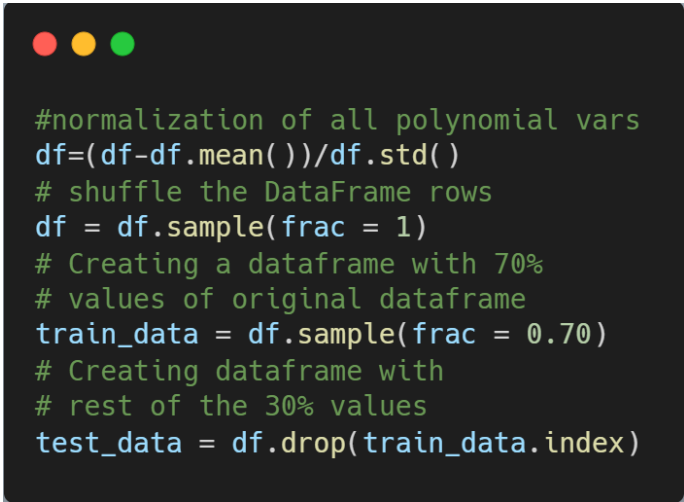


Assignment 1: Foundations of Data Science(CS F320)

Data preprocessing:

The scale of all variables in the data was not similar. Therefore, the first thing we did was normalize our dataframe before training any models so as to bring all the variable values to a common scale, so that they are easily comparable.

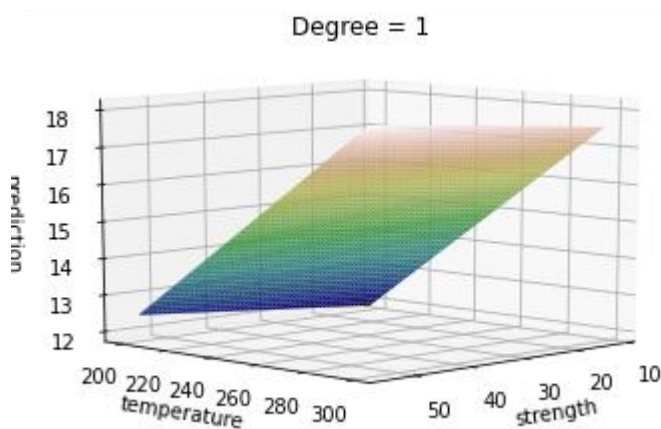
After that, we shuffled the data set rows randomly, using a built-in function. Next, we made the 70-30 split as mentioned, for training and testing data respectively, to run our models.



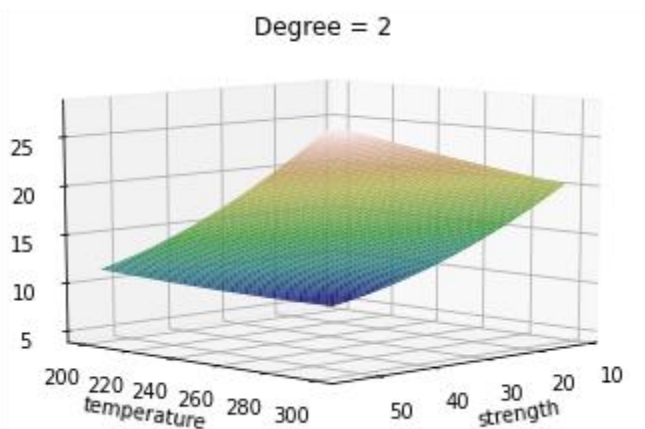
```
#normalization of all polynomial vars
df=(df-df.mean())/df.std()
# shuffle the DataFrame rows
df = df.sample(frac = 1)
# Creating a dataframe with 70%
# values of original dataframe
train_data = df.sample(frac = 0.70)
# Creating dataframe with
# rest of the 30% values
test_data = df.drop(train_data.index)
```

Polynomial Regression Using Gradient Descent Without Any Regularization

- Regression can be defined as, “Using the relationship between variables to find the best fit line or the regression equation that can be used to make predictions”.
- We are trying to use polynomial regression to fit a polynomial line so that we can achieve a minimum error or minimum cost function.
- This model contains no regularization term and hence there is no restriction on the values that the weights could take. The basic idea behind this algorithm is that we start at a random point on the loss function curve, and take small steps towards the minimum based on the current errors.
- The size of our steps is determined by a hyperparameter “learning rate”. A higher learning rate means we take bigger steps towards the minimum. However, this also means that there is a chance/possibility of overshooting and oscillating around the minimum.

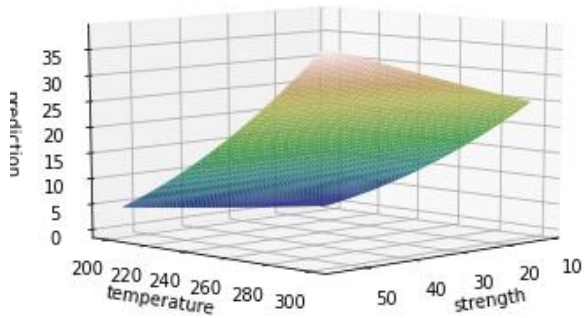


For degree 1
Train error = 0.22345417157238723
Test error = 0.26809713793131884



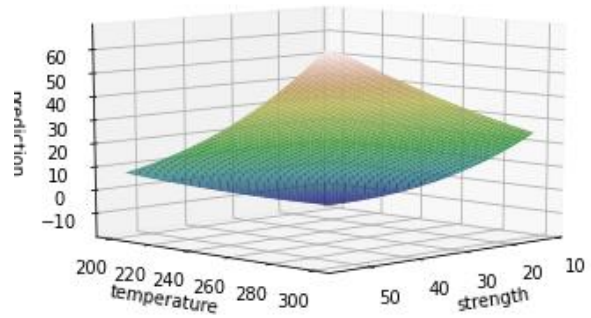
For degree 2
Train error = 0.19939892454031222
Test error = 0.23875819879437812

