**CS F320 ASSIGNMENT-1**

**Group member ID**

Anjan Neelisetty 2019A8PS0367H

Praneet Sai Madhu Surabhi 2019A7PS0060H

Yarramsetty Sanjeeva Sai Preetham 2019A3PS0485H

Under the supervision of

**Prof. N.L. Bhanu Murthy**

**SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS OF**

**CS F320: Foundations of Data Science**

**Assignment-1**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**HYDERABAD CAMPUS**

**(October 2021**)

**CS F320 Assignment 1 Report**

Polynomial Regression:  
Polynomial Regression is a type of linear regression that models the connection between the independent variable x and the dependent variable y as an nth degree polynomial. We use it when we hypothesize that the relations are curvilinear. In traditional multiple linear regression analysis, all of the independent variables are assumed to be independent. This assumption is not met in a polynomial regression model.

Preprocessing:

The given dataset has 2 column features named strength and temperature to predict

the pressure and around 1650 rows of data.Then we create a random 70-30 split to aid in training and testing respectively .Data is normalized using mean and variance method . This shuffled and normalised dataset .Then we generated a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to that of 9.

**Model, Algorithms and Regularization:**

We have two features which are strength and temperature and we predict the target attribute pressure using a polynomial regression model. We construct matured polynomial features from the given features in the dataset according to the degree of the polynomial required.

We use two algorithms to train our model and minimize the error.

**1.** **Batch Gradient Descent:**

We minimize the error function by updating the values of all the parameters after every iteration as shown below. Here the partial derivatives are calculated (updating is done) by taking the whole data.

2. **Stochastic Gradient Descent**:

We compute partial derivatives using a single datapoint in every iteration.

**Regularization:**

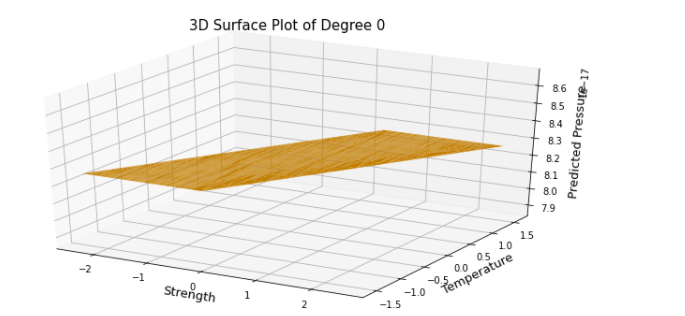
When a model is able to fit the training data well but fails to predict the testing data as well as training then we call it overfitting. We tend to overfit the data if we try to fit a higher order polynomial with a smaller number of training instances because parameters have the freedom to take very high or very low values. If we try to limit the freedom given to parameters, we can fit a higher order polynomial with a smaller number of training instances as well.

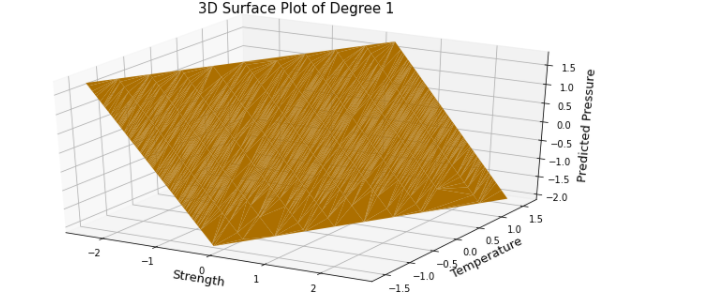
Ridge Regression (constraint on vector norm) and Lasso regression (Constraint is placed on sum of magnitudes of parameters) are two regularisation approaches. These are constrained optimization problems that can be transformed into unconstrained optimization problems with the help of Lagrange multipliers.

