Report for Assignment-1

-Navam Shrivastav 2019A7PS0092H
-Aayush Keval Shah 2019A7PS0137H
-Aditya Choudhari 2019AAPS0309H

Polynomial Regression is a form of regression in which the relationship between the independent variables and dependent variables is modeled as an nth degree polynomial by changing the value of the degree of polynomial.

For this assignment, we needed to implement Gradient Descent(GD) and Stochastic Gradient Descent(SGD) from scratch and then use them for finding the best fit polynomial of a given degree. We developed a vectorized algorithm for its implementation by making use of matrices via numpy in python. By making use of matrices we were able to find all the coefficients of the polynomials in one go and then by making them train on several iterations we fine tuned them to get the best fit curve.

We preprocessed the data and used min-max normalization to scale the features to range 0 to 1 using pandas and numpy. Then we split the data after shuffling. For each model, we use the functions implemented to carry out various steps like calculating cost and optimization algorithms like gradient descent and stochastic gradient descent. We plotted the cost vs iterations plot for both. Then after finding the trained coefficients we plot the surface of the curve using matplotlib modules. At last, we plot the final testing and training errors against appropriate measures to compare the performance of various models.

Following are the steps which we did for developing the algorithm:

1. Preprocessing:

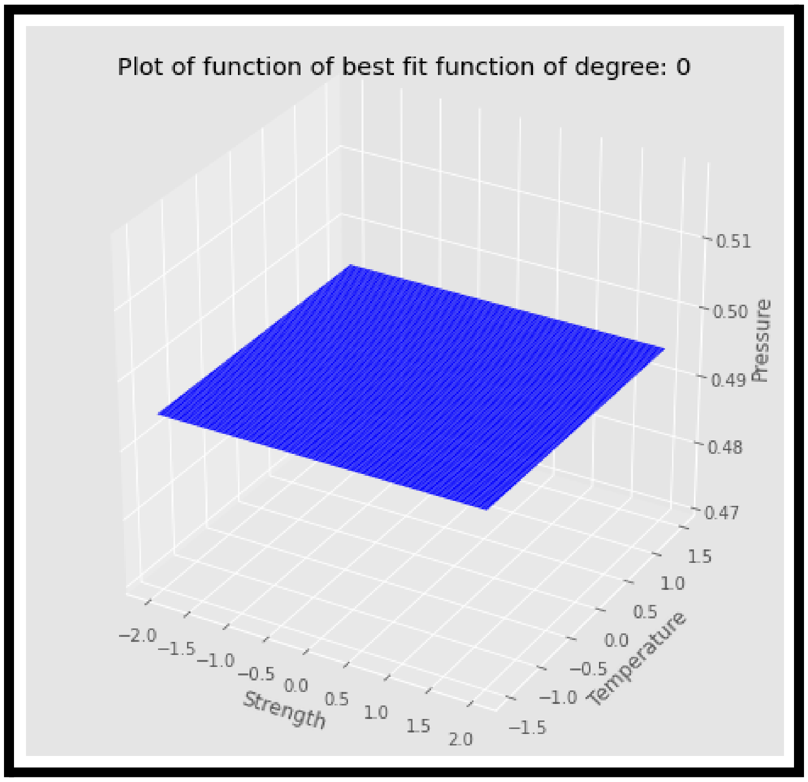
The given dataset has 2 input features named 'Strength' and 'Temperature' which are used to predict the 'Pressure'. The dataset has around 1650 rows of data. The dataset was first converted into an array with the help of numpy and also shuffled with the help of numpy's built-in function of random shuffling. Next, all the data features (input and output both) were min-max normalized. This shuffled and normalised dataset was split into 70% training data and 30% testing data.

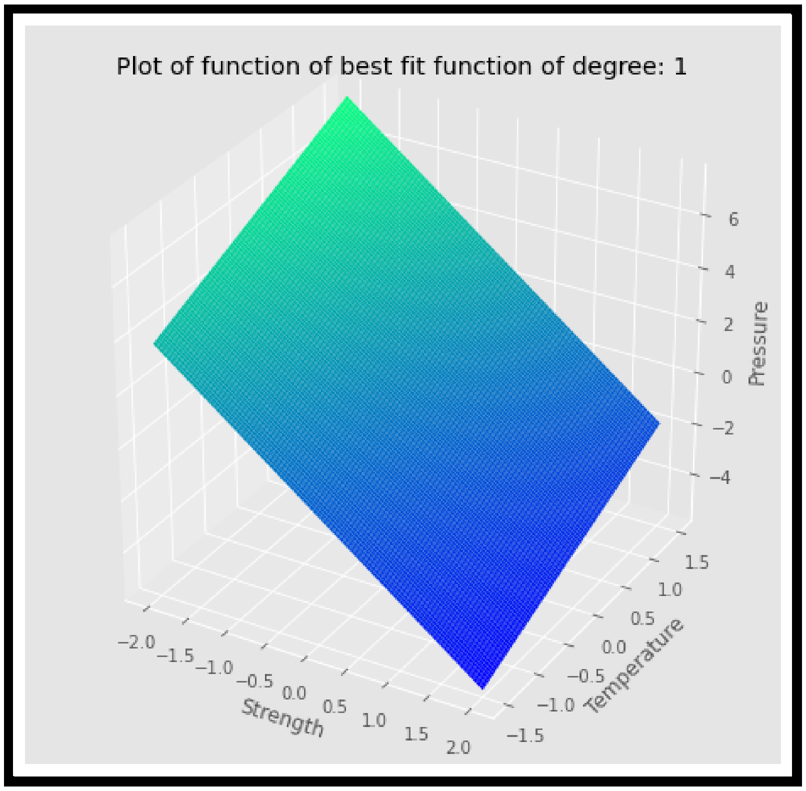
2. Part A:

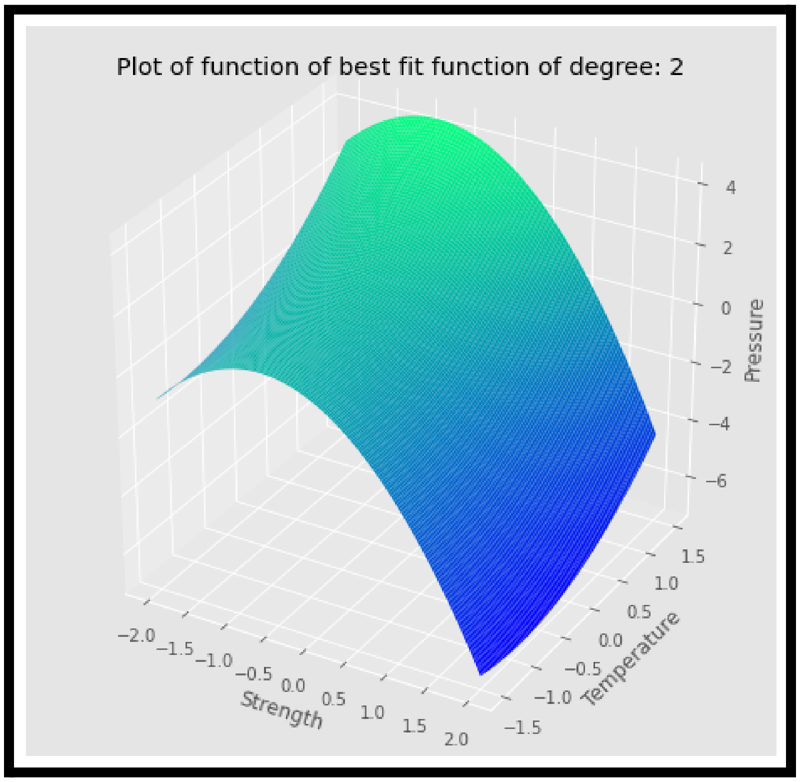
We implemented Gradient Descent and Stochastic Gradient Descent by passing the polynomial feature matrix for degrees varying from 0,1,2,..,9.
Below we tabulated are training and testing errors obtained for each degree.

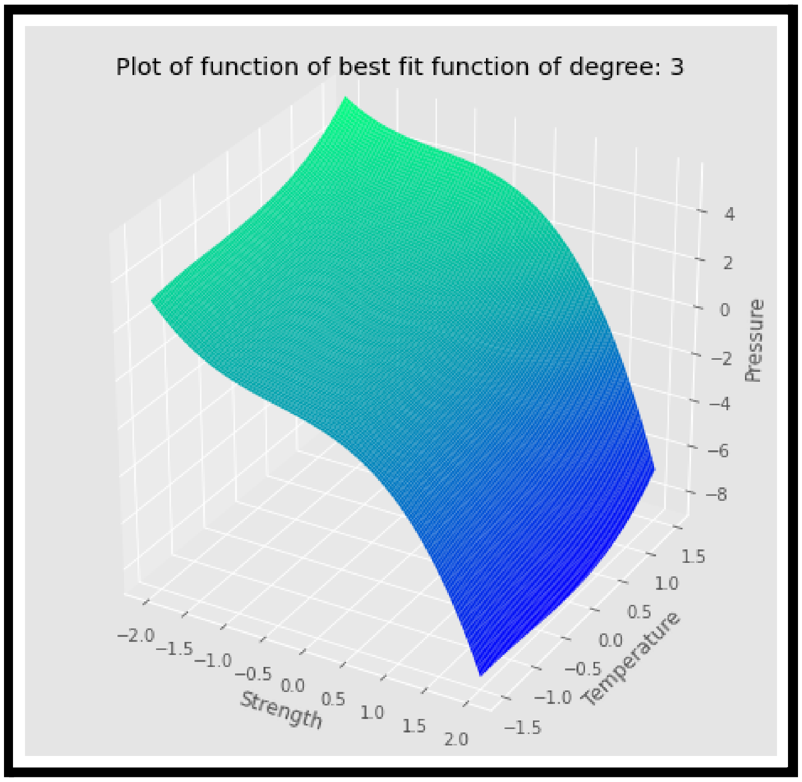
Degree	GD Training Error	GD Testing Error	SGD Training Error	SGD Testing Error
0	0.05925	0.05623	5.198	0.05661
1	0.01371	0.01318	1.625	0.01376
2	0.01392	0.01317	1.417	0.01330
3	0.01536	0.01450	1.777	0.01535
4	0.01532	0.01444	1.774	0.01589
5	0.01478	0.01392	1.673	0.01544
6	0.01433	0.01349	1.510	0.01479
7	0.01412	0.01328	1.651	0.01415
8	0.01411	0.01326	1.811	0.01388
9	0.01422	0.01334	1.426	0.01366

