A Project Report

On

Polynomial Regression (Assignment-3)

BY

KESHAV BERIWAL	2017B4A71301H
AMAN BADJATE	2017B3A70559H
GARVIT SONI	2017B3A70458H

UNDER THE SUPERVISION OF

DR. N.L.BHANU MURTHY

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF

CS F320: FOUNDATIONS OF DATA SCIENCE

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

HYDERABAD CAMPUS

OCTOBER - NOVEMBER, 2020

Contents

List of Figures

1 Objectives

The main focus of this assignment is to implement various Linear Regression techniques using Regularization. The following techniques are implemented in this assignment in python using numpy, pandas, matplotlib and sklearn.preprocessing libraries.

- Gradient Descent Algorithm using both Lasso and Ridge Regression
- Stochastic Gradient Descent Algorithm using both Lasso and Ridge Regression

2 Data Preprocessing

The data consists of 1338 rows and 4 columns. The 4 columns are age, bmi, children and charges. "age, bmi, children" are features (independent variables) and "charges" is the target variable (dependent variable). The children variable is dropped as given in the assignment. After this, we used the sklearn.preprocessing.PolynomialFeatures class to generate the polynomial features for the data. A graph is shown in Fig 1, which depicts how the number of features in design matrix changes on changing the polynomial degree.

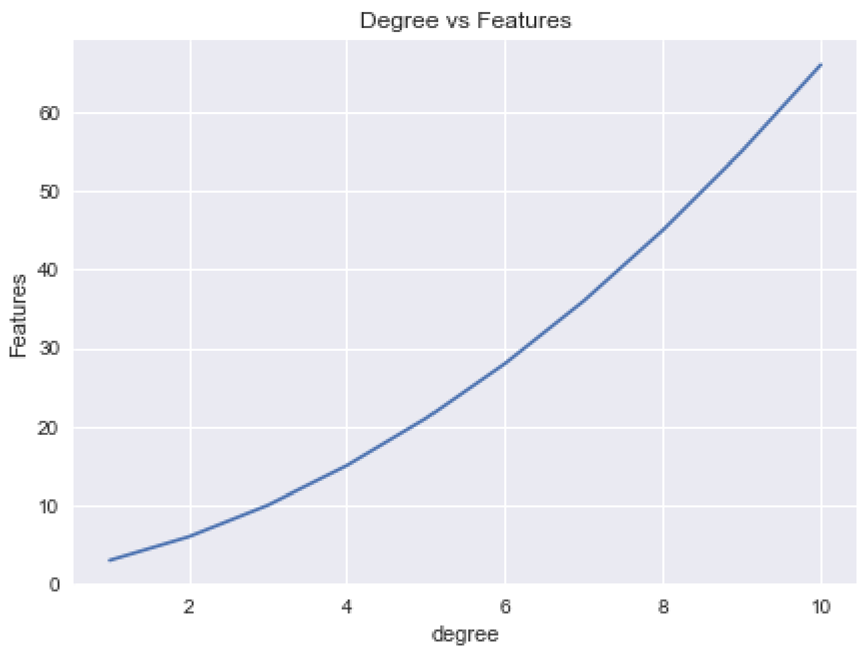


Figure 1: Polynomial Degree vs features Graph

Before training the model, the data was pre-processed by standardizing all the variables. Standardizing a variable means subtracting it from its mean value and then dividing it by standard deviation. The comparison between the original and standardised data is shown in figure 2. This is done to ensure that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