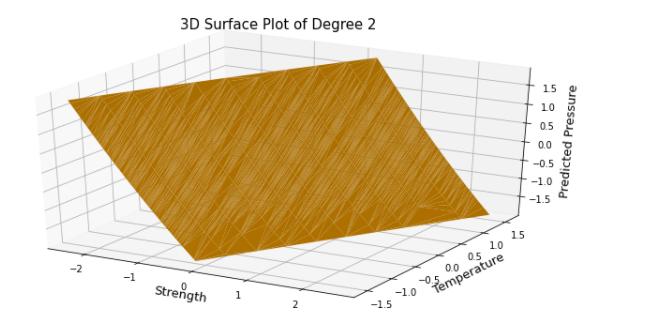
**Minimum training and testing error achieved by our model by using polynomials of degree 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 for Gradient descent**

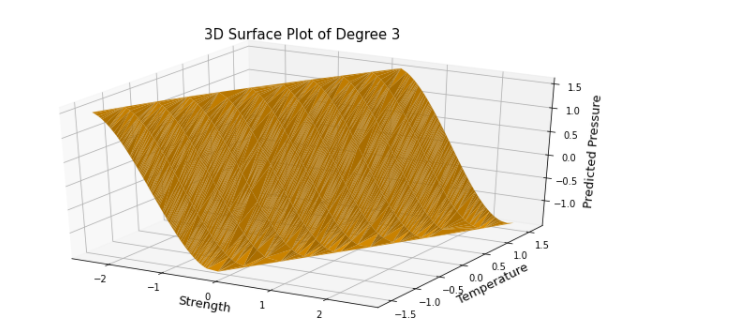
| **Degree** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0** | **0.9991341991342072** | **0.9914876777204766** |
| **1** | **0.2017556764902804** | **0.1933399200050467** |
| **2** | **0.20058393399328803** | **0.19021968091063876** |
| **3** | **0.1860374971964633** | **0.17691982903388376** |
| **4** | **0.19397362188994458** | **0.18322828561318702** |
| 5 | **0.1951512462363467** | **0.18567661935948387** |
| **6** | **0.23593736270388968** | **0.2214138853257371** |
| **7** | **0.3935417525904578** | **0.3815428739786793** |
| **8** | **0.5441078273452796** | **0.5340099605343164** |
| **9** | **0.6612300542381736** | **0.6557016207700365** |

