(CS-534) Machine Learning
Implementation Assignment #1

Submission By:

Akash Agarwal
OSU ID NUMBER: 933-471-097
Vishnupriya Nochikaduthekkedath Reghunathan
OSU ID NUMBER: 933-620-571
Anand P Koshy
OSU ID NUMBER: 933-617-060

Contribution by each member in percentage : Equal contribution by each member

Akash Agarwal - 33.33%
Vishnupriya Nochikaduthekkedath Reghunathan 33.33%
Anand P Koshy -33.33%
[Each member has established their own code and later its collaborated and 1 report is made]

Part 0: Preprocessing and simple analysis. Perform the following preprocessing of your data.

In this part, we are doing the preprocessing of the data by removing the id feature which we considered is not relevant in determining the price of the house, then we are splitting the date feature into day, month and year and make it into 3 different features for making the date feature more usable, the statistics of the features are calculated and certain features are predicted to be important to determine the price of the house.

A. Remove the ID feature. Why do you think it is a bad idea to use this feature in learning?
ID is a unique identification for each example or tuple. And it does not contribute to the final expected result . A powerful classifier could use the id column to create a perfectly fit on training set ignoring other features and that might make our model unreliable. And this might also create ooring other features and that might make our model unreliable. And this might also create overfitting .

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?
The Day and Month can be put together in a column and access it as a single feature. So that we can classify the data according to the period it occured, Ex. Late June or all events before Dec 20 etc . This will also help to reduce the no : of features to consider. The day alone has no advantage.

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.

NUMERICAL FEATURES

Numerical Features	MEAN	STD	RANGE
bedrooms	3.3752	0.943246485	32
bathrooms	2.118875	0.765128111	7.25
sqft_living	2080.2232	911.334358297	9520
sqft_lot	15089.2014	41203.894918348	1650787
floors	1.5037	0.542646991	2.5
sqft_above	1793.0993	830.865434458	8490
sqft_basement	287.1239	435.005263539	2720
long	-122.2132865	0.141404685	1.195
lat	47.55981419	0.13865059	0.6217
sqft_lot15	12746.3234	28241.243042566	870540
sqft_living15	1994.3261	691.900301027	5650
yr_built	1971.1249	29.480593795	115

CATEGORICAL FEATURES

grade	Percentage	condition	Percentage
7	41.3	3	65.3
8	28.38	4	25.69
9	11.82	5	8.12
6	9.33	2	0.76
10	5.47	1	0.13
11	2.1		
5	1.05	**waterfront**	**Percentage**
12	0.39	0	99.3
4	0.11	1	0.7
13	0.05		

zipcode	Percentage	zipcode	Percentage
98103	2.82	98168	1.32
98115	2.76	98031	1.32
98038	2.67	98107	1.28
98052	2.67	98003	1.27
98042	2.6	98040	1.27
98034	2.56	98136	1.26
98117	2.46	98112	1.24
98023	2.35	98166	1.22
98006	2.35	98146	1.21
98059	2.31	98178	1.15
98133	2.28	98055	1.15
98118	2.2	98030	1.14

98074	2.11	98045	1.13
98058	2.1	98105	1.12
98155	2.07	98022	1.12
98027	2.03	98008	1.09
98033	1.87	98177	1.09
98125	1.87	98077	0.94
98053	1.86	98011	0.92
98092	1.8	98002	0.88
98075	1.77	98019	0.83
98056	1.72	98119	0.79
98126	1.72	98108	0.77
98001	1.61	98005	0.75
98029	1.6	98188	0.66
98199	1.58	98007	0.64
98144	1.55	98070	0.53
98116	1.52	98102	0.52
98106	1.5	98014	0.52
98004	1.49	98032	0.48
98065	1.43	98109	0.47
98198	1.38	98010	0.42
98122	1.38	98024	0.31
98028	1.35	98148	0.27
98072	1.34	98039	0.24