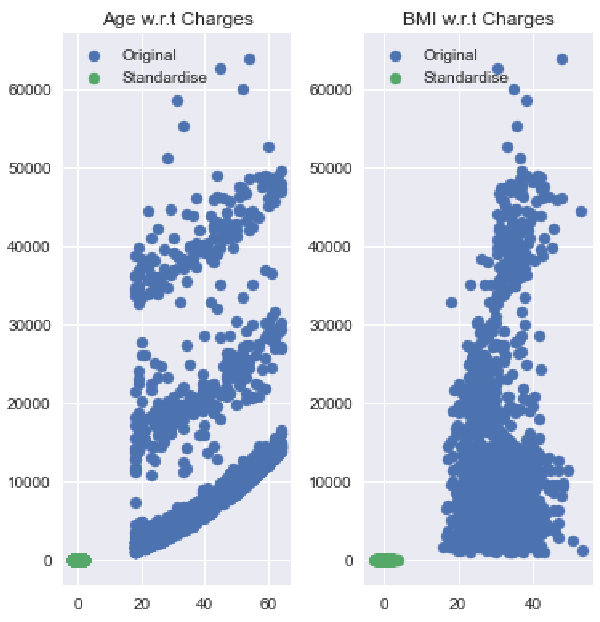


Figure 2: Comparison between Original and Standardised Data

3 Methodology

The data was shuffled once, transformed, standardised and split into training data (70%), validation data (20%) and testing data (10%). Training data was passed on to different algorithms to find out the weights of the features. After calculating these weights, for each algorithm we found out the predicted value of the target variable y$_{pred}$. The RMSE is given by the formula -

$$RMSE = \sqrt{\tfrac{1}{2} * \Sigma_{i=1}^{n} (y_{pred}^{i} - t^{i})^2}$$

The R^2 is given by the formula -

$$R^2 Score(\%) = (1 - \frac{\Sigma_{i=1}^{n} (y_{pred}^{i} - t^{i})^2}{\Sigma_{i=1}^{n} (y_{avg}^{i} - t^{i})^2}) * 100$$

Accuracy of each model is measured in terms of R^2 score and root mean squared error (RMSE). We then report the minimum training error, validation error and testing error for each degree from 1 to 10 for the most optimum regularization coefficient.

4 Optimisation Methods

Gradient Descent Algorithm using both Lasso and Ridge Regression and Stochastic Gradient Descent Algorithm using both Lasso and Ridge Regression has been performed in the following subsections.

4.1 Gradient and Stochastic Gradient Descent without Regularization

The hypothesis of is given by

$$y(w) = w_o + w_1x_1 + w_2x_2 + ... + w_nx_n.$$

The error function (loss function) that we used for the algorithm is

$$J(w) = \frac{1}{2} * \sum_{i=1}^{n} (y_{pred}^{i} - t^{i})^2$$

where,y_{pred}^{i} is the predicted value of target variable. We have tabulated the results of basic Gradient and Stochastic Gradient Descent algorithms without regularization below for all degrees from 1 to 10. The table below shows minimum training error, validation error and testing error for each degree. The initial weights are assigned random values from a normal distribution for both GD and SGD. 0.00001 is used as the learning rate for GD, whereas 0.001 is used for SGD.

Degree (Maximum Accuracy/Minimum Error)	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Best Method
1	10.5589	12.9390	15.5057	GD
2	9.1550	13.0653	15.9536	SGD
3	10.7820	11.2936	15.3290	GD
4	9.7062	11.4016	14.9601	GD
5	10.1131	11.1485	14.9052	GD
6	10.1169	13.2314	15.0077	GD
7	10.2943	11.2052	14.0276	GD
8	10.6538	10.8671	15.0072	GD
9	10.3260	10.5895	13.1216	GD
10	9.8296	11.7799	13.9752	GD

Figure 3: Comparison of Models having minimum training, testing and validation error

As we can see from the table, Degree 6 polynomial performs the best when performed using GD. The Loss and Accuracy vs Epoch graph along with surface plot is shown below. In the surface plot, we can see that not all the points lie on it, so we can say that for degree 6 there is no/less over-fitting.