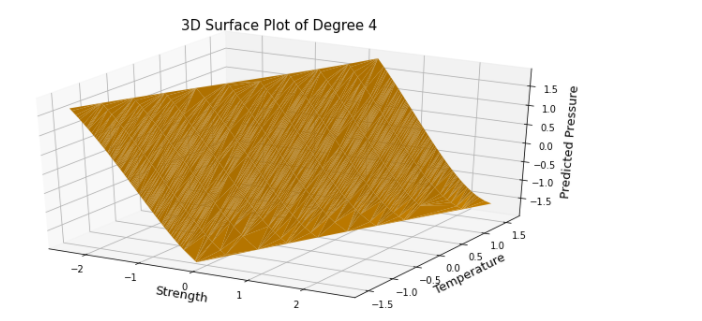
**Surface plots**

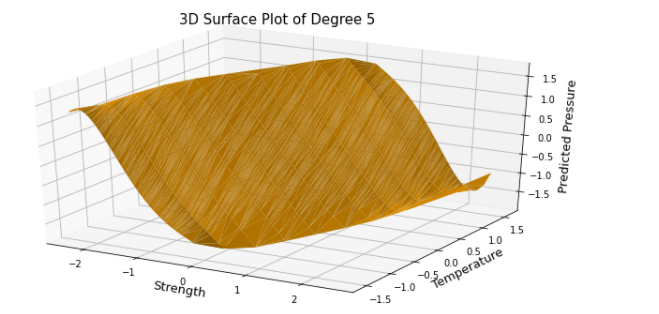


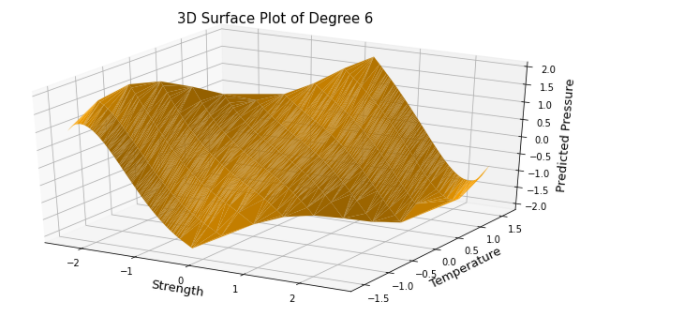


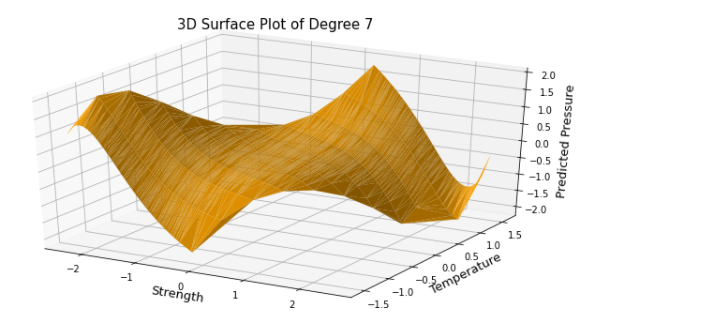


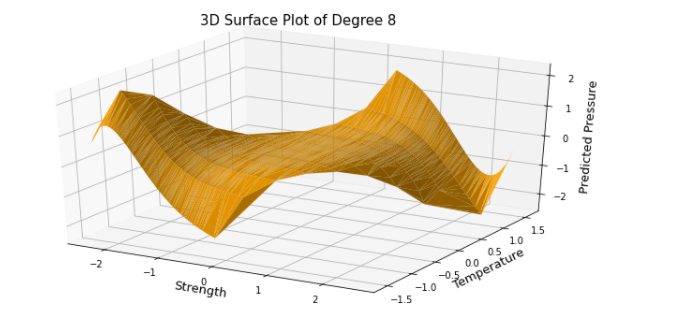


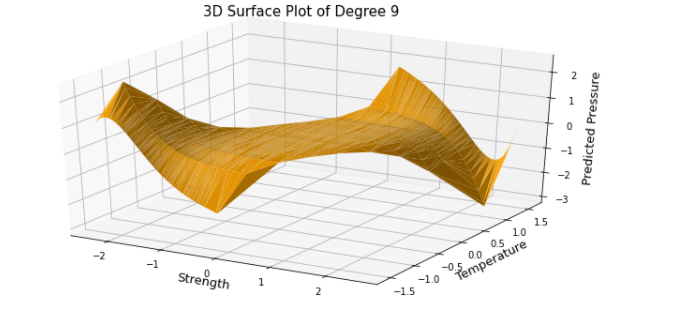




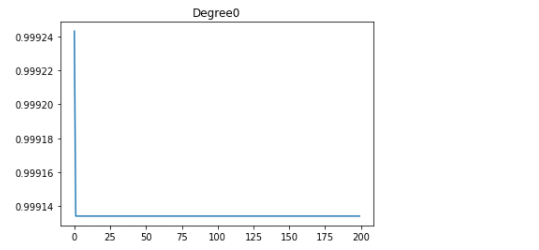


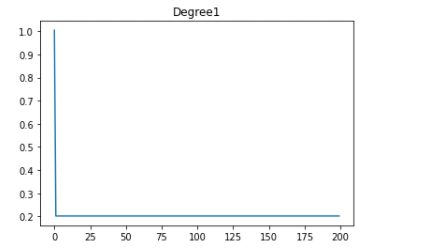


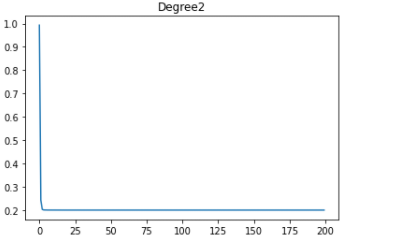


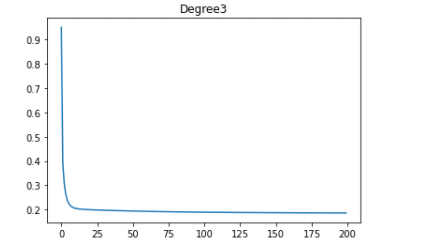


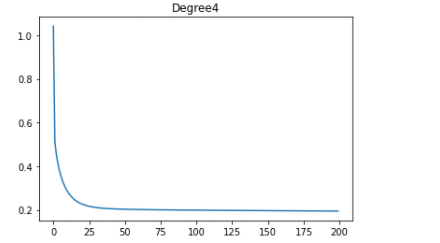
.

