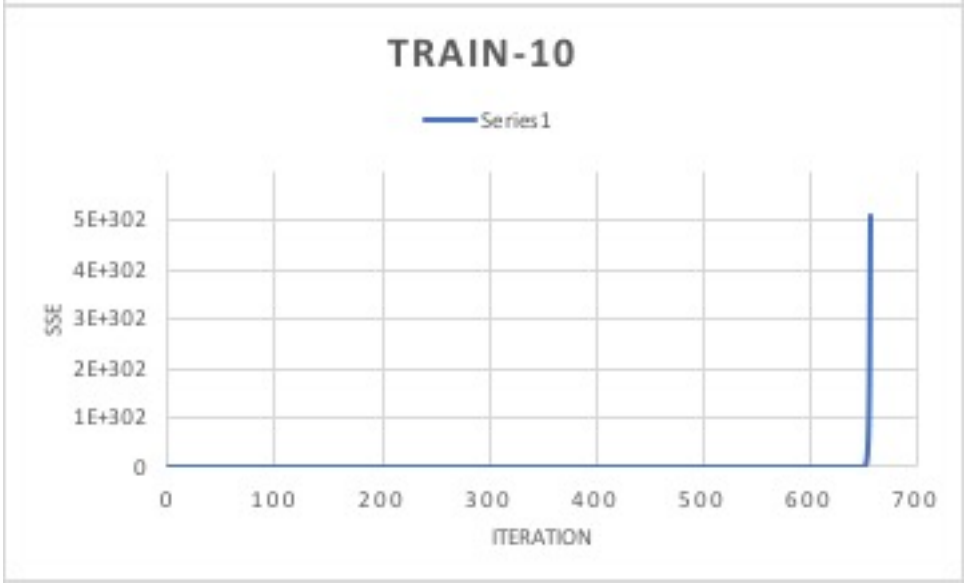
We think the most important feature is the grade because the weight value of this feature is the largest. This is the same as we think in part0 (d). We think the we are building up the overall model to predict the price. This model is similar with the grade feature, which covers more aspects, so it becomes more important to the indicator.

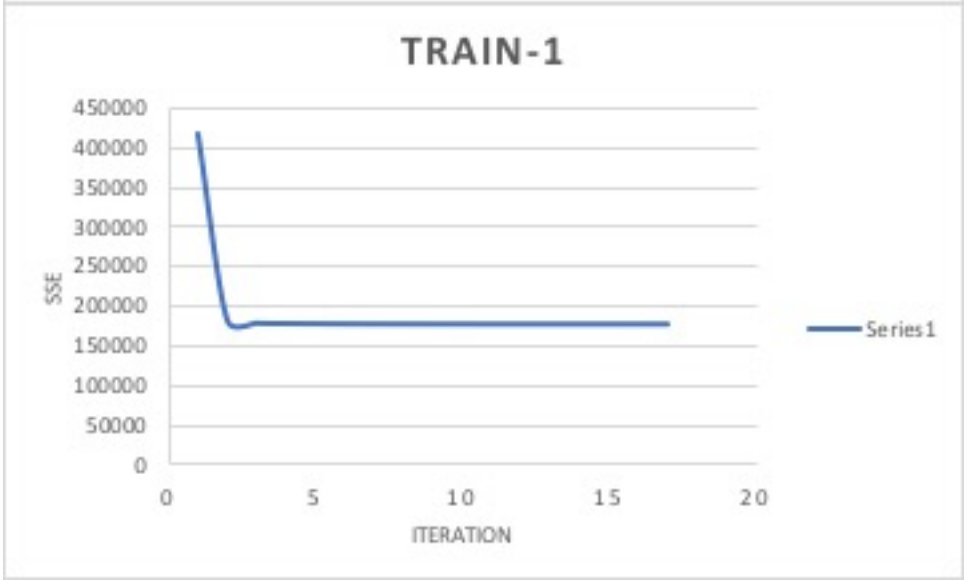
Part2 (30 pts). Experiments with different λ values. For this part, you will test the effect of the regularization parameter on your linear regressor. Please exclude the bias term from regularization. It is often the case that we don't really what the right λ value should be and we will need to consider a range of different λ values. For this project, consider at least the following values for λ: 0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100. Feel free to explore other choices of λ using a broader or finer search grid. Report the SSE on the training data and the validation data respectively for each value of λ. Report the weights you learned for different values of λ. What do you observe? Your discussion of the results should clearly answer the following questions:

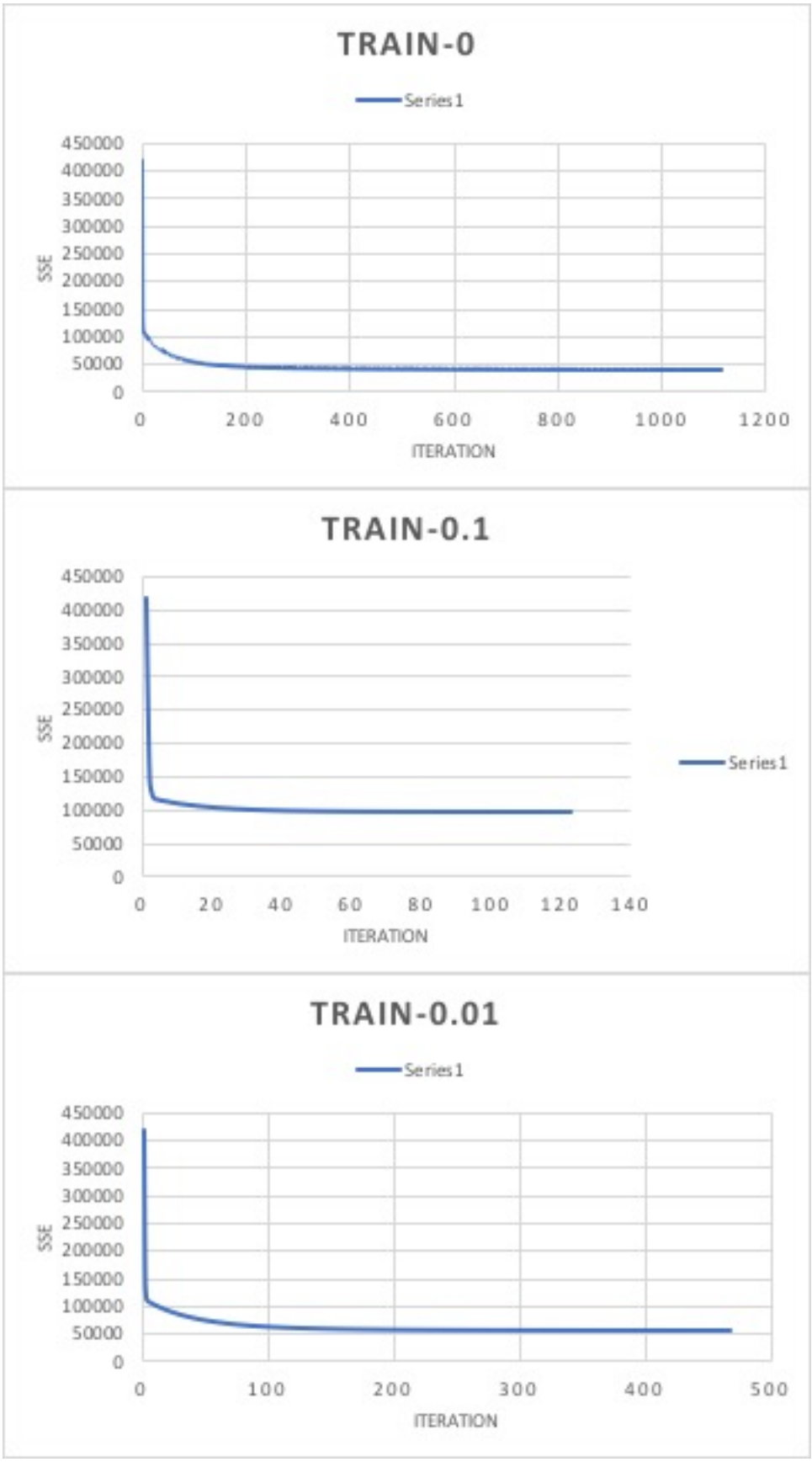
(a) What trend do you observe from the training SSE as we change λ value?
When λ value is bigger than 1, the result is infinity. Also, when λ value is 1, the speed of convergence is very fast. However, when λ value is smaller than 1, the smaller λ

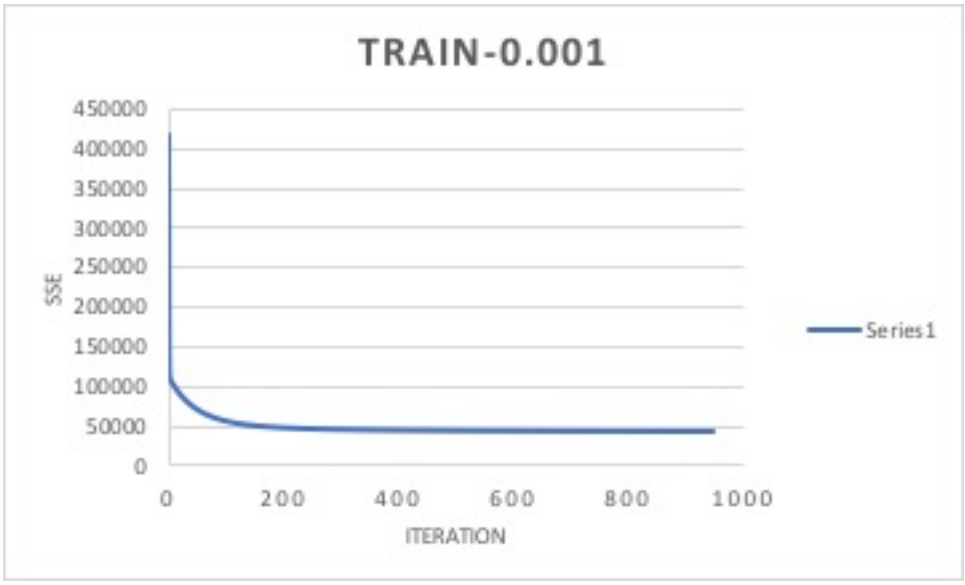
value is, the more iterations that it need to achieve convergence. The reason why we get this result is that 2* λ*w. Therefore, when λ is so small, the influence of λ*w is also small.