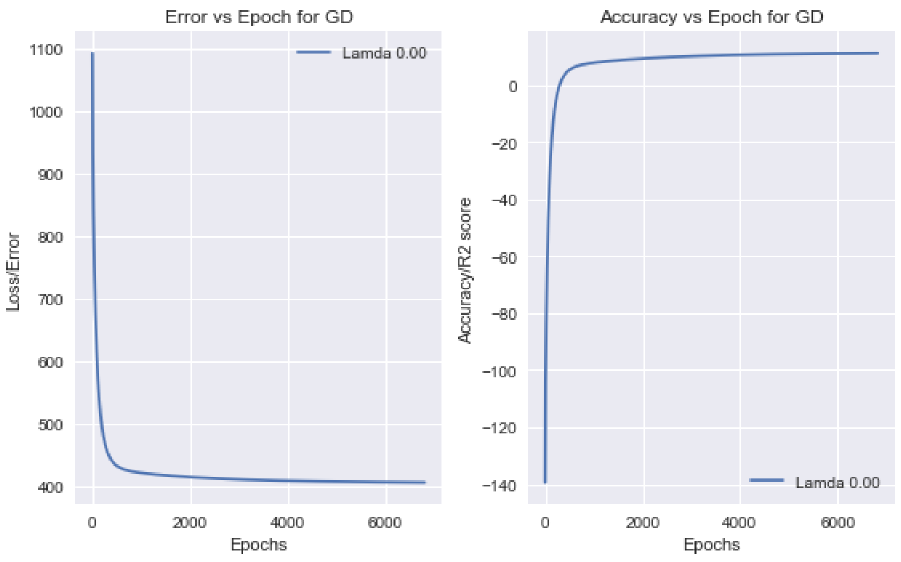


Figure 4: Error and Accuracy vs Epoch for Polynomial of Degree 6

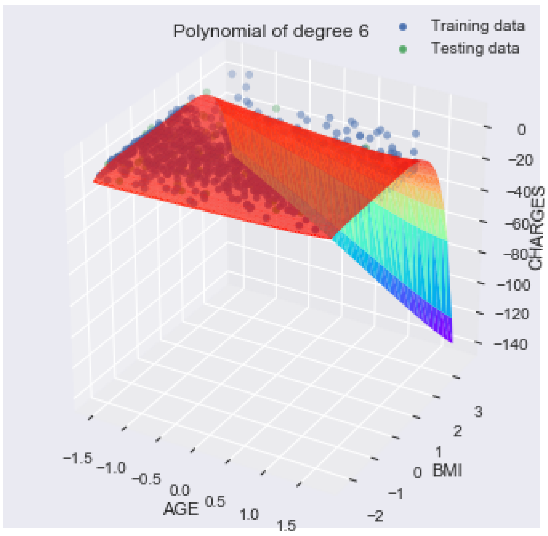


Figure 5: Surface Plot for Polynomial of Degree 6

As we can see from the table, Degree 9 polynomial performs the worst when performed using GD. The Loss and Accuracy vs Epoch graph along with surface plot is shown below. We can see from the table that the training accuracy is quite high and so is the testing accuracy. Maybe this is because of over-fitting of model. In the surface plot, we can see that most of the points lie on it so we can say there is over-fitting of model and maybe it would get better after regularization.