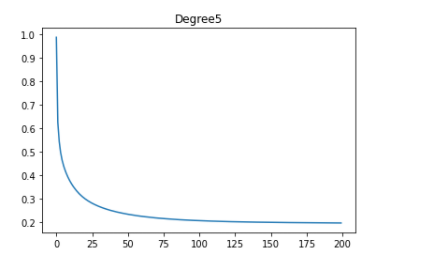
**Graphs of Training error vs number of iterations (\*50) for Gradient Descent  
**

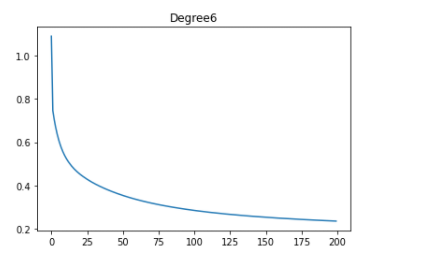
****

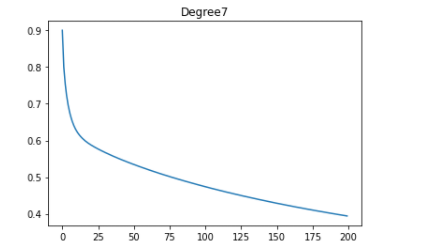
****

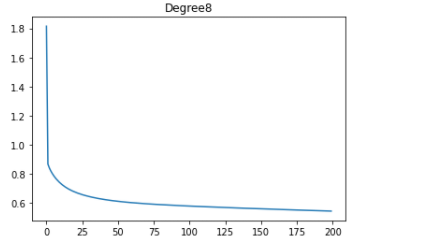
****

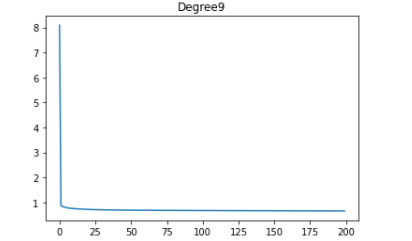
****

****

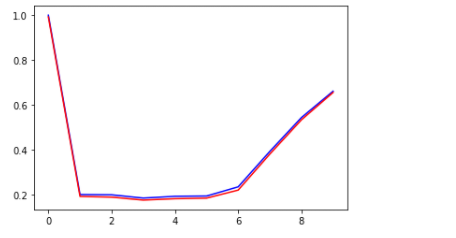
****

****

****

****

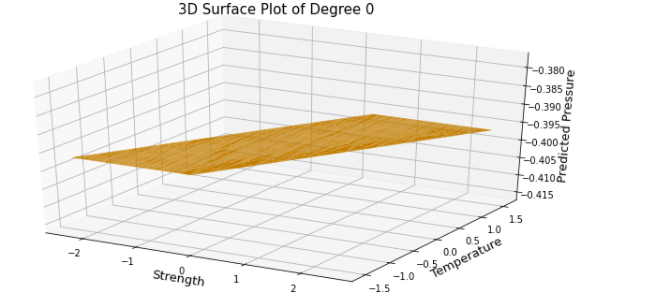
**Graph of Degree vs training(blue) and testing errors(red)**

