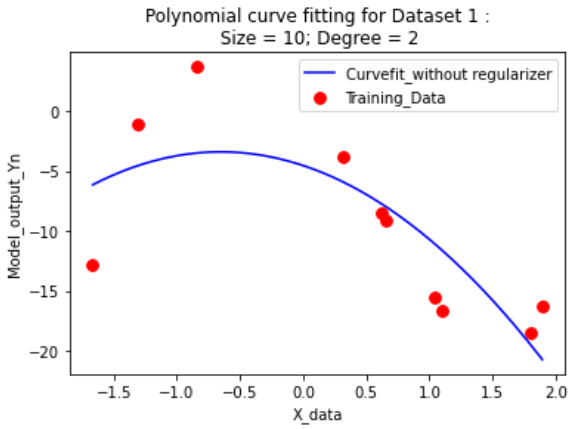
TASK 1:Polynomial curve fitting

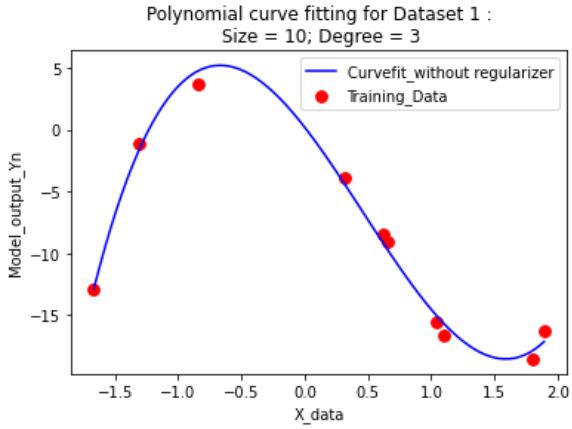
***Dataset 1**: 1-dimensional (Univariate) input data*: **function_2.csv**

Approximated functions obtained using training datasets of different sizes (10 and 200) for different model complexities (degrees 2, 3, 6 and 9) and different values of λ.

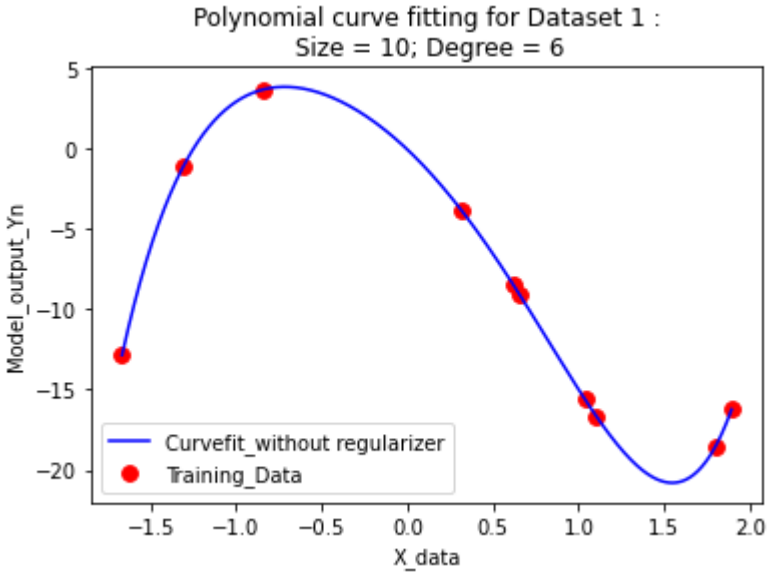
Without regularization: 10 training points



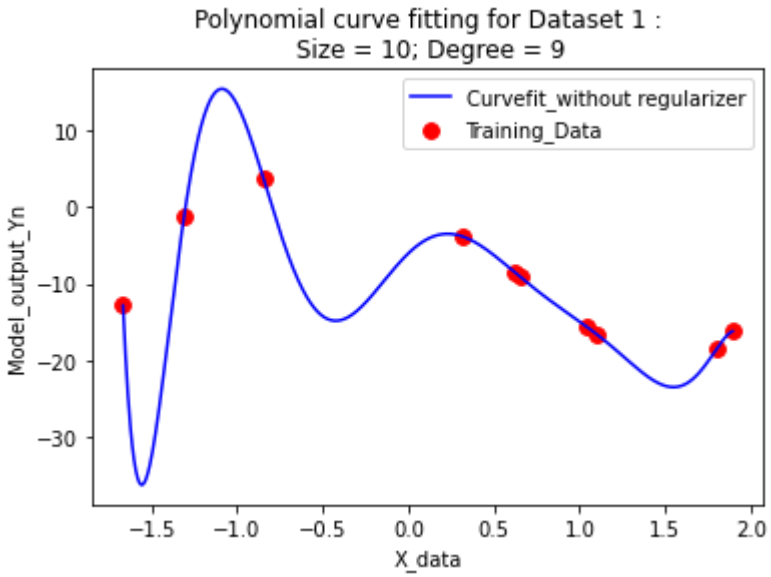
Plot 1.1



Plot 1.2



Plot 1.3

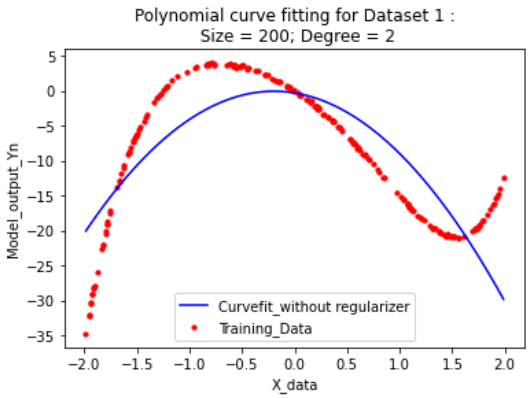


Plot 1.4

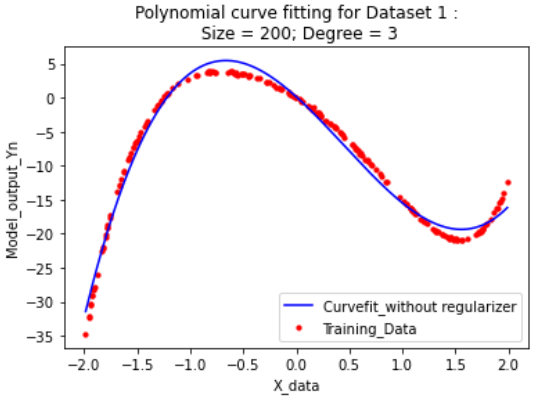
Inferences: (Size-10, no regularization)

- Here, we can see that a low complexity(degree) of 2 is not sufficient to fit the training points well. But Degree 3 performs significantly better but still doesn't pass through a few points.
- And degree 6 fits the curve amazingly well and passes through all the training points!
- A high complexity of 9, with just 10 training points, implies that our curve does pass through all the training points as evidenced by the graph, but there might be overfitting here, as we have not provided sufficient training points to train the 9th-degree model.

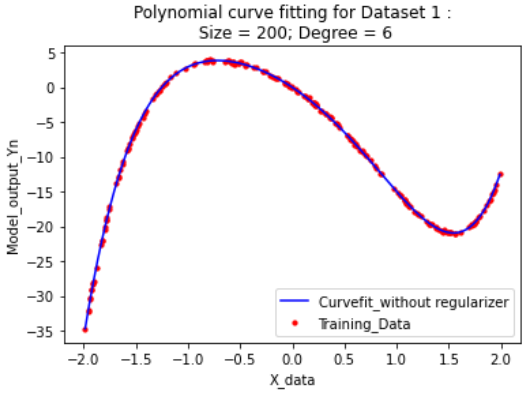
Without regularization: 200 training points



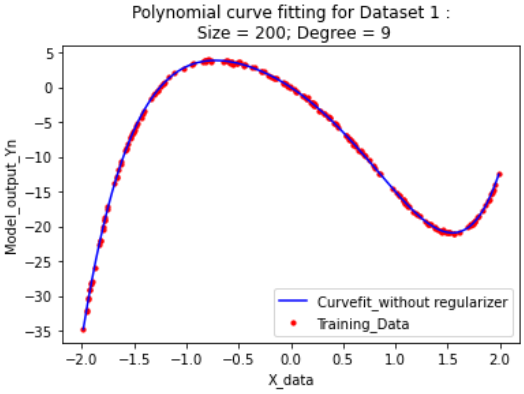
Plot 1.5



Plot 1.6



Plot 1.7

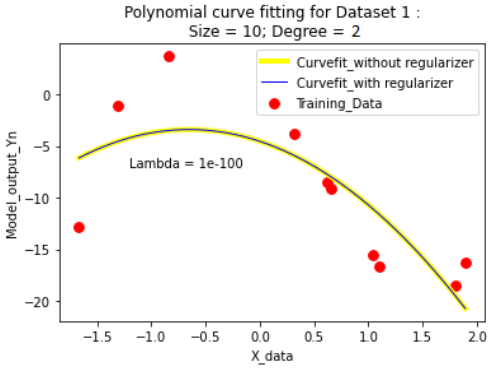


Plot 1.8

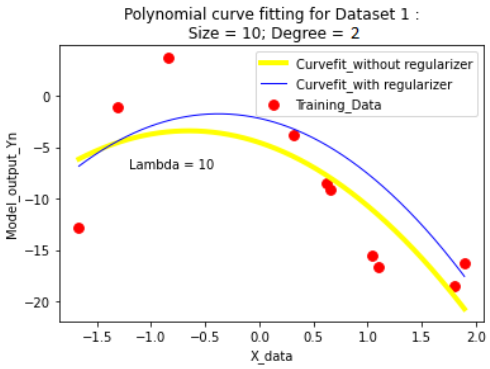
Inferences: (Size-200, no regularization)

- Here, again the degree-2 model is not able to approximate the training data well. While the degree-3 model performs okay, like the previous scenario.
- And again degree 6, fits the curve amazingly well and passes through all the training points! Here the degree-9 model also is able to perform very well and doesn't seem to overfit as we have given a lot of training points, and it resembles the degree-6 model, more or less.

With regularization: 10 training points

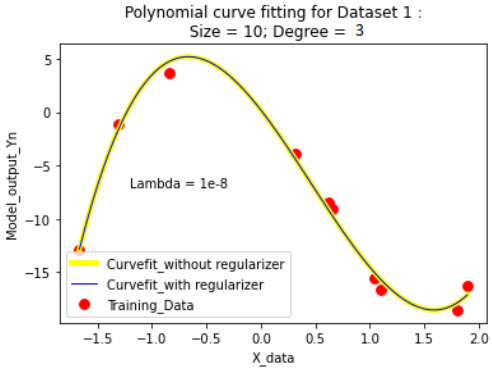


Plot 1.9

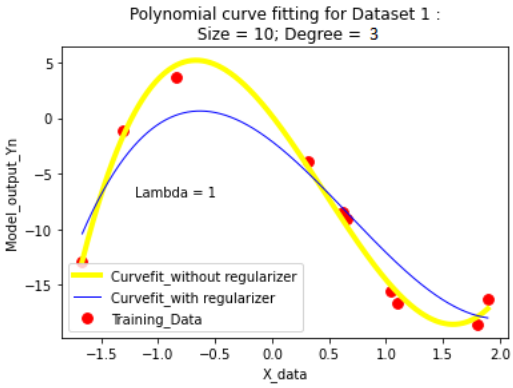


Plot 1.10

- For a very small Lambda, the regularized curve is almost the same as the non-regularized curve!
- For significantly large Lambdas, the regularized curve deviates from the non-regularized curve.
- Note that we cannot expect regularization alone to help us get a better fit! If we don't have a sufficiently complex model and a large enough training data set, we cannot fit the training points accurately irrespective of regularization!

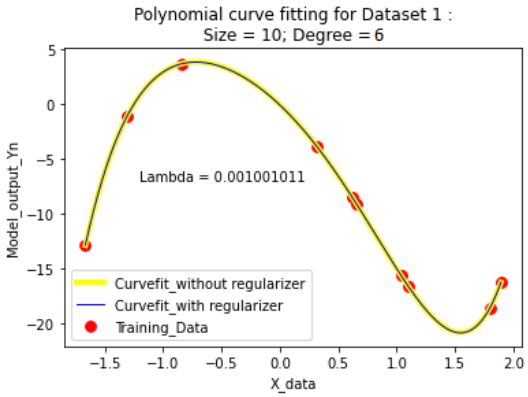


Plot 1.11

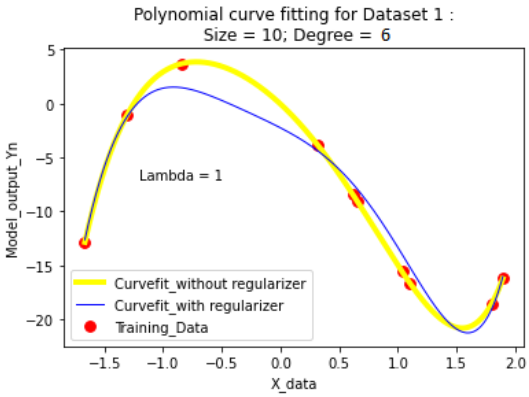


Plot 1.12

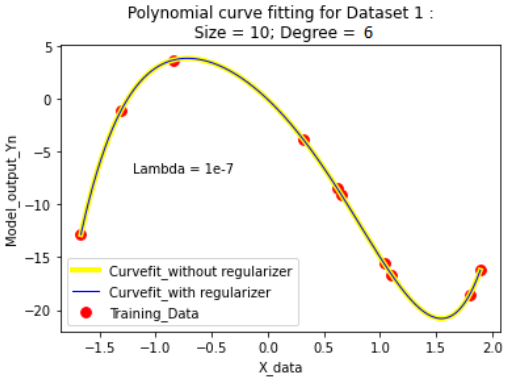
Again, for very small Lambda values, the effect of regularization is negligible, and for very large lambda values, we underfit the model and lose out on the information contained in the training set!



Plot 1.13



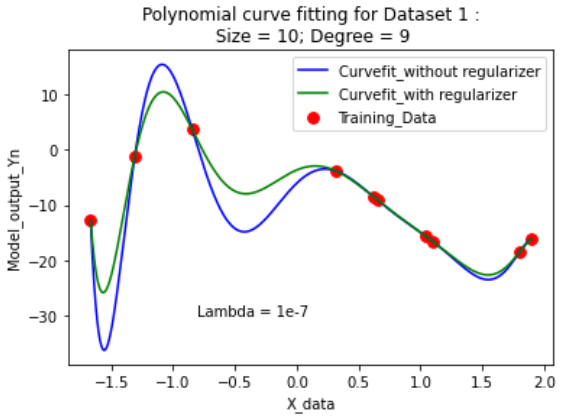
Plot 1.14



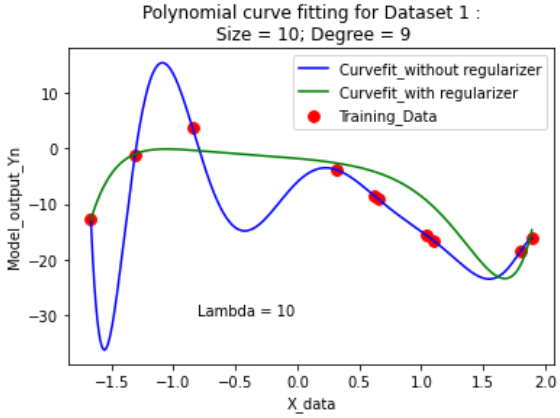
Plot 1.15

Seems like degree 6 is the actual degree of the curve! And it fits the training data perfectly without any regularization, And with moderate lambda and very low lambdas, the curve remains almost the same, indicating that the **non regularized curve** is smooth(Lambda effect) while also having low training errors!

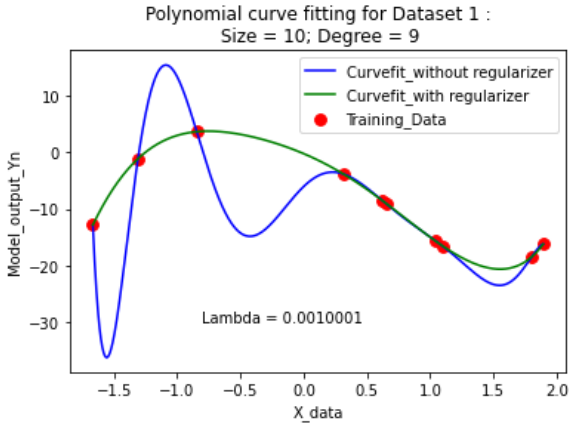
Although with a very large lambda, we start to lose information and it under-fits the training points!



Plot 1.16



Plot 1.17

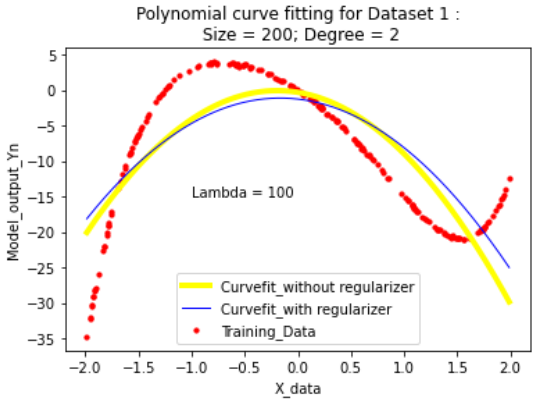


Plot 1.18

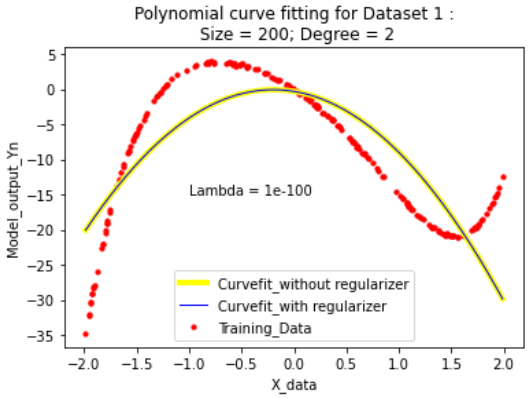
This case makes us appreciate the importance of regularization perfectly! When we have a few training data points and try to fit a complex model using them, we usually tend to overfit the training data! Regularization penalizes the fluctuations/roughness of the fitted curve. In this case, we can see that the regularized curve is much smoother than each of the non-regularized curves!

And the value of the regularization parameter determines how good we fit the curve. When we get the Lambda values just right as in **Plot 1.18.** We get a very neat curve, that perfectly fits all the training points, while also remaining smooth, generalizing very well! While, a very low lambda value, means that the curve is still overfitting the training data, as there is not enough importance given to the regularization. And in the case of extremely high lambda values, we underfit the training points and it doesn't even pass through all the training points!

