Foundations of DataScience

Assignment 1

1	ANIRUDH A	2018B4A70936H
2	VISHAL KUMAR NK	2019A4PS0693H
3	SHIVANI THIRUNAGARI	2019A4PS0754H

Table of Contents

Objectives of the Assignment:

1. To implement Gradient-Descent algorithm and Stochastic Descent algorithm for polynomials of degree 0-9.
2. Implementation of regularisation using ridge regression and lasso regression for a polynomial of degree 9.
3. Visualization of surface plots for all the polynomial degrees.

Data-Preprocessing:

For pre-processing, the data was shuffled initially, and then normalised using min-max normalisation. Then the data was split into training and testing portions of 70:30 proportion.

Description of the Model:

Our method is about developing 10 different models of degrees from 0 to 9. Each model of degree d contains all polynomial combinations of the features with degree less than or equal to d as independent variables. The dataset contains 2 features, Strength and Temperature which are used to predict the pressure. The assumption of Strength as x1 and temperature as x2 has been considered. For a model of degree 1, we have three different terms in our feature matrix, for a model of degree 2, we have 6 different terms. The resulting model would be:

For a polynomial of degree 1: $y = w0 + w1x1 + w2x2$

For a polynomial of degree 2:

$y = w0 + w1x1 + w2x2 + w3x1^2 + w4x1x2 + w5x2^2$

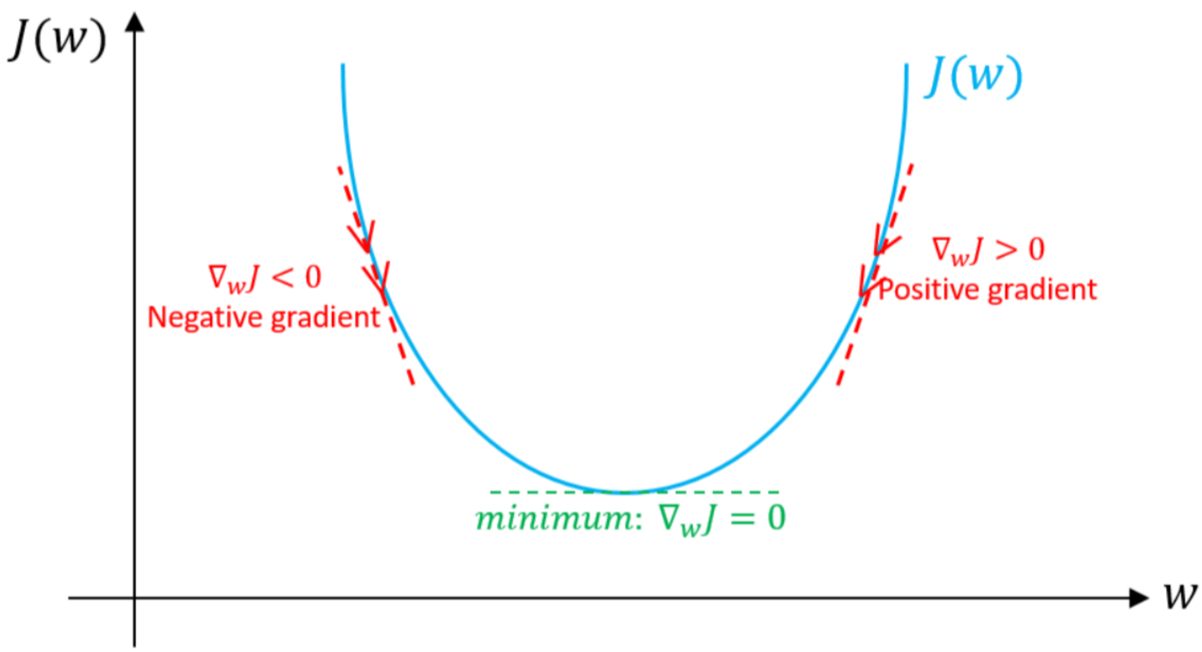
For a polynomial of degree n:

$$y = w0 + \Sigma\Sigma Wk(x1)^i(x2)^j \ where\ i + j <= n, \text{i,j belongs to (0,n)} ,$$

To generate the feature matrix, we have defined a function named featureVector with input parameters degree and the training data. This processes the featureVector taking the degree as input by using a nested-for loop using the principle that the sum of powers of the features x1, x2 will be less than or equal to that of the degree d. This helped us generate the featureVector as required for each degree of the polynomial.

Gradient Descent Algorithm:

Gradient Descent is an iterative approach used in optimization to predict the values of the coefficients of the features in order to minimize the cost function. By taking steps of sizes proportional to the negative gradient, the gradient descent iteratively finds the optimal value. This in fact is the global minimum since a convex function is being considered.



For the implementation of this algorithm, we define a function, gradientDescent(deg,eta), where deg stands for degree of the polynomial, eta is the learning rate, which has a while loop that runs till there convergence of the error values. The error values are computed using the function compute_cost(). We initialised the W, cost from previous iteration, error from this iteration. At each iteration of the loop, we calculate the predicted y by taking the dot product of the featureVector with respect to the degree given and current model. We then calculate the derivative by finding the dot product of the featureVector and the difference between the predicted y and the actual. The equations are vectorised to save computation time. Then we update our model(W) by subtracting the derivative found and the learning rate in each iteration of the current model. This is in accordance with the gradient descent algorithm.
After every 50 iterations, we make a note of the cost corresponding to the 50th iteration, and print them. We utilised these arrays to plot the required graphs after every 50ths iteration.

In order to obtain the models of degrees 0-9, we added a for-loop which gives the input of degree deg to the above mentioned gradientDescent() function. It plots the graphs for no. of iterations v/s cost for degrees 0-9.

Stochastic Gradient Descent Algorithm:

As the name suggests, there is some stochasticity involved in this algorithm and is almost the same as how gradient descent works. Here, we take any random set of values in each iteration to evaluate the gradient and find the global optimum. This algorithm shows a faster convergence than the normal Gradient Descent algorithm, thus helping in saving time to train the data.

In our model, we have defined the function Stochastic_GD(). We set the number of iterations to 2000 and inside the while-loop, we generate a random value of "i" in the range (number of rows) which selects a training data sample randomly. Here, we use the derivation value based on only this one observation and calculate the cost of each iteration. Then at a step of 50 iterations, we print the error to obtain its minimum value and plot the required graphs.

In addition to this, for the ridge and lasso forms of the Stochastic Gradient Descent algorithm, we have calculated the RMSE separately for training and testing data and plotted it against log lambda.

Finally, we incorporated a loop that gives the models for polynomials from degrees 0 to 9, along with their training and testing errors, RMSE, and minimum errors.