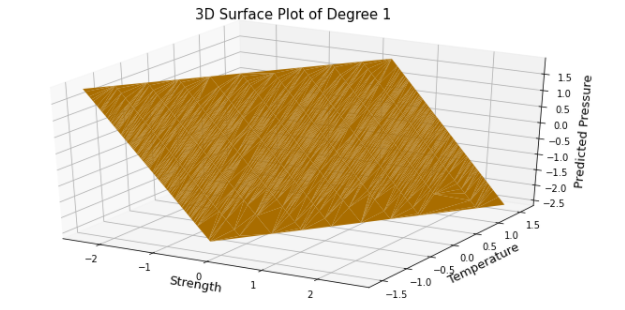
****

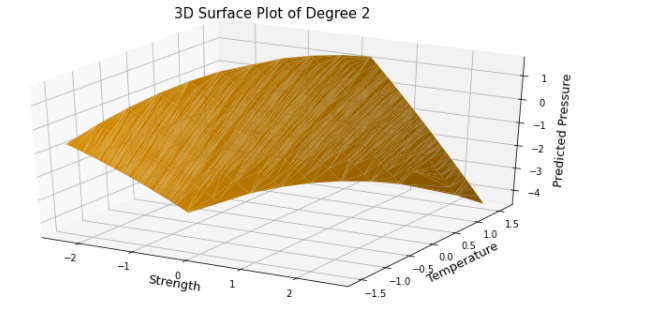
**Minimum training and testing error achieved by our model by using polynomials of degree 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 for stochastic Gradient descent**

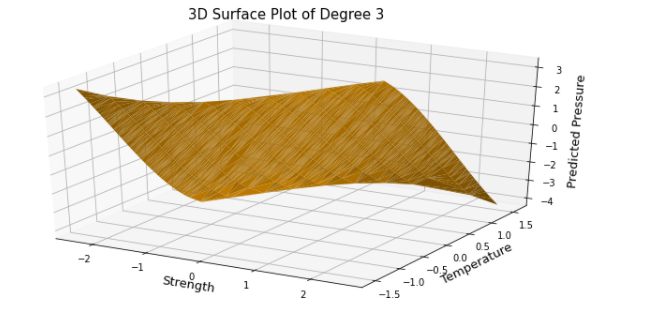
| **Degree** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0** | **1.1564896086441794** | **1.0826053521817256** |
| **1** | **0.24897264028848842** | **0.245029938577069** |
| **2** | **1.0054638892557402** | **0.9530942099394092** |
| **3** | **0.4189133134081476** | **0.40289505697030287** |
| **4** | **0.3098572530197008** | **0.2974428238286299** |
| **5** | **0.4187898824945824** | **0.3677001393460043** |
| **6** | **0.3697723879823114** | **0.3556881925508624** |
| **7** | **0.6837781277609277** | **0.6770300786341963** |
| **8** | **0.7550940364225152** | **0.7474921404412469** |
| **9** | **0.7809072327433831** | **0.7685686600777419** |

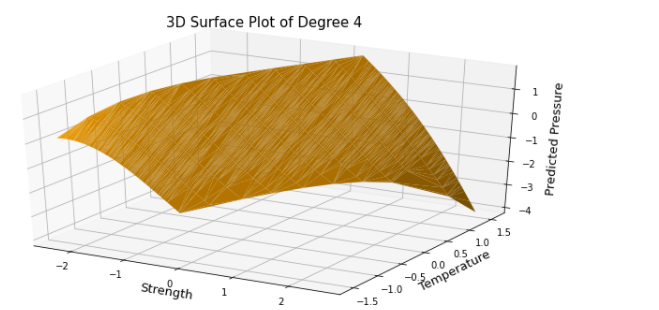
**Surface plots for stochastic Gradient descent**