With regularization: 200 training points

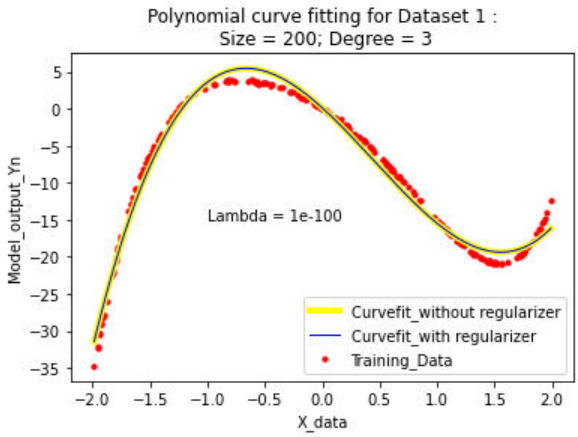


Plot 1.19

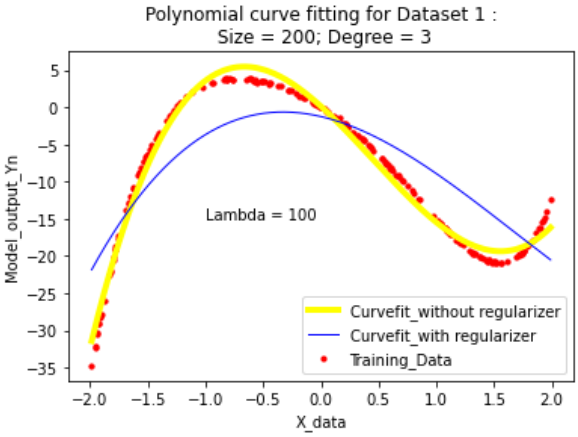


Plot 1.20

In the case of 200 training points, we have sufficient training points to fit a model of degree (approximately) up to 20 well! So, choosing a low complexity model such as degree=2 implies that we are not using the training data sufficiently well to make an appropriate model out of it. Here regularization too cannot help in fitting better, as we do not have sufficient flexibility to start with! Here we underfit the model, due to a lack of model complexity and the training data has come from a complex distribution.

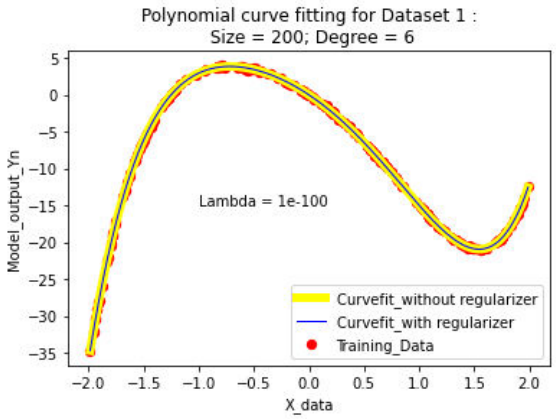


Plot 1.21

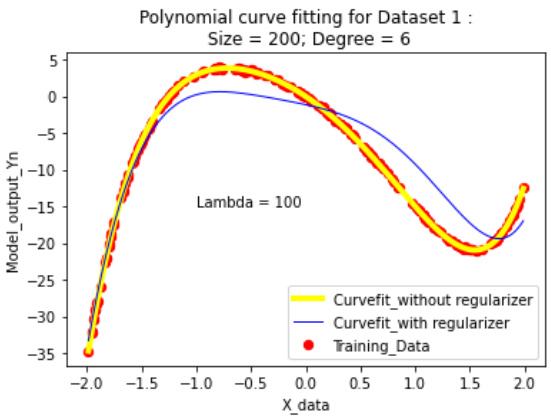


Plot 1.22

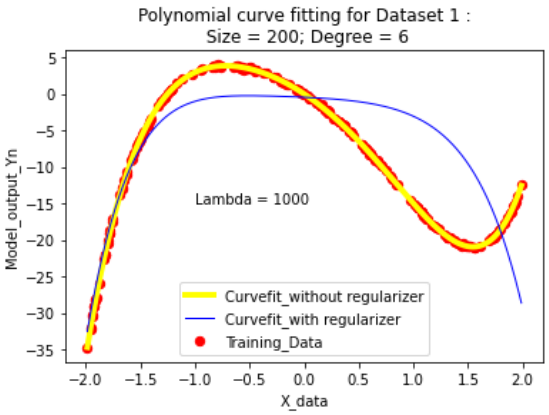
Degree 3, does significantly better! Still, it lacks the flexibility to fit the training points completely! When our model under-fits the data, regularization cannot help us improve our model! Here, we see that a very high lambda value implies we further underfit the data!



Plot 1.23

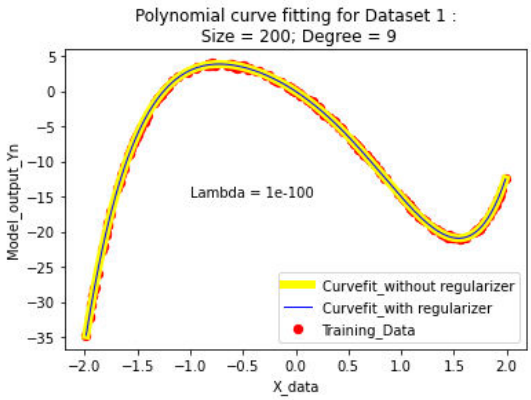


Plot 1.24

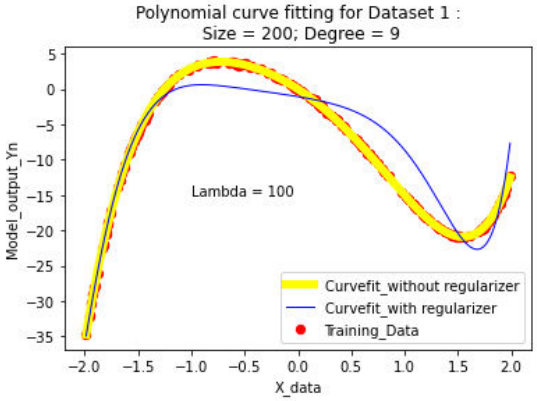


Plot 1.25

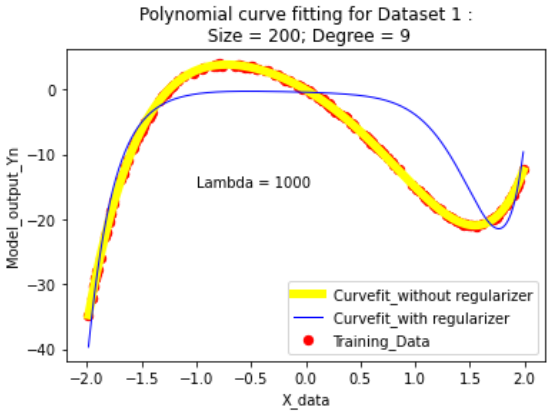
As we had seen earlier, the training points are actually from a curve of degree 6, and the no-regularization model fits them perfectly. A moderate regularization value curve will more or less coincide with the unregularized curve, so does a very low regularization value curve. When we increase the lambda values further and further we start underfitting the training points as seen from **Plots 1.24 and 1.25.**



Plot 1.26



Plot 1.27



Plot 1.28

For degree=9, now that we have enough training points(200), we do not overfit the training data like when we had only 10 points! Here, the non-regularized curve itself is really good and fits all the training points while also being pretty smooth. The curves with appropriate lambda values also will more or less coincide with the non-regularized curve. Once we start putting extremely large Lambda values, we begin underfitting the training data!

E_{RMS} Table Task 1:

DEGREE	SIZE	LAMBDA (λ)	Erms_train	Erms_vald	Erms_test
Degree = 2	Size = 10	0	4.257861	6.82934	6.71979
		1e-05	4.25786	6.82935	6.7198
		0.00101099	4.25786	6.82911	6.71956
		2.602	4.38651	**6.61277**	6.4942
		10	4.97738	6.97817	6.79214
	Size = 200	0	5.553233	5.14930	5.7277603
		1e-05	5.55323	**5.1493**	5.72776
		0.00101099	5.55323	5.14931	5.72776
		7.60762	5.55639	5.17892	5.7252
		32.032	5.59501	5.28739	5.7431
Degree = 3	Size = 10	0	0.714142	1.48161	1.44513
		1e-05	0.714143	**1.48160**	1.44515
		0.00101099	0.714154	1.48345	1.4466
		0.650654	1.82618	3.20657	3.11779
		1	2.29203	3.83104	3.74027
	Size = 200	0	1.255532	1.15778	1.24176
		1e-05	1.25553	1.15779	1.24177
		0.01002	1.25553	1.15742	1.24157
		1.20121	1.27316	**1.13584**	1.23948
		2.30231	1.31541	1.15209	1.27037
Degree = 6	Size = 10	0	0.0285123	0.123553	0.1323651
		1e-05	0.0285125	0.123512	0.0285125
		0.00101099	0.0299723	**0.121001**	0.0299723
		0.970971	1.1028	1.60045	1.1028
		1	1.12152	1.61236	1.12152
	Size = 200	0	0.0973563	0.100985	0.103180
		1e-05	0.0973564	0.100986	0.10318
		0.0150249	0.0974568	**0.100862**	0.103014
		0.970971	0.248849	0.269296	0.26403
		1	0.253927	0.275009	0.269549
Degree = 9	Size = 10	0	1.058e-05	122.9542	110.9337
		1e-05	0.0177835	3.86715	3.48627
		0.00101099	0.117265	**0.292113**	1.96296
		0.994995	1.08961	2.45197	2.32044
		1	1.09288	2.44347	2.31361
	Size = 200	0	0.09631739	0.1019634	0.1030177
		1e-05	0.0963174	**0.101964**	0.103018
		0.00101099	0.0963186	0.101971	0.103003
		0.900902	0.234391	0.258741	0.26104
		1	0.249599	0.27565	0.261201

Table 1.1

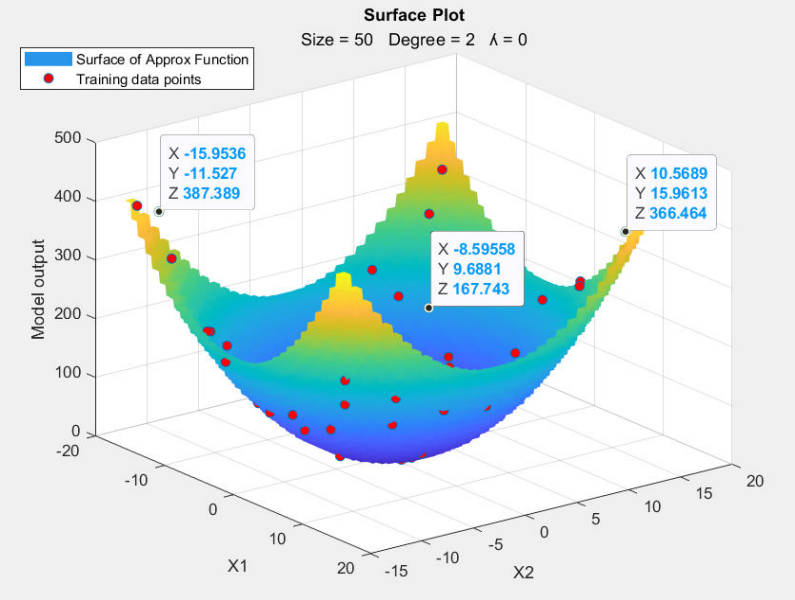
- The highlighted values in the **Erms_vald** column denotes the Lambda values for which we got the minimum validation error! These will be lambda values that will get chosen if we iterate over a large range of values.
- As a general trend, we can observe that the training errors are considerably small, and the training and validation errors are more or less similar.
- The only model where regularization has improved the model drastically over the non-regularized model is the **degree 9, Size 10 model.** In this model, we can clearly see how much effect the addition of appropriate Lambda values has on reducing the test error of prediction!

Conclusion:

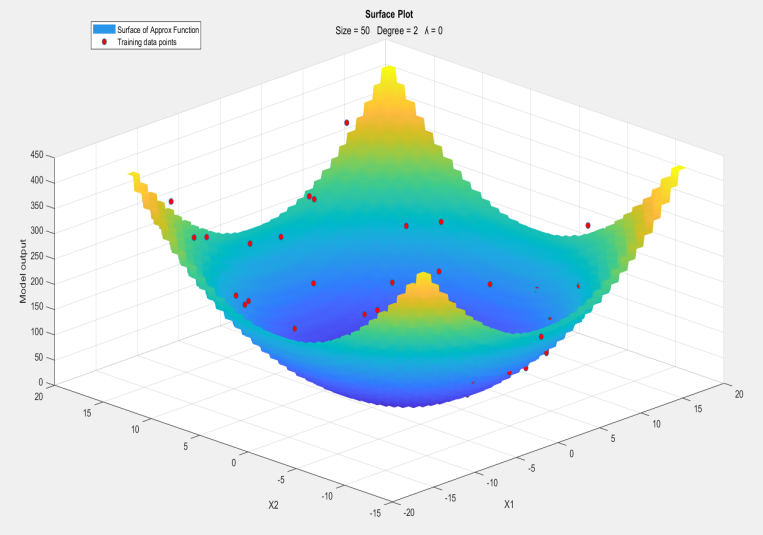
We conclude that the given data points were taken from a 6th degree curve. It can be observed that the curve with 10 points for degree 6, has comparable training and test errors!Which means it generalizes to all the points based on the information contained in the 10 training points alone! Also, from Plot 1.3 we see that it passes through all the training points!

TASK 2: Linear regression using polynomial basis functions

Note: Here, for 3D plots, we had to export the data onto an excel sheet from python variable explorer, then imported it and plotted it using Matlab as the Python surface plots were not good enough.

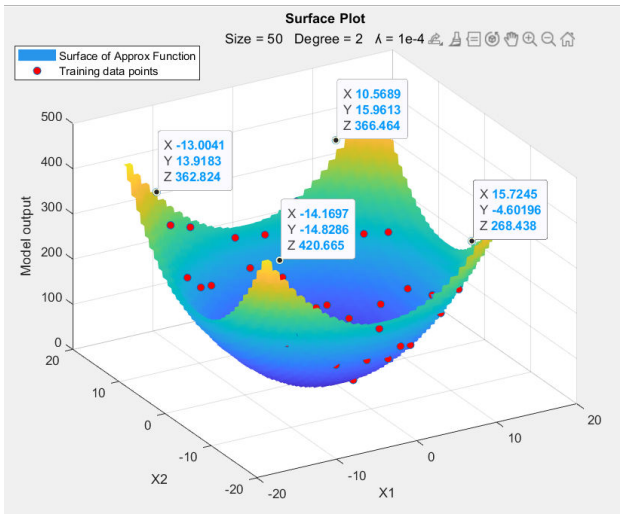


Plot 2.1a

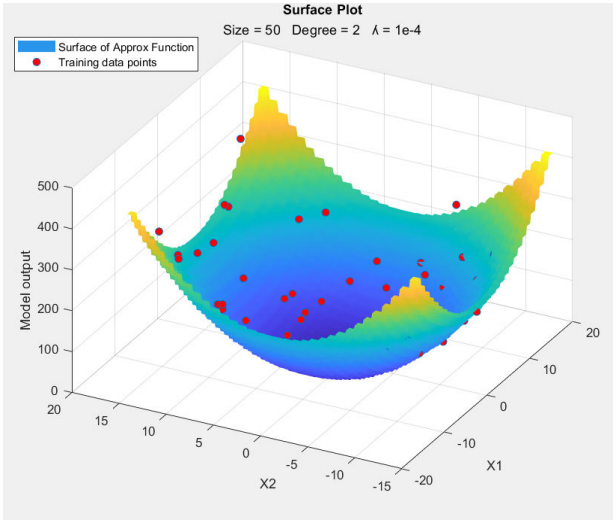


Plot2.1b

Non-regularized model of **degree 2** and **size 50** itself seems to fit the training data points fairly well!

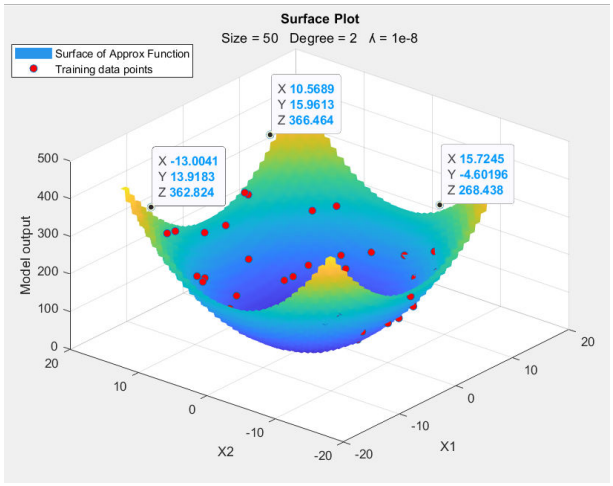


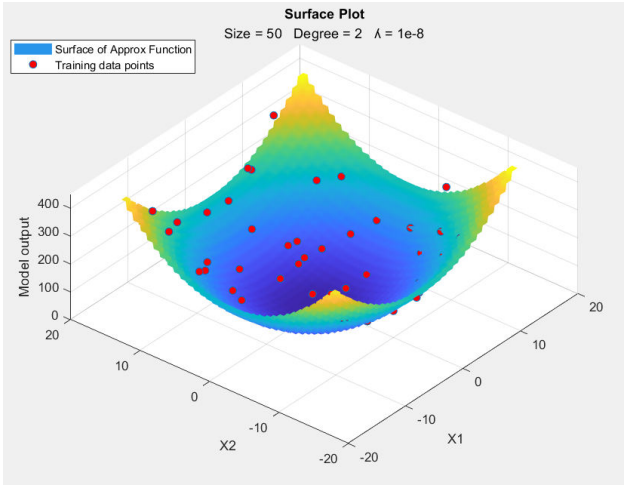
Plot 2.2a

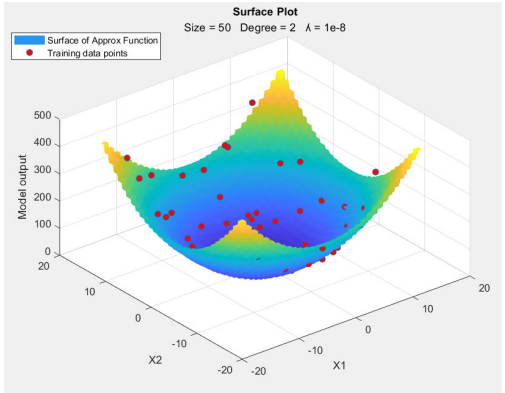


Plot 2.2b

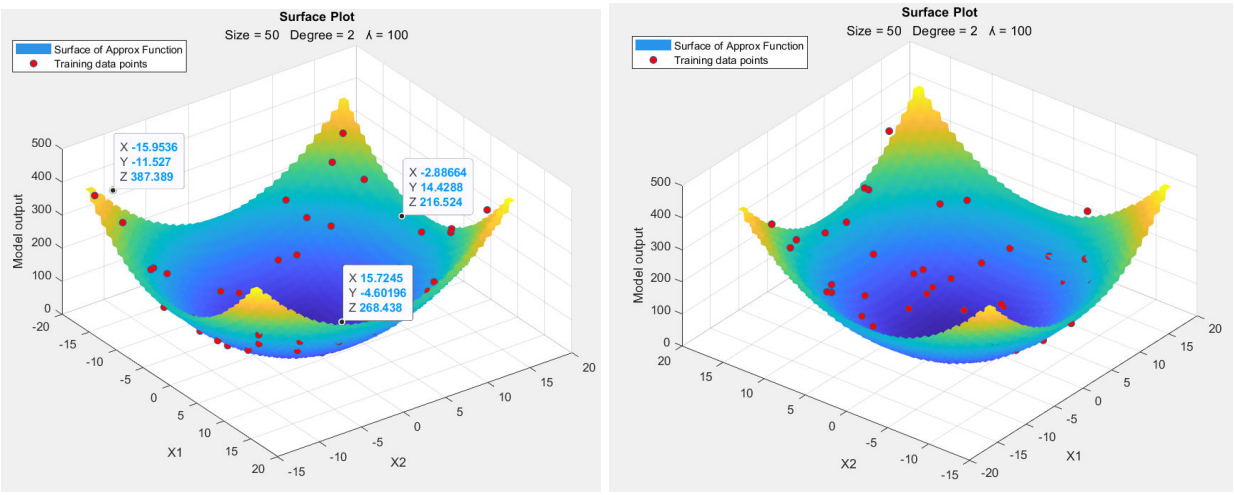
Degree 2, size-50, Lambda= 1e-4. Decent fit!