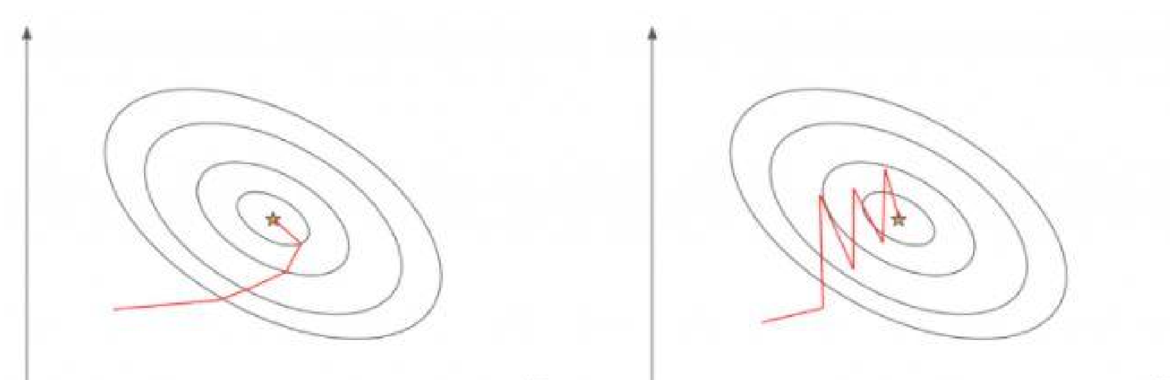


Fig: Comparison of Gradient Descent and Stochastic Gradient Descent algorithms.

REGULARIZATION:

Regularization is a technique to obtain different models of a set of data by changing the values of a parameter called "lambda".

Ridge regularization:

In the ridge regularization technique, the cost function gets modified by adding a square of the absolute value of the coefficients whole multiplied by lambda. Different values of lambda give us different models. The cost function **without regularisation** is:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2$$

The **cost function with ridge regularisation** is as follows:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

Thus, we have defined a new cost function called cost_function_ridge() that evaluates this cost and functions called gradientDescent_ridge() and Stochastic_GD_ridge() for the model. The regularization parameter lambda has been incorporated into the function to modify it accordingly (as $d/dx(lambda * (w ** 2) = 2 * lambda * w)$.
Again, after every 50 iterations, we print out the error values and plot the respective graphs. A plot of log($lambda$) vs RMSE has also been shown.

Lasso Regularization:

When the term added to the error function is lambda times the sum of absolute values of the coefficients, the regularization is known as lasso regularization. We use different values of lambda to get different models. The cost function using lasso regularization is:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

A function canned cost_function_lasso() has been defined to evaluate this cost and functions called gradientDescent_lasso() and Stochastic_GD_lasso() have been defined for the models.
The models have also been updated by adding lambda to the derivative(as $d/dx(lambda * abs(w)) = lambda)$.20 different values of lambda have been used to obtain the different models and RMSE for training and testing has been calculated

accordingly. At a step of 50 iterations, the relevant error values are printed and graphs have been plotted accordingly. A plot of log($lambda$) vs RMSE has also been shown.

Errors : Without Regularisation

(a) Gradient Descent

Degree of the polynomial	Minimum Training Error	Minimum Testing Error
0	66.123709285827	30.16854042476644
1	13.934476949748674	6.603623992313004
2	15.923729583766763	7.497449964102895
3	15.8131670343265	7.325443698852643
4	15.004249176538965	7.020358370894085
5	14.788130681679464	6.970498463852234
6	14.838305223907017	7.027752353628535
7	15.026731673159844	7.130305854345865
8	15.269975209673264	7.239002932405782
9	15.488068604155732	7.318209202503303

Conclusion: For polynomial regression using gradient descent, the minimum training and testing error can be observed for a polynomial of degree 1. Hence, a linear polynomial best fits the model considering a learning rate of 0.1.

(b) Stochastic Gradient Descent

Degree of the polynomial	Minimum Training Error	Minimum Testing Error

0	172.49577394231972	68.87068590149865
1	505.5911585695828	202.02435222053836
2	1014.2494176270469	405.27239778460444
3	1744.750570541063	696.8266506430223
4	2656.345692944137	1060.8236235596203
5	3840.921899039334	1533.536072399309
6	5209.762445168131	2080.200727712149
7	6821.695159972138	2723.8676409864975
8	8617.624574971656	3440.208864367946
9	10719.874652655957	4280.705715149436

Conclusion: For polynomial regression using stochastic gradient descent, the minimum training and testing error can be observed for a polynomial of degree 0. However, this is a case of underfitting the model and hence regularization is required.
Hence, minimum training and testing errors have been tabulated for polynomials of degree 0-9 using gradient descent and stochastic gradient descent.

Errors : With Regularization

In this section, we tabulate the minimum training and testing errors along with the RMSE values for 20 different values of lambda taken from the range 0 to 0.1. A polynomial of degree 9 has been considered.

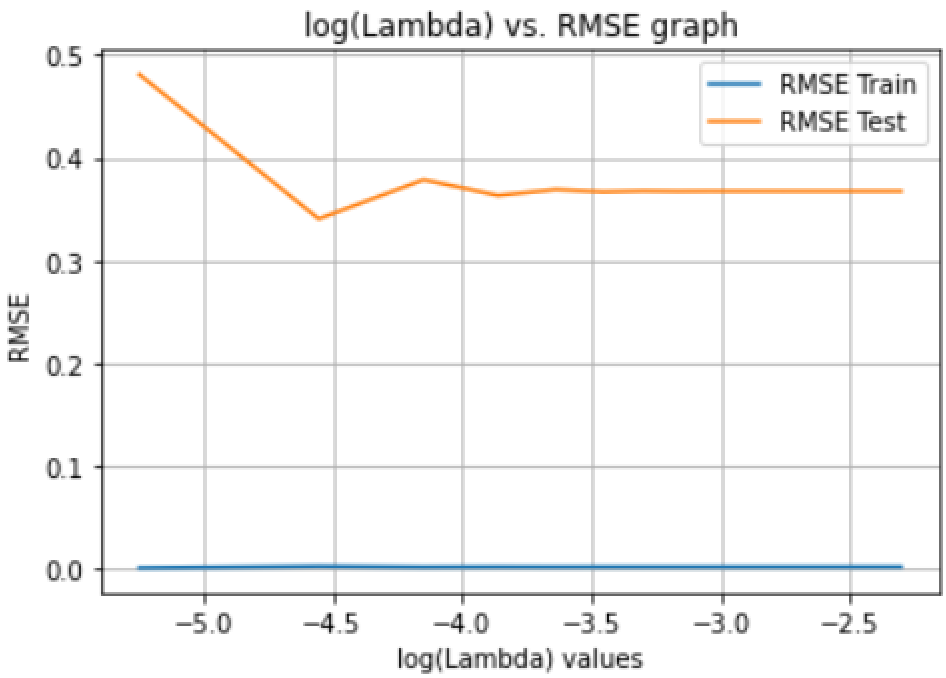
(a) Gradient Descent with Ridge Regularisation