Degree = 3



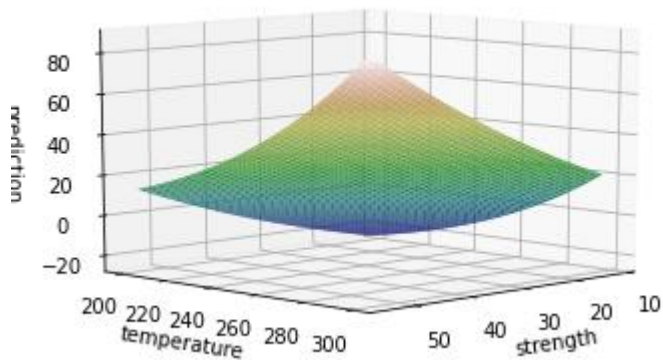
For degree 3
Train error = 0.2116889502669954
=0.22112125409604938
Test error = 0.2495997684587392

Degree = 4



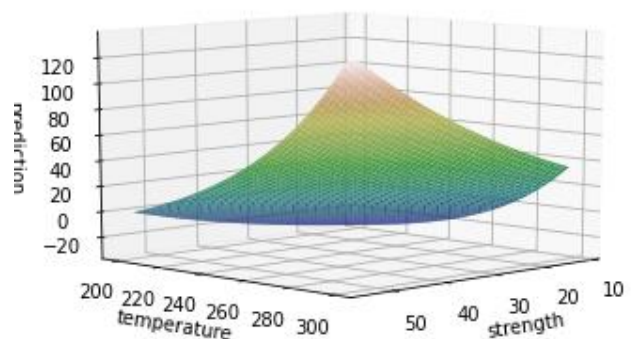
For degree 4
Train error
Test error = 0.25807902956981654

Degree = 5

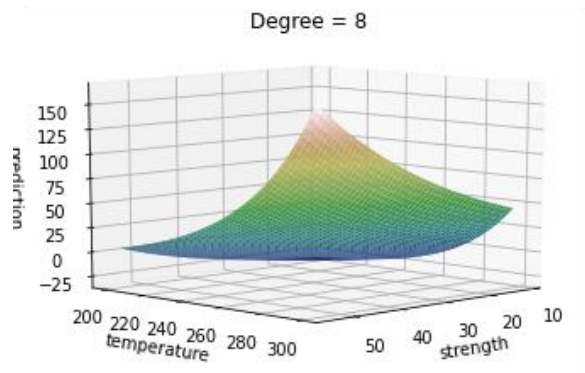
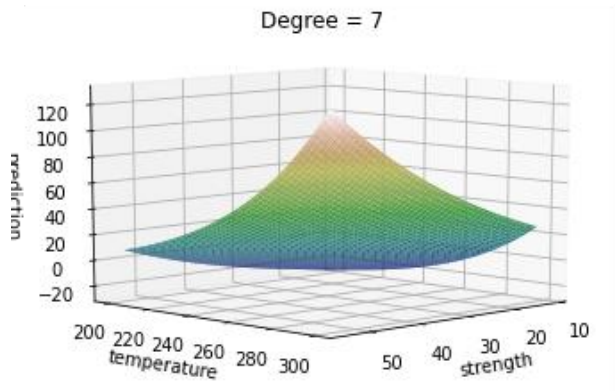


For degree 5
Train error = 0.21788402035259977
Test error = 0.2541479860074944

Degree = 6



For degree 6
Train error = 0.22695201766091289
Test error = 0.2651238607883532



For degree 7

Train error = 0.20808812310859975

Test error = 0.24456620931448453

For degree 8

Train error = 0.19773156328276054

Test error = 0.23616019222005513

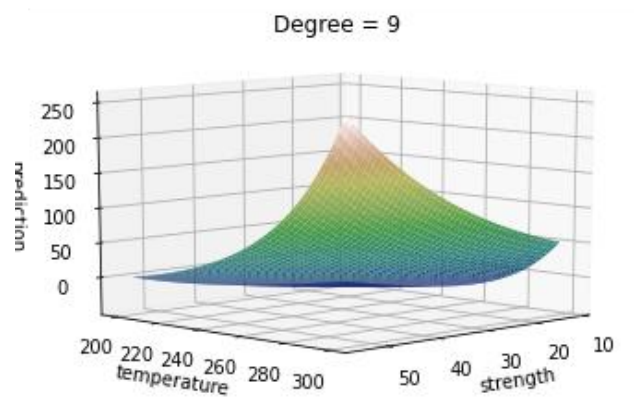


Figure 1 For degree 9, training error = 0.21527995942530426
Test error = 0.24510184077712666

Polynomial Regression Using Stochastic Descent Without Any Regularization

- It is an iterative method used to optimize an objective function.
- It is the stochastic approximation of Gradient Descent optimization, since it replaces the actual gradient with an approximation.