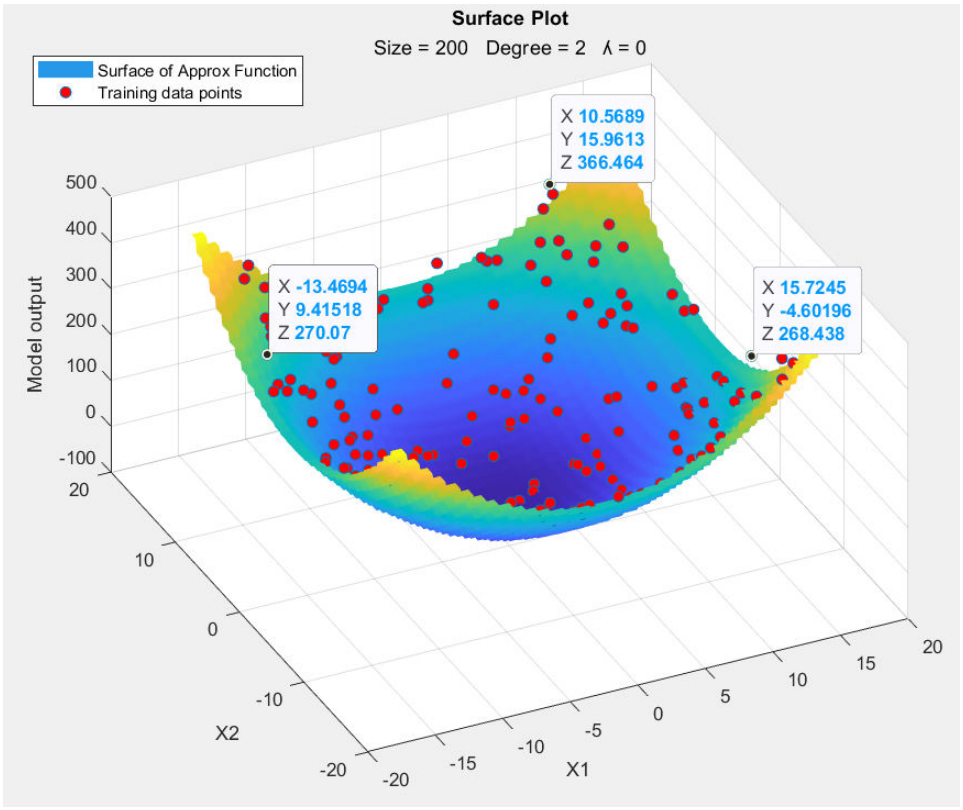


Plot 2.3 {Size-50, Lamda= 1e-8, degree =2 }

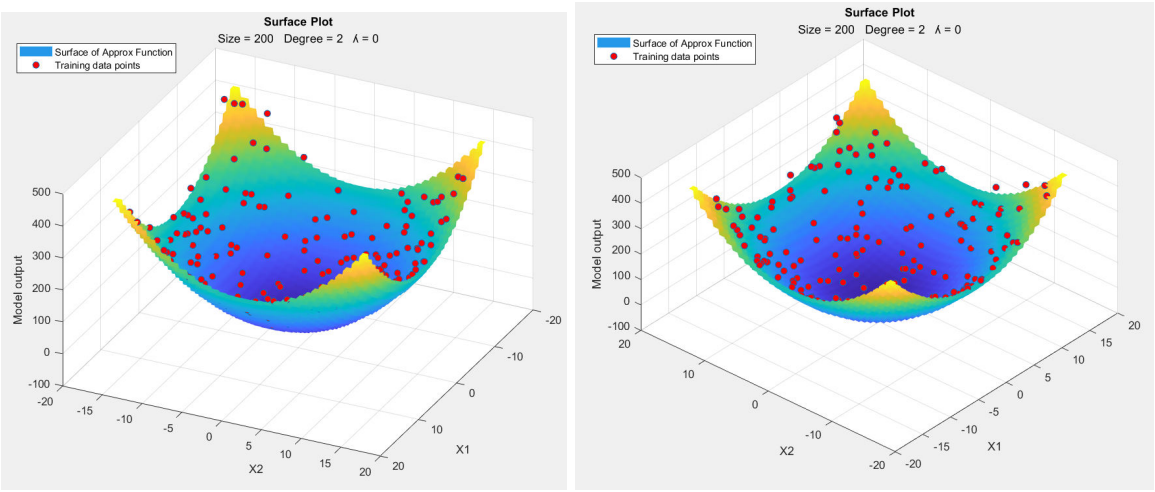


Plot 2.4 {Size-50, Degree-2, Lambda- 100}

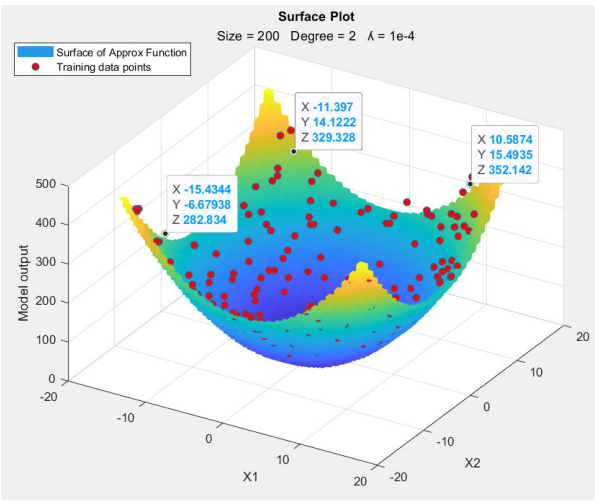
SIZE 200:

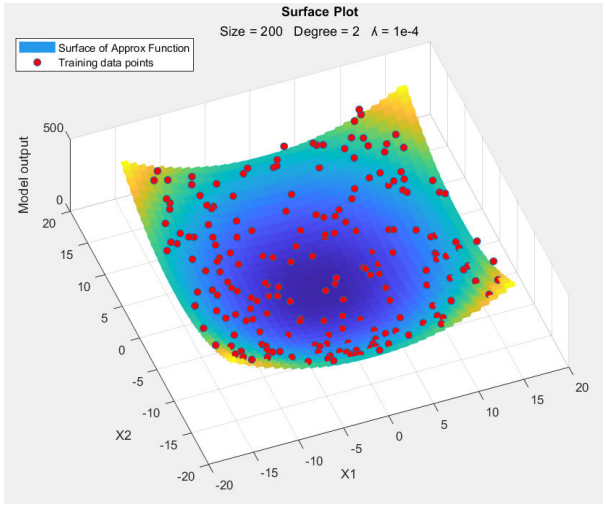


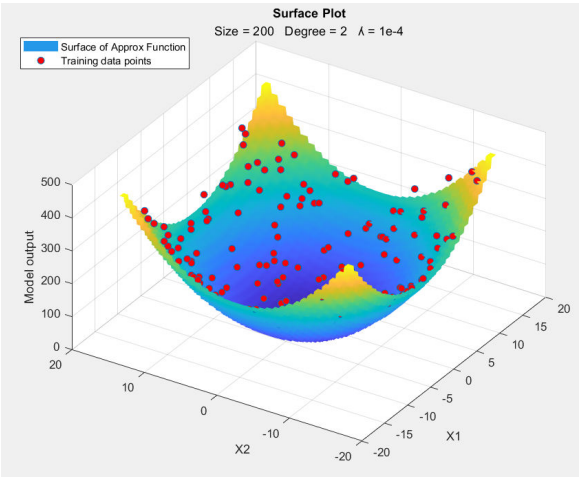
Plot 2.5a



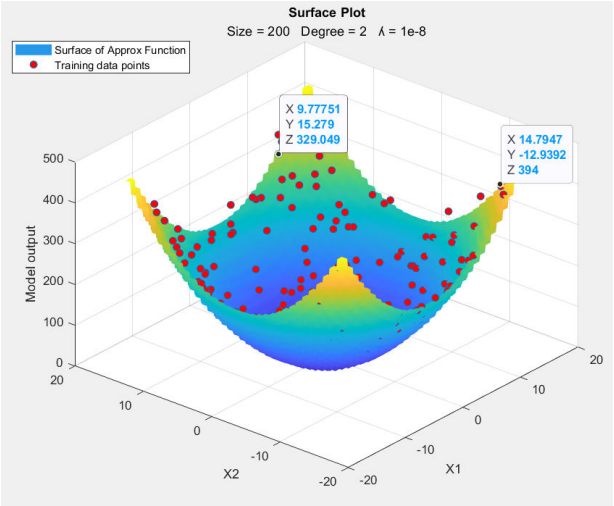
Plot 2.5 {Size-200, Degree-2, Lambda-0} Good fit!

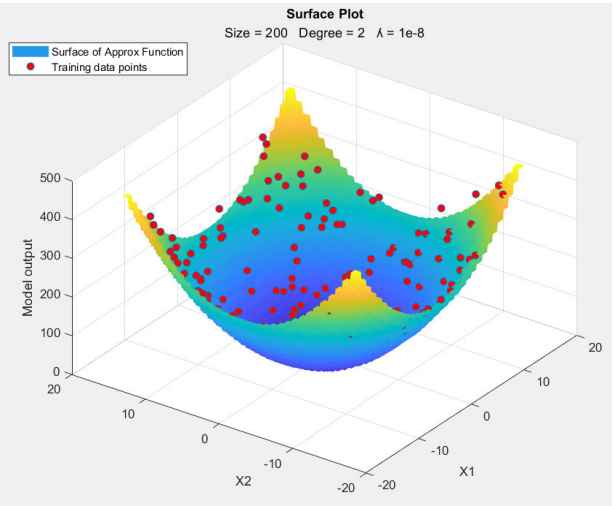




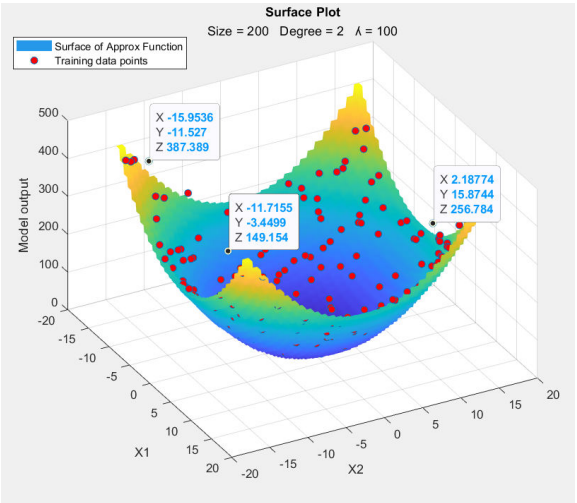


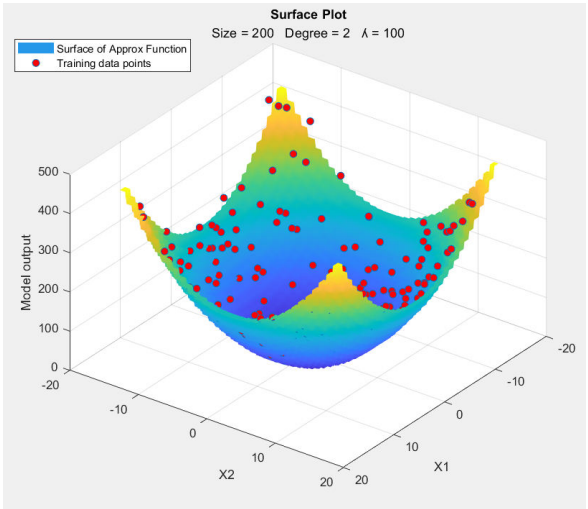
Plot 2.6 {Size-200, Degree-2, λ=1e-4}

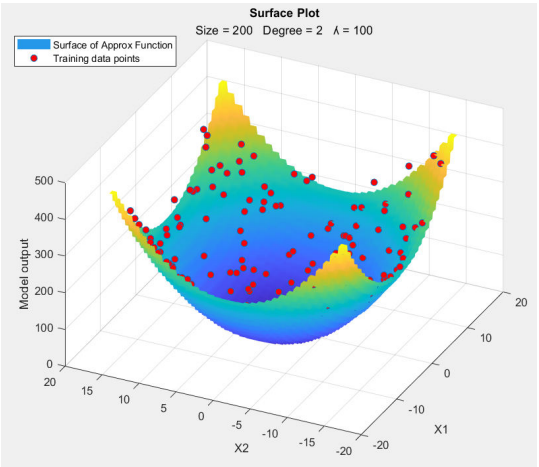




Plot 2.7 {Size-200, Degree-2, λ=1e-8}

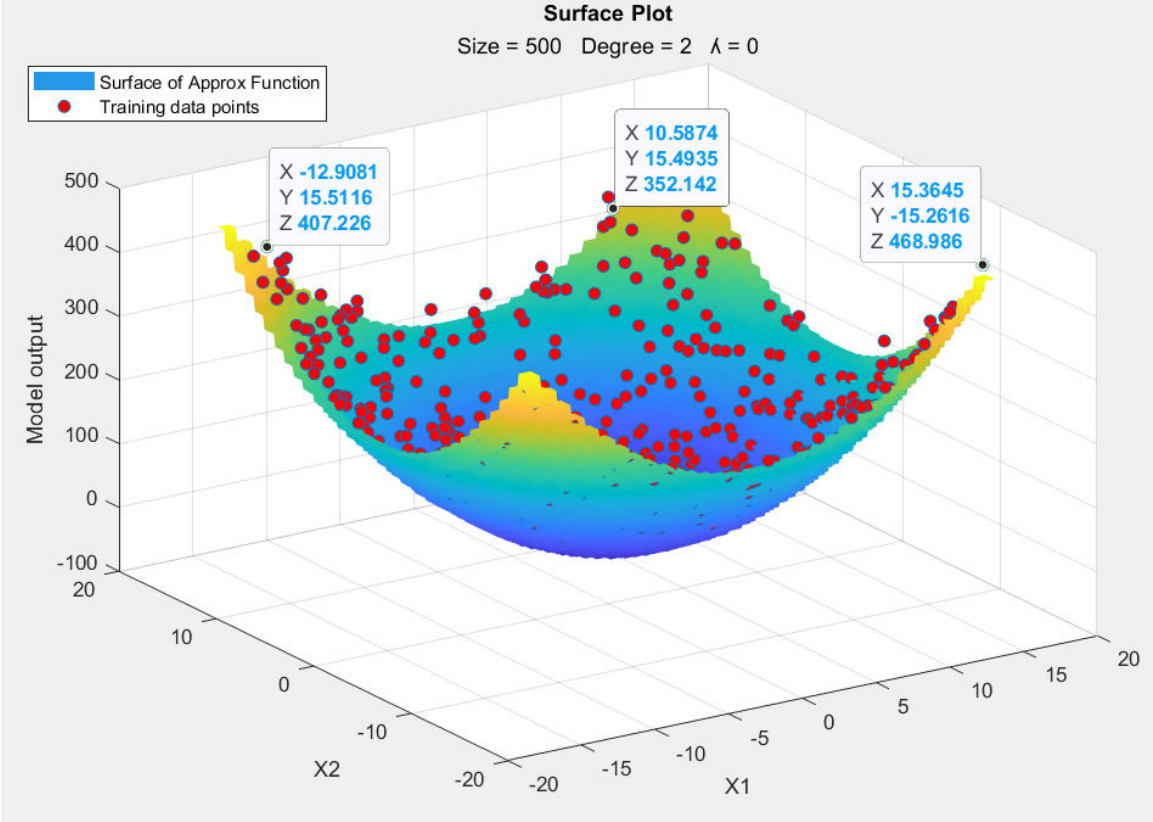


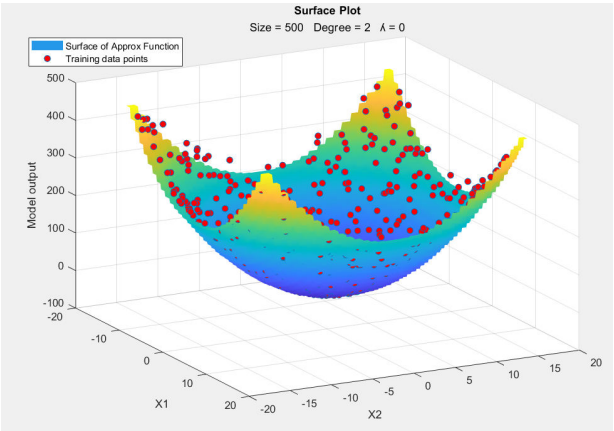


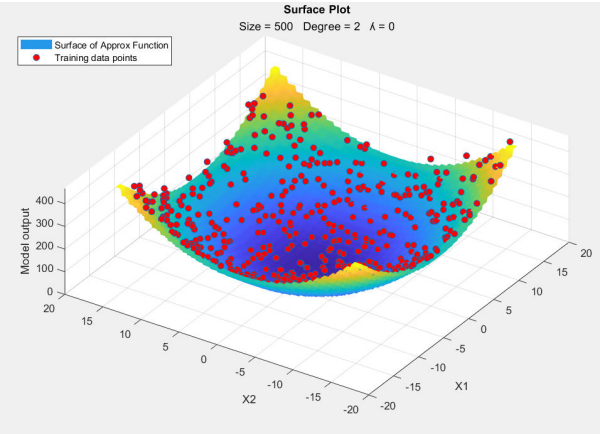


Plot 2.8 {Size-200, Degree-2, λ=100}

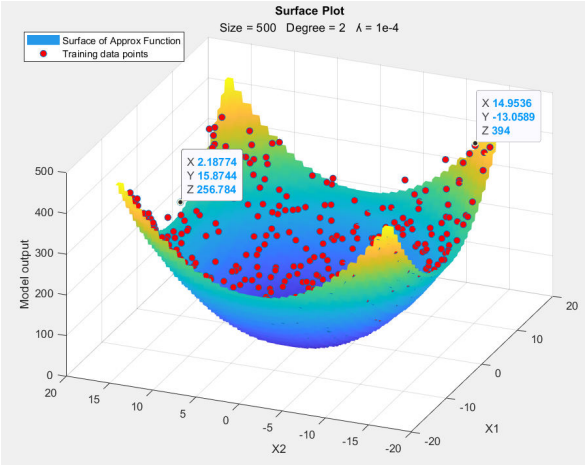
SIZE 500:

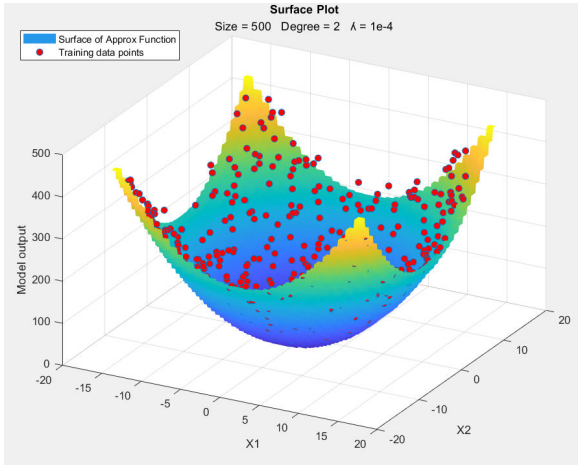


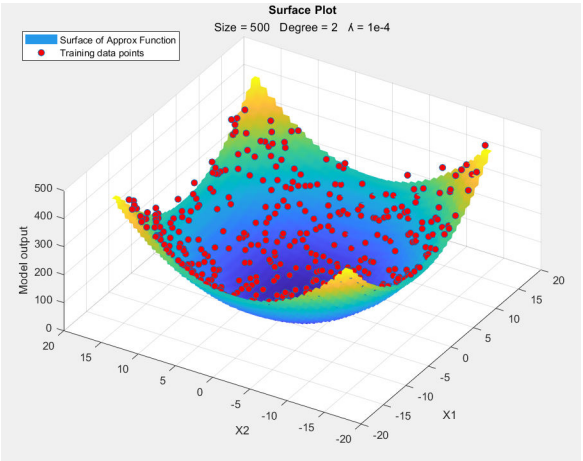




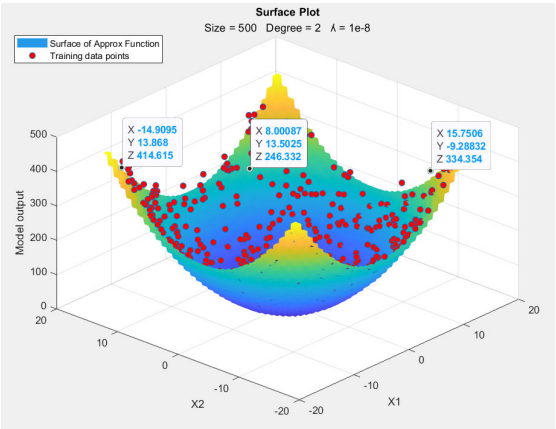
Plot 2.9 {Size-500, Degree-2, λ=0}

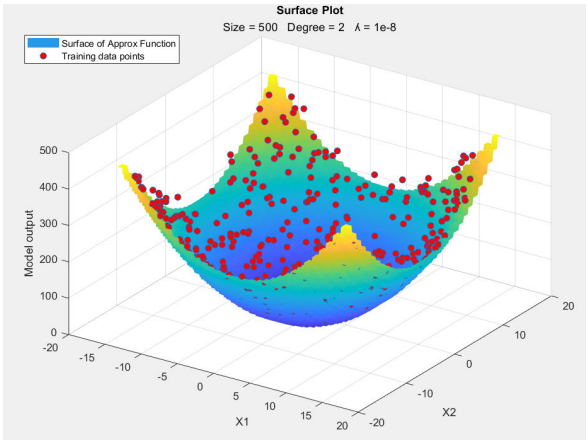


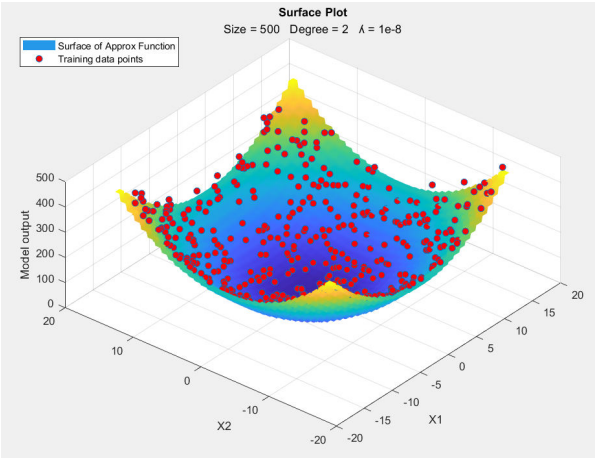




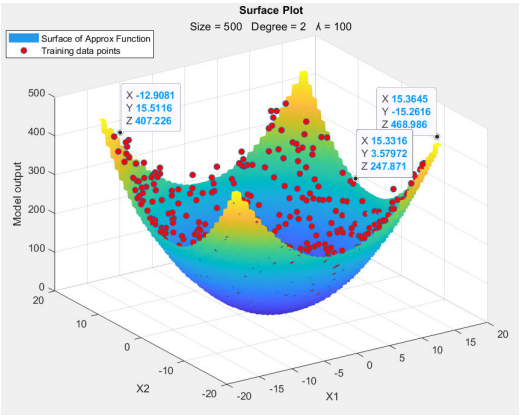
Plot 2.10 {Size-500, Degree-2, λ=1e-4}

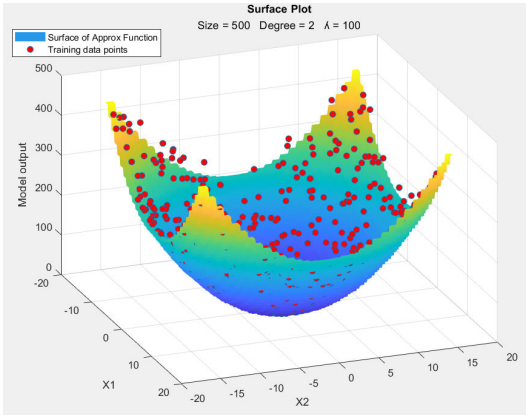


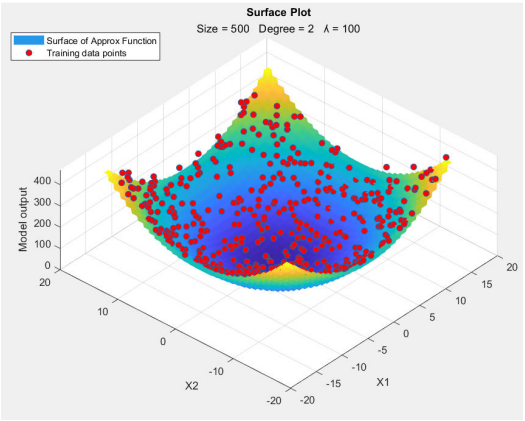




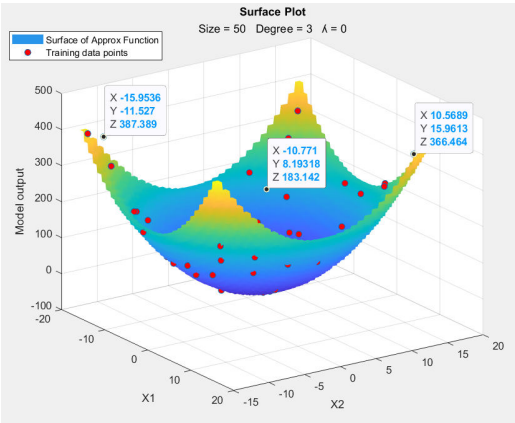
Plot 2.11 {Size-500, Degree-2, λ=1e-8}

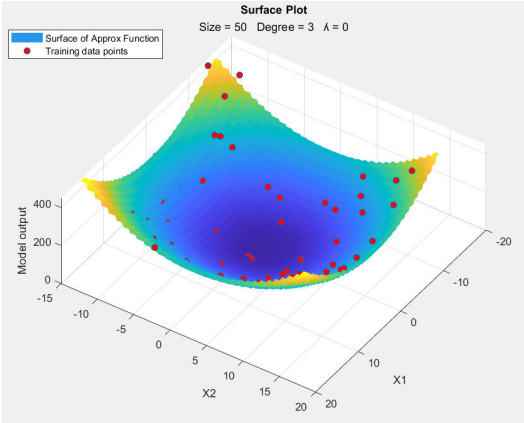


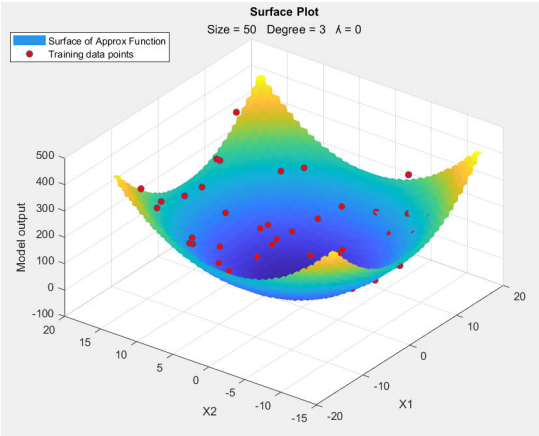




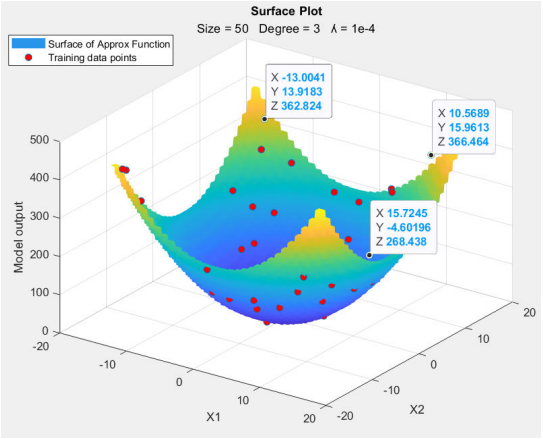
Plot 2.12 {Size-500, Degree-2, λ=100}

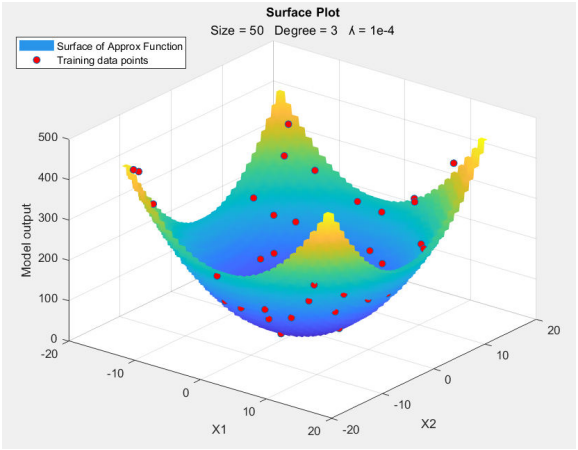


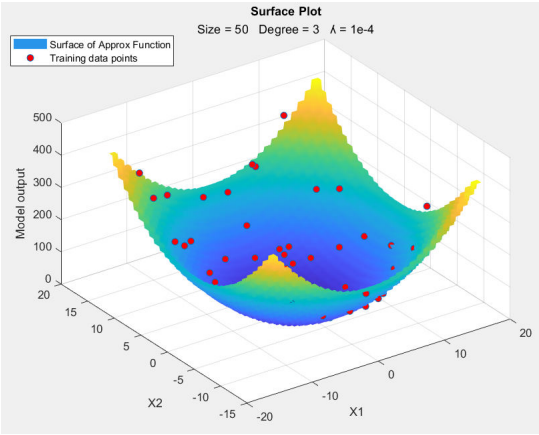




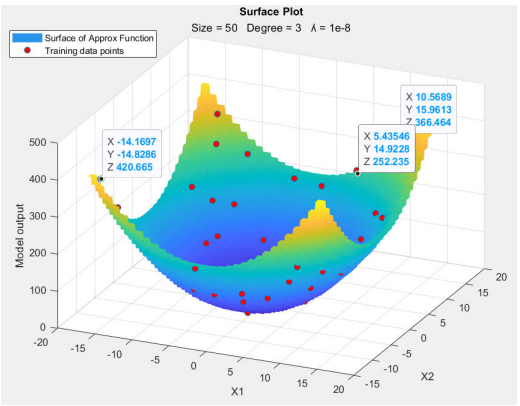
Plot 2.13 {Size-50, Degree-3, λ=0}

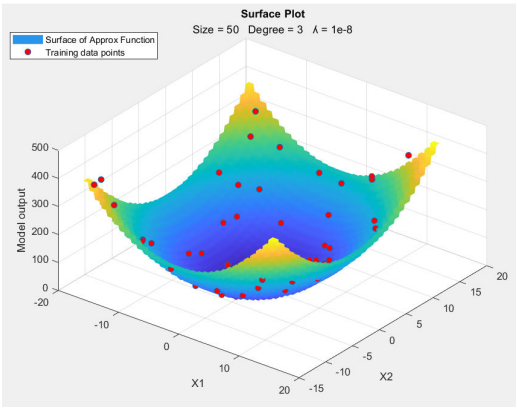


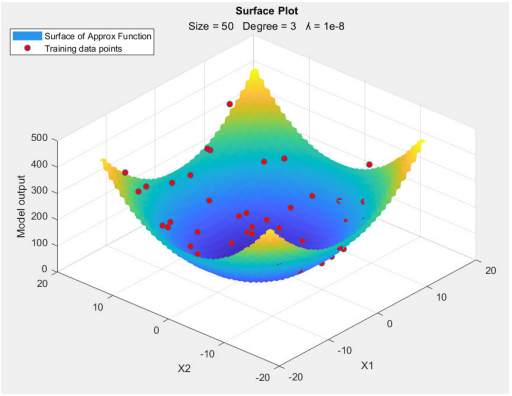




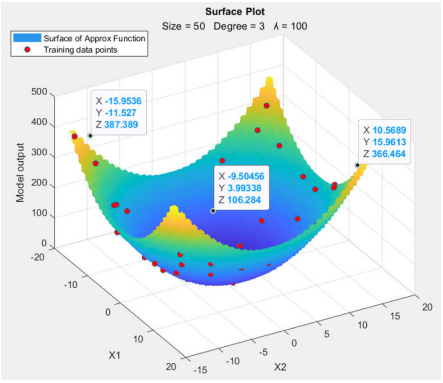
Plot 2.14 {Size-50, Degree-3, λ=1e-4}

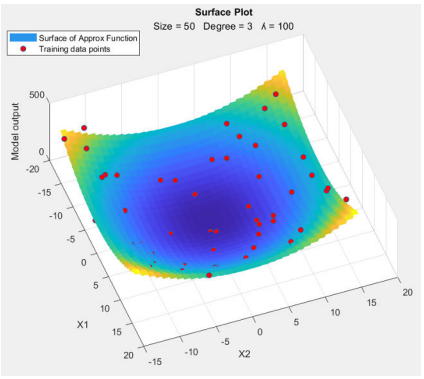


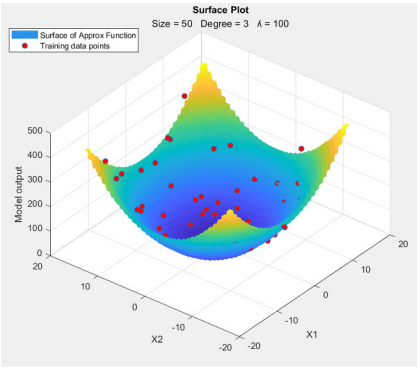




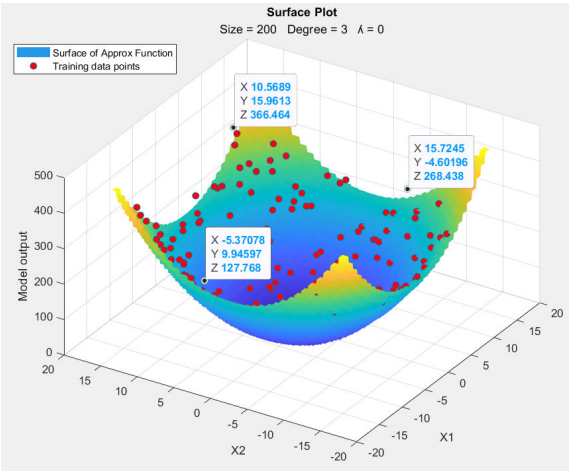
Plot 2.15 {Size-50, Degree-3, λ=1e-8}

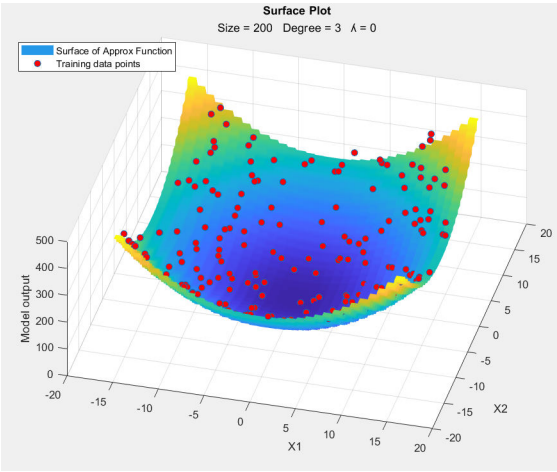




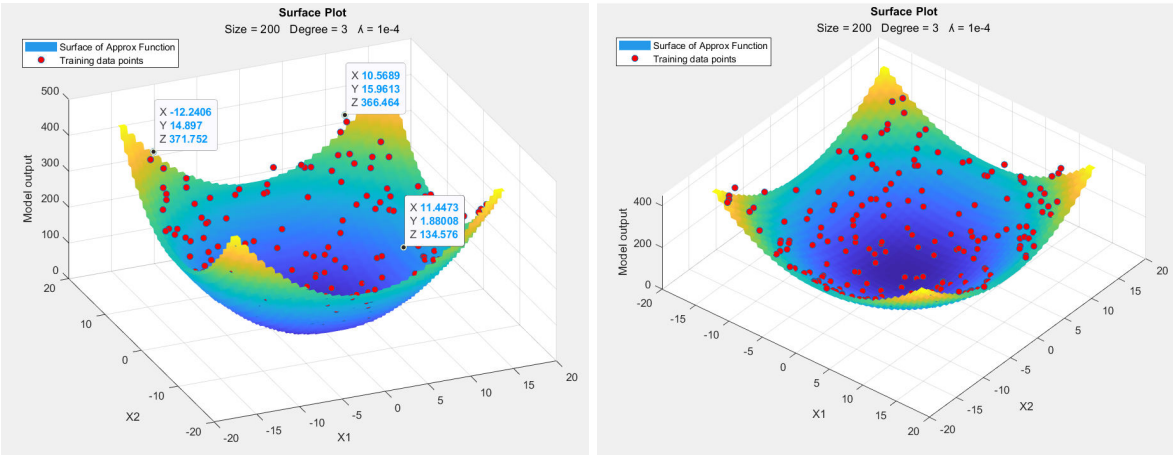


Plot 2.16 {Size-50, Degree-3, λ=100}

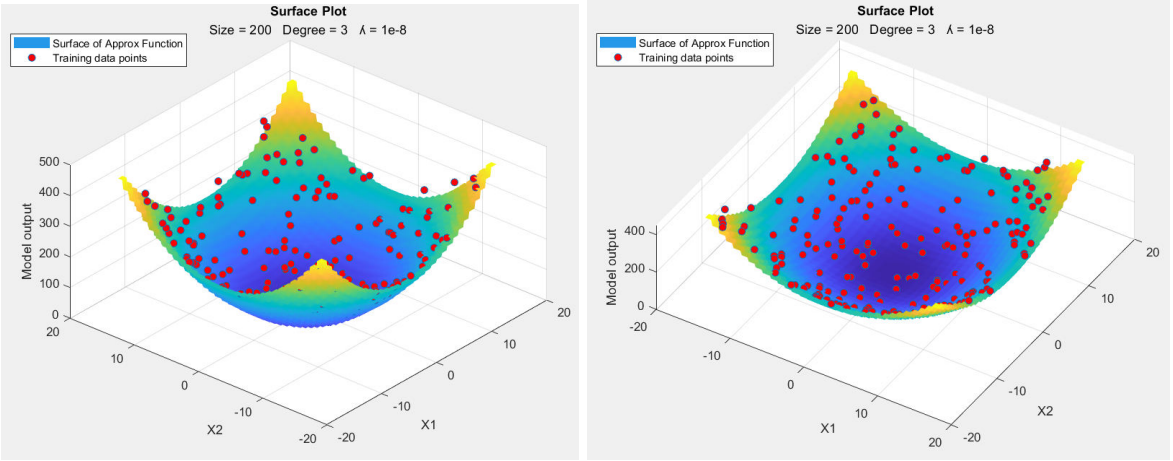




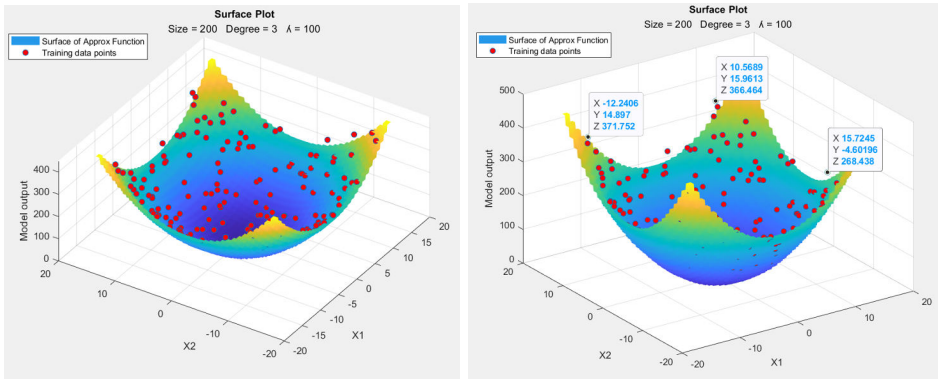
Plot 2.17 {Size-200, Degree-3, λ=0}



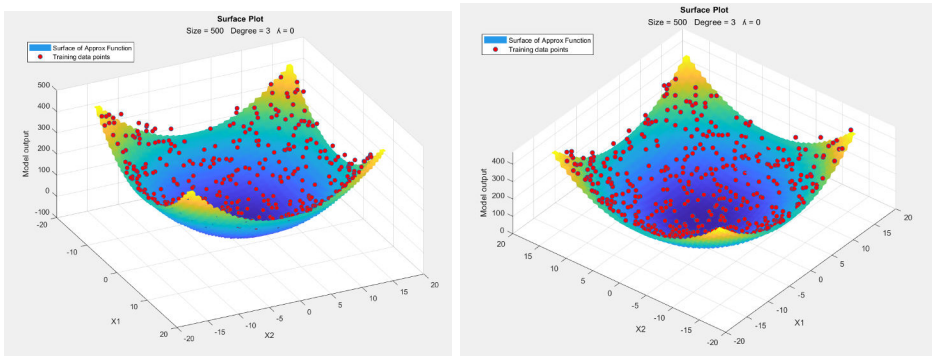
Plot 2.18 {Size-200, Degree-3, λ=1e-4}



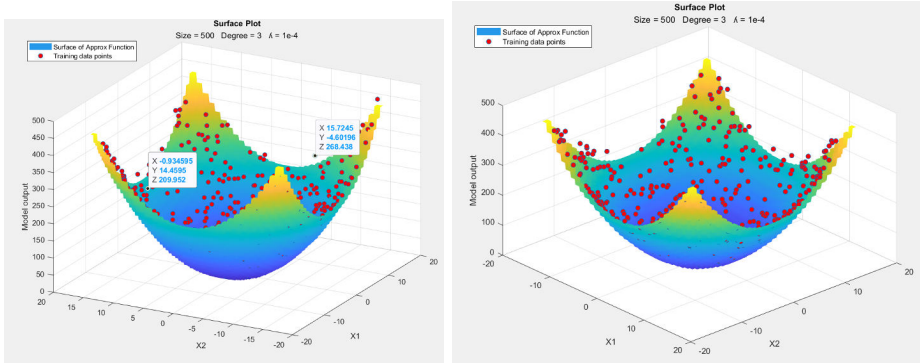
Plot 2.19 {Size-200, Degree-3, λ=1e-8}



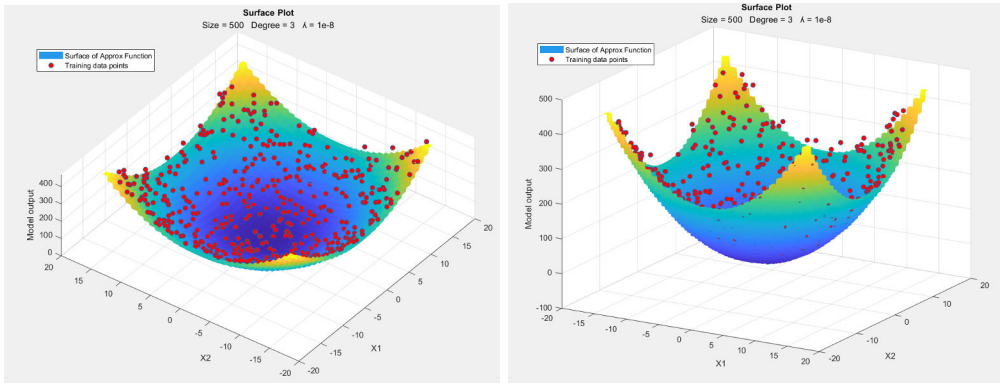
Plot 2.20 {Size-200, Degree-3, λ=100}



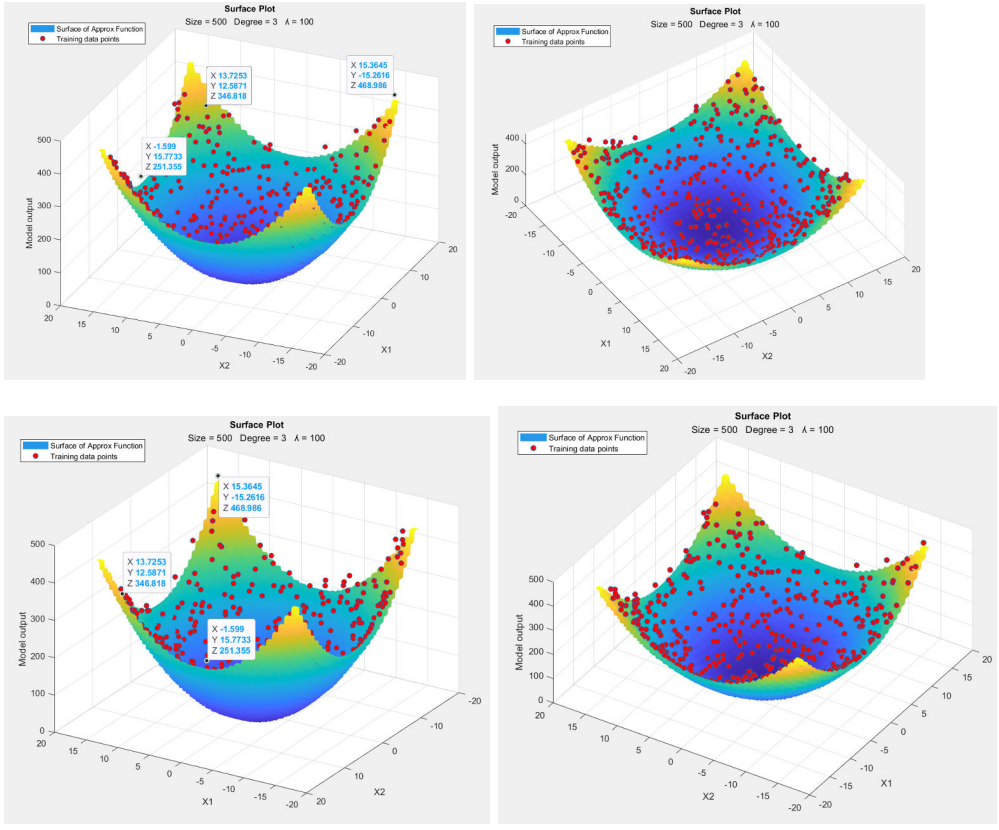
Plot 2.21 {Size-500, Degree-3, λ=0}



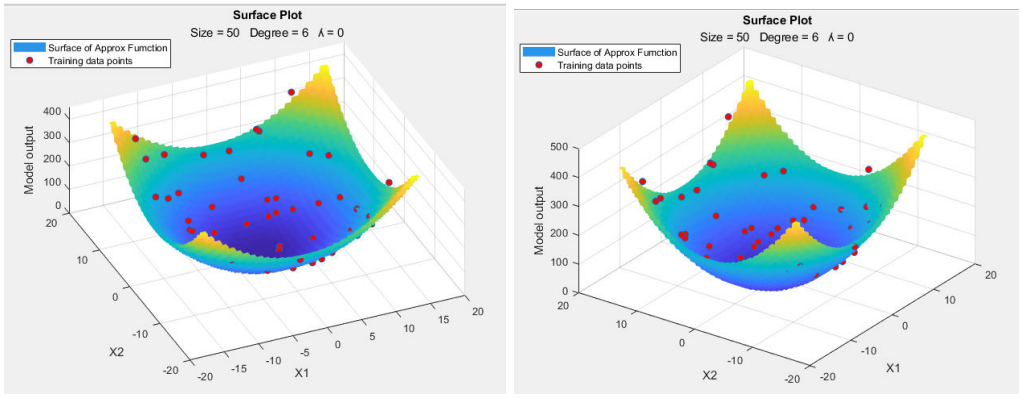
Plot 2.22 {Size-500, Degree-3, λ=1e-4}



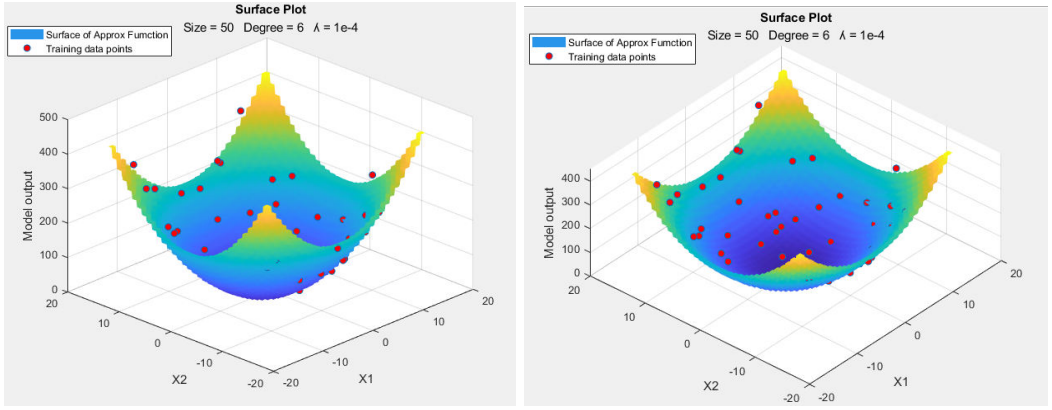
Plot 2.23 {Size-500, Degree-3, λ=1e-8}



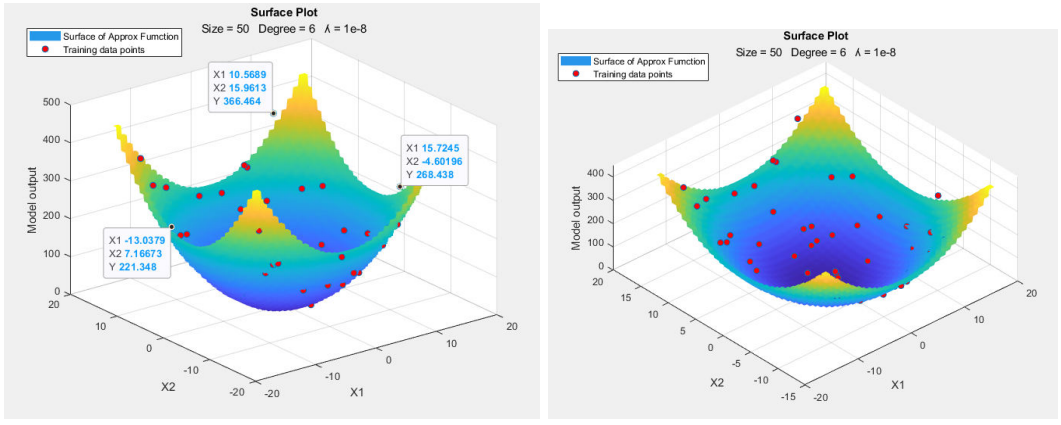