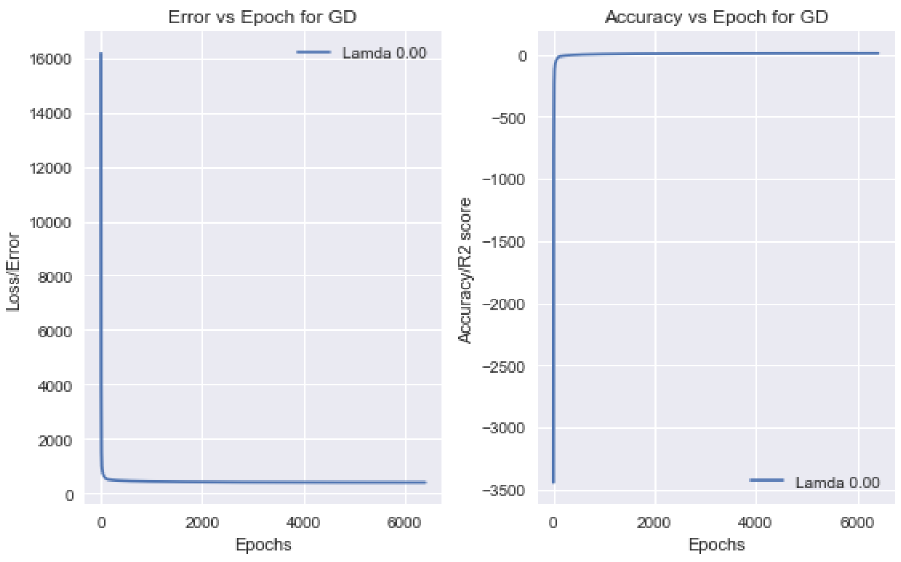


Figure 6: Error and Accuracy vs Epoch for Polynomial of Degree 9

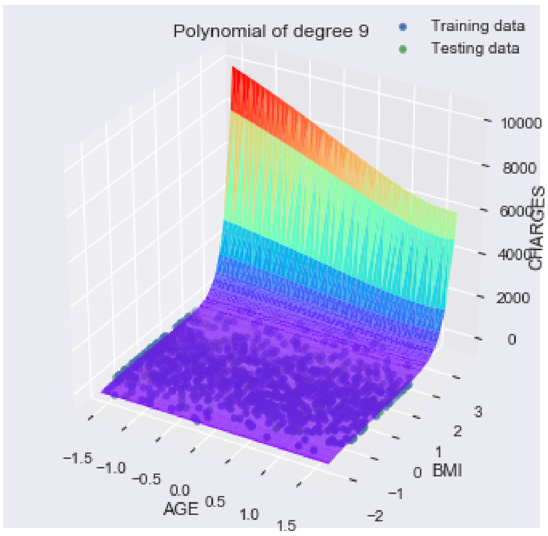


Figure 7: Surface Plot for Polynomial of Degree 9

4.2 Gradient Descent Algorithm with Lasso Regression

This is the basic Gradient descent algorithm with Lasso regularization technique. The error function (loss function) that we used for the algorithm is sum of squares of errors plus the regularization term multiplied by sum of absolute weights. We performed this method for all 10 degree polynomial by varying the regularization parameters from 0 to 1.The initial weights are assigned random values from a normal distribution. 0.00001 is used as the learning rate. On running our python code for this method on the given data data, we get the following results-

- Degree 2 polynomial performs the best with regularisation parameter as 0.15 and validation, testing, and training accuracy as 12.7728, 13.3672, 10.6214 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 2. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