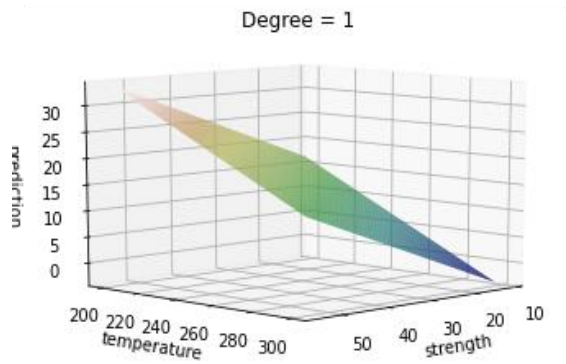


Figure 1: Train error = 0.18771667657266755
Test error = 0.22843062267199815

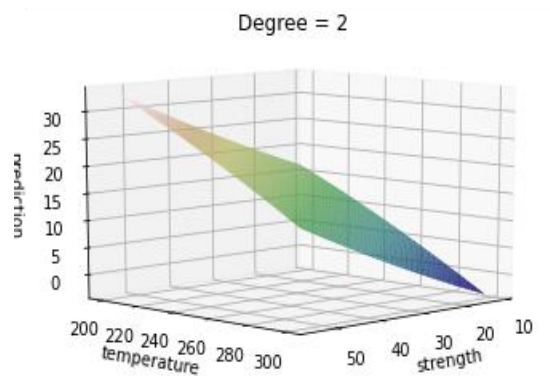


Figure 2 Train error = 0.1898797479593413,
Test error = 0.2307464156660114

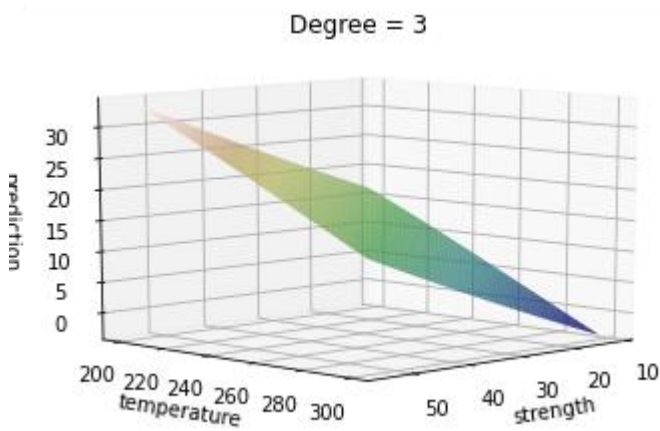


Figure 3: Train error = 0.1903929876221352,
Test error = 0.23124879410266955

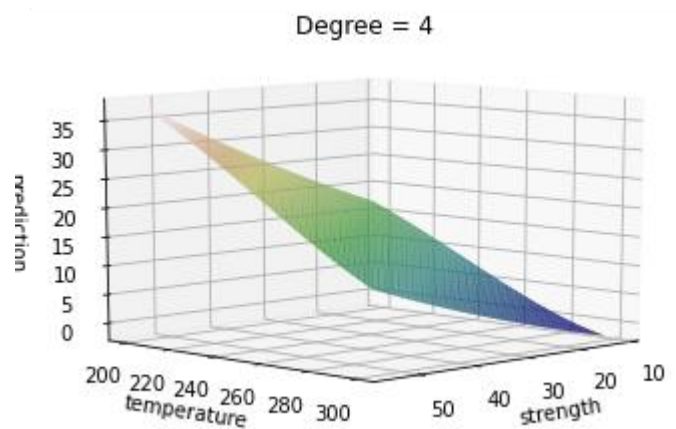


Figure 3: Train error = 0.18643886810763258,
Test error = 0.22643717783857503

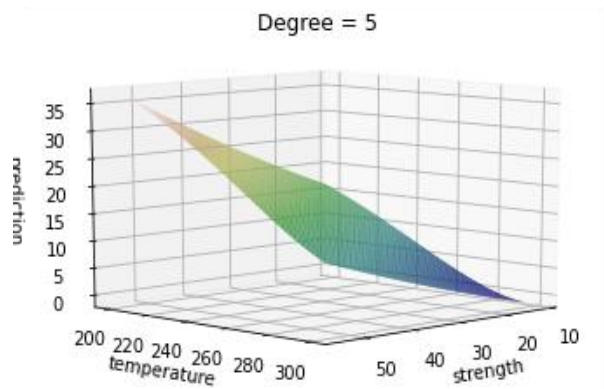


Figure 5: For degree 5: Train error = 0.18657940538095352, Test error = 0.2273425580811136

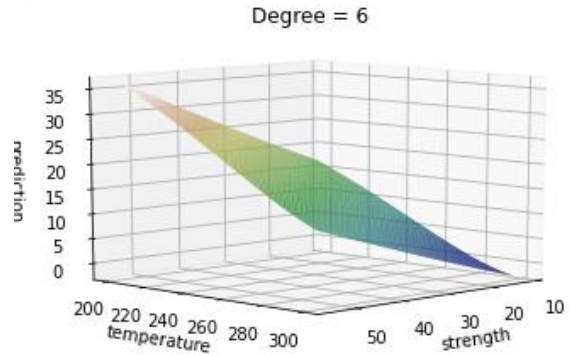


Figure 6: For degree 6 - Train error = 0.18662457713421088, Test error = 0.22934508504959455

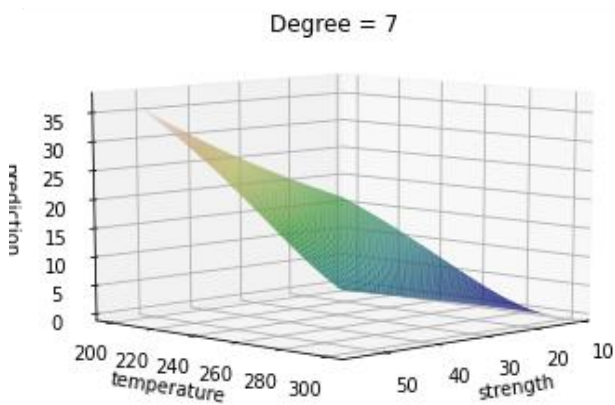


Figure 5: For degree 7 - Train error = 0.18486605355178937, Test error = 0.22645805116035686