Date	Percentage	Month	Percentage
23	4.4	5	11.23
24	3.88	7	10.37
20	3.77	6	10.3
9	3.67	4	10.08
5	3.53	10	8.87
16	3.52	8	8.74
2	3.47	3	8.46
17	3.45	9	8.38
8	3.44	12	6.84
12	3.42	11	6.44
13	3.4	2	5.75
18	3.38	1	4.54
27	3.32		
10	3.32	Year	Percentage
25	3.3	14	68.15
22	3.27	15	31.85
4	3.24		
26	3.21		
7	3.2		
21	3.16		

14	3.13		
6	3.08		
19	3.04		
3	3.03		
30	3.03		
29	2.98		
11	2.95		
15	2.88		
28	2.87		
1	2.47		
31	1.19		

view	Percentage
0	90.3
2	4.25
3	2.5
1	1.62
4	1.33

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?

bedrooms - From the Statistics, the range is high for the feature and the no of bedrooms is also an important feature that people consider while buying a house.
bathrooms - From the meaning of Feature, the no of bathrooms can be considered as an important value determining the price of a house.
sqft_living - From Statistics, the standard deviation and range are high for this feature, so we expect sqft_living has an impact on the price of the house.
sqft_lot - Square footage of the lot has higher standard deviation and range, so it might affect the predicted value.
condition - Overall condition 1 indicates worn out property and 5 excellent. This feature gives us an idea of the worn out condition of the house.
grade - Overall grade given to the housing unit. 1 poor ,13 excellent. This can be considered as an important feature that in 1 value, it gives the total condition and price of the house can easily determined from this value.
yr_built - The year built gives us an idea of how old the house is and can predict the price of house based on it. As the house is old the price may decrease
lat, long - This feature gives us the location of the house , a main feature considered determining the price of a house

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.
All the features have been normalized to range: [0, 1]. Please refer the code for the same. The normalizing eq used is (x - x(min)) / (x(max)-x(min).

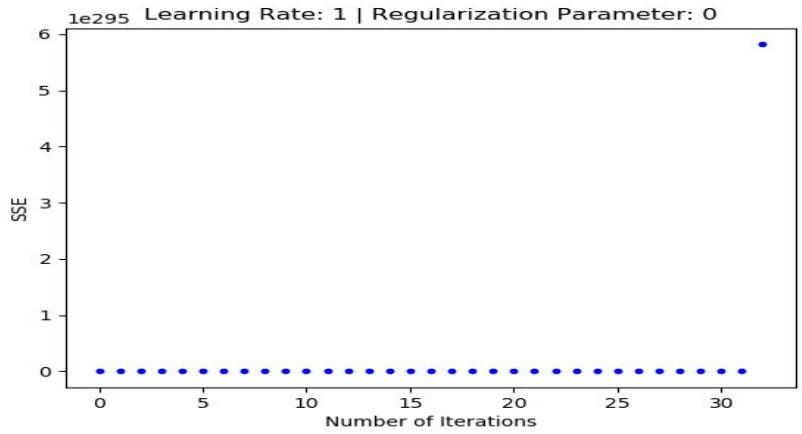
Part 1. Explore different learning rate for batch gradient descent. For this part, you will work with the preprocessed and normalized data and fix λ to 0 and consider at least the following values for the learning rate: 10^{0} , 10^{-1} , 10^{-2} , 10^{-3} , 10^{-4} , 10^{-5} , 10^{-6} , 10^{-7} .