Plot 2.24 {Size-500, Degree-3, λ=100}



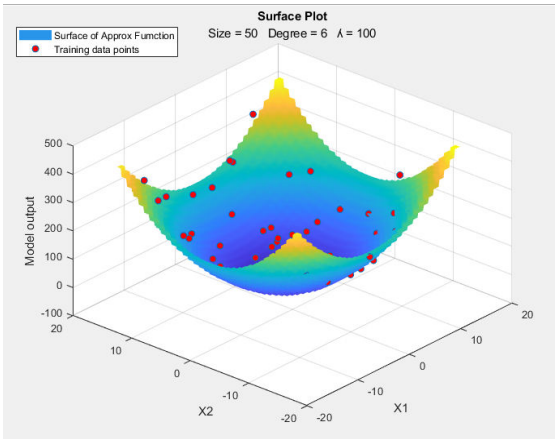
Plot 2.25 {Size-50, Degree-6, λ=0}

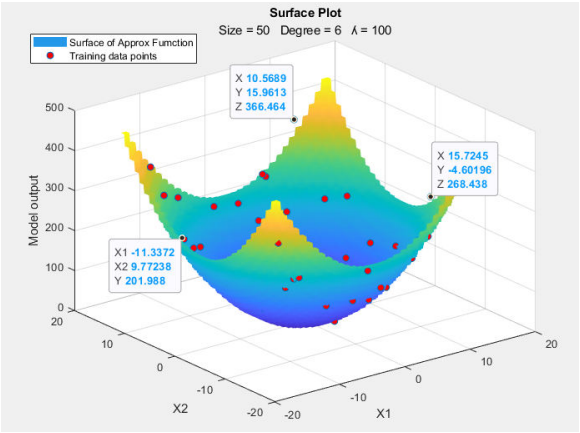


Plot 2.26 {Size-50, Degree-6, λ=1e-4}

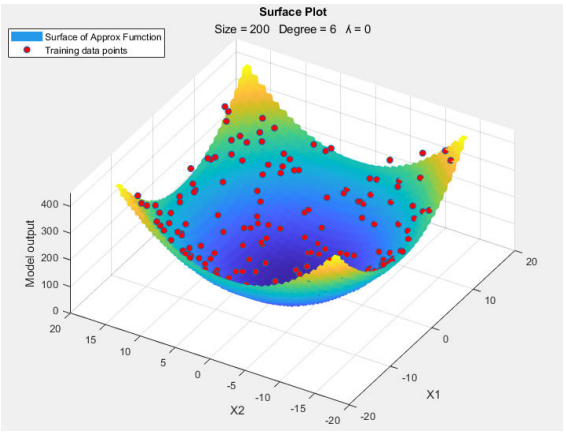


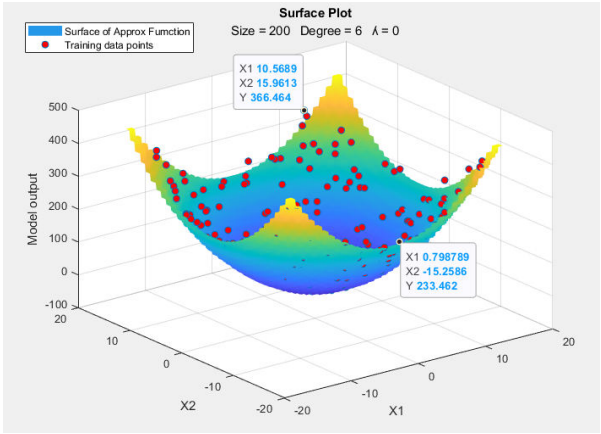
Plot 2.27 {Size-50, Degree-6, λ=1e-8}



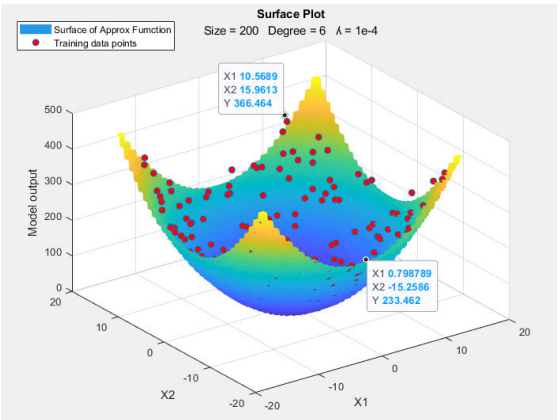


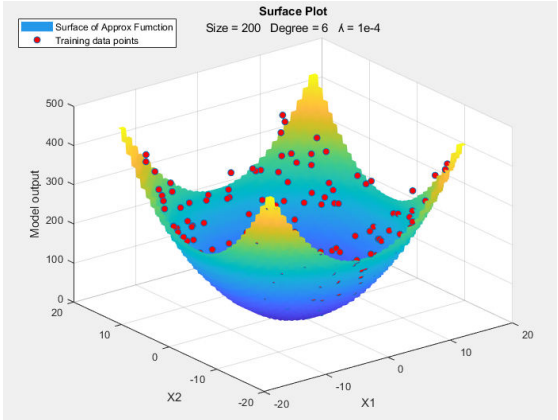
Plot 2.28 {Size-50, Degree-6, λ=100}



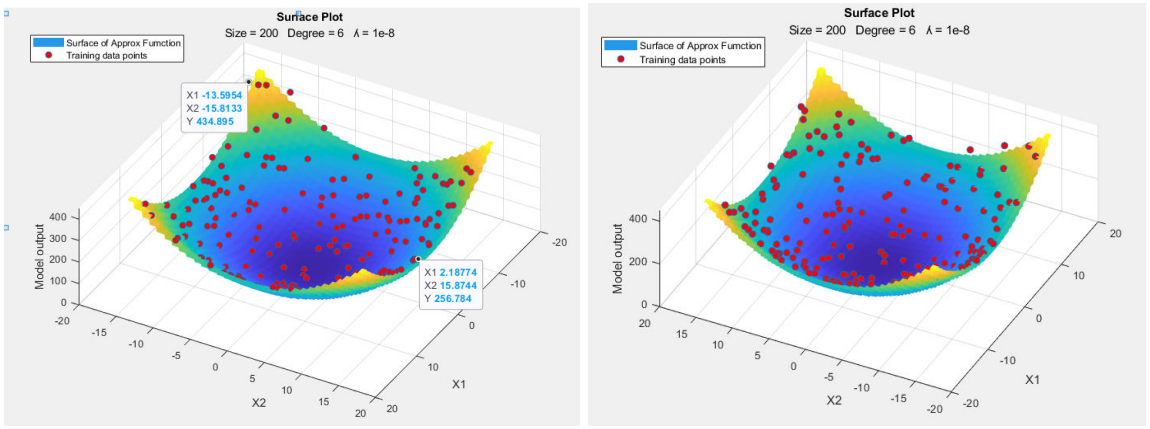


Plot 2.29 {Size-200, Degree-6, λ=0}

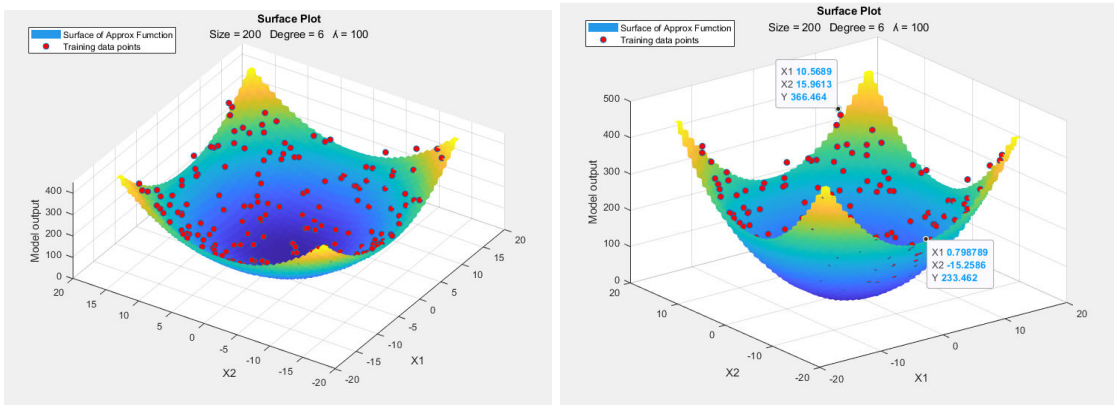




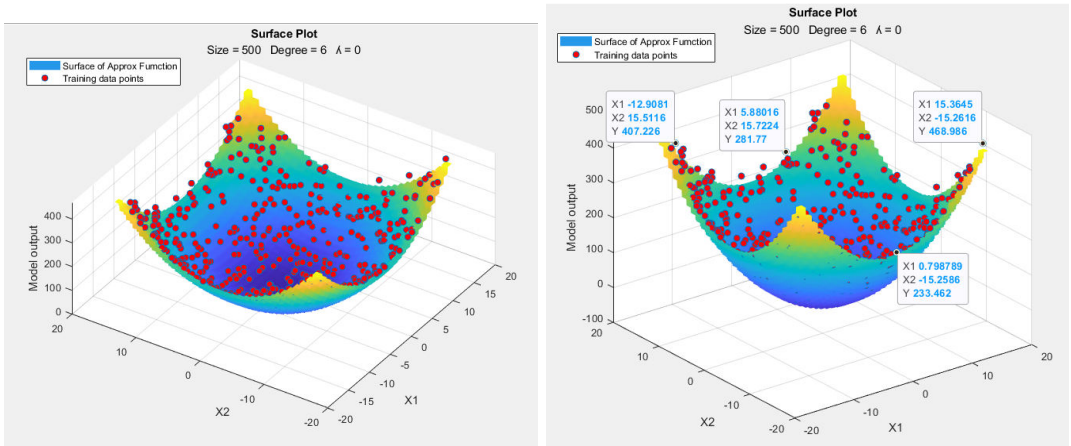
Plot 2.30 {Size-200, Degree-6, λ=1e-4}



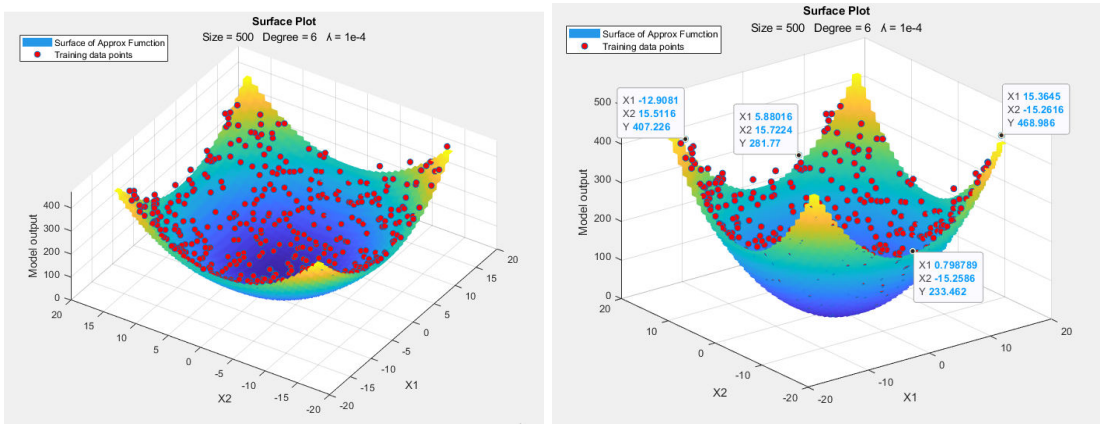
Plot 2.31 {Size-200, Degree-6, λ=1e-8}



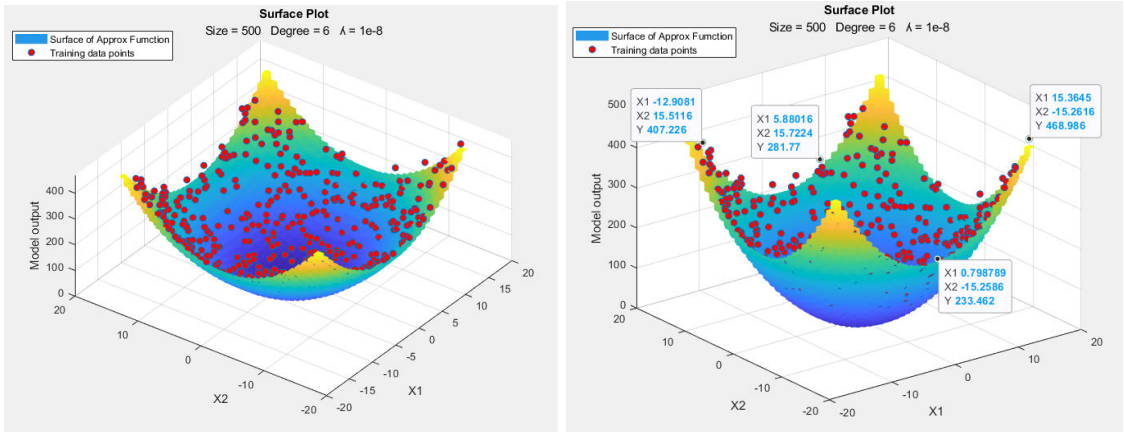
Plot 2.32 {Size-200, Degree-6, λ=100}



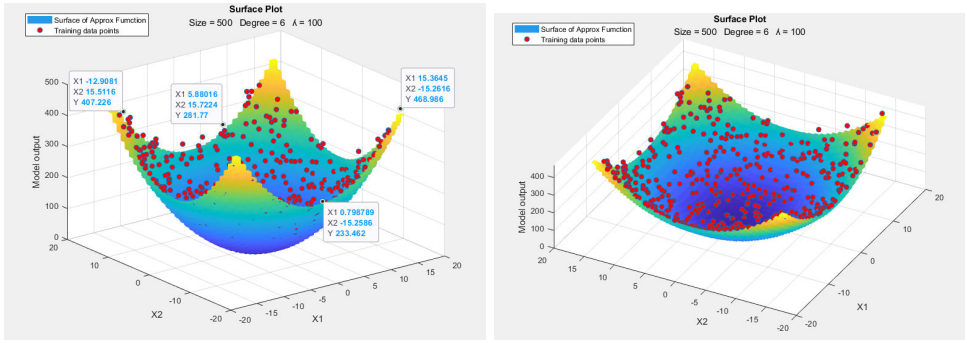
Plot 2.33 {Size-500, Degree-6, λ=0}



Plot 2.34 {Size-500, Degree-6, λ=1e-4}



Plot 2.35 {Size-500, Degree-6, λ=1e-8}



Plot 2.36{Size-500, Degree-6, λ=100}

All of the above plots across all degrees and training sizes seemed to have fit the training data points pretty well!

TABLE 2.1

DEGREE	SIZE	LAMBDA (ʎ)	ERMS_train	ERMS_val	ERMS_test
2	50	0	2.03E-13	**2.27E-13**	1.72E-13
		1.00E-08	4.05E-12	4.79E-12	5.06E-12
		1.00E-04	4.05E-08	4.78E-08	5.09E-08
		100	0.02172533	0.026019816	0.02648662
	200	0	1.60E-13	**1.60E-13**	2.72E-13
		1.00E-08	8.70E-13	9.23E-13	1.38E-12
		1.00E-04	9.54E-09	1.00E-08	1.36E-08
		100	0.00578511	0.006114265	0.007801402
	500	0	8.69E-14	**8.58E-14**	1.49E-13
		1.00E-08	3.88E-13	3.95E-13	6.48E-13
		1.00E-04	3.76E-09	3.84E-09	6.09E-09
		100	0.002715695	0.002779631	0.004320347
3	50	0	7.36E-13	**8.05E-13**	1.04E-12
		1.00E-08	4.54E-12	6.86E-12	8.41E-12
		1.00E-04	4.49E-08	6.81E-08	8.53E-08
		100	0.023573808	0.035301382	0.04115471
	200	0	1.75E-13	**1.81E-13**	3.90E-13
		1.00E-08	9.02E-13	9.79E-13	1.54E-12
		1.00E-04	9.70E-09	1.05E-08	1.42E-08
		100	0.005870899	0.006442308	0.008008707