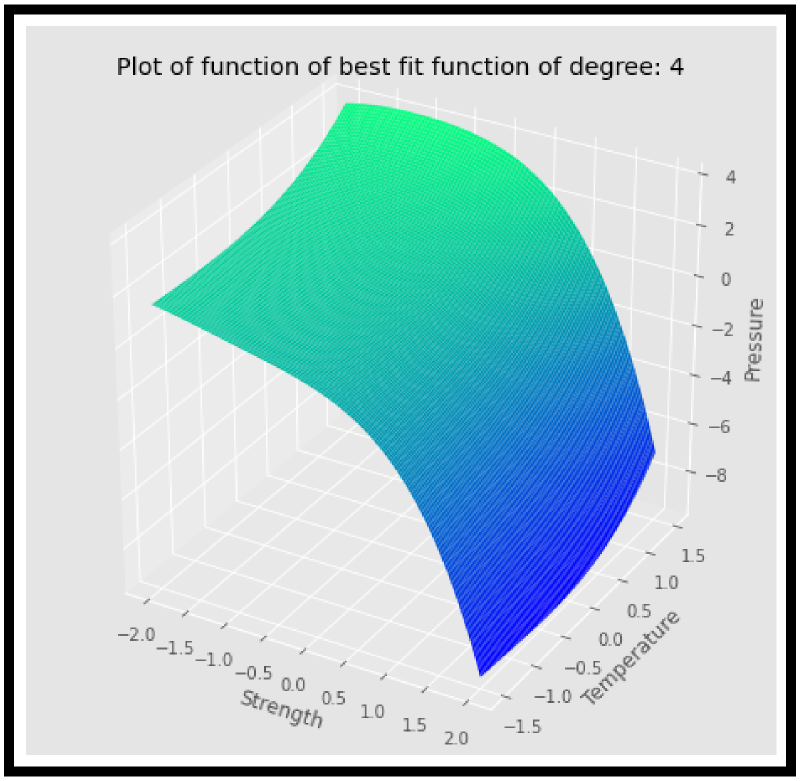
Plots for each degree:

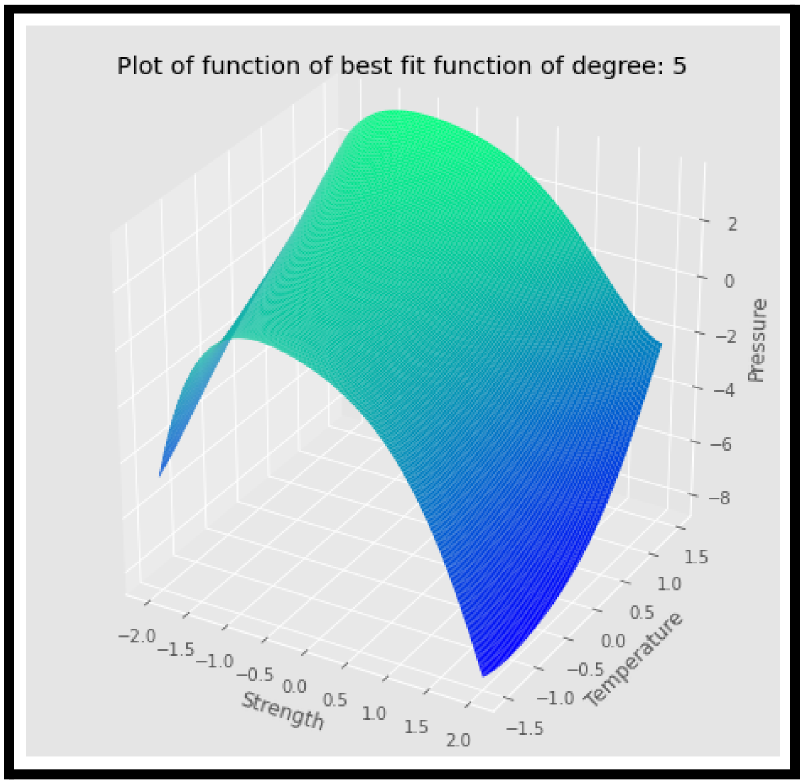


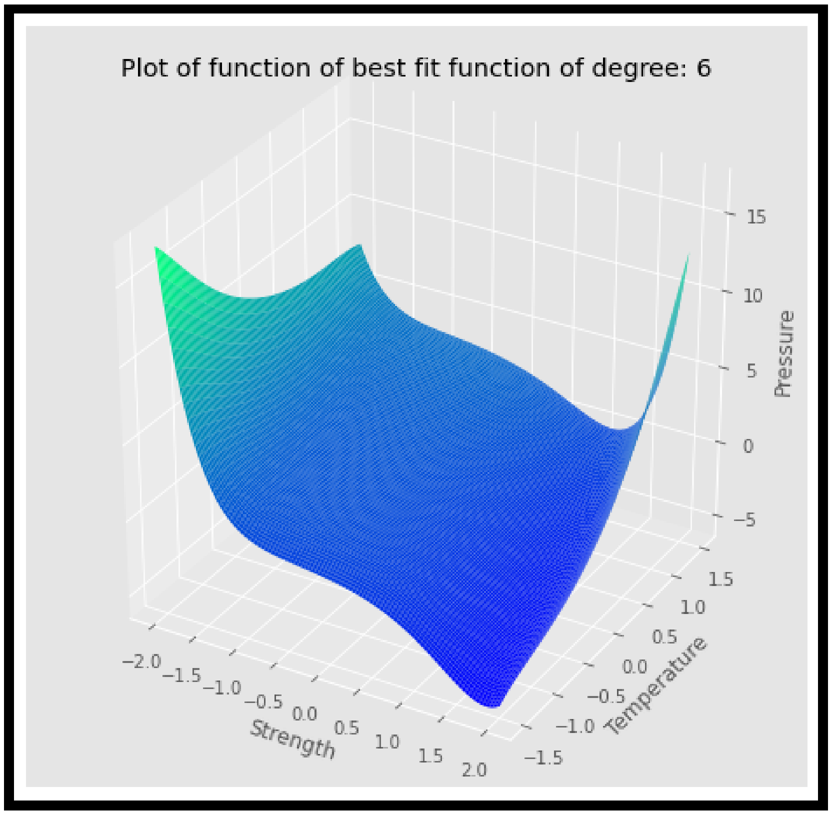


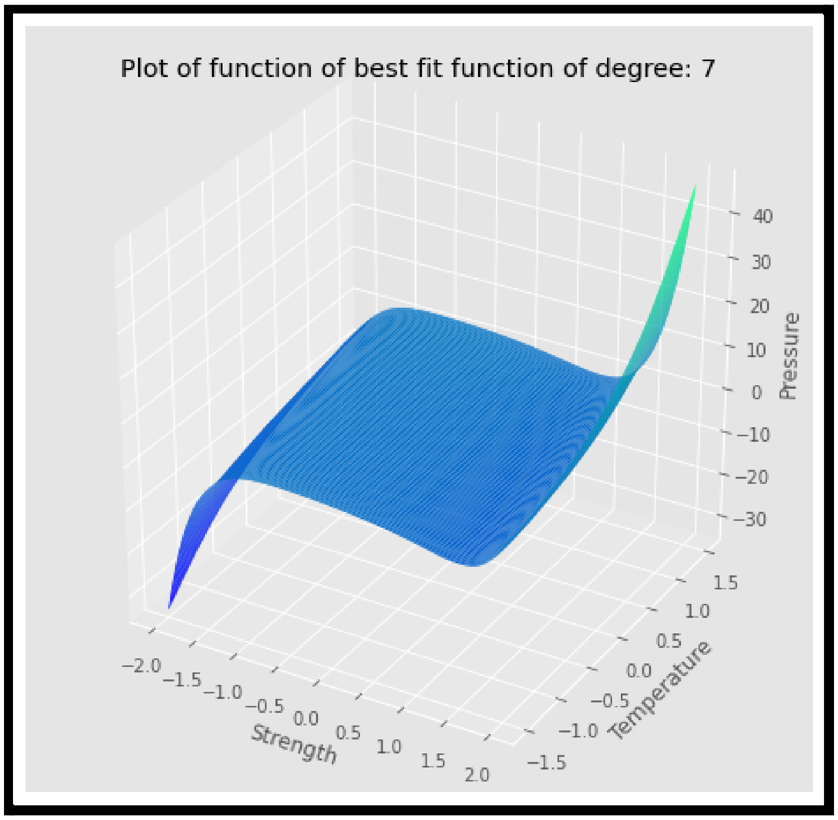

Plot of function of best fit function of degree: 2
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Pressure
4
2
0
−2
−4
−6