In this part, we try to find the best learning rate for our model on training data. Model is tested with different learning rate and graphs are plotted for easy evaluation. Here threshold is taken as 0.05. For the learning rate that converges, we note down the SSE on training data and SSE on validation data and the number of iterations for evaluating our model.
The best features that affect our data is also calculated from the weights of each feature and compared it to the preprocessing result

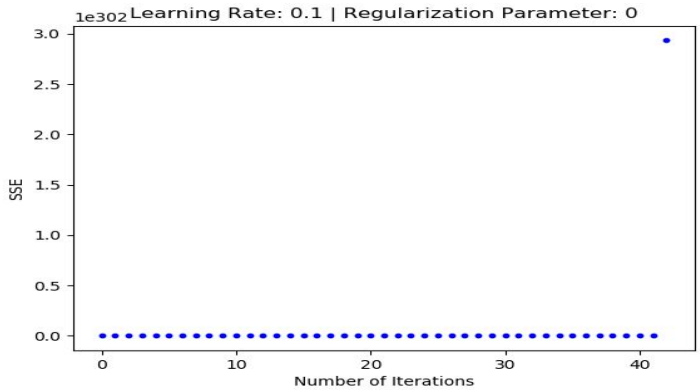
A. Which learning rate or learning rates did you observe to be good for this particular dataset? What learning rates make the gradient decent explode? Report your observations together with some example curves showing the training SSE as a function of training iterations and its convergence or non-convergence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

Learning Rates 10^{-5} , 10^{-6} , 10^{-7} were found to be good for the dataset. Among these we observe that 10^{-5} is the best learning rate, as it causes convergence at lesser number of iterations. For learning rates, 10^{0} , 10^{-1} , 10^{-2} , 10^{-3} , 10^{-4} the Gradient descent is observed to explode.
Following are the graphs observed with different learning rates:

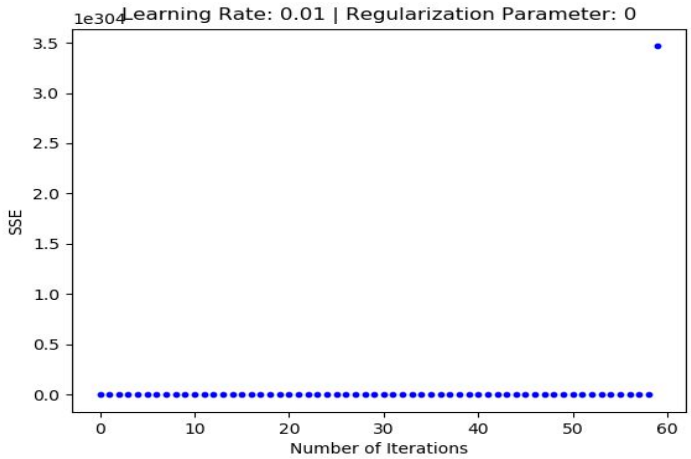
Learning Rate = 10^{0}



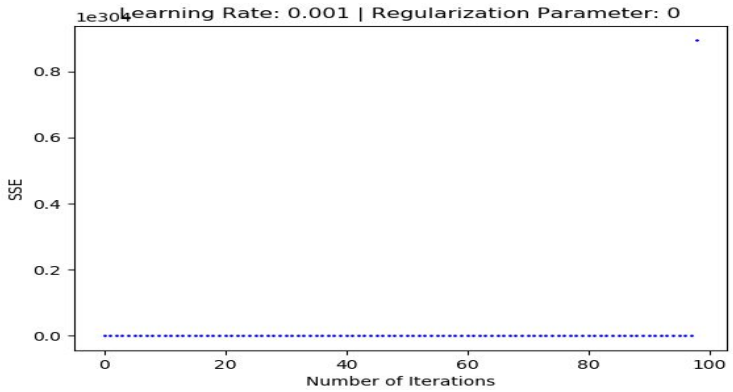
Learning Rate = 10^{-1}



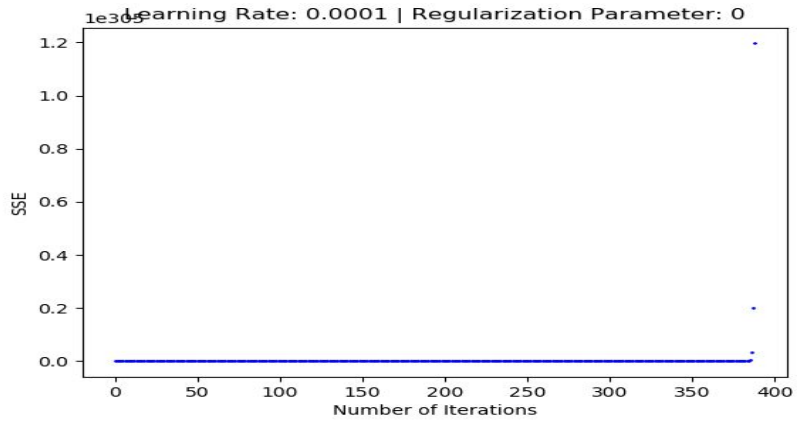
Learning Rate = 10^{-2}



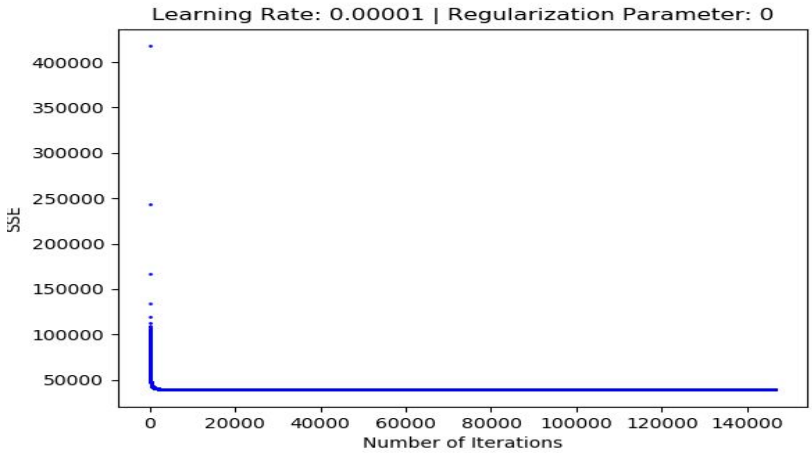
Learning Rate = 10^{-3}



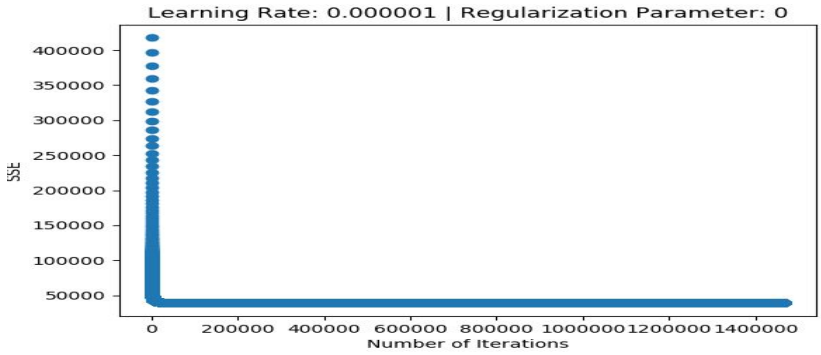
Learning Rate = 10^{-4}



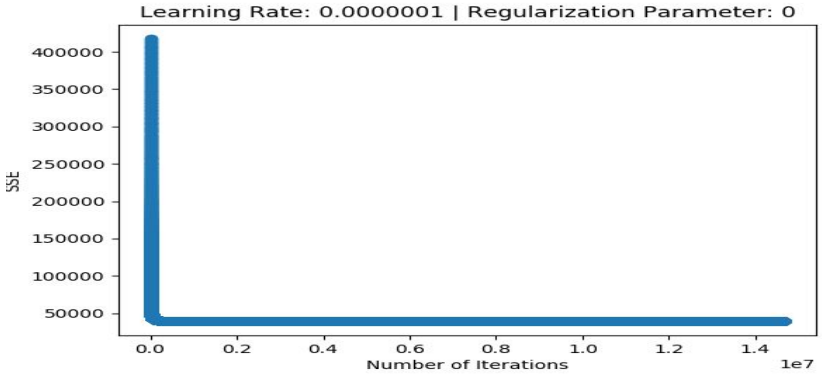
Learning Rate = 10^{-5}



Learning Rate = 10^{-6}



Learning Rate = 10^{-7}



Observation : From the graphs, we can observe that for learning rates 10^{0} , 10^{-1} , 10^{-2} , 10^{-3} , 10^{-4} the gradient descent explodes and for values 10^{-5} , 10^{-6} , 10^{-7} the gradient descent converges when threshold is 0.05.