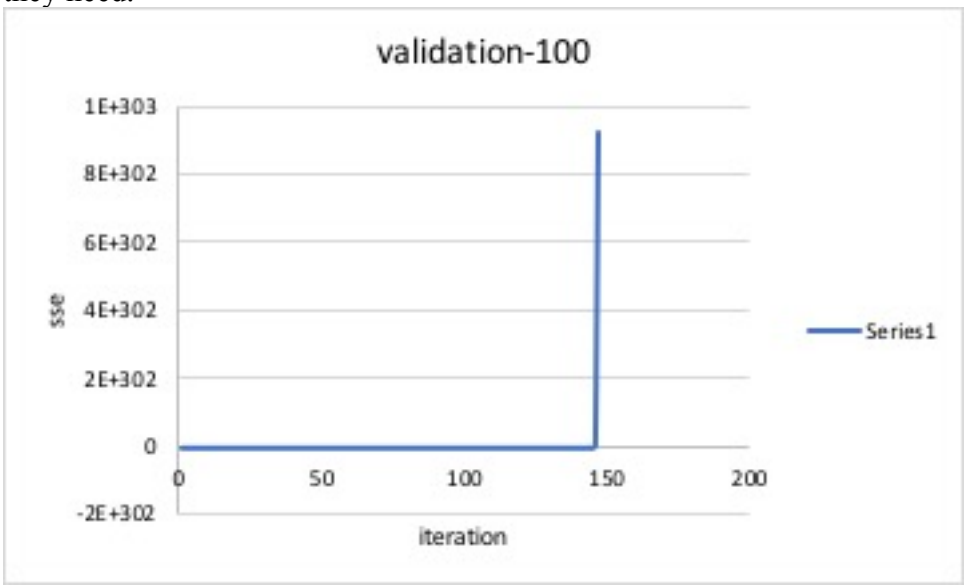





TRAIN-0
Series1
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
SSE
0
200
400
600
800
1000
1200
ITERATION
TRAIN-0.1
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
SSE
0
20
40
60
80
100
120
140
ITERATION
Series1
TRAIN-0.01
Series1
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
SSE
0
100
200
300
400
500
ITERATION