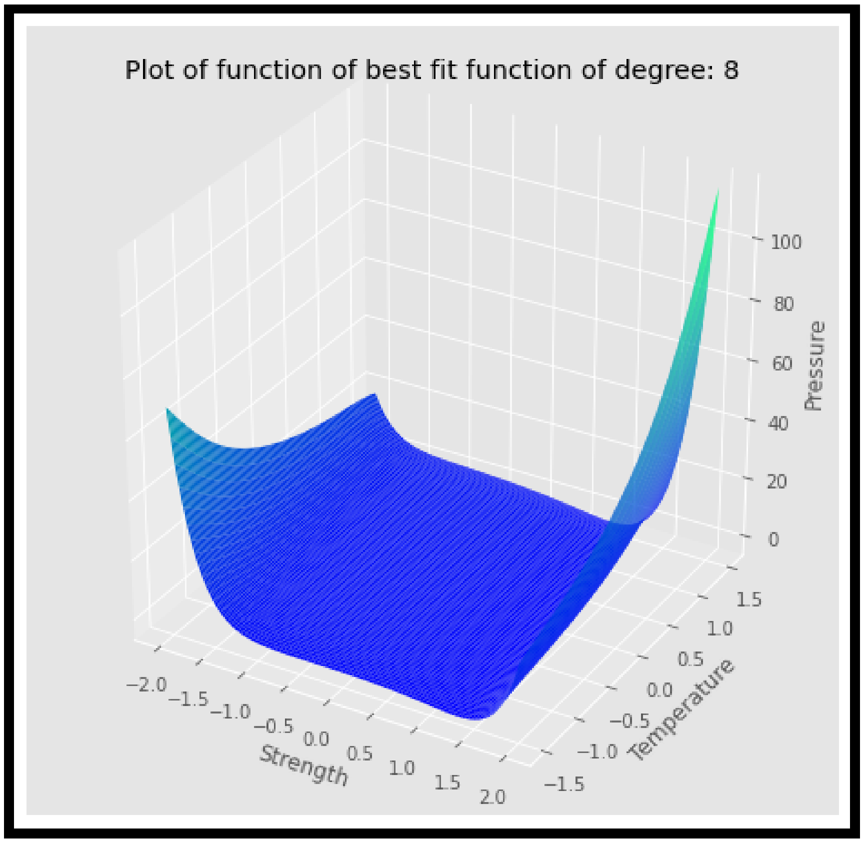

Plot of function of best fit function of degree: 3
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Pressure
4
2
0
−2
−4
−6
−8