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?

Learning Rate	SSE on Training Data	SSE on Validation Data	No. of iterations for convergence
0.00001	38991.518046	55026.267338962	146683
0.000001	38991.518048	55026.257447032	1466838
0.0000001	38996.172339	Program taking too long to converge	5730986

Observation : For Learning Rate 10^{-5}, the program converges faster with lesser number of iterations . So 10^{-5} can be considered as the best learning rate among all the values of learning rate tested. Difference in SSE on Validating data and training data when learning rate is 10^{-5} is 16,034.749 (approx) when threshold is 0.5.

For 10^{-7} the program is taking too long to converge, the value shown in table is the observed value when norm is 3.645032 and threshold is 0.5.

It is observed that as the learning rate decreases, the number of iterations for convergence increases. On comparing the SSE on validation and training data we observe that SSE on validation data

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)).

For LEARNING RATE: 10^{-5} and lambda = 0 , SSE on training: 38991.518046, SSE on validation: 55026.2673389624

Features	**Weights**
dummy	-2.02634726
day	-0.17278705
month	0.18228678
year	0.38104872
bedrooms	-9.1811135

bathrooms	3.33191851
sqft_living	7.26277625
sqft_lot	2.31366786
floors	0.11770157
waterfront	4.71107643
view	2.32284388
condition	1.25636608
grade	8.8744548
sqft_above	7.72517044
sqft_basement	1.3069606
yr_built	2.9363435
yr_renovated	0.28851062
zipcode	-1.02065174
lat	3.76772534
long	-2.67437416
sqft_living15	1.33430012
sqft_lot15	-2.91738238

From the weights observed for each Feature, we can conclude that, the most important features are (here |weight| > 3 is taken as a standard to select features):

- bedrooms
- bathrooms
- Sqft_living
- waterfront
- grade
- sqft_above
- lat

The features predicted in the preprocessing almost matches with the important features we got by calculating weights with the following mis- matches.

In preprocessing we predicted features (sqft_lot ,condition, yr_built, long) have a direct and great impact on the price of the house, but these were observed to be irrelevant when we Compare it with the weights of the feature.
The features waterfront, sqft_above has higher weights and contribute greater to the price of the house and in the preprocessing analysis we ignored these feature, considering the statistics and general meaning of the feature which was a mistake in our pre analysis

Part 2. 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?
   SSE on training: 38991.518046, for lambda = 0
   SSE on training: 38991.923541 for lambda = 10^{-3}
   SSE on training: 38995.200945 for lambda = 10^{-2}
   SSE on training: 39027.707857 for lambda = 10^{-1}
   SSE on training: 39331.439084 for lambda = 1
   SSE on training: 41626.413091 for lambda = 10
   SSE on training: 55302.249289 for lambda = 100
   The SSE for training data is increasing as we increase the value of lambda. For lambda = 0, lambda = 10^{-3} and lambda = 10^{-2}; the values of SSE are almost similar with a minute difference. There is a huge difference between values of SSE for lambda = 10 and lambda = 100

B. What tread do you observe from the validation SSE?
   SSE on validation: 55026.2673389624, for lambda = 0
   SSE on validation: 55153.875909494964, for lambda = 10^{-3}
   SSE on validation: 55030.86264954815, for lambda = 10^{-2}
   SSE on validation: 53836.82239130535, for lambda = 10^{-1}
   SSE on validation: 44725.109954456595, for lambda = 1
   SSE on validation: 25849.413590700195, for lambda = 10
   SSE on validation: 26019.751367557816, for lambda = 100
   The SSE for validation data is decreasing as we increase the value of lambda. For lambda = 0, lambda = 10^{-3} and lambda = 10^{-2}; the values of SSE are almost similar with just a minute difference. There is a huge difference in SSE for lambda = 1 and lambda = 10. For lambda = 100 the value of SSE started increasing again and SSE(on lambda = 10) < SSE(on lambda = 100)