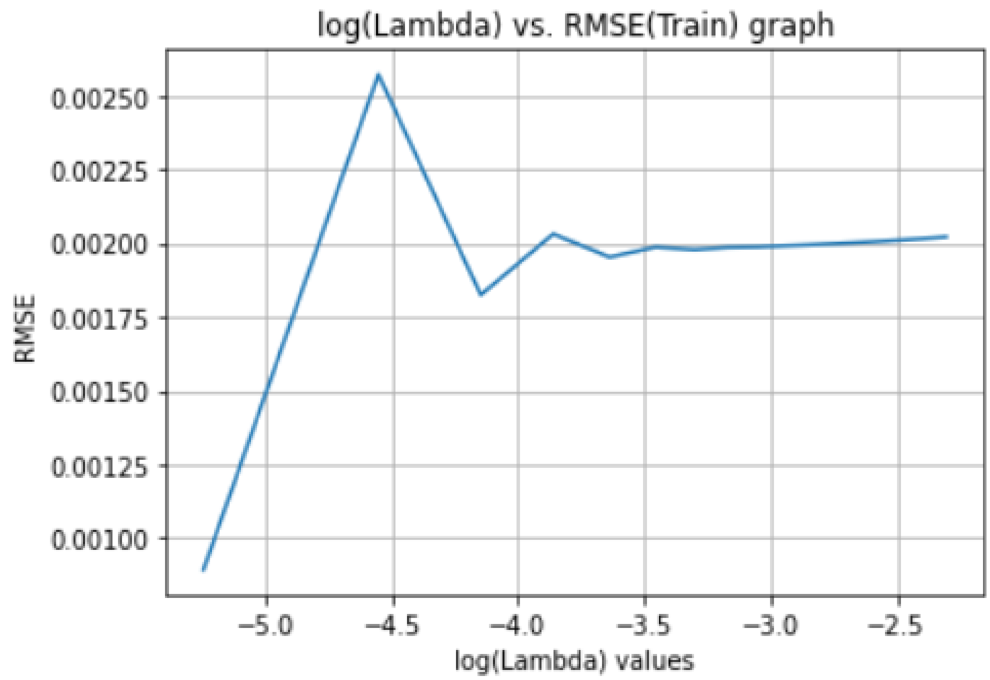
Penalty (Lambda)	RMSE training	RMSE testing	Minimum Training Error	Minimum Testing Error

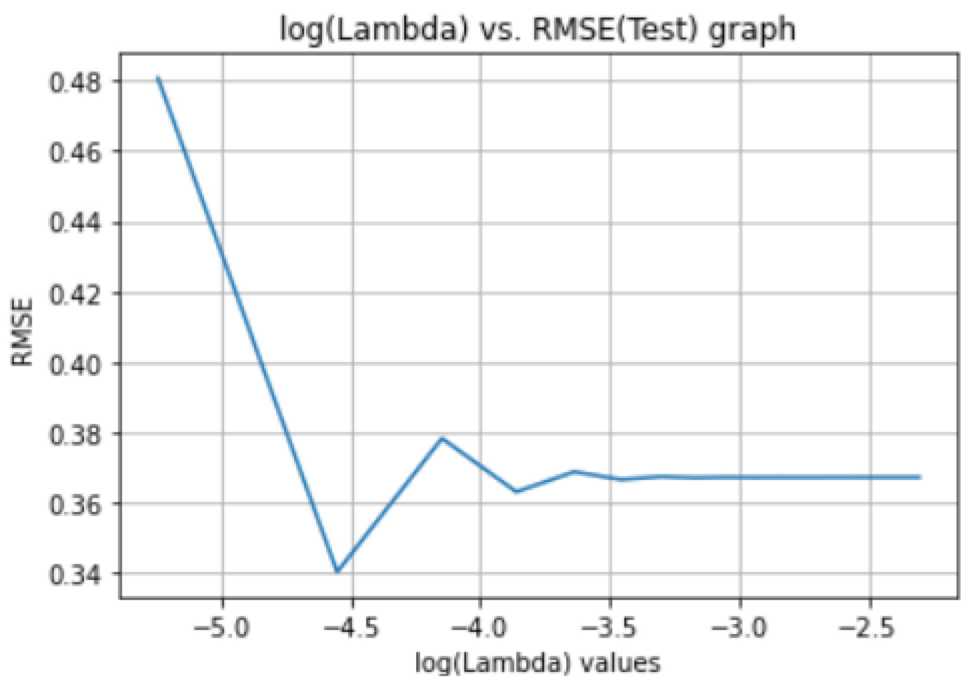
0.	0.002959822941240777	0.35760481865054716	0.2956569659605091	0.12600365423793083
0.00526316	0.001909373502703186	0.3684403634332047	0.00015060001484298665	1.0401124771370023
0.01052632	0.0019703225973550977	0.36509443129325453	0.0024166652773281398	0.5237960626876018
0.01578947	0.001955514808122437	0.36610076924320234	0.0018000993817877764	0.646088522640739
0.02105263	0.0019722150063530485	0.3657992721107745	0.0029511220958116867	0.5964863922354798
0.02631579	0.001965529492154176	0.36589343223881976	0.0033878592948978726	0.61569816918168
0.03157895	0.001969184098542376,	0.365868135930915	0.004183249017209606	0.6092338384659439
0.03684211	0.0019722150063530485	0.36587892464305294	0.004811752148001734	0.6126445409642288
0.04210526	0.001975458793790997	0.36587881308386744	0.005513332846471726	0.6122990868465671
0.04736842	0.0019786621926774876	0.36588199215254397	0.0061805865869152595	0.6133883123980713
0.05263158	0.0019819016686619404	0.36588417574806503	0.006860713451772674	0.6139302338734418
0.05789474	0.0019851539377191297	0.3658866583597918	0.007533543048285398	0.6146808109633879
0.06315789	0.001988425892378603	0.36588904903751834	0.008190525576160615	0.6153516029894314
0.06842105	0.001991715296886712	0.3658914658514681	0.008837916037052584	0.6160526456461533
0.07368421	0.0019950226772641414	0.3658938731467393	0.009483526978979806	0.6167419612337254
0.07894737	0.001998347724542667	0.36589628169242516	0.010126897645738252	0.6174355633067768
0.08421053	0.002001690382781046	0.3658986882382593	0.0107684029671625	0.6181273435303892
0.08947368	0.0020050505199608873	0.3659010937674944	0.0114080291126534	0.6188196321533176

0.09473684	0.0020084280 27543965	0.3659034979 84992	0.0120458945 47349233	0.6195115405 700207
0.1	0.0020118227 903673353	0.3659059009 816345	0.0126820463 541897	0.6202034079 488877

Conclusion: For ridge regression using gradient descent, the minimum RMSE training error can be observed for a lambda value of 0.00526316. Hence, among the 20 chosen values, this best fits the model considering a learning rate of 0.1.