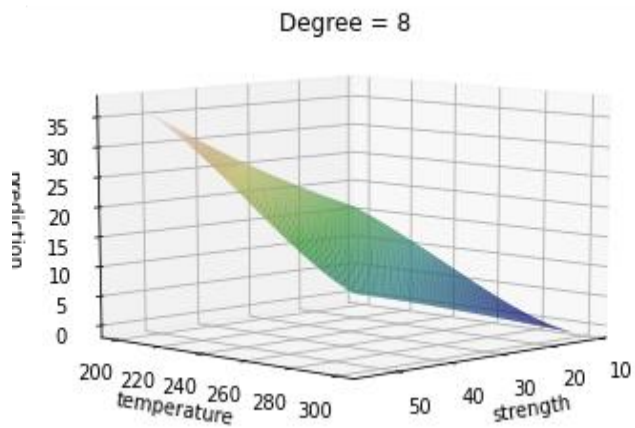


Figure 4: Degree 8 - Train error = 0.18555387579940635, Test error = 0.22899212301530222

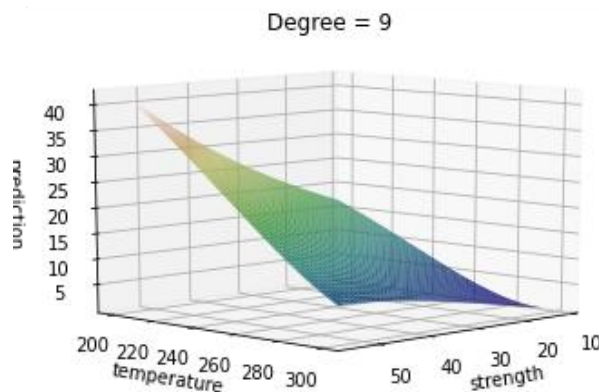
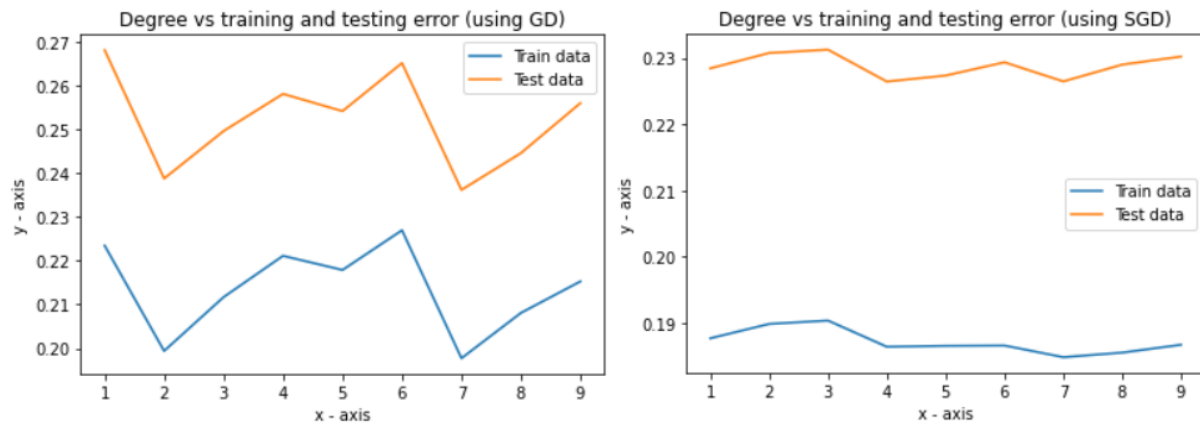


Figure 6: For degree 9 - Train error = 0.1867369634137333, Test error = 0.23020660566188503

Degree vs training and testing errors



To determine which degree polynomial provides the best fit to the data, we will find the degree for which the cost is minimum by looking at the graph. For GD, we can clearly see that the cost is least for degree 7, because the training RMSE and testing RMSE are minimum here.

For SGD, it is difficult to find the right degree for the polynomial because the Root Mean Square Errors are almost the same for each degree, with slight variations.

Overfitting is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. In case of overfitting, the difference between the training error and the testing error becomes high. The training error is minimised in this case. The testing error grows abnormally.

Here, we cannot observe overfitting, because the degree of the polynomial is much less than the number of training data points.

Polynomial Regression with regularization

Lasso Regression

- The problem with plain regression is the model can have a tendency to overfit the training data, not making it generalise well.
- To combat this problem, we impose a restriction on the sum of absolute values of the weights.
- The degree of regularisation is determined by the hyperparameter λ
- Cost function:

$$J(\theta) = \frac{1}{2} \left(\sum_{i=1}^n (x_i^T \theta - y_i)^2 + \lambda \sum_{j=0}^n |\theta_j| \right)$$

Ridge Regression:

- Another regularisation technique follows the same principle of imposing restrictions on the possible weight values but has a slightly different cost function
- Here, we use the squared value of the weights instead of the absolute value.
- The degree of regularisation is determined by the hyperparameter λ
- Cost function:

$$J(\theta) = \frac{1}{2} \left(\sum_{i=1}^n (x_i^T \theta - y_i)^2 + \lambda \sum_{j=0}^p \theta_j^2 \right)$$