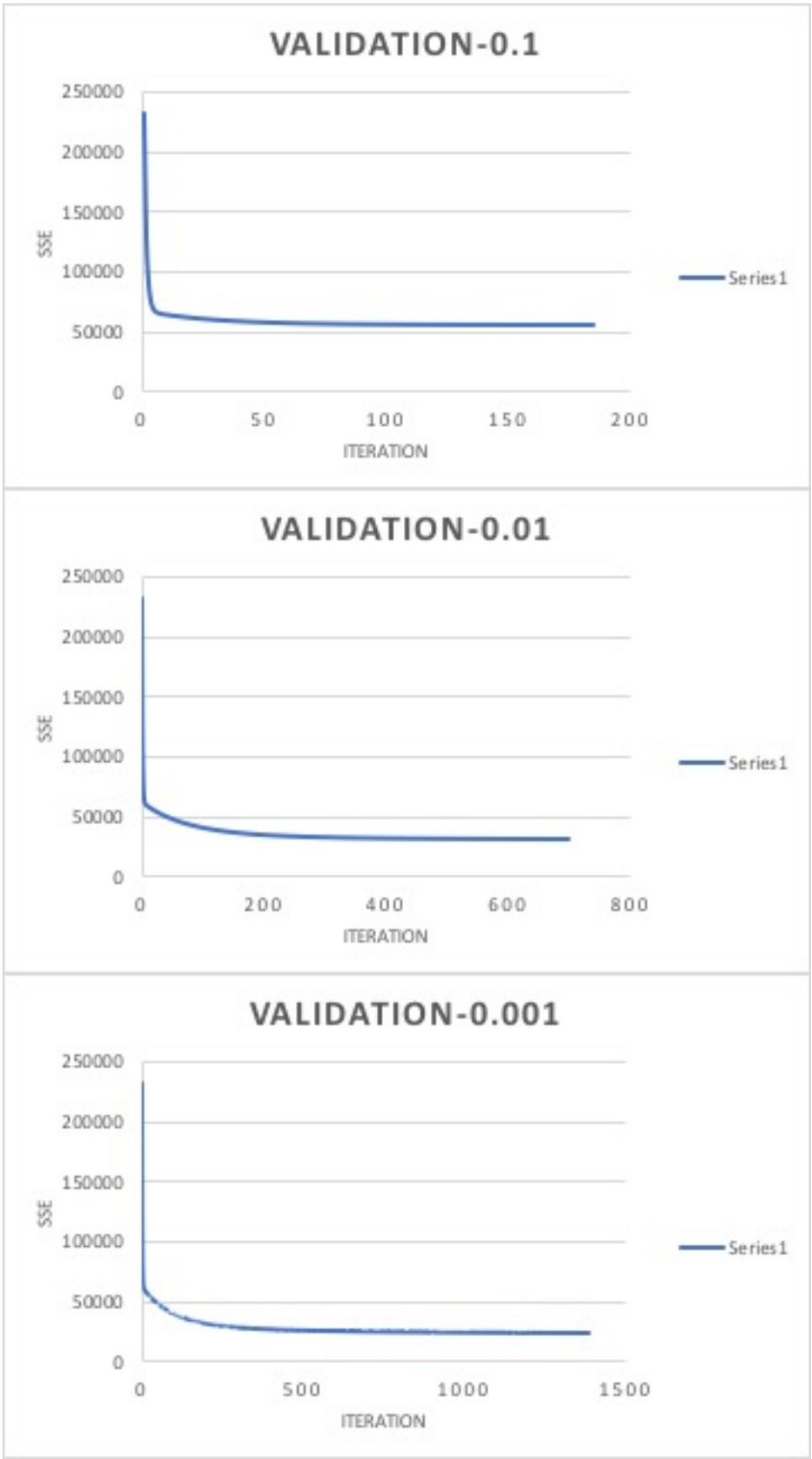
(b) What tread do you observe from the validation SSE?
According to pictures, we observe that when the smaller λ it is, the smaller SSE value it is when it achieves convergence. Also, the smaller λ it is, the less times of iterations they need.




VALIDATION-10
250000
200000
150000
100000
50000
0
SSE
0
2
4
6
8
10
12
ITERATION
Series1


validation-1
250000
200000
150000
100000
50000
0
sse
0
5
10
15
20
25
30
iteration
Series1


validation-0
250000
200000
150000
100000
50000
0
sse
0
500
1000
1500
2000
iteration
Series1


VALIDATION-0.1
250000
200000
150000
100000
50000
0
SSE
0
50
100
150
200
ITERATION
Series1
VALIDATION-0.01
250000
200000
150000
100000
50000
0
SSE
0
200
400
600
800
ITERATION
Series1
VALIDATION-0.001
250000
200000
150000
100000
50000
0
SSE
0
500
1000
1500
ITERATION
Series1

(c) Provide an explanation for the observed behaviors.
Both training data and validation data get infinity result when λ is 100. However, training data needs more iterations than validation data until it achieves convergence.

(d) What features get turned off for λ = 10, 10^{-2} and 0?

	dummy	date(year)	date(month	date(day)	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basem	yr_built	yr_renovate	zipcode	lat	long	sqft_living1	sqft_lot15		
validation(λ=10)	0.395189	0.132403	0.195698	0.189926	0.128772	0.09323	0.067304	0.006708	0.095143	0.011589	0.046664	0.238552	0.210043	0.07998	0.033949	0.248031	0.028517	0.146071	0.280375	0.099099	0.117619	0.012855	0.006708	sqft_lot
validation(λ=0)	-2.084531	0.16023	-0.303935	-0.110281	-0.808863	4.260712	5.318908	0.135851	0.541937	4.078475	2.187761	0.766338	8.091189	6.061896	3.167083	-2.911081	0.571164	-1.107801	3.784589	-2.23817	4.565524	-0.326367	-2.911081	yr_built
validation(λ=0.01)	-0.80577	-0.015621	-0.523033	-0.196216	0.934474	3.031633	3.257643	0.178145	1.149277	2.044534	2.863791	0.550799	4.960355	3.598084	2.154209	-1.730516	0.990098	-0.892436	3.444689	-0.899937	3.933977	0.102726	-1.730516	yr_built