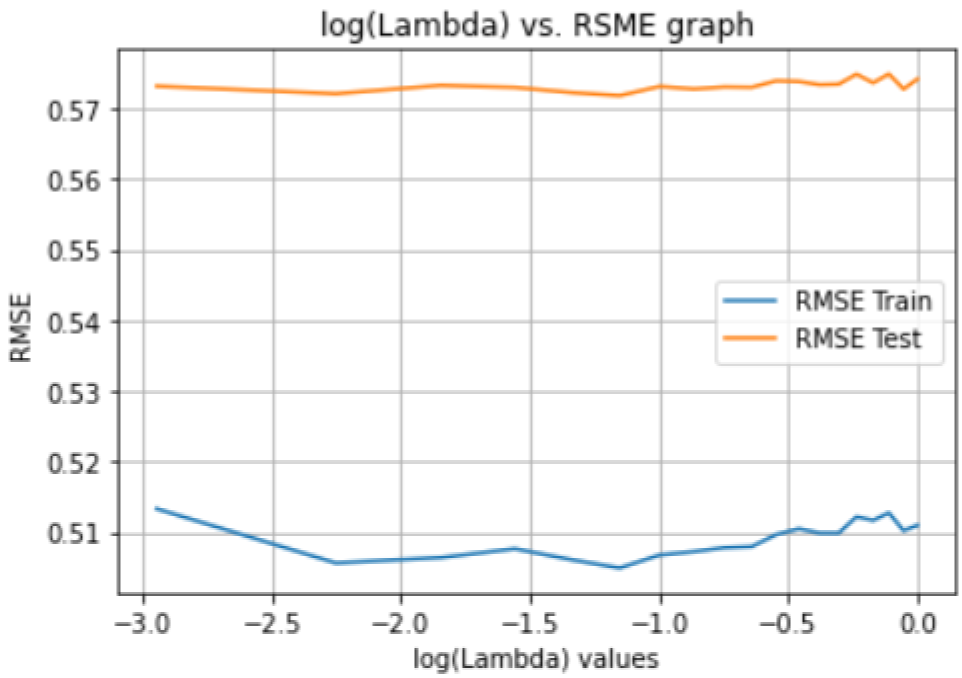
(b) Stochastic Gradient Descent with Ridge Regularisation

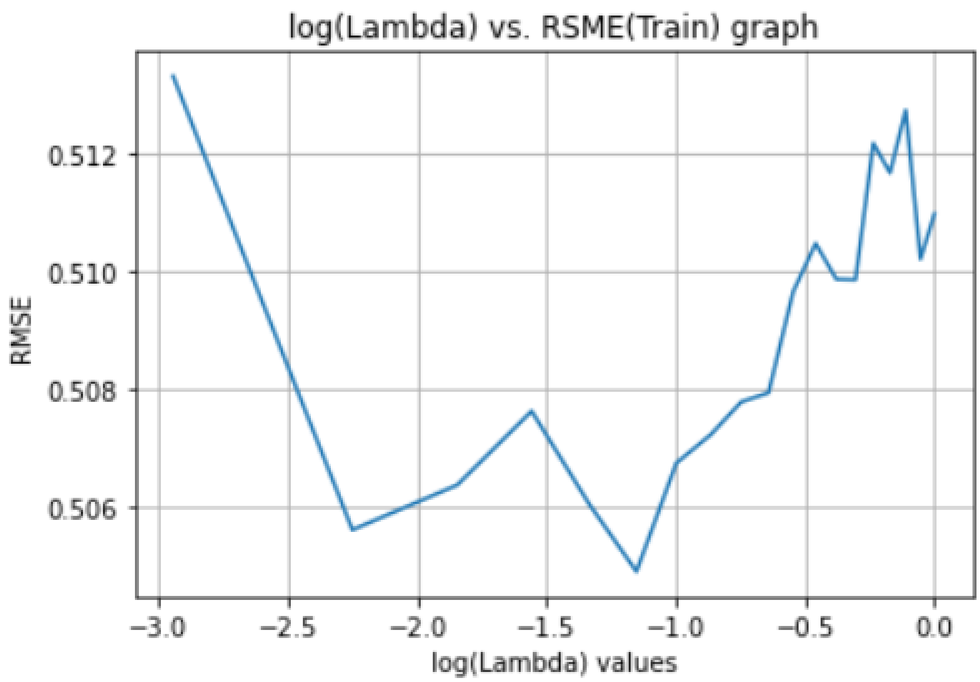
Penalty (Lambda)	RMSE training	RMSE testing	Minimum Training Error	Minimum Testing Error

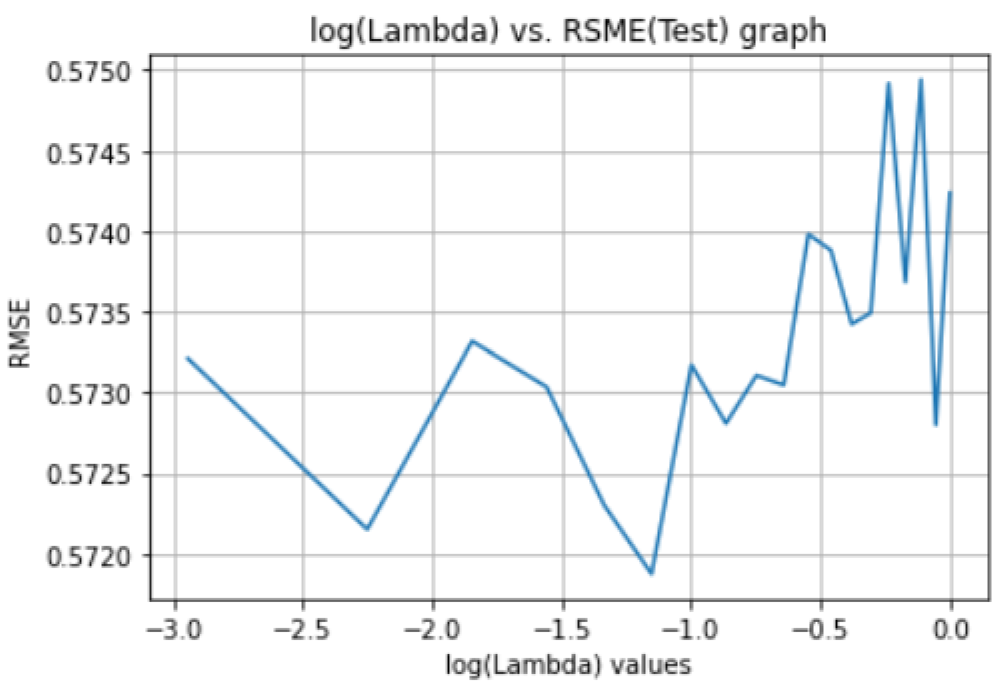
0.	0.3232768691952058	0.5801618925605911	60.35278290721532	83.30548584092301
0.05263158	0.5133023860872065	0.5732109503410885	152.18456478337836	81.35146200013483
0.10526316	0.5056077355131638	0.572149823961949	147.68611240596823	81.07785941621476
0.15789474	0.5063686973676841	0.5733167911728955	148.16170303757212	81.43735807846046
0.21052632	0.5076226090082823	0.5730342057181997	148.91832215845494	81.380118620204
0.26315789	0.5060418874375047	0.5722979712831239	148.02392630097566	81.20108457647581
0.31578947	0.5049006014115437	0.5718727135462962	147.38206377600548	81.10510140539554
0.36842105	0.5067399172613061	0.5731709484527802	148.47485513676614	81.49524919731253
0.42105263	0.5072243742868127	0.5728080092852189	148.79343827509572	81.42320282136674
0.47368421	0.5077792145670067	0.5731058607398886	149.1326697855303	81.52427957721714
0.52631579	0.5079321256243928	0.5730464811495236	149.25196862393378	81.5344422901432
0.57894737	0.5096470545723275	0.5739806918233357	150.27569721241758	81.81560183717255
0.63157895	0.5104660022976055	0.573880436860377	150.77903553550217	81.81009731829869
0.68421053	0.509859299069625	0.5734221587205475	150.4294506315116	81.69872483154745
0.73684211	0.509850185138831	0.5734914730858356	150.44701769433834	81.73100591486116
0.78947368	0.5121651060743913	0.5749189128282666	151.82301822626334	82.1568911799266
0.84210526	0.5116635764025407	0.5736843882813348	151.5476188386171	81.83002377077828
0.89473684	0.512727389603675	0.5749415800875519	152.1666701861574	82.19511341956066

0.94736842	0.5102028772773332	0.5727993065560243	150.75497420038852	81.63220935453127
1.	0.5109688868212708	0.574237567728231	151.18856307683987	82.03179798151098

Conclusion: For ridge regression using stochastic gradient descent, the minimum RMSE training error can be observed for a lambda value of 0.31578947. Hence, among the 20 chosen values, this best fits the model considering a learning rate of 0.1.