1. Stochastic Gradient Descent with Ridge regression

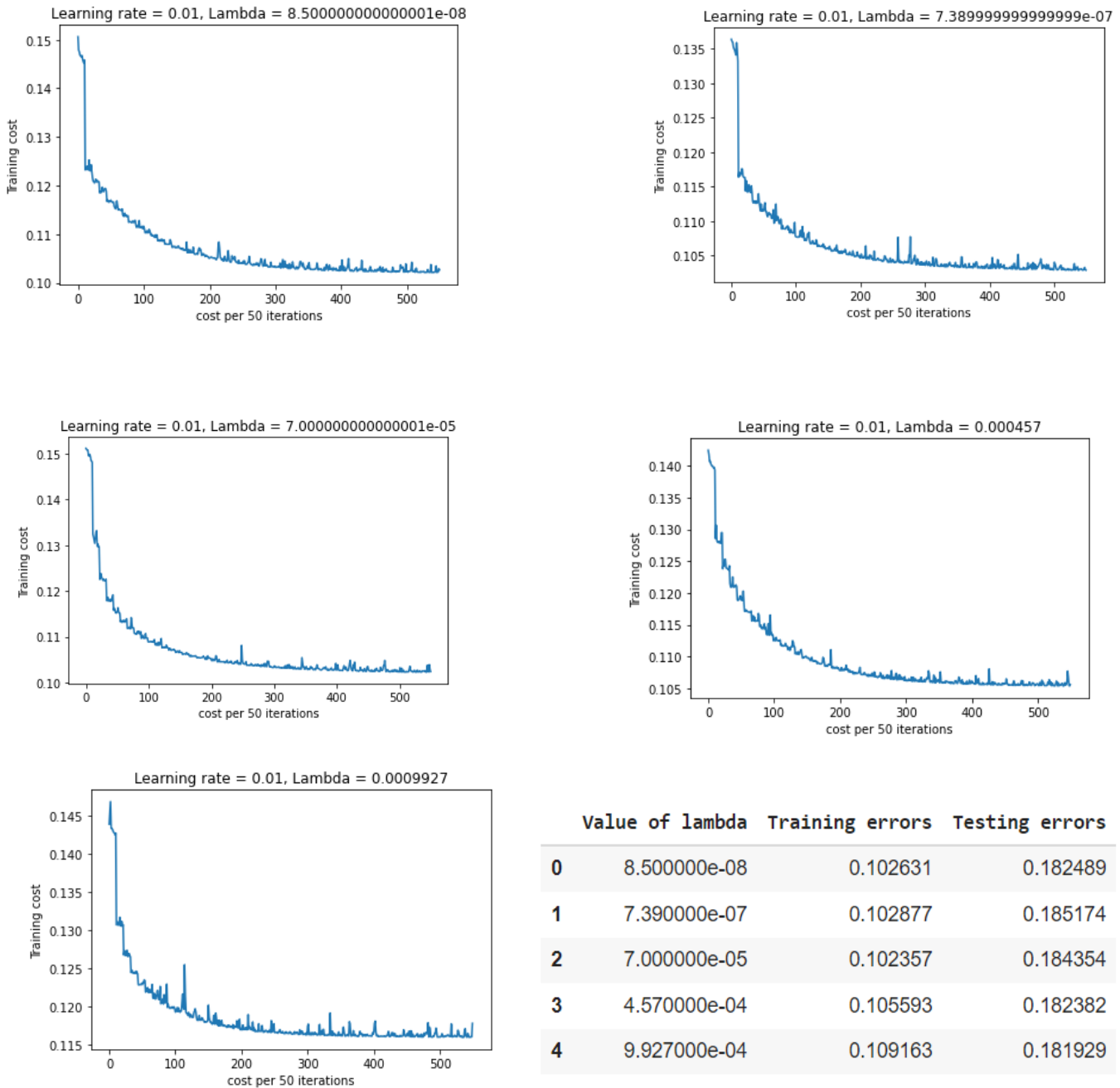
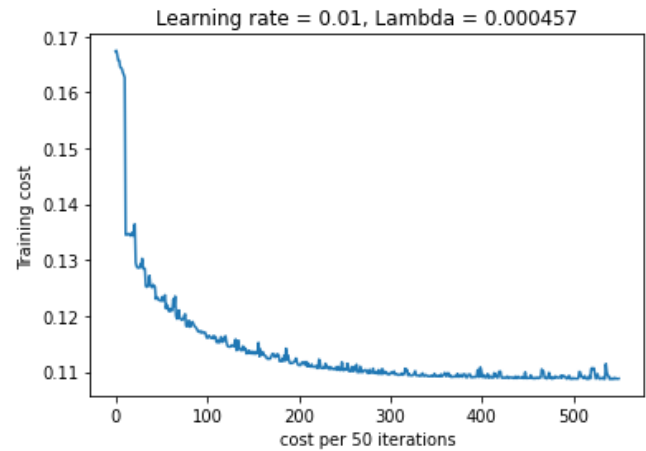
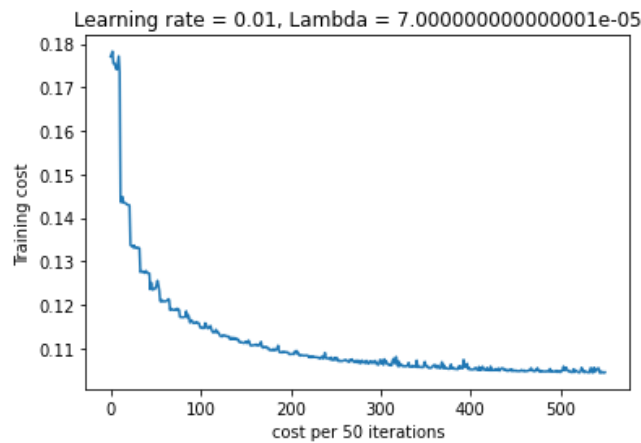
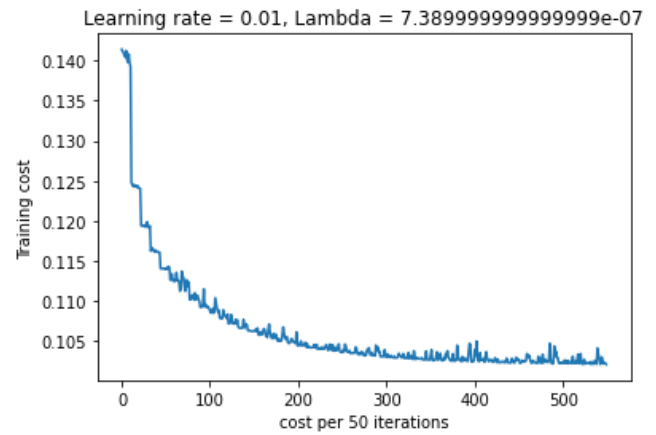
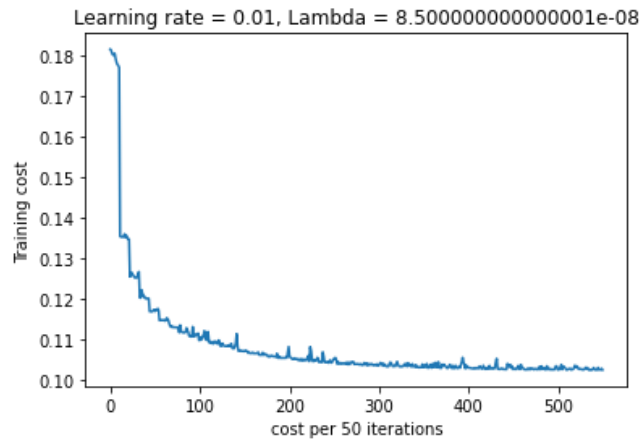
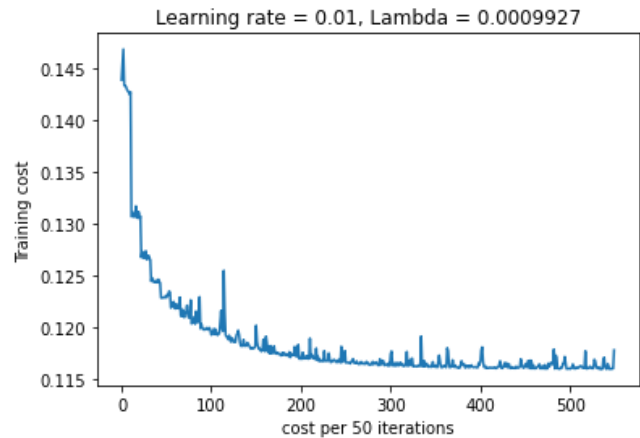