	500	0	4.98E-13	**4.89E-13**	1.08E-12
		1.00E-08	7.33E-13	7.05E-13	1.87E-12
		1.00E-04	3.80E-09	3.96E-09	6.23E-09
		100	0.002741912	0.00288435	0.004437105
6	50	0	1.04E-09	1.82E-09	9.77E-10
		1.00E-08	7.77E-10	**1.56E-09**	9.13E-10
		1.00E-04	6.14E-07	5.13E-06	3.14E-06
		100	0.294006598	2.063225034	2.024934328
	200	0	2.21E-10	2.31E-10	3.51E-10
		1.00E-08	2.00E-10	**2.08E-10**	8.34E-11
		1.00E-04	1.00E-07	1.81E-07	3.08E-07
		100	0.054192246	0.095302096	0.141133089
	500	0	2.38E-10	2.38E-10	2.15E-10
		1.00E-08	2.36E-10	**2.33E-10**	3.48E-10
		1.00E-04	3.71E-08	5.00E-08	8.75E-08
		100	0.021478194	0.02836786	0.050275502

Observations:

The highlighted values in the E_rms_val column denotes the best Lambda value that will get selected for that particular model.

A general observation is that the Error values are very low across all degrees and models. This means that even the simplest model that we chose(degree 2) with the minimum number of observations(50) was good enough to fit the training data points very well.

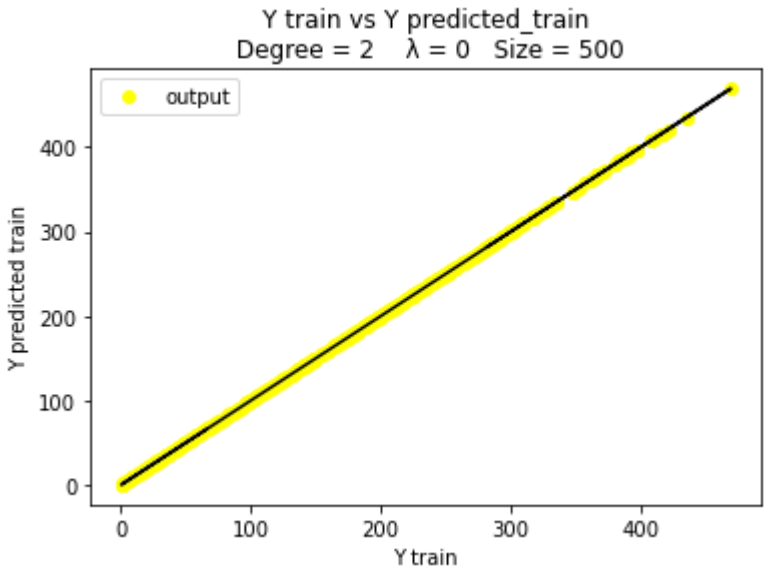
While when we increased the model complexity it still did not overfit and perform badly on the validation set! That means we both had good enough complexity and good enough #points all the way through!

The models did underfit the data to some extent for very large lambda values.

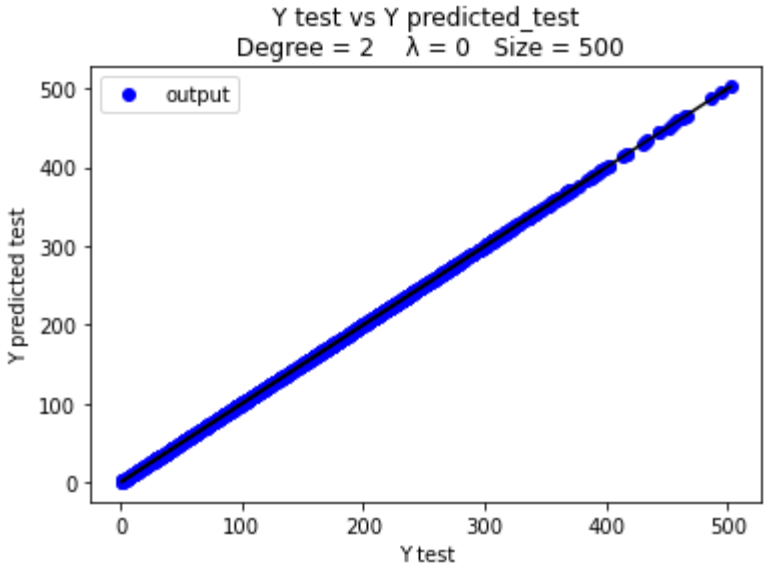
From the table, the best overall model is the model with

{Size-500, Degree-2, λ=0}

Best Overall Model {Size-500, Degree-2, λ=0}



Plot 2.37



Plot 2.38

The model performs extremely well for both training and testing data points!

Task 2 Conclusion:

The actual data seems to have been taken from a second degree curve. This is the reason why the degree 2 model is good enough to fit the overall data while also outperforming the higher complexity models on Validation as well as training errors!
All this it can do without even needing regularization, which further affirms the claim.