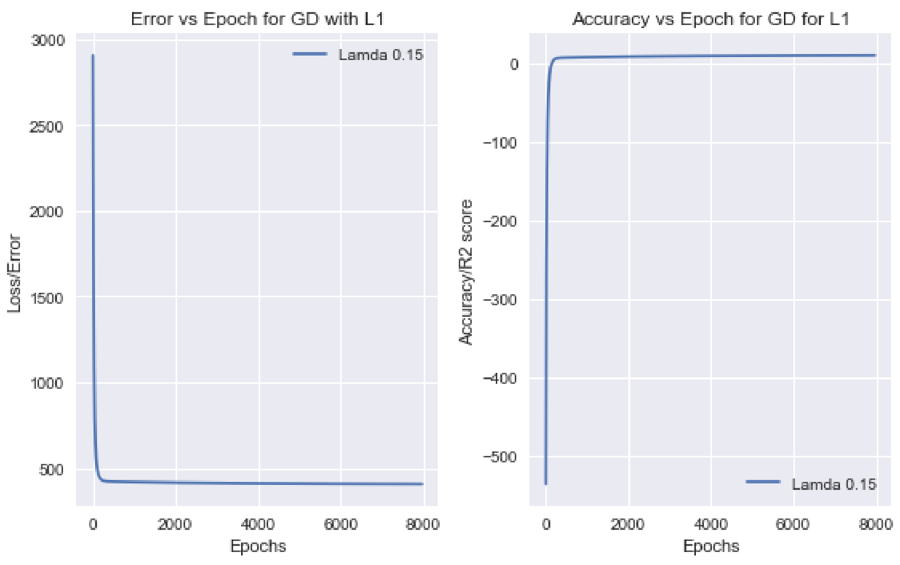


Figure 8: Error and Accuracy vs Epoch for Polynomial of Degree 2

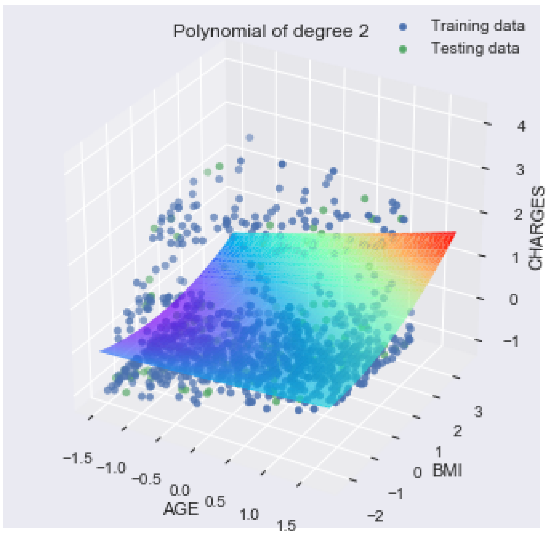


Figure 9: Surface Plot for Polynomial of Degree 2

- Degree 9 polynomial performs the worst with regularisation parameter as 0.35 and validation, testing, and training accuracy as 7.1521, 14.8401, 11.0955 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 9. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much. But the accuracy is less than that of degree 2.