Figure 7: Training and testing errors for 5 diff. values of lambda (SGD and Ridge Regression)

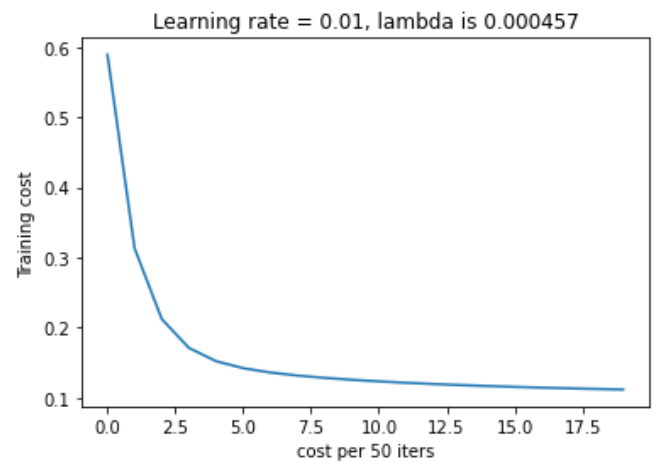
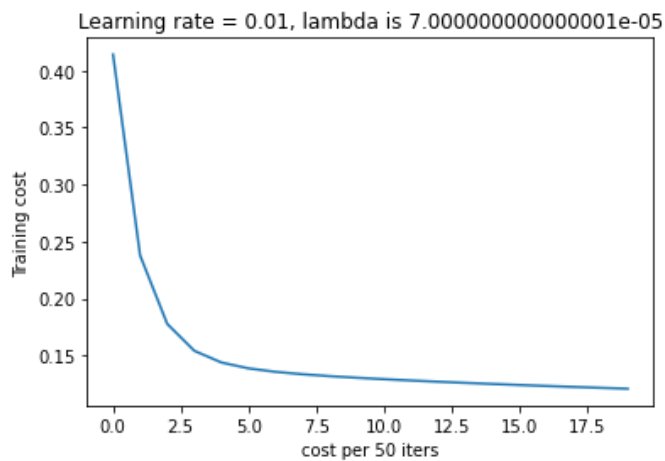
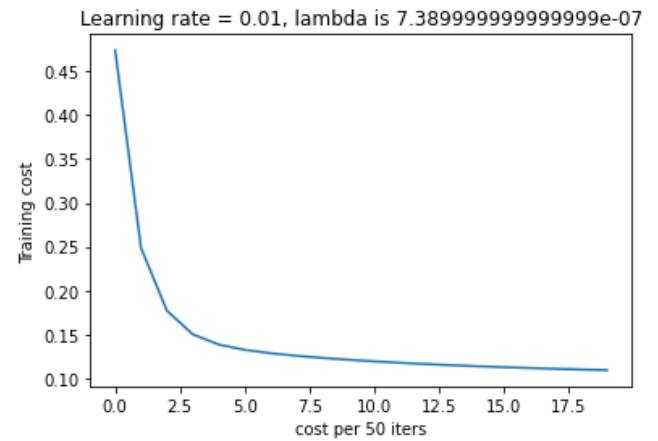
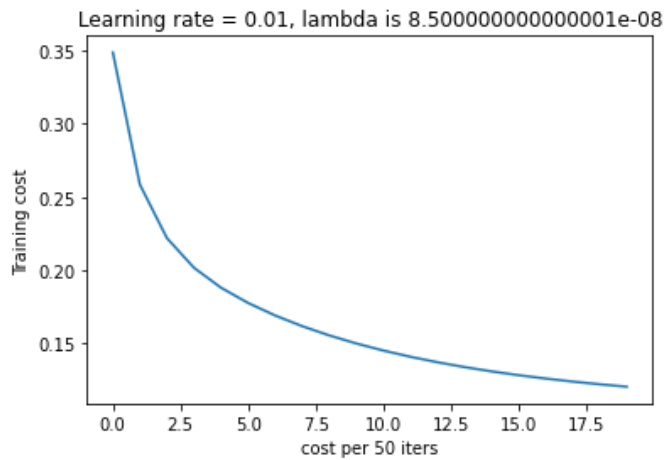
2. Stochastic Gradient Descent with Lasso regression