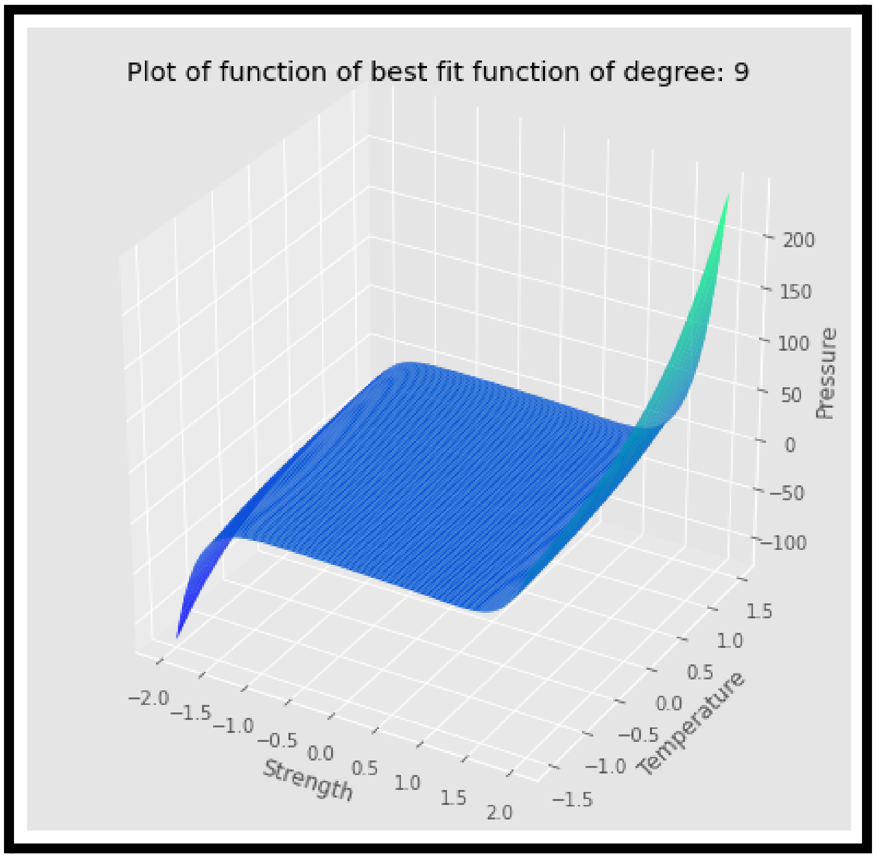

Plot of function of best fit function of degree: 4
Pressure
Strength
Temperature
4
2
0
−2
−4
−6
−8
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Plot of function of best fit function of degree: 5
Pressure
Strength
Temperature
2
0
−2
−4
−6
−8
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

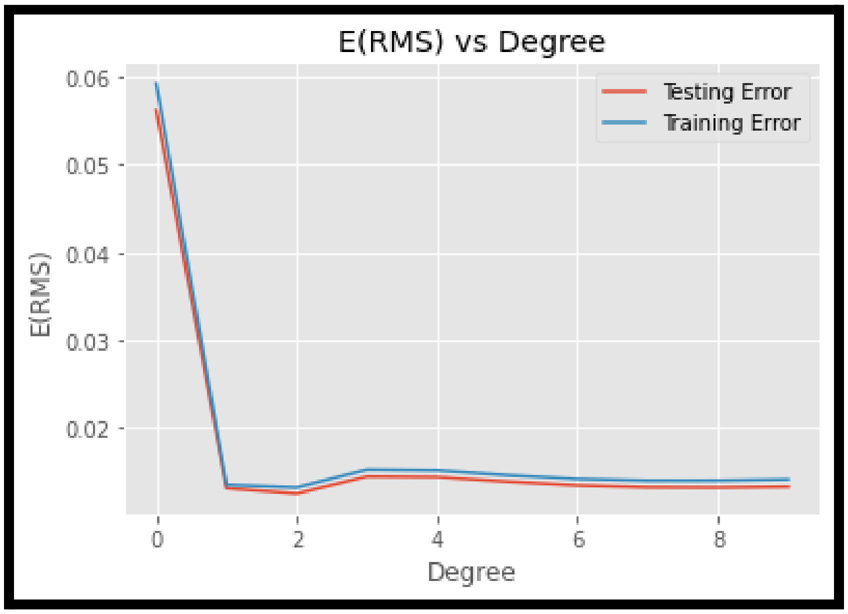

Plot of function of best fit function of degree: 6
Pressure
15
10
5
0
−5
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Plot of function of best fit function of degree: 7
Pressure
40
30
20
10
0
−10
−20
−30
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Plot of function of best fit function of degree: 8
Pressure
100
80
60
40
20
0
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5


Plot of function of best fit function of degree: 9
Pressure
200
150
100
50
0
−50
−100
Strength
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Temperature
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

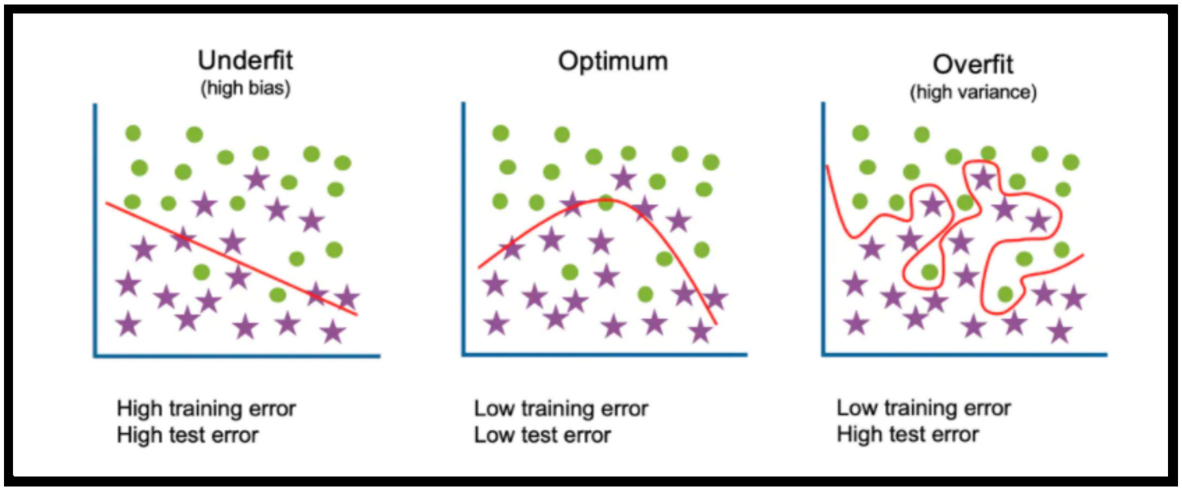
Based on all the values obtained from polynomials of degree 0,1,2,...,9 we plot the training and testing root mean squared error values against degree using gradient descent.