C. Provide an explanation for the observed behaviors.
   Since the lambda is regularization parameter, as we increase the value of lambda the value of SSE on training set is increasing, because it is preventing the model to be overfitted on the training examples. Therefore, the more the values of lambda the more it prevents the model to be overfitted. The SSE on validation set on the other hand, starts

to decrease as we increase lambda, this is because, since the model is less overfitted, the model is better suited for new values of features and thus, provides better prediction on Y values. But on very large values of lambda, for example lambda = 100, the model SSE on validation will also start increasing, this is because, large lambda regularizes the features such that the curve fitted on the training example is starts getting worse. This in turn makes a worse model for prediction of Y as compared to lesser values of lambda. Thus, this will also increase the SSE for validation data.

D. What features get turned off for λ= 10, 10^{-2} and 0 ?

- The features and corresponding weights when λ= 10 is given in the table below:

Features	Weight
dummy	-2.30162163
day	-0.17665345
month	0.16986205
year	0.36451656
bedrooms	-2.60010639
bathrooms	3.07401812
sqft_living	5.99413607
sqft_lot	0.37431906
floors	0.4000786
waterfront	4.13930535
view	2.51382939
condition	1.22080617
grade	8.46154188
sqft_above	6.32555623
sqft_basement	1.23536873
yr_built	-2.77073519
yr_renovated	0.35356061
zipcode	-0.90694042
lat	3.73070421
long	-2.40195666
sqft_living15	2.23262639
sqft_lot15	-0.5092292

From the table we observe that the features day, month, year, sqft_lot, floors, yr_renovated, zipcode, sqft_lot15 can be considered as turned off as |weight| is approximately equal to 0.

- The features and corresponding weights when λ= 10 -2 is given in the table below:

Features	Weight

dummy	-2.00846633
day	-0.17441448
month	0.17733064
year	0.37827103
bedrooms	-9.18125146
bathrooms	3.33236338
sqft_living	7.26394362
sqft_lot	2.29491333
floors	0.11912888
waterfront	4.70999854
view	2.32358296
condition	1.24858153
grade	8.86886866
sqft_above	7.72633215
sqft_basement	1.30742035
yr_built	-2.9404617
yr_renovated	0.28664443
zipcode	-1.02417363
lat	3.76507714
long	-2.68048338
sqft_living15	1.333717
sqft_lot15	-2.90230302

From the table we observe that the features day, month, year, floors, yr_renovated, zipcode, sqft_living15 can be considered as turned off as |weight | is approximately equal to 0.

- The features and corresponding weights when λ= 0 is given in the table below :

Features	**Weight**
dummy	-2.02634726
day	-0.17278705
month	0.18228678
year	0.38104872
bedrooms	-9.1811135
bathrooms	3.33191851
sqft_living	7.26277625
sqft_lot	2.31366786
floors	0.11770157
waterfront	4.71107643
view	2.32284388

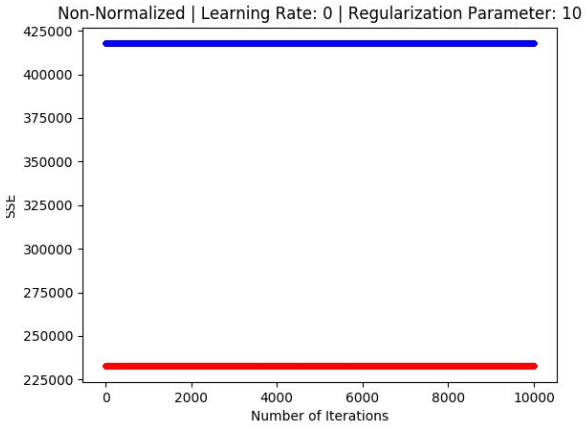
condition	1.25636608
grade	8.8744548
sqft_above	7.72517044
sqft_basement	1.3069606
yr_built	-2.9363435
yr_renovated	0.28851062
zipcode	-1.02065174
lat	3.76772534
long	-2.67437416
sqft_living15	1.33430012
sqft_lot15	-2.91738238