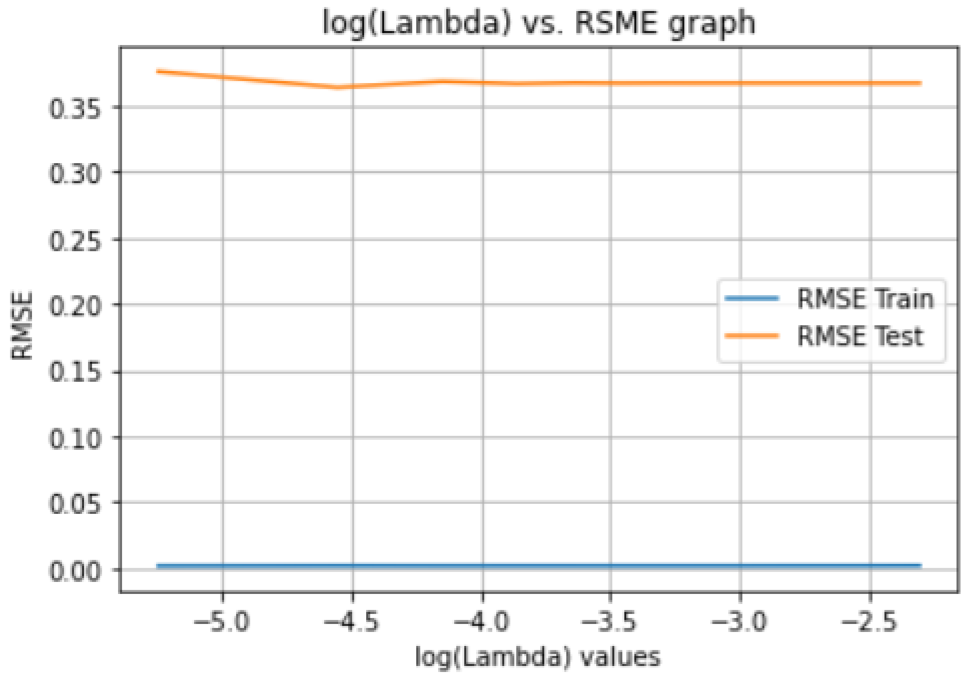
(c) Gradient Descent with Lasso Regularisation

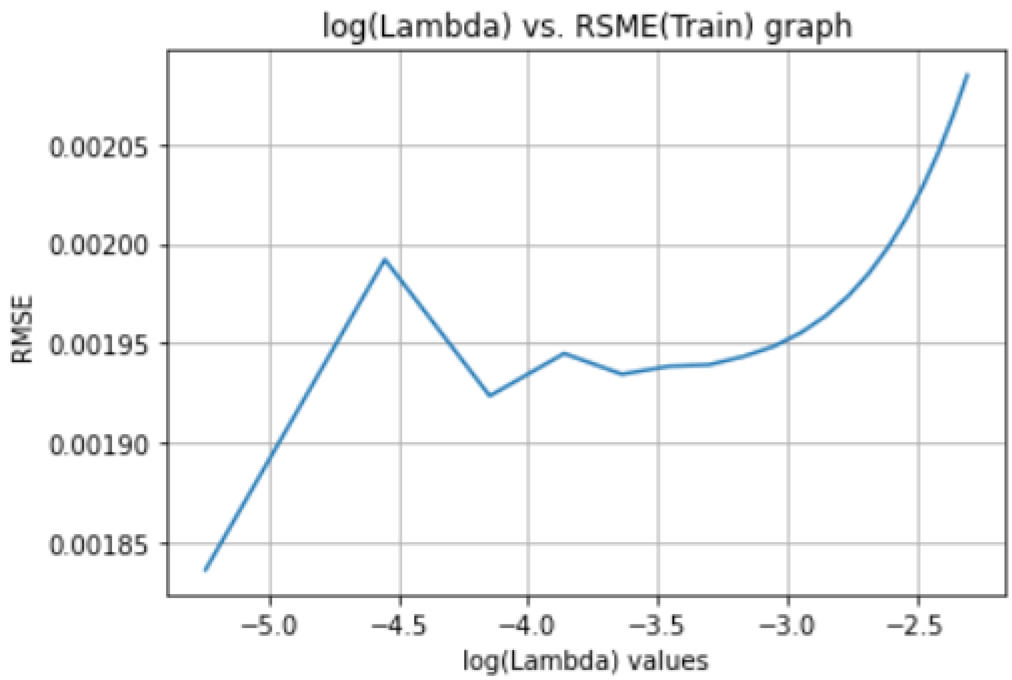
Penalty (Lambda)	RMSE training	RMSE testing	Minimum Training	Minimum Testing Error