We can see that the training and testing errors both are less for the polynomial of degree 2. So we can say that for given trained models the polynomial regression model with degree 2 is the best fit polynomial for the given dataset.

In data science, overfitting occurs when a statistical model fits exactly against its training data. As a result, the algorithm unfortunately cannot perform accurately against the testing data, which defeats its purpose.

Overfitting usually occurs when the model trains for too long on sample data or when the model is too complex and memorizes the noise in the training set. Such models experience low bias and high variance within their predictions. When fitting a model, the goal is to find the “sweet spot” in between underfitting and overfitting, so that it can establish a dominant trend and apply it broadly to new datasets.



3. Part B:

Overfitting is a phenomenon that occurs when a machine learning or statistics model is tailored to a particular dataset and is unable to generalise to other datasets. This usually happens in complex models, like deep neural networks. Regularisation is a process of introducing additional information in order to prevent overfitting. Lasso (L1) and Ridge (L2) regularisation owes its name to the L1 and L2 norm of a vector w respectively. ‘λ’ is a Hyper-parameter used in regularisation. If ‘λ’ was a parameter, Gradient Descent would nicely set it to 0 and travel to the global minimum. Hence, the control on ‘λ’ cannot be given to Gradient Descent and needs to be kept out. Hence, it will not be a parameter but a Hyper-parameter.

A linear regression model that implements L1 norm for regularisation is called lasso regression, and one that implements L2 norm for regularisation is called ridge regression. So in our model firstly, to implement these two, the linear regression model stays the same.

$$Loss = Error(y, \hat{y})$$

Loss function with no regularisation

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Loss function with L1 regularisation

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Loss function with L2 regularisation

Further, we need to find partial derivatives of error function w.r.t. each coefficient of the polynomial regression function in order to apply SGD and GD. In the image below we show how it is done when regularisation terms are also included.

L:

$$\begin{aligned} w_{\text{new}} &= w - \eta \frac{\partial L}{\partial w} \\ &= w - \eta \cdot \left[2x(wx + b - y)\right] \end{aligned}$$

L1:

$$\begin{aligned} w_{\text{new}} &= w - \eta \frac{\partial L_1}{\partial w} \\ &= w - \eta \cdot \left[2x(wx + b - y) + \lambda \frac{d|w|}{dw}\right] \\ &= \begin{cases} w - \eta \cdot \left[2x(wx + b - y) + \lambda\right] & w > 0 \\ w - \eta \cdot \left[2x(wx + b - y) - \lambda\right] & w < 0 \end{cases} \end{aligned}$$

L2:

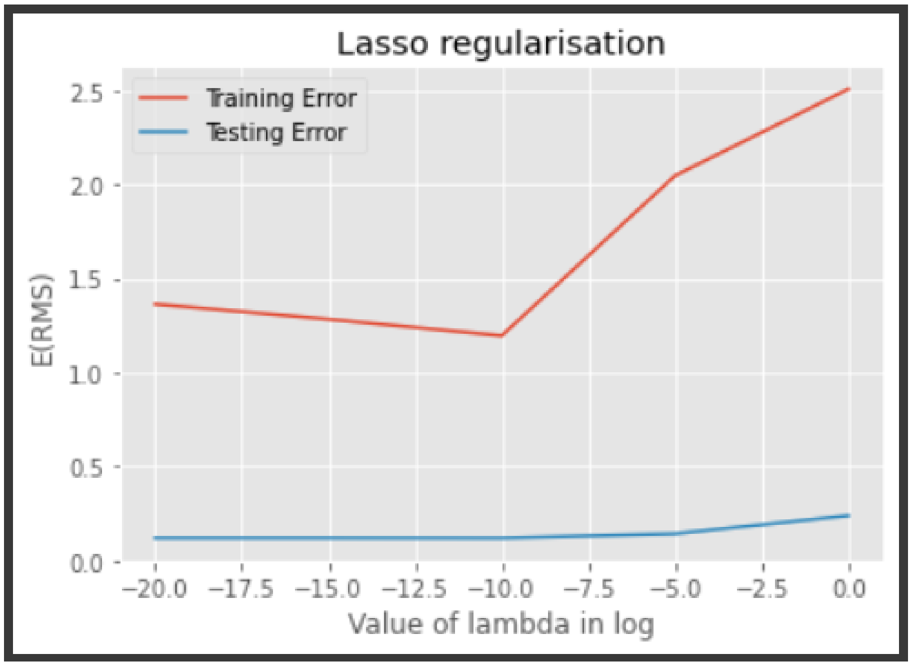
$$\begin{aligned} w_{\text{new}} &= w - \eta \frac{\partial L_2}{\partial w} \\ &= w - \eta \cdot \left[2x(wx + b - y) + 2\lambda w\right] \end{aligned}$$