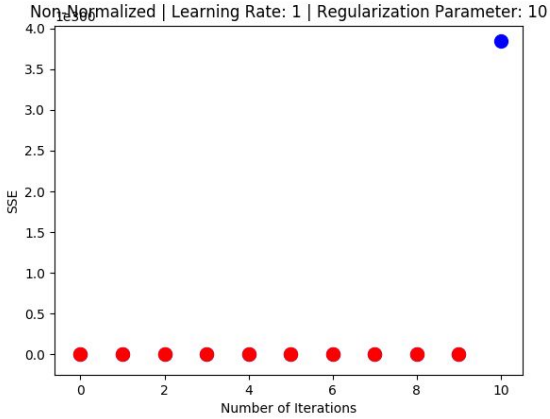
From the table we observe that the features day, month, year, floors, yr_renovated can be considered as turned off as |weight| is approximately equal to 0.

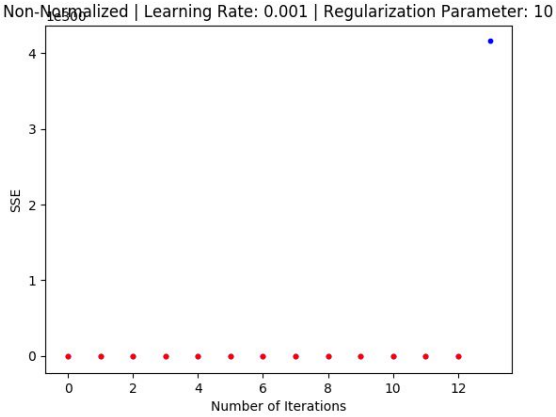
For different λ , different features are turned off, which means that weight of the features depends on λ. For all three cases we noticed that certain values like day, month, year,floors and yr_renovated is turned off and is expected to not create impact on prediction of price of the house.

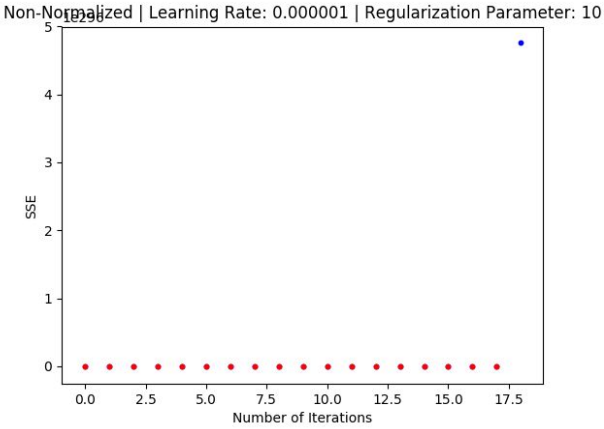
Part 3. Training with non-normalized data Use the preprocessed data but skip the normalization. Consider at least the following values for learning rate: 1, 0, 10^{-3}, 10^{-6} 10^{-9} , 10^{-15} . For each value , train up to 10000 iterations (Fix the number of iterations for this part). If training 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?