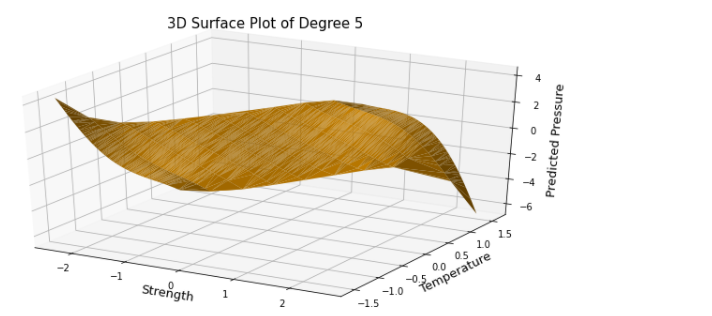
****

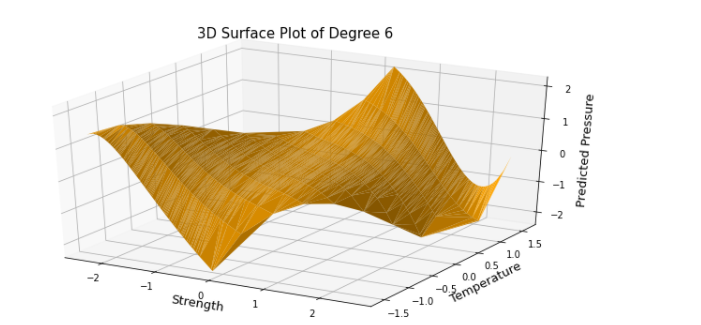
****

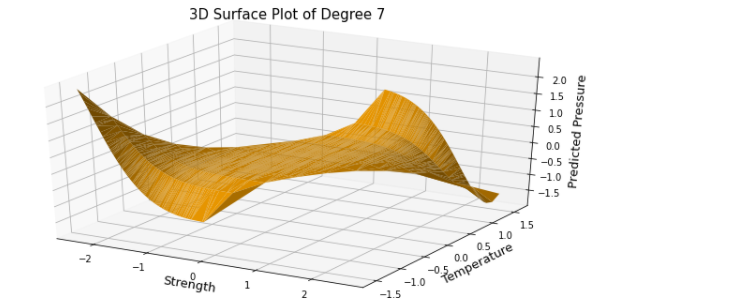
****

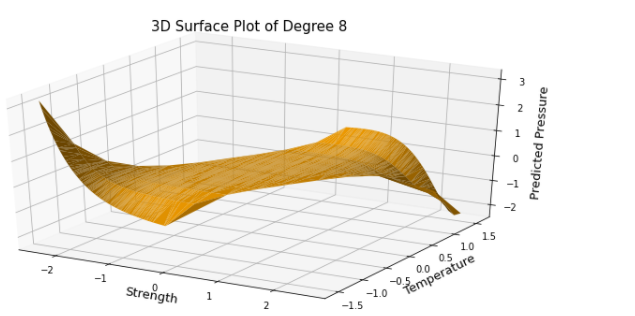
****

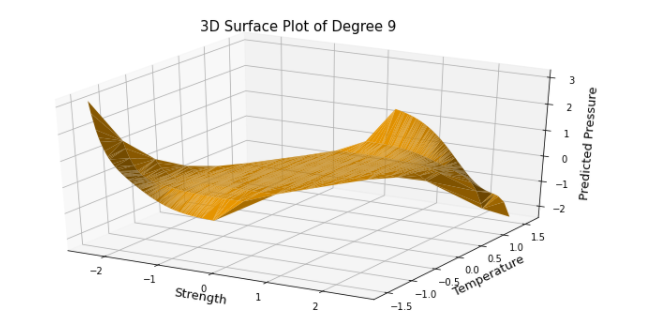
****

****

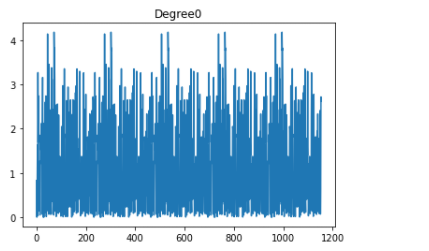
****

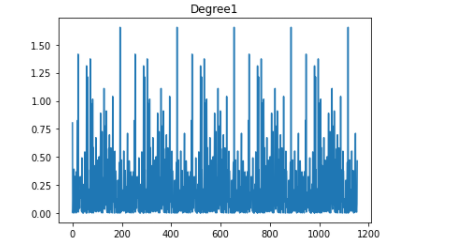
****

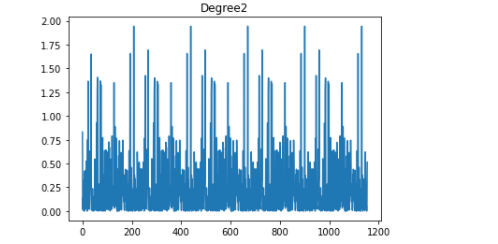
****

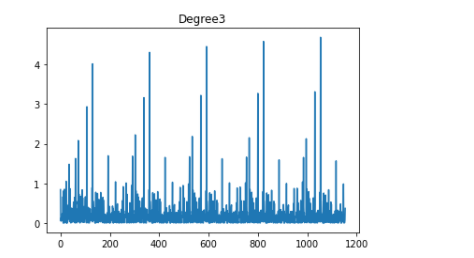
****

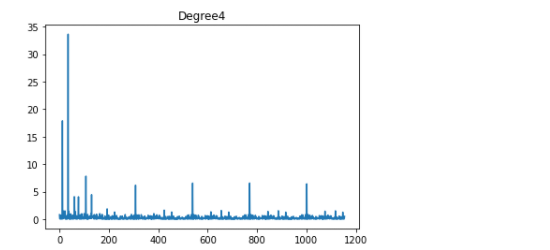
**Graphs of Training error vs number of iterations (\*50) for stochastic Gradient Descent**

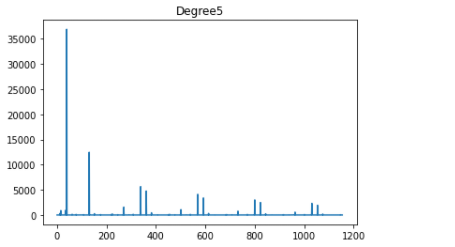
****

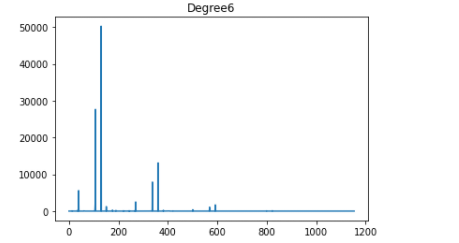