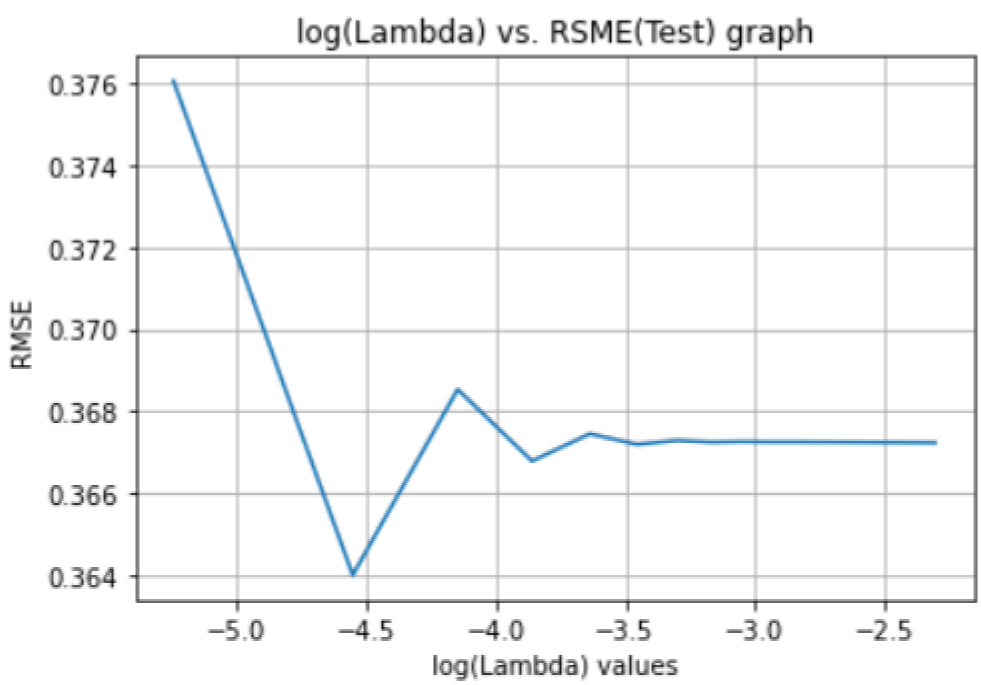
			Error	
0.	0.003062643977942353	0.34575968491936665	9.848777542407888e-05	0.5379739187199286
0.00526316	0.0018366181083689377	0.37604341902425015	0.002583219481021006	0.6388867396987381
0.01052632	0.001992077049222626	0.36401183034140117	0.0055670366465625405	0.6017961255796137
0.01578947	0.0019237002315991177	0.36853677568395765	0.008033052045218604	0.6191812931499923
0.02105263	0.0019449803282015718	0.3667907306991781	0.010798534147236865	0.616168293758994
0.02631579	0.0019345989732409054	0.3674544695968518	0.013380950886727451	0.6209441953384277
0.03157895	0.001938568668056153	0.36719691704425356	0.016028059941361732	0.6227396919229335
0.03684211	0.0019392957886821467	0.36729234182214276	0.01862024948476774	0.6256472499945015
0.04210526	0.0019434752073149173	0.36725263958019383	0.021215504709511602	0.6281111009565452
0.04736842	0.0019485449437108041	0.36726470195362226	0.023813583776360534	0.6307488429435586
0.05263158	0.0019554636527468155	0.3672569512785462	0.026423061163295913	0.6333324180441312
0.05789474	0.0019638452919350886	0.36725680116480713	0.02904598981339794	0.6359545055940562
0.06315789	0.0019738102829525445	0.36725375140648203	0.03166678826007826	0.6385648116758342
0.06842105	0.0019852820636579284	0.36725182413378665	0.03429431337952899	0.6411854897415419
0.07368421	0.001998254656945496	0.3672494785423012	0.03694728981596327	0.6438301707915508
0.07894737	0.002012691479534199	0.3672473049267678	0.03961576893489343	0.6464908575933642
0.08421053	0.0020285641312360234	0.36724507712345367	0.04228856960197177	0.6491556213627605

0.08947368	0.0020458381097653096	0.3672428817895564	0.04498631362023679	0.6518453703706788
0.09473684	0.0020644786482884895	0.36724068571834345	0.04769143897295541	0.6545424328269744
0.1	0.002084448929944153	0.3672385016347565	0.05040336885151298	0.6572462739876614

Conclusion: For lasso regression using gradient descent, the minimum RMSE training error can be observed for a lambda value of 0.00526316. Hence, among the 20 chosen values, this best fits the model considering a learning rate of 0.1.







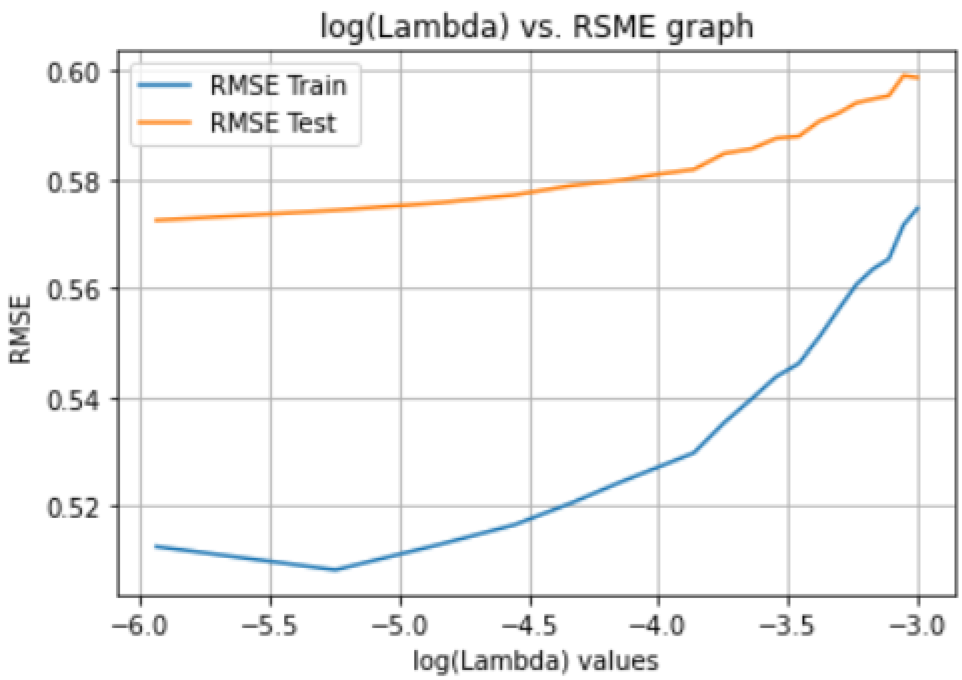
(d) Stochastic Gradient with Lasso Regularisation

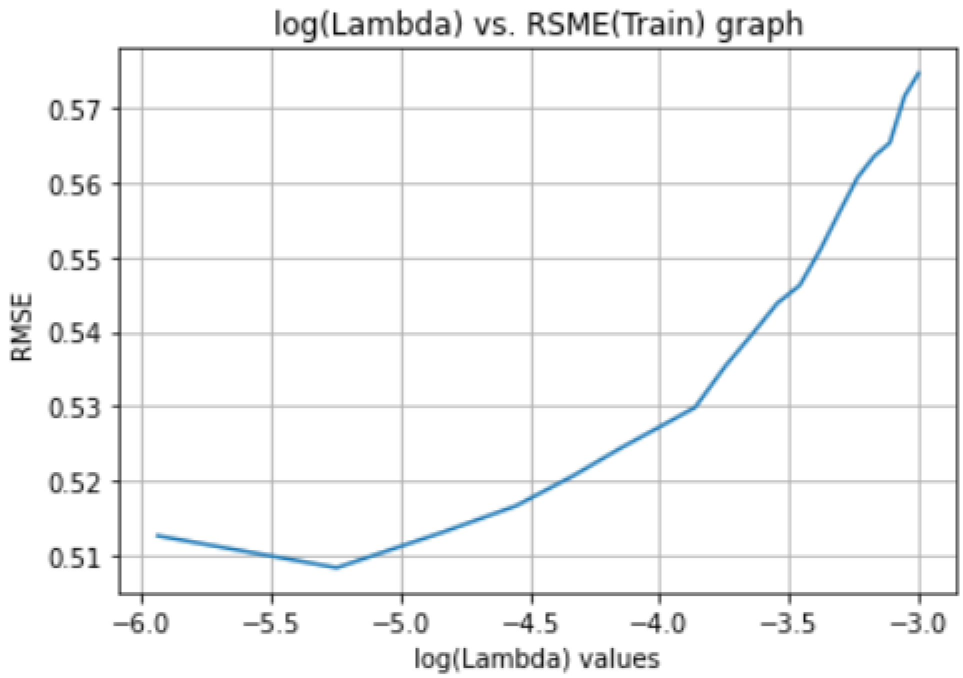
Penalty (Lambda)	RMSE training	RMSE testing	Minimum Training Error	Minimum Testing Error