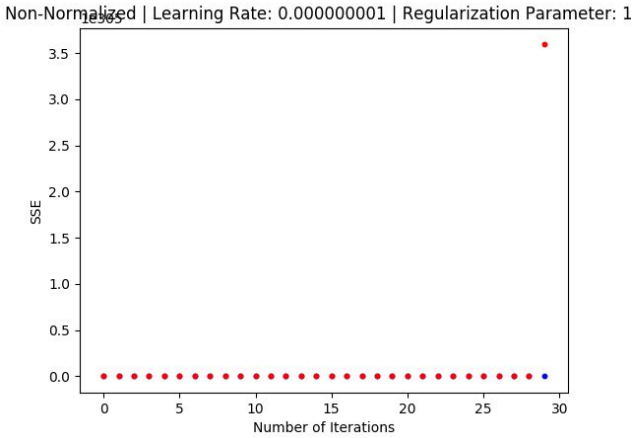
In this part, we take our take the training data, pre-process it but train the model without normalizing our dataset. We train our model for different values of learning rate and plot the curve for SSE v/s No. of iterations for the training and validation set. We then compare the results obtained with the results previously obtained for SSE v/s No. of iterations obtained on normalized data. We then write our analysis for the trend observed.

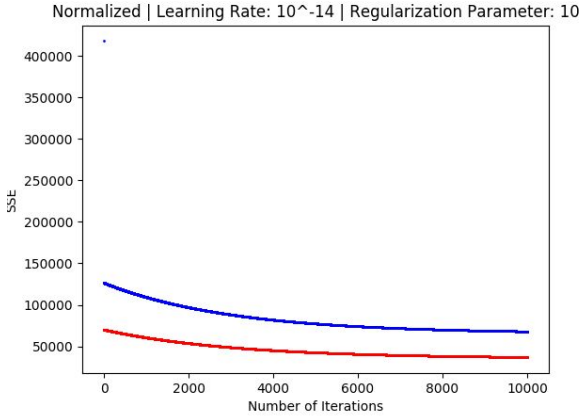

Non-Normalized | Learning Rate: 0 | Regularization Parameter: 10
SSE
Number of Iterations

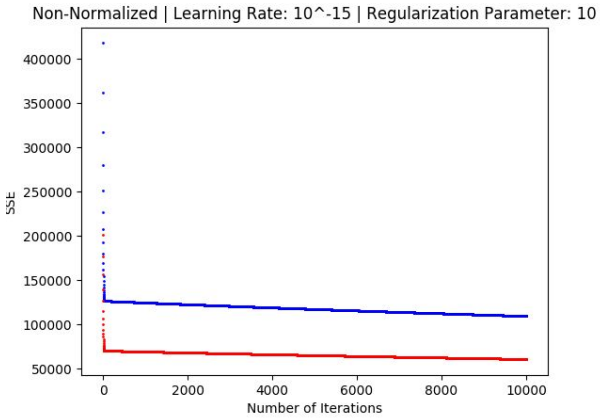

Non-Normalized | Learning Rate: 1 | Regularization Parameter: 10
1e380
SSE
Number of Iterations


Non-Normalized | Learning Rate: 0.001 | Regularization Parameter: 10
1e300
SSE
Number of Iterations


Non-Normalized | Learning Rate: 0.000001 | Regularization Parameter: 10
1e298
SSE
Number of Iterations







Learning rate = 0: No change in value of SSE
Learning rate = 1: Diverging
Learning rate = 10^{-3}: Diverging
Learning rate = 10^{-6}: Diverging
Learning rate = 10^{-9}: Diverging
Learning rate = 10^{-14}: Converging
Learning rate = 10^{-15}: Converging

We observed that, while for the normalized data, the gradient descent started converging at learning rate = 10^{-5}, the non-normalized dataset was diverging for almost all values of learning rate greater than 10^{-14}.

The gradient descent stopped exploding, i.e. diverging for learning rate = 10^{-14}

On comparing the normalized and non-normalized versions of the data, we observe that the normalized data reaches convergence for greater values of learning rate and is easier to predict as compared to the non-normalized data, the non-normalized data needs very minute values of learning rate. Also, the point at which it starts converging is also difficult to find in case of non-normalized data.

Thus, it is easier to train normalized data, because, it is easier to predict the learning rate and easier to find the point at which the gradient descent starts to converge.