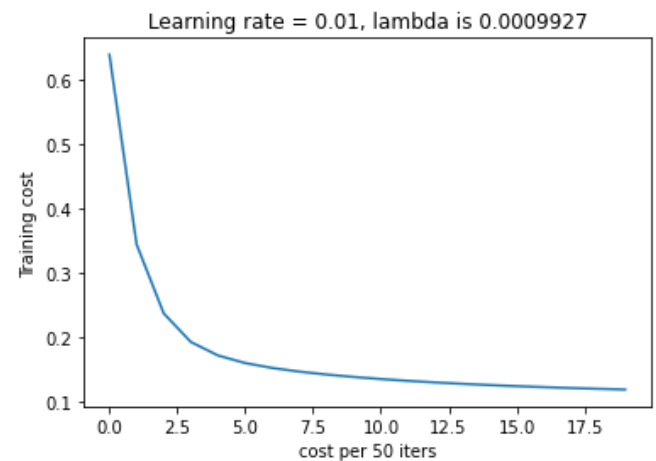
	Value of lambda	Training errors	Testing errors
0	8.500000e-08	0.102383	0.183663
1	7.390000e-07	0.102070	0.182055
2	7.000000e-05	0.104498	0.181308
3	4.570000e-04	0.108789	0.185555
4	9.927000e-04	0.117857	0.181904



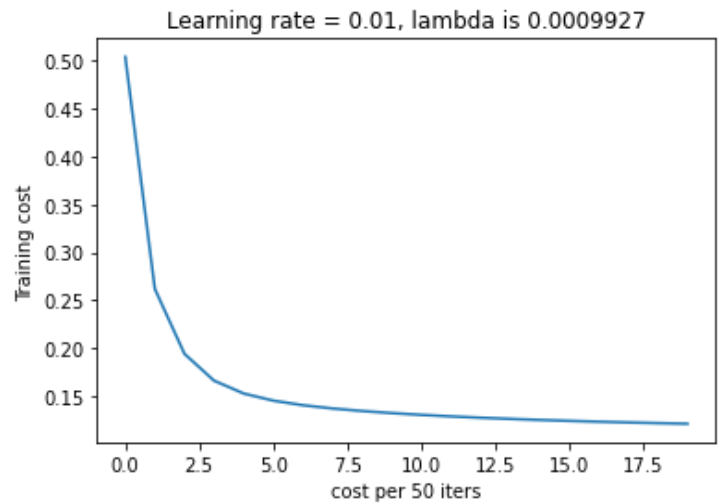
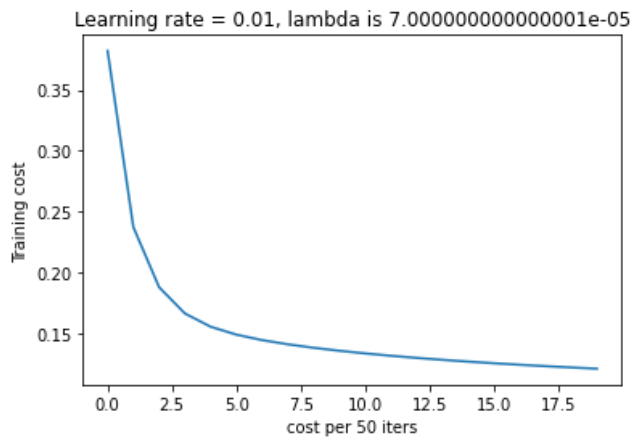
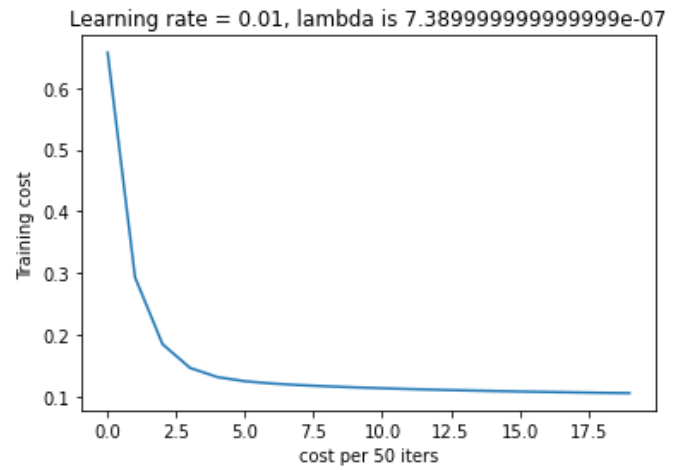
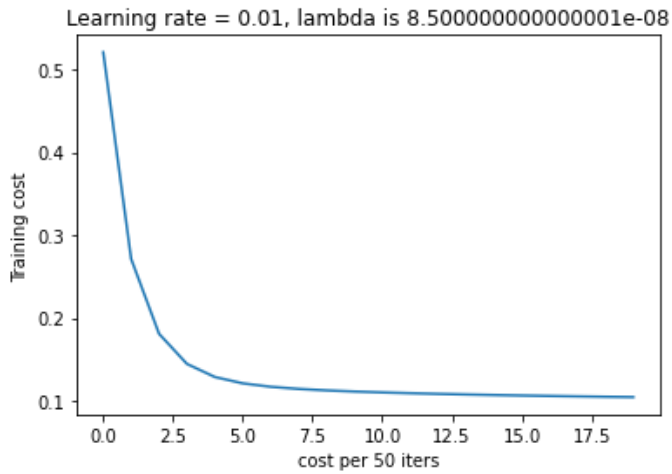
3. Gradient Descent with ridge regression