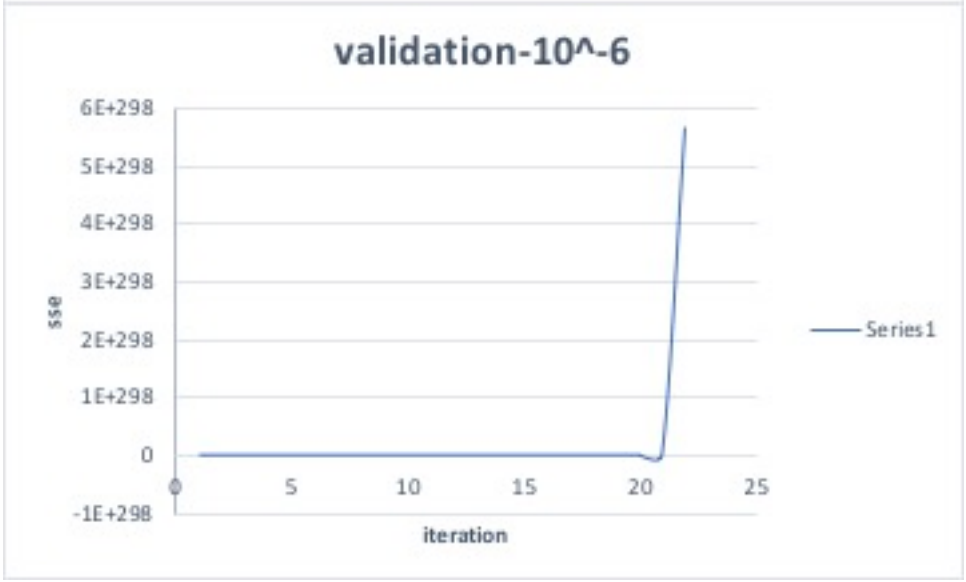
According to this picture, we compare different weights of features for λ=10, 10^{-2}, and 0. We observe that yr_built feature is the smallest weight for λ=0 and 10^{-2}. Thus, we strongly believe that we can turned off yr_built feature due to smallest weight.

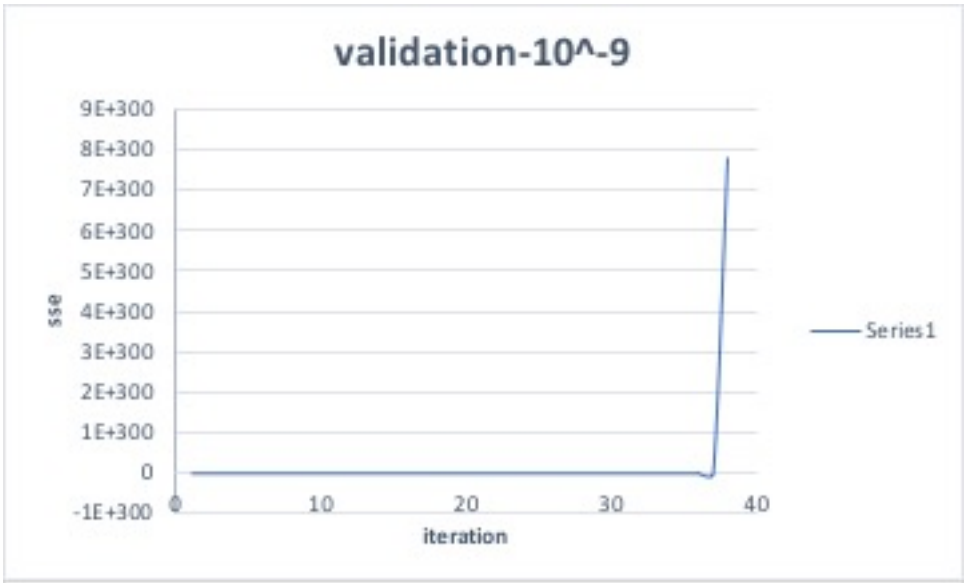
Part 3 (10 pts). Training with non-normalized data. Use the preprocessed data but skip the normalization. Consider at least the following values for learning rate: 1, 0, 10^{-3}, 10^{-6}, 10^{-9}, 10^{-15}. For each value, train up to 10000 iterations (Fix the number of iterations for this part). If training is clearly diverging, you can terminate early. Plot the training SSE and validation SSE respectively as a function of the number of iterations. What do you observe? Specify the learning rate value (if any) that prevents the gradient descent from exploding? Compare between using the normalized and the non-normalized versions of the data. Which one is easier to train and why?