Lasso Regression:

Below we have tabulated (with particular precision) the minimum training and testing errors achieved using regularisation for GD, SGD for 9th degree polynomial for 5 values of λ for lasso regularisation:

ln(λ)	GD training error	GD testing error	SGD training error	SGD testing error
0	0.01898	0.02390	6.471	0.05664
-5	0.01454	0.01359	4.194	0.02002
-10	0.01450	0.01347	1.431	0.01394
-15	0.01450	0.01347	1.647	0.01406
-20	0.01450	0.01347	1.860	0.01408

Based on this table we can see that this model works better with λ = e^{-10} value. Thus we can see that the regularization works better with that particular value of penalty applied to penalize the weights of the model.
Following is the plot of E(rms) vs ln(λ) for the above mentioned lasso regression using the Stochastic Gradient Descent optimization algorithm.

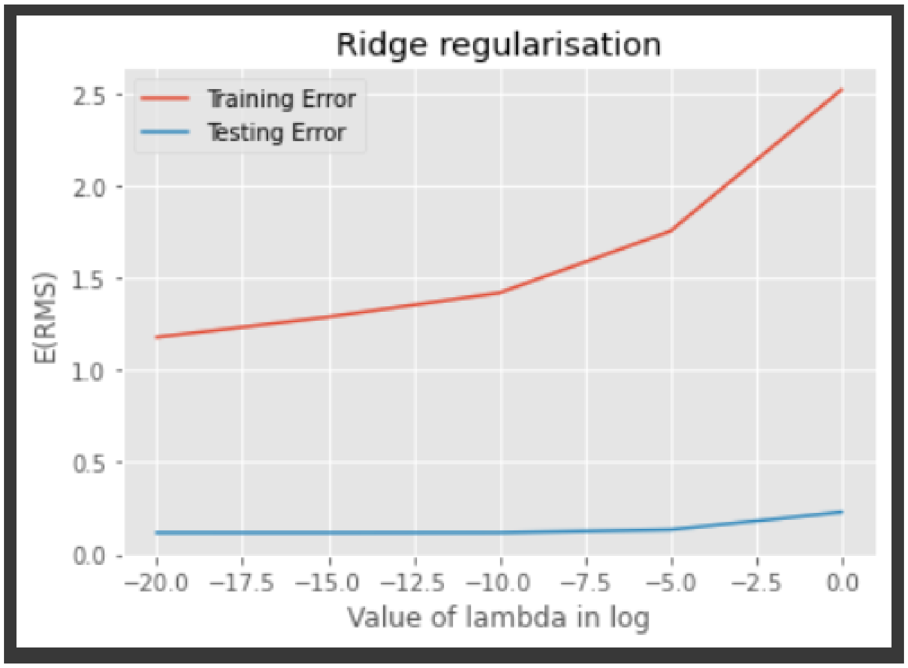


Ridge Regression:

Below we have tabulated (with particular precision) the minimum training and testing errors achieved using regularisation for GD, SGD for 9th degree polynomial for 5 values of λ for ridge regularisation:

ln(λ)	GD training error	GD testing error	SGD training error	SGD testing error
0	0.01611	0.01731	6.353	0.05267
-5	0.01424	0.01336	3.075	0.01790
-10	0.01422	0.01334	2.012	0.01378
-15	0.01422	0.01334	1.659	0.01366
-20	0.01422	0.01334	1.388	0.01368

Based on this table we can see that this model works better with λ = e^{-20} value. Thus we can see that the regularization works better with that particular value of penalty applied to penalize the weights of the model.
Following is the plot of E(rms) vs ln(λ) for the above mentioned ridge regression using the Stochastic Gradient Descent optimization algorithm.



Comparison between best models of part A and part B:

The best model of the part A (to train models without regularization) based on both Gradient Descent and Stochastic Gradient descent is when the model is a polynomial of degree 2 with given features and target.

The best model of the part B (to train model of degree 9 with regularization)

1) For lasso regression based on both Gradient Descent and Stochastic Gradient descent is when the ln(λ) is -10.
2) For ridge regression based on both Gradient Descent and Stochastic Gradient descent is when the ln(λ) is -20

We can see that:

Model	GD training error	GD testing error	SGD training error	SGD testing error
Polynomial regression with degree 2	0.01392	0.01317	1.417	0.01330
Polynomial lasso regression with degree 9 ln(λ) = -10	0.01450	0.01347	1.431	0.01394
Polynomial ridge regression with degree 9 ln(λ) = -20	0.01422	0.01334	1.388	0.01368

Based on above values we can see that the polynomial regression model with degree 2 works better than other models based on the training and testing errors calculated with gradient descent and stochastic gradient descent optimization algorithms.