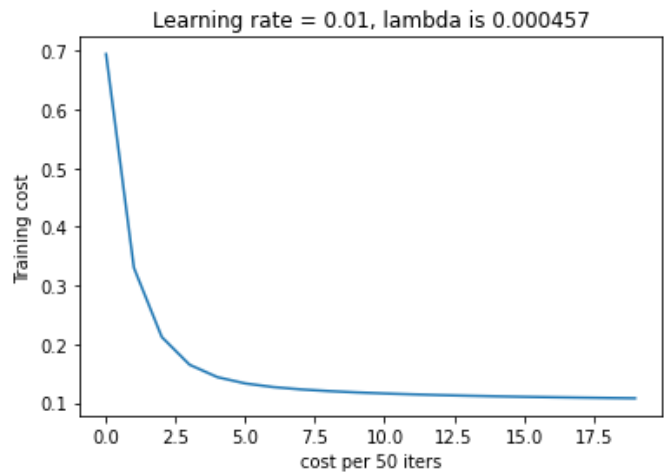
	Value of lambda	Training errors	Testing errors
0	8.500000e-08	0.120127	0.278749
1	7.390000e-07	0.109116	0.257151
2	7.000000e-05	0.120703	0.278030
3	4.570000e-04	0.111859	0.260228
4	9.927000e-04	0.119403	0.270107



4. Gradient descent with lasso regression

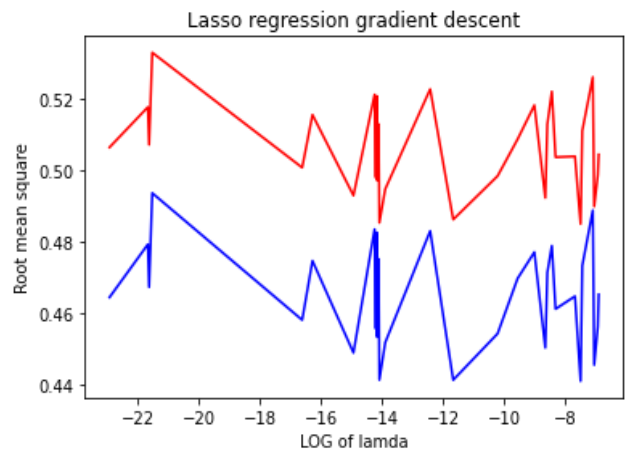
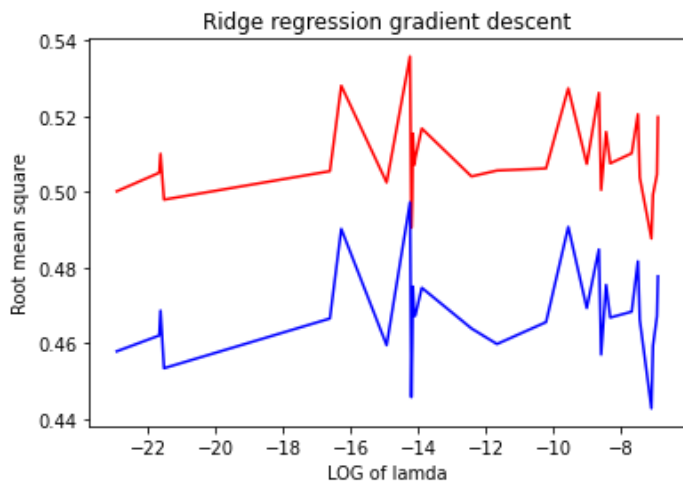
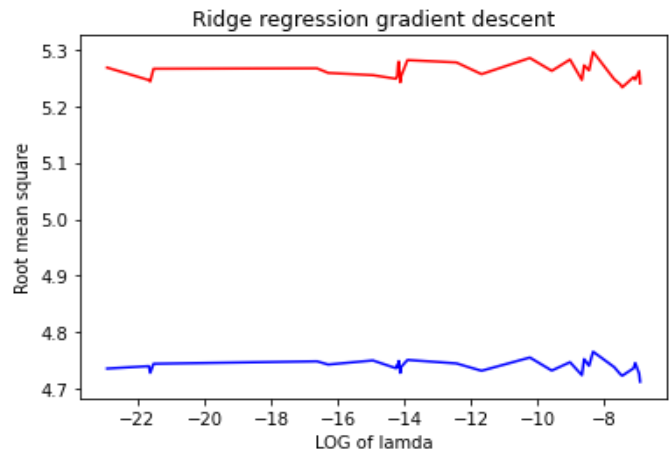
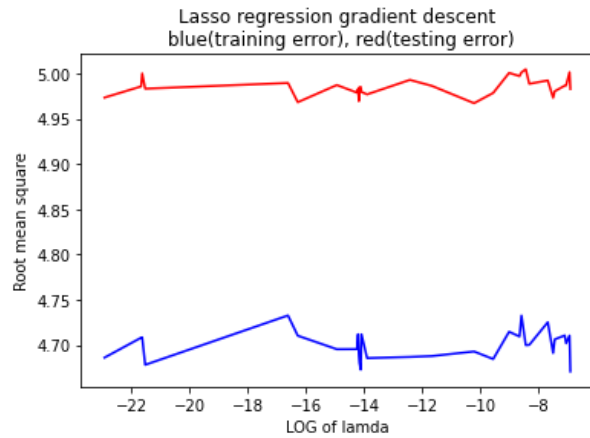


	Value of lambda	Training errors	Testing errors
0	8.500000e-08	0.106013	0.250773
1	7.390000e-07	0.115520	0.271849
2	7.000000e-05	0.102416	0.242093
3	4.570000e-04	0.110584	0.247294
4	9.927000e-04	0.120256	0.253820



In case of Ridge Regression gradient descent, the most appropriate lambda would be e^{-9} where the testing error is 0.49 and the training error is 0.44.

In case of Lasso Regression gradient descent, the most appropriate lambda would be e^{-9} where the testing error is 0.49 and the training error is also 0.49.



In part (a) the best model was with degree 7, where RMSE for testing was 0.2

In part(b) the best model was with lambda e^{-9} with RMSE 0.49.

A is better