TASK 3:

PART1: Dataset 2: ERMS Tables

Table 3.1- Gaussian Linear Regression with Tikhonov Regularizer

Train size	K	Sigma sq	λ	Erms train	Erms val	Erms test
800	30	600	0.0001	0.495569	**0.476995**	0.459831
1200	25	700	0.0001	0.591722	**0.478456**	0.537016
800	30	600	0	0.479409	**0.490525**	0.476862
1200	25	650	0	0.525846	**0.492147**	0.500723
800	25	650	0	0.485678	**0.511444**	0.480248
800	25	700	0.0001	0.499128	**0.516028**	0.464036
1200	25	700	0	0.552708	**0.523051**	0.509056
1200	25	650	0.0001	0.556088	**0.532242**	0.528417
600	25	700	0.0001	0.518299	**0.540831**	0.524057
800	30	750	0.0001	0.513416	**0.543627**	0.478024
1200	30	600	0.0001	0.55156	**0.544049**	0.532311
800	25	900	0.0001	0.55172	**0.558635**	0.520988
1200	25	650	0.0001	0.601063	**0.562748**	0.55848
800	25	650	0.0001	0.568936	**0.582565**	0.540566
800	25	600	0	0.5964	0.61686	0.589043
1200	25	600	0.0001	0.637571	0.619663	0.602365
1200	30	600	0	0.617834	0.638085	0.598624
1200	100	60	0.0001	0.639525	0.638971	0.614985
450	25	700	0.0001	0.644943	0.642206	0.538727
1200	25	600	0	0.651439	0.664833	0.6494
800	25	650	0.0001	0.689259	0.670935	0.668183
800	25	700	0	0.645921	0.674658	0.638381
350	25	700	0.0001	0.680547	0.680886	0.480835
1200	35	300	0.0001	0.744059	0.71279	0.666598
800	60	200	0.0001	0.687325	0.733817	0.698152
500	20	1000	0.0001	0.834162	0.784352	0.736471
800	35	300	0.0001	0.816468	0.794598	0.757719
500	20	1000	0	0.828626	0.799534	0.68752
1200	35	300	0.0001	0.888467	0.801926	0.7773
800	25	600	0.0001	0.883551	0.810878	0.799729
800	100	60	0.0001	0.578819	0.817884	0.661365
800	35	300	0.0001	0.788258	0.83639	0.714523
1200	25	900	0.0001	0.958745	0.890969	0.966241
1200	60	100	0.0001	1.104933	1.082995	1.085037
1200	30	650	0.0001	1.078219	1.103341	1.018224
800	30	650	0.0001	1.175916	1.152803	1.181769
800	30	300	0.0001	1.181734	1.166569	1.087355
150	20	1000	0	0.726021	1.208479	1.049952
500	60	100	0.0001	0.941237	1.221017	1.3089
500	10	10000	0.0001	1.514794	1.378068	1.240848
150	20	1000	0.0001	0.859049	1.380015	0.948163
800	60	100	0.0001	1.302896	1.409292	1.295624
1200	30	700	0.0001	1.393208	1.412215	1.422468
500	60	100	0	1.223336	1.452919	1.295475
150	10	10000	0	1.417587	1.565855	1.351901

150	10	10000	0.0001	1.583727	1.663103	1.213904
800	30	600	0.0001	1.685101	1.689775	1.67381
1200	30	600	0.01	2.069404	1.712	1.595481
1200	27	600	0.01	2.07287	1.720455	1.599369
1200	25	600	0.01	2.126472	1.751096	1.631597
600	25	1200	0.0001	1.761566	1.783504	1.033279
1200	25	650	0.01	2.178628	1.784046	1.691051
1200	27	650	0.01	2.159452	1.784838	1.680068
1200	30	650	0.01	2.136545	1.78604	1.669152
1200	60	200	0.0001	1.855403	1.87927	1.890631
1200	30	600	0.0001	1.898038	1.886086	1.912222
1200	25	700	0.01	2.272972	1.898571	1.800861
1200	30	700	0.01	2.246081	1.899515	1.78499
1200	27	700	0.01	2.252827	1.907767	1.793824
1200	30	300	0.0001	1.966589	1.933223	1.863075
1200	100	100	0.0001	2.167725	1.989923	2.17456
800	35	200	0.0001	2.195647	2.038762	2.028146
800	30	600	0.01	2.535677	2.209065	1.943408
800	27	600	0.01	2.540797	2.213149	1.945855
800	25	600	0.01	2.664528	2.240437	2.013843
1200	30	700	0	2.159788	2.30207	2.205105
1200	35	600	0.0001	2.335573	2.333861	2.343341
800	30	650	0.01	2.676328	2.350361	2.088875
800	27	650	0.01	2.680825	2.357843	2.092689
800	25	650	0.01	2.68922	2.381149	2.099355
1200	30	750	0.0001	2.464353	2.478672	2.402789
800	30	700	0.01	2.856333	2.534566	2.276972
800	25	700	0.01	2.885834	2.544544	2.296364
800	27	700	0.01	2.860753	2.54633	2.280009
800	30	700	0.0001	2.453861	2.63518	2.483711
800	30	650	0	2.970486	2.878744	2.884145
1200	35	200	0.0001	3.359188	3.210749	3.151195
500	50	100	0	2.61312	3.245619	2.995146
150	60	100	0	0.586404	3.424328	2.71862
500	50	100	0.0001	2.917791	3.472269	3.035789
150	60	100	0.0001	0.635055	3.85537	2.669028
500	10	10000	0	4.661845	4.568505	1.282357
1200	30	700	0.0001	4.607419	4.682208	4.654261
450	25	1200	0.0001	4.881644	4.801382	5.397005
800	100	100	0.0001	5.161629	4.878693	5.038782
800	60	60	0.0001	4.694155	5.238986	4.529393
150	50	100	0	1.420054	5.548838	3.14662
1200	60	60	0.0001	5.091024	5.57431	4.672849
1200	30	650	0	5.652046	5.618246	5.605369
150	50	100	0.0001	1.485076	5.845893	4.259197
1200	35	100	0.0001	7.482469	5.979712	5.981633
800	30	700	0	6.822912	6.824454	6.836059
800	30	700	0.0001	6.970713	7.167205	7.205557

800	35	100	0.0001	7.913293	7.268514	7.150685
350	25	1200	0.0001	8.426925	8.343335	8.222555
800	35	600	0.0001	8.80529	8.841216	8.94691
500	10	1000	0.0001	9.23587	8.981405	7.834402
500	10	1000	0	11.05842	10.52314	10.44172
150	10	1000	0.0001	12.1006	13.2562	12.26563
1200	35	60	0.0001	17.24294	13.54658	14.31536
150	10	1000	0	12.68613	13.79649	12.47278
20	10	10000	0.0001	6.788344	16.35818	19.15863
500	60	100	1	18.51442	17.48224	14.55316
500	50	100	1	18.60314	17.60816	14.61809
800	35	60	0.0001	21.09303	20.20265	18.71563
500	20	100	0.0001	22.86746	22.25267	21.41005
500	20	100	0	24.71459	23.89176	21.448
20	20	1000	0.0001	2.920547	24.07468	27.10542
20	20	1000	0.0001	2.741	24.21389	26.95155
20	20	1000	0.0001	2.668161	24.42053	26.9311
20	20	1000	0.0001	3.00214	24.55745	26.97748
20	20	1000	0.0001	2.77242	24.83232	27.29707
500	20	1000	1	28.63203	25.98452	24.16664
500	10	1000	1	28.86618	26.18407	24.32983
500	20	100	1	28.90652	28.12035	25.32407
20	10	10000	0	25.83874	28.23834	5.902286
20	10	1000	0.0001	3.934607	29.20948	35.8148
20	10	1000	0	3.654321	31.03214	37.60459
800	35	750	0.0001	32.49887	32.89979	32.94371
150	20	100	0	28.97771	33.3489	22.29449
800	60	300	0.0001	38.50698	36.74489	38.33888
150	20	100	0.0001	31.83172	37.55284	32.63277
150	60	100	1	35.04534	38.06867	26.33333
150	150	100	1	42.09013	45.86343	27.12518
150	20	100	1	43.49625	47.68163	37.62456
1200	35	700	0.0001	49.79543	48.95642	49.15887
150	20	1000	1	59.02744	62.94478	54.63449
150	10	1000	1	59.15882	63.06322	54.72655
500	10	100	0	67.33376	63.0733	59.55045
500	10	100	0.0001	68.74491	64.36352	58.74058
500	10	100	1	69.45988	65.39148	60.04523
150	10	100	0	64.02748	65.62196	58.9775
150	10	100	0.0001	64.70699	66.6094	64.72774
150	50	1000	1	63.56927	67.51695	57.78496
150	10	100	1	65.5534	67.94193	59.88454
20	20	100	0	1.37E-05	77.41073	96.26472
800	35	700	0.0001	83.82196	83.37251	83.24759
10	5	10000	0	26.73013	85.45262	39.16713
500	5	10000	0	85.03403	85.9077	88.28929
500	5	10000	0.0001	85.12677	85.96314	89.17156
10	5	10000	0.0001	29.69652	86.86033	40.06292

1200	60	300	0.0001	86.53278	89.19835	82.71523
20	5	10000	0	43.03906	89.55339	128.6281
600	40	3000	0.0001	91.05492	89.92574	95.87664
20	5	10000	0.0001	43.38474	90.63752	128.6961
500	5	1000	0.0001	92.29196	91.90868	99.937
10	5	1000	0.0001	31.67305	94.65654	45.89152
10	5	1000	0	30.84443	95.29483	45.79144
500	5	1000	0	97.77144	96.25403	98.55213
500	5	1000	1	99.19886	96.41802	98.15266
20	10	100	0.0001	23.67825	98.73115	106.4418
20	10	100	0	23.56521	98.96967	104.6854
20	5	1000	0.0001	49.24313	100.2992	136.8977
500	10	10000	1	106.5097	102.0172	99.80223
500	5	10000	1	110.2286	105.6432	103.6167
500	20	10000	1	110.323	106.5082	99.84461
150	5	10000	0	100.7654	108.9527	98.49261
150	20	10000	1	102.0722	109.1559	103.9344
150	10	10000	1	102.0486	109.1876	103.925
150	5	10000	0.0001	101.2642	109.4213	102.6885
150	5	10000	1	103.4948	110.7279	105.614
20	20	10000	1	93.63975	111.8118	108.3149
20	10	10000	1	92.50317	111.8661	108.2764
20	5	10000	1	92.52425	111.9026	108.3521
20	5	1000	0	52.33349	112.6498	136.8985
20	20	10000	1	94.61511	113.9	110.0823
10	10	10000	1	90.16798	113.9419	114.4767
20	20	1000	1	83.47792	121.0018	128.506
20	20	1000	1	83.47795	121.0018	128.506
20	10	1000	1	83.48384	121.0177	128.5139
150	5	1000	0	113.029	121.3589	110.4785
20	10	100	1	62.42982	121.8017	133.0547
150	5	1000	0.0001	113.344	121.9301	110.0499
150	5	1000	1	114.0196	122.4127	112.7509
20	20	10000	1	104.9034	123.8933	107.7935
20	20	10000	1	104.7212	123.9904	109.656
20	5	1000	1	85.45714	124.2574	134.0991
500	5	100	0	132.3064	125.6526	124.6417
500	5	100	1	132.3946	125.8059	124.9247
500	5	100	0.0001	132.4193	125.8228	125.3435
10	10	1000	1	82.51264	128.1533	128.0564
20	5	100	0.0001	69.35978	132.0365	150.1791
20	5	100	0	69.89705	132.4465	150.1789
450	25	3000	0.0001	132.9108	133.6142	18.33461
800	60	300	0.0001	132.1891	134.4152	129.0893
600	40	700	0.0001	132.5842	134.5801	70.17689
10	5	100	0	49.61858	135.644	136.0709
10	5	100	0.0001	49.61858	135.6445	136.0733
150	5	100	0.0001	130.7027	138.1848	130.1207

350	25	3000	0.0001	138.2587	138.2945	80.86969
150	5	100	1	131.2296	138.3824	130.5129
20	5	100	1	82.86248	138.5705	152.8408
150	5	100	0	131.1771	138.8167	130.1206
10	10	100	1	73.00516	142.1984	150.3778
10	10	100	1	73.00516	142.1984	150.3778
500	20	10000	0	142.8825	142.6148	157.5792
10	5	100	1	82.33743	147.8292	153.1546
1200	60	900	0.0001	158.6771	150.3929	159.2373
350	40	700	0.0001	153.6933	152.3178	37.00474
150	20	10000	0.0001	155.5971	154.0905	54.09049
450	40	700	0.0001	165.3512	170.4462	243.3441
800	60	700	0.0001	181.9979	176.0911	177.0957
10	10	10000	0	179.6353	176.9247	38.61768
10	10	10000	0	196.7844	191.7258	50.62519
20	20	1000	0	221.7117	209.1955	284.4257
450	40	3000	0.0001	241.4422	241.8242	305.3317
20	20	1000	0	250.0569	243.1265	72.33526
150	50	1000	0.0001	278.5931	277.1965	217.2009
1200	60	750	0.0001	287.9389	280.1157	285.0759
10	10	10000	0	296.3121	293.317	36.21377
1200	60	650	0.0001	292.7246	300.263	302.2514
10	10	10000	0	350.4589	350.9016	104.8242
20	20	1000	0	359.441	352.4343	151.2306
150	50	10000	1	359.8472	362.6059	309.9337
150	20	10000	0	366.4093	365.7417	413.6657
450	60	1200	0.0001	365.7715	379.7929	321.8721
600	25	3000	0.0001	396.7062	397.0638	666.3294
500	20	10000	0.0001	419.5661	419.8835	280.606
1200	60	300	0.0001	427.1653	425.6299	414.8449
10	10	10000	0	431.0814	427.8714	5.019442
450	60	3000	0.0001	429.78	429.5577	236.2315
150	60	1000	0.0001	425.592	429.7363	247.556
500	50	1000	0.0001	433.2395	438.0926	648.3868
450	40	1200	0.0001	439.984	438.5667	542.1427
20	20	10000	0.0001	469.4586	468.9134	287.1033
500	60	1000	0.0001	478.018	469.9048	711.8502
10	10	10000	0	478.3581	477.3882	12.54779
500	50	10000	0.0001	480.0183	480.4815	905.4292
20	20	1000	0	529.0935	490.784	435.1765
500	200	100	1	517.8835	518.9597	697.1243
500	60	1000	1	533.6962	537.3547	172.5593
800	40	650	0.0001	551.5863	554.7274	549.9755
1200	40	650	0.0001	606.395	596.7847	605.3099
800	60	750	0.0001	613.6236	609.7	606.3075
20	20	10000	0.0001	625.3212	624.5896	73.45709
20	20	10000	0.0001	636.0962	634.7386	242.183
800	60	600	0.0001	637.0285	640.2956	632.1347

150	60	10000	0.0001	831.8155	832.0475	101.686
500	60	10000	0.0001	864.4786	866.9549	355.002
20	20	10000	0	917.6818	920.6663	124.906
20	20	10000	0	928.5626	925.2597	2009.867
800	40	900	0.0001	925.6044	925.7996	920.0477
500	60	10000	0	953.1199	953.9844	969.0327
500	50	1000	0	967.5034	982.6344	628.1739
600	40	1200	0.0001	1037.94	1016.892	1065.213
150	50	10000	0.0001	1091.842	1089.249	4624.514
1200	60	700	0.0001	1200.222	1144.887	1222.337
500	200	100	1	1301.968	1251.09	3065.473
800	60	650	0.0001	1339.076	1307.728	1316.282
500	200	10000	1	1305.935	1309.964	603.4571
600	60	700	0.0001	1304.529	1310.178	1153.541
450	60	700	0.0001	1414.17	1429.28	623.7209
1200	100	300	0.0001	1508.089	1495.813	1500.783
800	100	600	0.0001	1551.227	1566.621	1556.673
150	60	1000	1	1576.454	1578.598	248.5271
1200	100	300	0.0001	1700.781	1657.743	1768.646
800	60	900	0.0001	1760.62	1743.945	1762.564
150	150	1000	0.0001	1813.273	1797.636	1811.317
500	200	100	0.0001	2179.81	1988.255	727.9152
800	100	300	0.0001	1908.185	2012.502	1954.933
500	200	1000	1	2000.014	2014.918	619.3801
150	150	100	0.0001	1905.688	2044.525	879.0903
800	100	750	0.0001	2080.799	2082.846	2065.538
150	150	10000	0.0001	2130.893	2129.788	2479.113
600	60	3000	0.0001	2429.311	2416.013	1418.449
800	100	200	0.0001	2748.949	2418.817	2564.578
500	200	1000	0.0001	2537.354	2553.475	5457.948
600	60	1200	0.0001	2567.101	2569.559	637.145
150	150	10000	0.0001	2577.921	2571.433	26.04645
150	150	10000	1	2809.416	2798.491	371.2753
150	60	1000	0	2807.057	2821.94	909.305
150	150	10000	0	3101.128	3103.599	2674.12
500	200	10000	0	3102.585	3108.636	920.4047
150	150	1000	0.0001	3218.607	3192.565	1673.065
500	200	10000	0.0001	3190.922	3195.198	2669.226
500	60	10000	1	3275.976	3271.85	277.0501
150	60	10000	0	3448.225	3441.041	361.5681
1200	100	600	0.0001	4221.653	4162.57	4129.692
150	150	100	0.0001	4301.813	4327.276	1081.442
350	60	700	0.0001	4494.507	4534.769	2521.621
150	150	10000	0	4743.056	4731.765	25587.16
500	200	1000	1	4913.239	4876.219	1581.92
20	20	10000	0	5121.043	5029.728	8181.689
800	100	300	0.0001	5782.29	5249.593	5238.412
150	150	1000	1	5306.957	5260.708	2346.784

800	100	700	0.0001	5253.341	5278.071	5276.68
1200	60	600	0.0001	5844.54	5669.459	5717.319
500	200	1000	0	6201.888	6149.037	1026.69
150	150	1000	1	6876.59	6878.37	13809.02
350	60	3000	0.0001	8689.439	8649.027	9597.157
1200	100	700	0.0001	10629.56	10258.91	10820.11
1200	100	200	0.0001	11914.22	11369.38	12113.01
500	200	100	0	12198.2	11803.9	3409.076
500	200	1000	0.0001	16218.06	16169.87	13400.65
500	200	10000	0.0001	16938.46	16918.78	8058.508
500	200	10000	1	17700.45	17752.37	16599.73
1200	100	750	0.0001	22211.38	22684.67	22277.19
150	150	10000	1	27530.35	27381.24	589.4361
150	150	1000	0	37752.9	37099.18	516.6847
500	200	100	0.0001	263803.2	283727.6	813166.5

We systematically iterated over different values of Number of clusters(k), Sigma_sq and Lambdas.