****

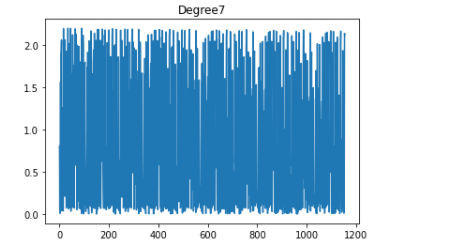
****

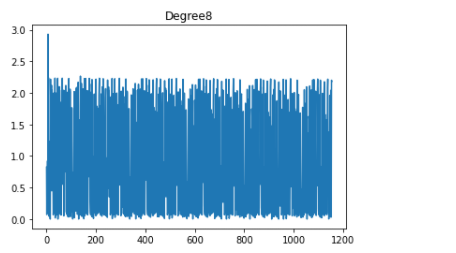
****

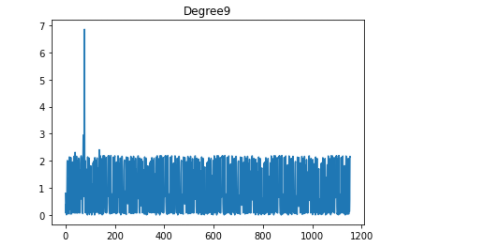
****

****

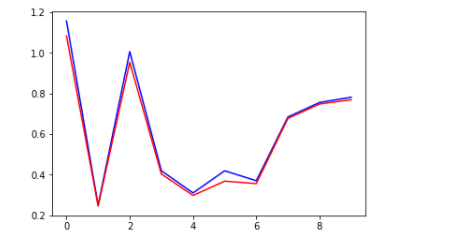
****

****

****

****

**Graph of Degree vs training(blue) and testing errors(red)**

****

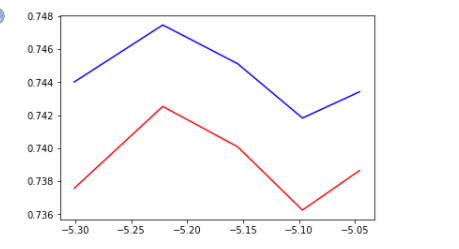
**Comment on how overfitting actually works:**When a model learns the information and noise in the training data to the point where it degrades the model's performance on fresh data, this is known as overfitting. This means that the model picks up on noise or random fluctuations in the training data and learns them as ideas.