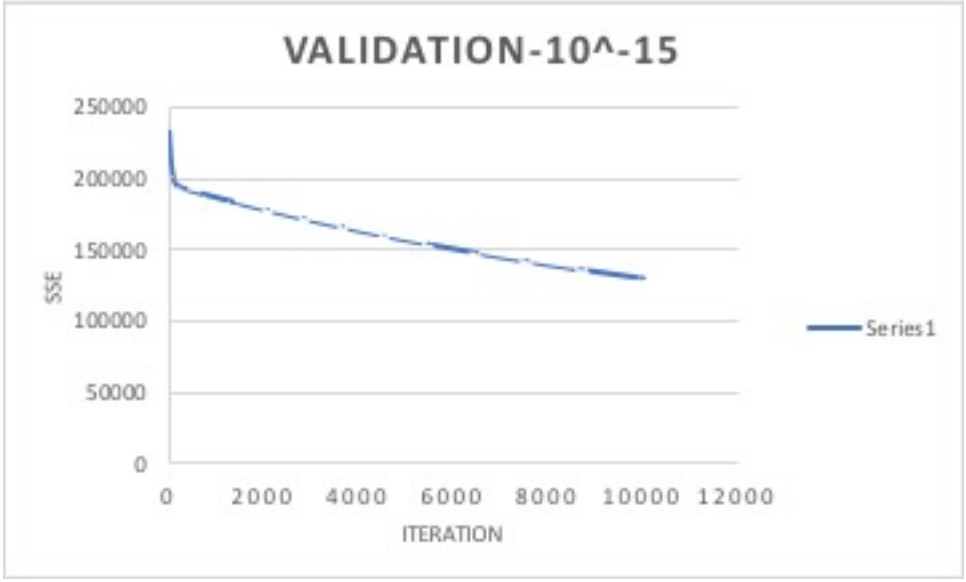
Validation:


validation-1
5E+290
4E+290
3E+290
2E+290
1E+290
0
-1E+290
sse
2
4
6
8
10
12
14
iteration
Series1

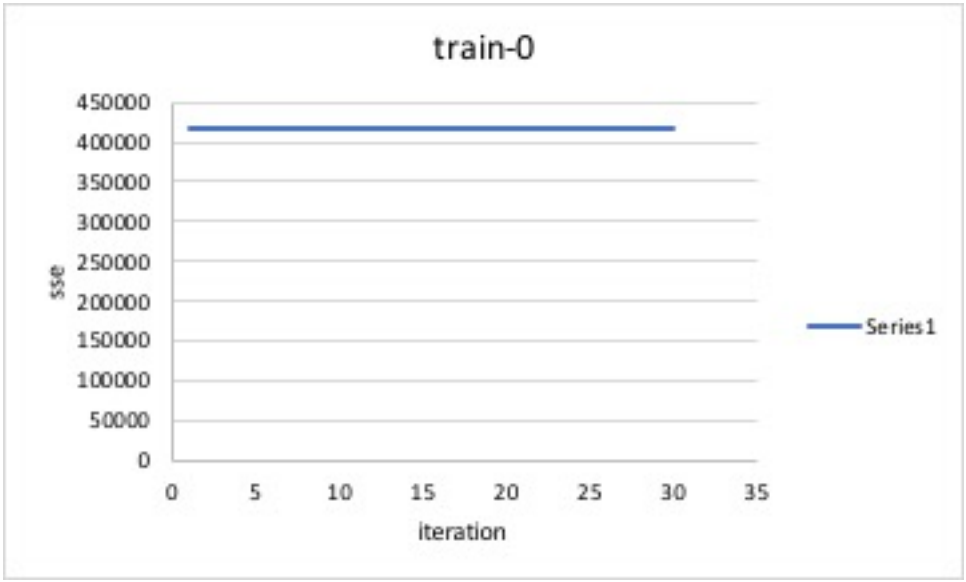

validation-10^-3
6E+304
5E+304
4E+304
3E+304
2E+304
1E+304
0
-1E+304
sse
5
10
15
20
iteration
Series1

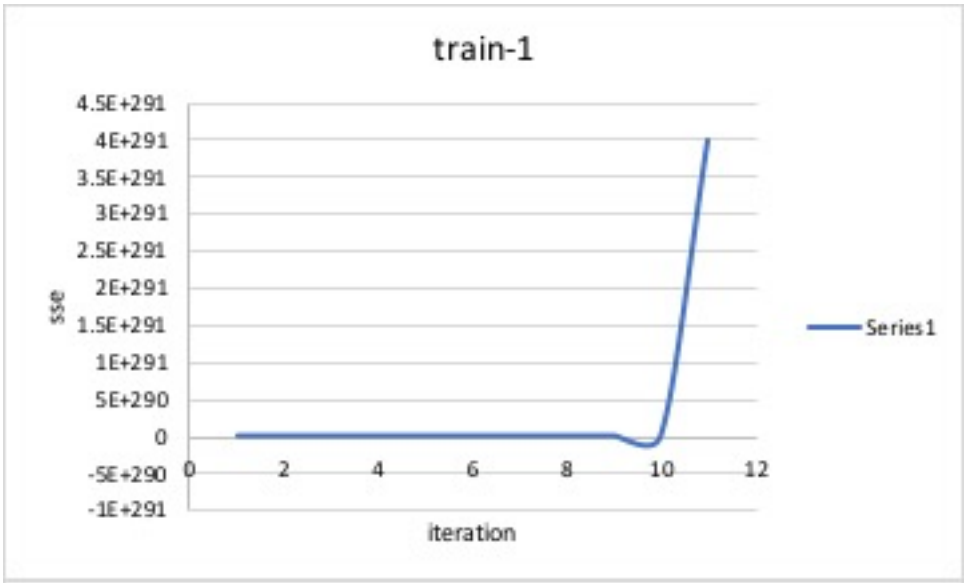

validation-10^-6
6E+298
5E+298
4E+298
3E+298
2E+298
1E+298
0
-1E+298
sse
5
10
15
20
25
iteration
Series1