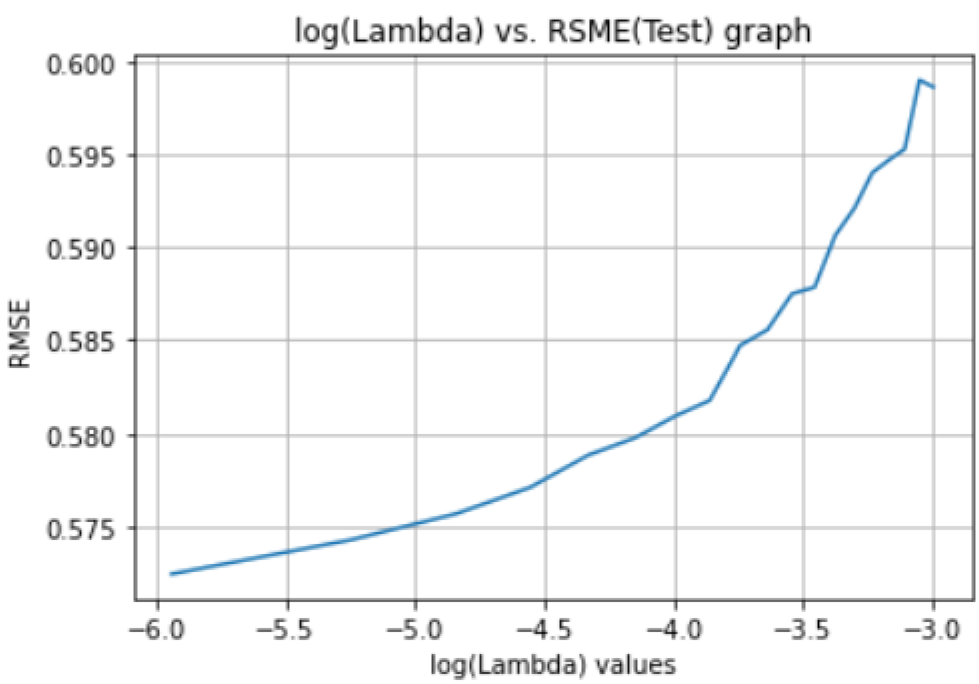
0.	0.32355923926517616	0.5803662019220462	60.45881070875387	83.36416976252178
0.00263158	0.5126226503275542	0.5724393996735007	151.8350520372166	81.18733491462798
0.00526316	0.5083369546323548	0.5742787645689261	149.38749288898376	81.78635340962273
0.00789474	0.5130456540204075	0.5756781051740151	152.23726567062573	82.25292326729101
0.01052632	0.5166281490541821	0.5771072609331792	154.4359340606776	82.72906760137305
0.01315789	0.5206975855628655	0.5788364899324561	156.933840979512	83.2858796886147
0.01578947	0.5243869239295973	0.5797738050902412	159.21166859157248	83.60879360062474
0.01842105	0.5272919615968767	0.5809425061700967	161.0264283631437	83.99813549870642
0.02105263	0.5298319976487207	0.5817851714035112	162.6385239933143	84.2970967834184
0.02368421	0.5353958071816015	0.5847247445283491	166.08311806380135	85.17359067217849
0.02631579	0.5397340608160104	0.5855632400812566	168.8270881988225	85.46660289887699
0.02894737	0.5438205394533024	0.5875041399208681	171.33562468079967	86.05712128988652
0.03157895	0.5462263556633338	0.5878547426013558	172.96920954330903	86.22075068071929
0.03421053	0.5510436297913769	0.5906514322700394	176.0354297238667	87.05243890752412
0.03684211	0.5560987053700288	0.5921169035045885	179.28252203953446	87.52083901744456
0.03947368	0.5606629329049276	0.5940231556331397	182.27418284211404	88.09938498806912
0.04210526	0.5635170907325913	0.5947093607485944	184.0031004471282	88.37200078283331
0.04473684	0.5653364922766713	0.5952999646097221	185.4176176163358	88.59131481008075

0.04736842	0.571604380344485	0.599016521852475	189.5746784004326	89.69534447411218
0.05	0.5746684597675228	0.5986349064360172	191.53741533107453	89.64822834073384

Conclusion: For lasso regression using stochastic gradient descent, the minimum RMSE training error can be observed for a lambda value of close to 0. Hence, among the 20 chosen values, this best fits the model considering a learning rate of 0.1.



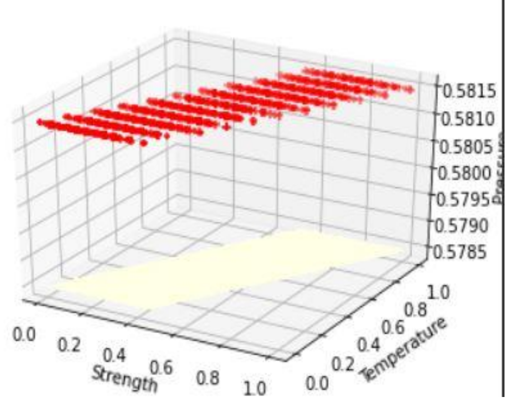

log(Lambda) vs. RSME(Train) graph
RMSE
0.57
0.56
0.55
0.54
0.53
0.52
0.51
−6.0
−5.5
−5.0
−4.5
−4.0
−3.5
−3.0
log(Lambda) values


log(Lambda) vs. RSME(Test) graph
RMSE
0.600
0.595
0.590
0.585
0.580
0.575
−6.0
−5.5
−5.0
−4.5
−4.0
−3.5
−3.0
log(Lambda) values

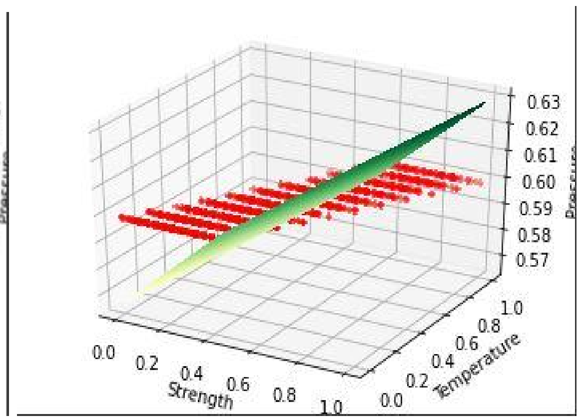
Surface Plots

In this section, surface plots have been drawn for polynomial regression with degrees 0-9 based on the gradient descent algorithm.