**b)  
Minimum training and testing error achieved by your model for 5 different values of lambda for gradient descent with ridge regression**

**In ROOT MEAN SQUARE of errors**

| **lambda** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0.000005** | **0.744014** | **0.737577** |
| **0.000006** | **0.747466** | **0.742534** |
| **0.000007** | **0.745121** | **0.740088** |
| **0.000008** | **0.741829** | **0.736253** |
| **0.000009** | **0.743420** | **0.738654** |

**root-mean square error vs the logarithm of lambda**

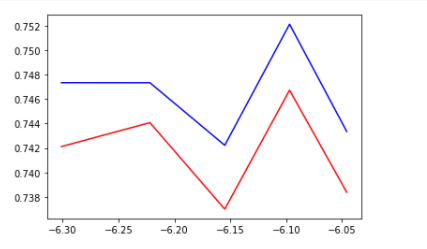
****

**Minimum training and testing error achieved by your model for 5 different values of lambda for gradient descent with lasso regression**

**In ROOT MEAN SQUARE of errors**

| **lambda** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0.0000001** | **0.747336** | **0.742114** |
| **0.0000004** | **0.747332** | **0.744070** |
| **0.0000007** | **0.742216** | **0.737001** |
| **0.000001** | **0.752117** | **0.746724** |
| **0.0000013** | **0.743341** | **0.738375** |

**root-mean square error vs the logarithm of lambda**

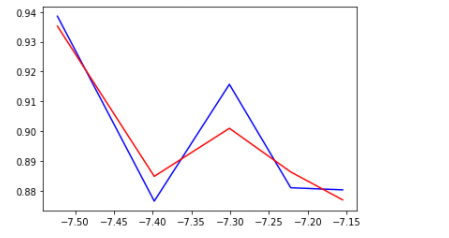
****

**Minimum training and testing error achieved by our model for 5 different values of lambda for stochastic gradient descent with Ridge regression**

**In ROOT MEAN SQUARE of errors**

| **lambda** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0.00000003** | **0.938573** | **0.935232** |
| **0.00000004** | **0.876492** | **0.884807** |
| **0.00000005** | **0.915709** | **0.900938** |
| **0.00000006** | **0.880978** | **0.886281** |
| **0.00000007** | **0.880266** | **0.876919** |

**Root-mean square error vs the logarithm of lambda**

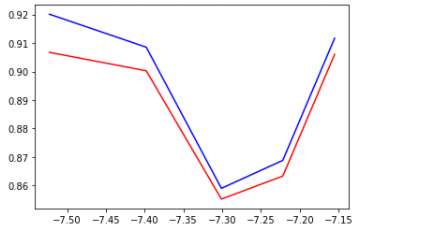
****

**minimum training and testing error achieved by our model for 5 different values of lambda for stochastic gradient descent with lasso regression**

**In ROOT MEAN SQUARE of errors**

| **lambda** | **Training error** | **Testing error** |
| --- | --- | --- |
| **0.00000003** | **0.920119** | **0.906729** |
| **0.00000004** | **0.908540** | **0.900271** |
| **0.00000005** | **0.859037** | **0.855248** |
| **0.00000006** | **0.868833** | **0.86332** |
| **0.00000007** | **0.911709** | **0.906051** |

**Root-mean square error vs the logarithm of lambda**

****

**Comparison of best models**

The best model obtained from part ‘a’ is of Degree 4 with Gradient Descent and the best model obtained from part ‘b’ is Degree 9 with Gradient Descent and Ridge Regularization. The first model has a test error of 0.177 and the second model has a test root mean square error of 0.737. Hence the model fit with Degree 4 seems to be working better.