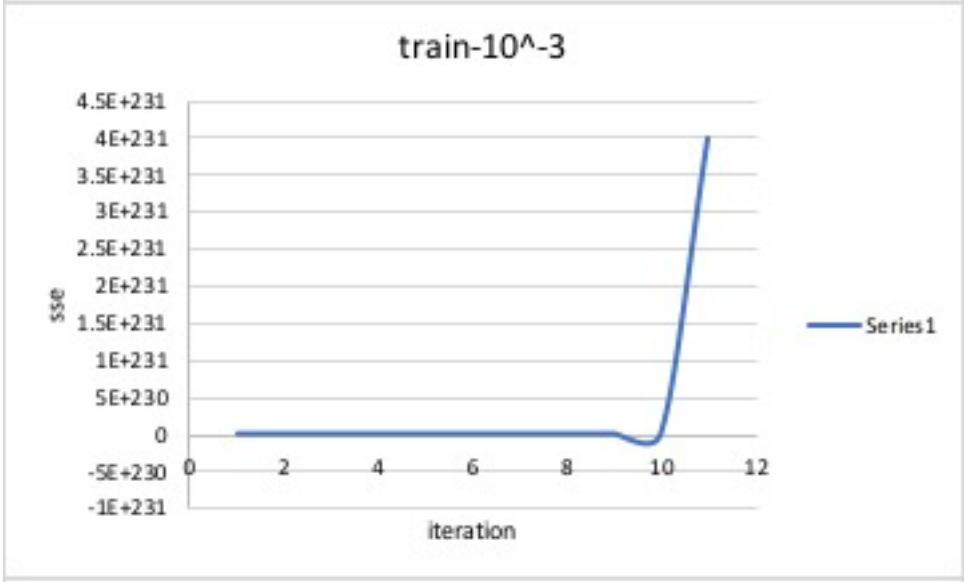

validation-10^-9
9E+300
8E+300
7E+300
6E+300
5E+300
4E+300
3E+300
2E+300
1E+300
0
-1E+300
sse
0
10
20
30
40
iteration
Series1