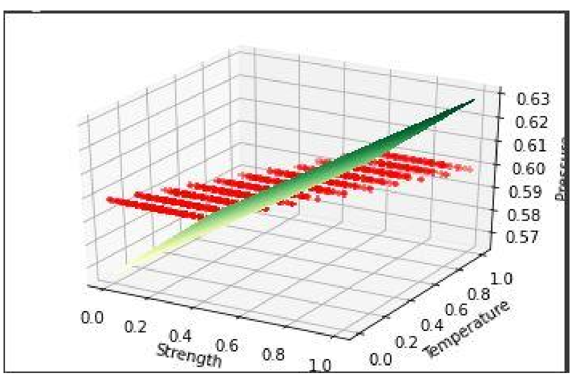
Degree 0



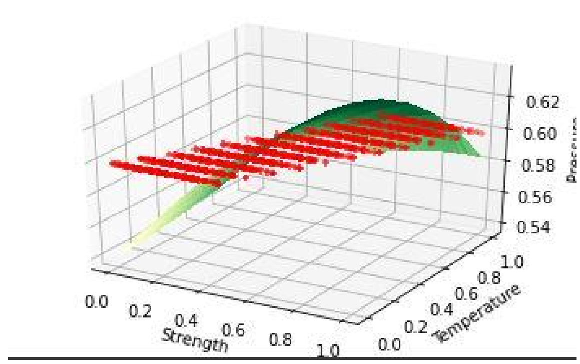
Degree 1



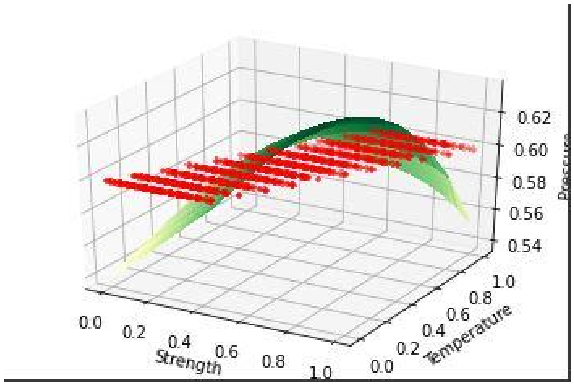
Degree 2



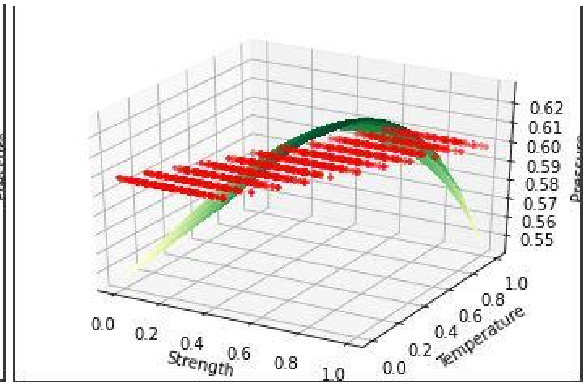
Degree 3



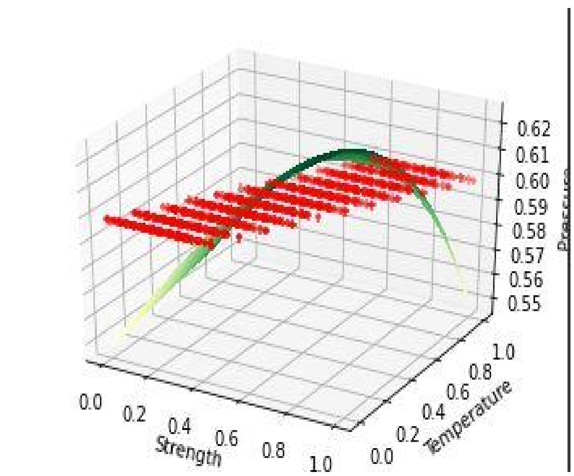
Degree 4



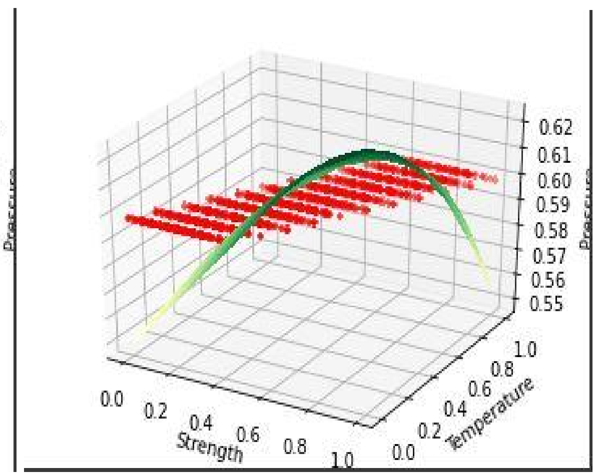
Degree 5



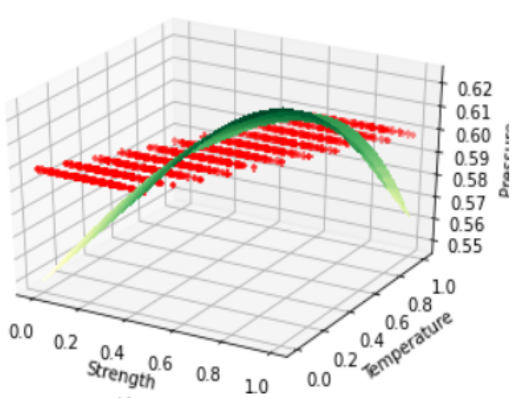
Degree 6



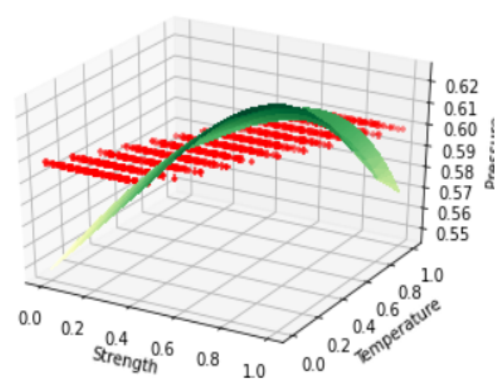
Degree 7



Degree 8



Degree 9



How does overfitting work?

For the model described above, we can clearly see that the model covers more data points as the number of degrees of the polynomial increases. For the zeroth-degree polynomial, the model looks flat but as the number of the degrees increases, the model tries to fit more data points in order to reduce the training error but this results in an increase in testing error. This is called overfitting and mainly happens when we have a large number of features but less number of data points.

Compare between the best model obtained in part a) and the best model obtained in part b).

- **Gradient Descent: -**
  Without regularization, the testing error values for gradient descent algorithm are in the range of 6.5 to 7.5. However, after regularization, the testing error values (in both ridge and lasso regressions) can be seen to be in the range of 0.5-0.7. Hence, the model based on regularization proves to have a better predictive ability.

- **Stochastic Gradient Descent: -**
  Without regularization, the testing error values for stochastic gradient descent algorithm are above 1000. However, after regularization, the testing error values (in both ridge and lasso regressions) can be seen to be around 80. Hence, the model based on regularization proves to have a better predictive ability.

Conclusion: -

The models built using regularization better predict the values of pressure given the values of strength and temperature applied to a certain piece of plastic. Regularization decreases the training and testing error, thereby building a more reliable model.