Inferences:

- **C**hanging Lambda values don't seem to have much of an effect on the optimization process!

- Number of clusters and Sigma_sq have to be tuned together appropriately, in order to get good models.
- When we decrease the number of clusters, we need to give a higher value of sigma_sq for the model to still be able to fit the curve. This is due to the fact that for sparsely distributed clusters, each cluster should be able to control a larger range of input values around their centers.

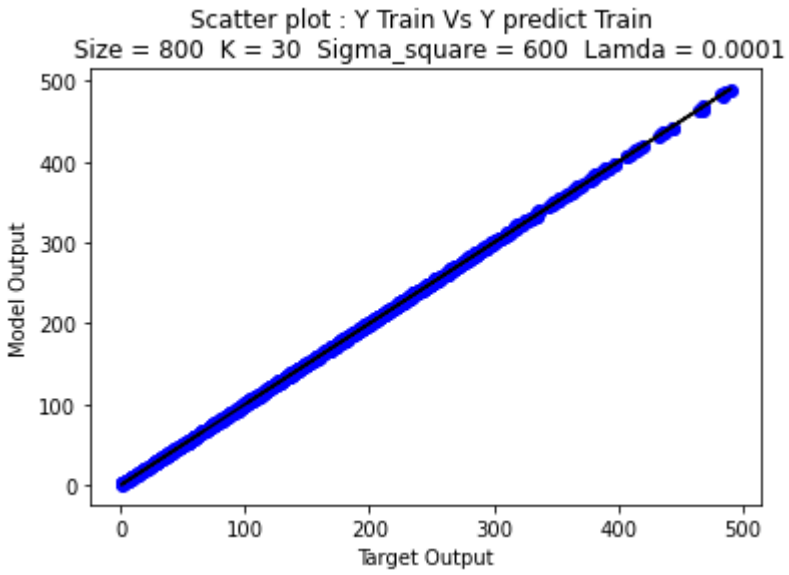
Now we start our iterations for Gaussian Linear models with Quadratic regularization parameters. We need not iterate over every possible value of k, sigma_sq now and only need to see the effect that the Quad_regularization parameter has on the best performing models upto now.

Table 3.1- Gaussian Linear Regression with Quadratic Regularizer

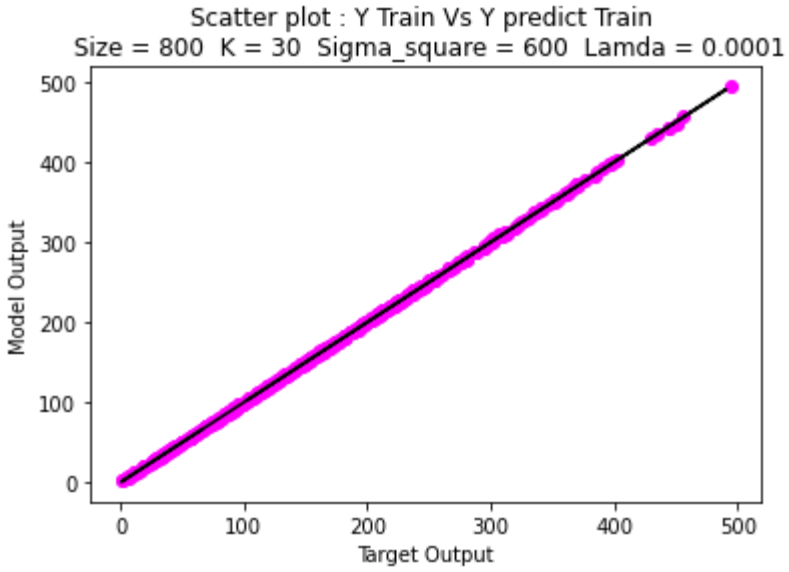
Train_size	K	sigma_sq	lamda	Erms_train	Erms_val	Erms_test
1200	30	600	0.0001	7.185659	6.643011	6.549088
1200	27	600	0.0001	7.838015	7.189716	7.108339
1200	30	650	0.0001	7.789258	7.191632	7.081769
800	30	600	0.0001	7.728277	7.399358	6.954012
1200	27	650	0.0001	8.110947	7.493109	7.436105
1200	30	700	0.0001	8.167644	7.494165	7.493713
1200	25	600	0.0001	8.322918	7.732057	7.550324
1200	27	700	0.0001	8.593409	7.936166	7.871048
800	30	650	0.0001	8.606617	8.15954	7.740935
800	30	700	0.0001	8.719973	8.30339	7.865671
800	25	600	0.0001	8.902417	8.358623	7.938476
1200	25	700	0.0001	9.186158	8.488676	8.405557
800	27	600	0.0001	8.841186	8.515238	8.016788
800	27	650	0.0001	9.241788	8.652266	8.316212
800	25	650	0.0001	9.054732	8.660876	8.208709
800	27	700	0.0001	9.265093	8.66534	8.305043
1200	25	650	0.0001	9.493559	8.802804	8.771881
800	25	700	0.0001	9.906378	9.16138	8.834984
1200	30	600	0.01	15.69652	14.09411	13.57414
1200	30	650	0.01	15.79494	14.2394	13.64567
1200	30	700	0.01	15.89091	14.31528	13.8126
1200	27	650	0.01	16.01661	14.45005	13.85392
1200	27	700	0.01	16.01533	14.48037	13.85771
1200	25	700	0.01	16.16365	14.6662	13.97587
1200	27	600	0.01	16.23319	14.82809	13.92461
800	30	700	0.01	16.53774	15.02724	14.01413
1200	25	600	0.01	16.68785	15.05374	14.35807
1200	25	650	0.01	16.78743	15.15141	14.60312
800	27	700	0.01	16.79336	15.18583	14.30521
800	30	650	0.01	16.86033	15.21251	14.34338
800	25	700	0.01	16.84126	15.23534	14.3447
800	30	600	0.01	16.98472	15.32266	14.41713
800	27	650	0.01	17.08743	15.4955	14.57609
800	25	650	0.01	17.28918	15.68611	14.68075
800	27	600	0.01	17.47492	15.80133	14.87636
800	25	600	0.01	17.86094	16.03775	15.18282

We can see that the quadratic regularization parameter only acts to worsen the overall efficiency of the model! Hence, we conclude that Tihknov regularization is superior to the quadratic regularization in Gaussian Linear models. We now plot the best performing overall model.

Best Overall Model {Size-800, k-30, sigma_sq=600}



Plot 3.1



Plot 3.2

The chosen Gaussian model has fitted the training points as well as generalized to the test data very very well. Even though this model is not as accurate as the best Polynomially fitted model. This is a very versatile and powerful model capable of generalizing to any sort of distribution!

TASK 3 GAUSSIAN: Real World Data:

Train_size	K	Sigma_sq	Lambda	Erms_train	Erms_val	Erms_test
3000	450	500	0.0001	5.886108	6.568659	15.71505
14000	1500	100	0.0001	12.88688	13.85884	13.84803
4000	700	150	0.0001	12.97162	14.09478	14.65809
4000	700	170	0.0001	12.9712	14.23293	14.62805
4000	600	180	0.0001	13.61615	14.68948	13.83797
4000	600	150	0.0001	13.74357	14.79884	13.93215
14000	600	180	0.0001	14.48762	14.98731	14.73059
4000	400	300	0.0001	14.26458	15.02005	14.25065
4000	800	100	0.0001	13.43186	15.04607	15.35932
4000	500	200	0.0001	14.52269	15.15988	15.35391
4000	450	250	0.0001	14.67335	15.24864	15.47099
3000	450	500	0.0001	13.74039	15.3125	14.8648
4000	450	300	0.0001	14.70879	15.93125	15.38836
4000	450	500	0.0001	14.81458	16.06392	15.55992
4000	450	600	0.0001	14.93425	16.12239	15.61031
4000	450	800	0.0001	15.00122	16.14179	15.65666
4000	200	300	0.0001	16.13886	16.43393	16.8071
4000	200	800	0.0001	16.28818	16.60832	16.97067
4000	200	700	0.0001	16.34936	16.93959	16.51868
4000	100	800	0.0001	17.40778	17.45342	17.53674
4000	100	700	0.0001	17.44415	17.8173	17.34654
4000	70	900	0.0001	18.05134	18.27168	17.91309
4000	35	1000	0.0001	19.71846	19.25722	19.77545
4000	35	1000	0.0001	19.7139	19.91526	19.66028
4000	5	1500	0.0001	22.6294	22.60328	22.58955
4000	5	1500	0.0001	24.49957	25.10881	23.98718

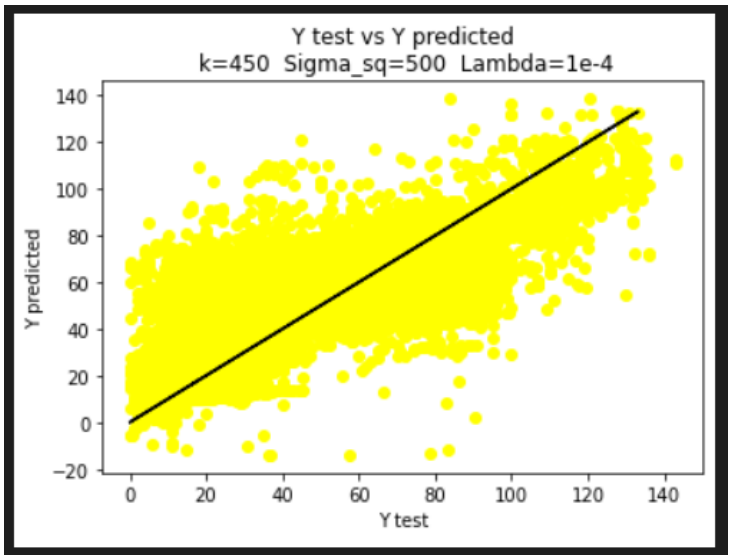
Table 3.2

We had to pre-process the data, before regressing. We normalized the input features using inbuilt sklearn functions in python. And then iterated for different values of K, sigma_sq to see the performance of the prediction. It turned out the models thus predicted were extremely biased working decently on training data and types of test data, but performed much worse on the train data from some other region. We realized that this was due to the fact that in the original data sheet, similar points were grouped together. And based on what data range we give as test and train data, the errors will significantly vary! And the

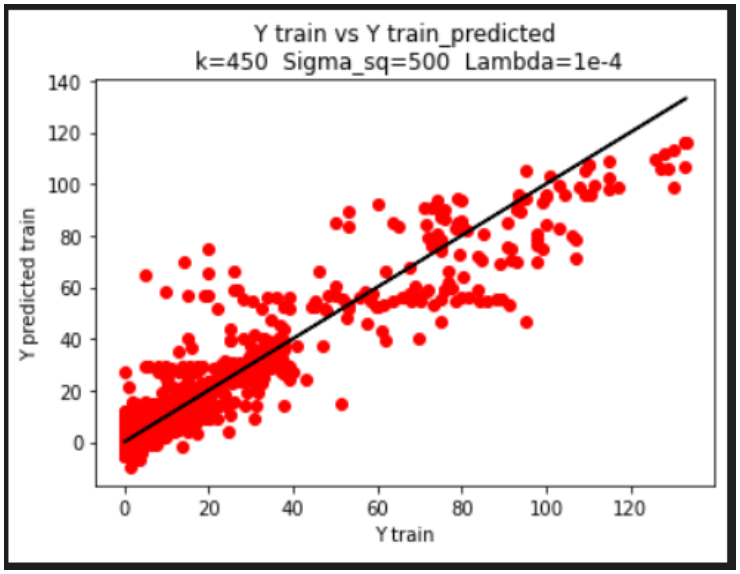
model does not generalize well to all unseen points. To overcome this, we had to shuffle the entire data frame and then run the whole process! Albeit this, the first row of the above table was still kept in the table, as it was the best performing model amongst all the unshuffled models!

We now plot the best performing models on both unshuffled and shuffled data sets!

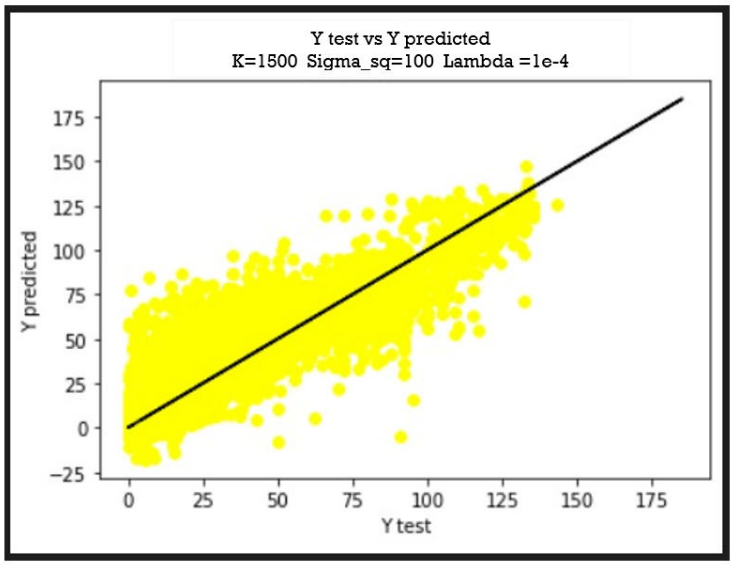
Best Unshuffled Model:



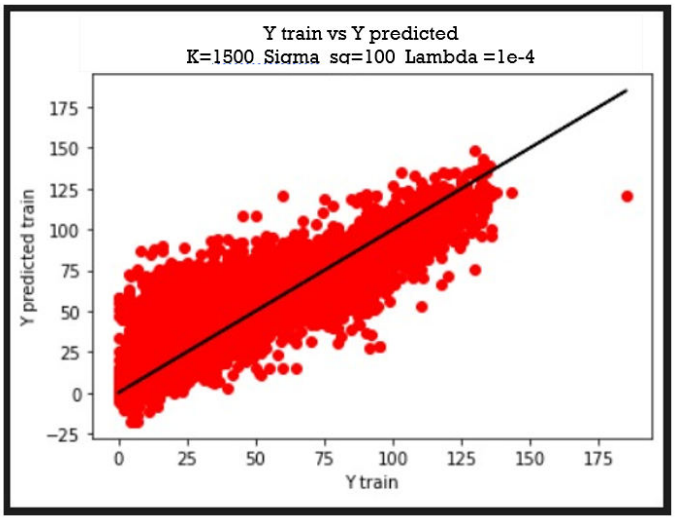
Plot 3.3 a,b



Best Model obtained after shuffling the data frame to remove selection bias:



Plot 3.4 a,b



This model seems to be less biased than the previous model and the width of this model also is lesser than the previous model=> Lesser variance.

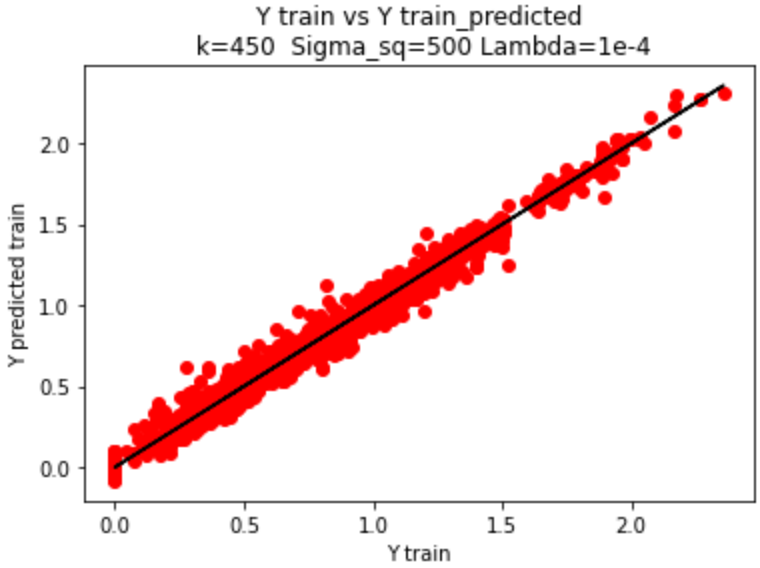
These were the best two models that we could obtain from iterating over a lot of values.

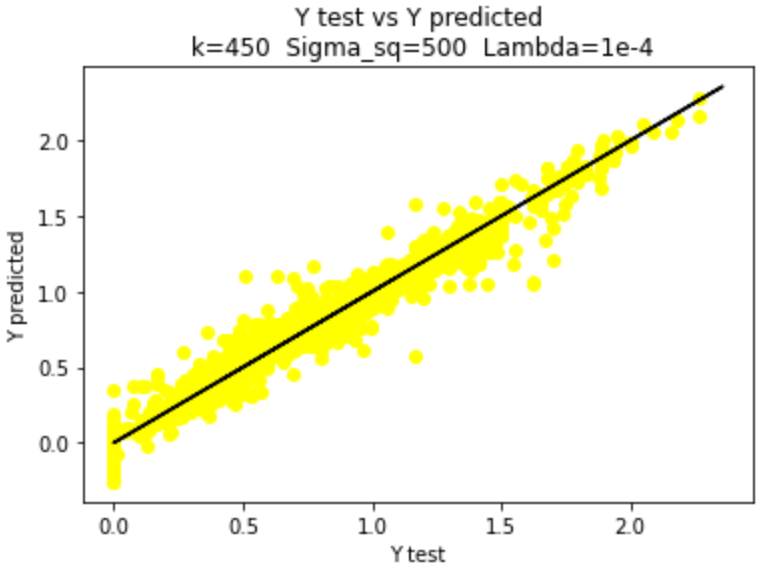
Table 3.3

Train_size	K	Sigma_sq	Lambda	Erms_train	Erms_val	Erms_test
3000	450	500	0.0001	0.047887	**0.459562**	0.070239
3000	250	500	0.0001	0.0803	0.4588	0.0934
2500	250	500	0.0001	0.07718	0.471536	0.101577
1500	400	300	0.0001	0.046366	0.525365	0.085463
1500	500	200	0.0001	0.038512	0.548313	0.081299
1500	600	180	0.0001	0.0254	0.64658	0.0804
3000	600	150	0.0001	0.04526	0.66791	0.075365
1500	700	150	0.0001	0.0209	0.6854	0.0844
1500	600	150	0.0001	0.03186	0.73192	0.07959

Now, we do the same for the last but one column which is the prediction of wtd_std_valence from all the remaining columns! The prediction of this feature worked significantly better than the critical_temp prediction and the error values are very small and the fit is also very good, as can be seen from the graph below!

Best Shuffled Model for wtd_std_val:





Plot 3.5 a,b

We can see that the fit for the wtd_std_valence is very good in our best model!