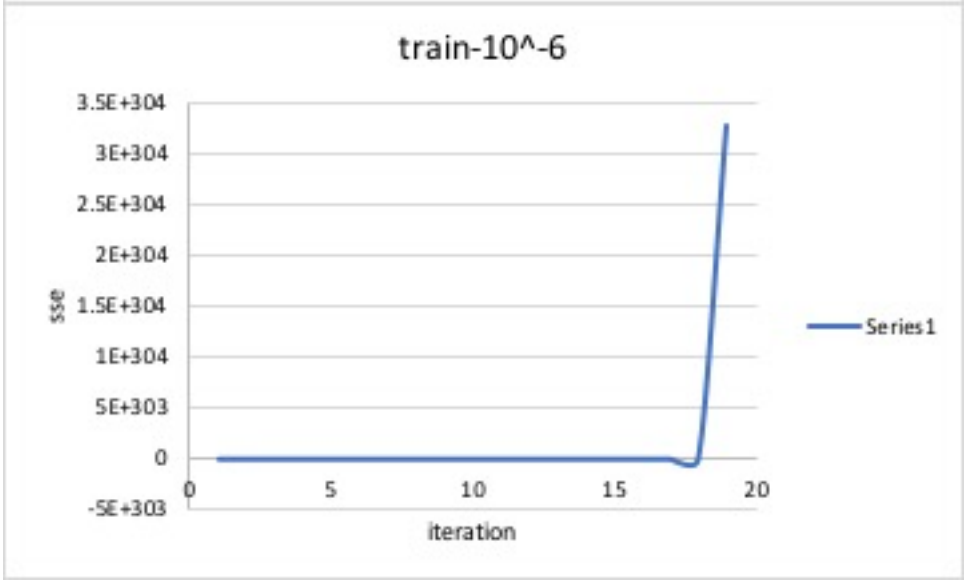

VALIDATION-10^-15
250000
200000
150000
100000
50000
0
SSE
0
2000
4000
6000
8000
10000
12000
ITERATION
Series1

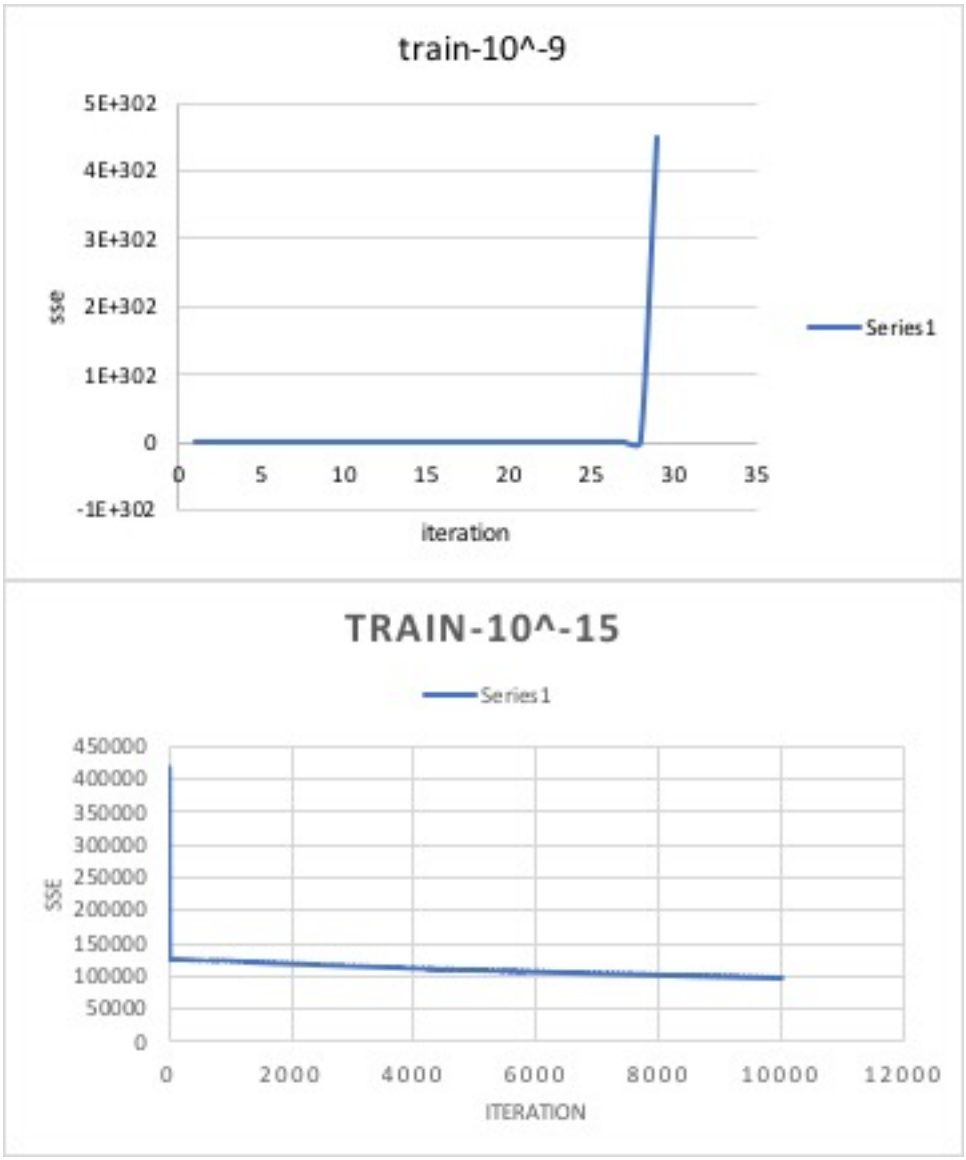
Train:


train-0
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
sse
0
5
10
15
20
25
30
35
iteration
Series1


train-1
4.5E+291
4E+291
3.5E+291
3E+291
2.5E+291
2E+291
1.5E+291
1E+291
5E+290
0
-5E+290
-1E+291
sse
0
2
4
6
8
10
12
iteration
Series1


train-10^-3
4.5E+231
4E+231
3.5E+231
3E+231
2.5E+231
2E+231
1.5E+231
1E+231
5E+230
0
-5E+230
-1E+231
sse
0
2
4
6
8
10
12
iteration
Series1


train-10^-6
3.5E+304
3E+304
2.5E+304
2E+304
1.5E+304
1E+304
5E+303
0
-5E+303
sse
0
5
10
15
20
iteration
Series1



For this part, we train with non-normalized data. It is obvious to observe that when learning rate is 0, both Training SSE and Validation SSE have fixed number due to algorithm ($w_{new} = w_{old}$ – learning rate * $\nabla E(w)$). Therefore, we always get the same w. Also, we observe that when learning rate is until 10^{-15}, the SSE value is not infinity. According to pictures, we know that the infinity speed of training data is faster than validation data. Therefore, because it is hard to get convergence result when we train with non-normalized data, we can easily know that normalized data is better for us to train.