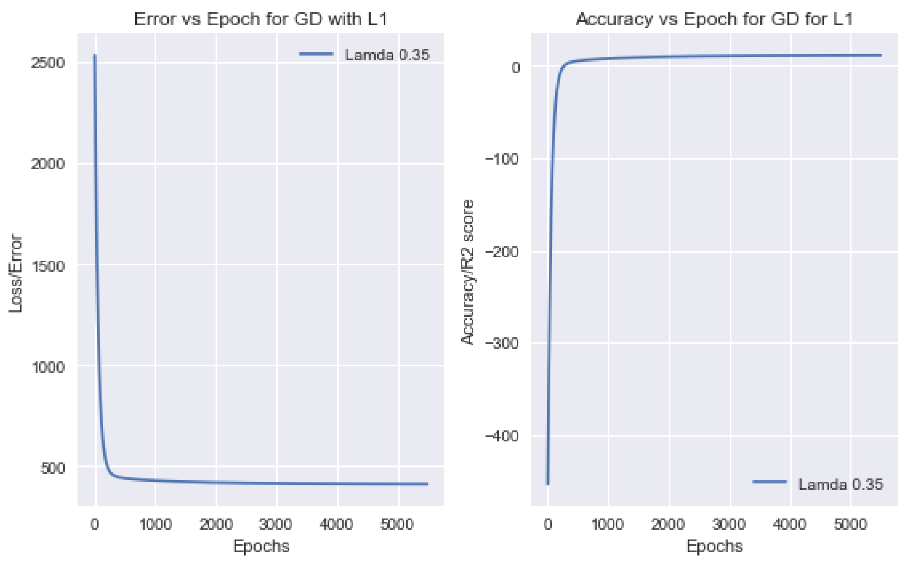


Figure 10: Error and Accuracy vs Epoch for Polynomial of Degree 9

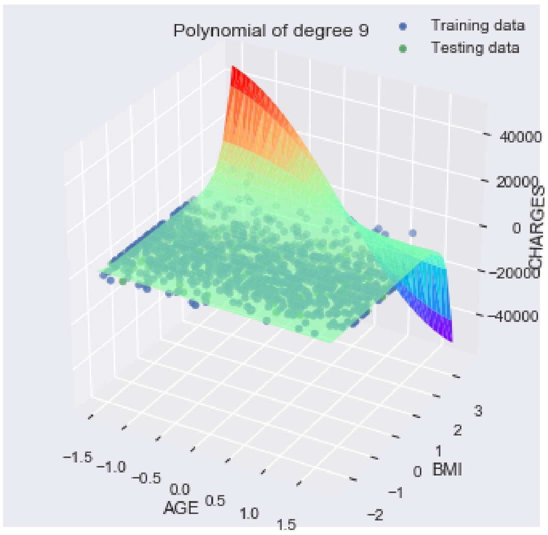


Figure 11: Surface Plot for Polynomial of Degree 9

4.3 Gradient Descent Algorithm with Ridge Regularization

This is the basic Gradient descent algorithm with Ridge regularization technique. The error function (loss function) that we used for the algorithm is sum of squares of errors plus the regularization term multiplied by sum of square of weights. We performed this method for all 10 degree polynomial by varying the regularization parameters from 0 to 1.The initial weights are assigned random values from a normal distribution. 0.00001 is used as the learning rate. On running our python code for this method on the given data, we get the following results-

- Degree 2 polynomial performs the best with regularisation parameter as 0.05 and validation, testing, and training accuracy as 12.7181, 13.2391, 10.6008 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 2. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

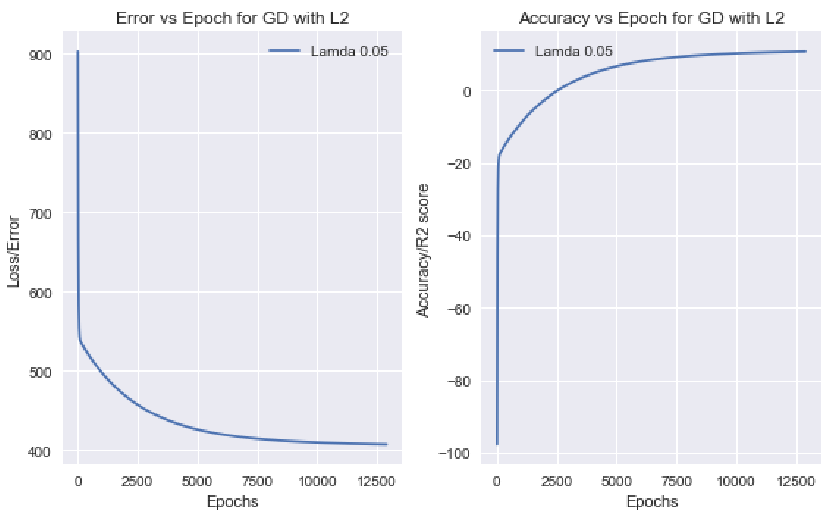


Figure 12: Error and Accuracy vs Epoch for Polynomial of Degree 2

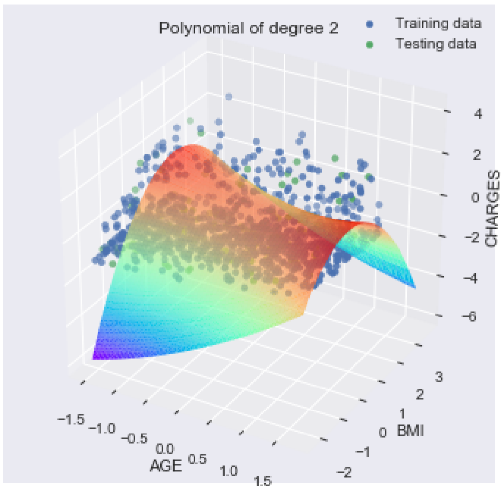


Figure 13: Surface Plot for Polynomial of Degree 2

- Degree 8 polynomial performs the worst with regularisation parameter as 0.35 and validation, testing, and training accuracy as 7.4171, 14.8452, 10.8050 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 8. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

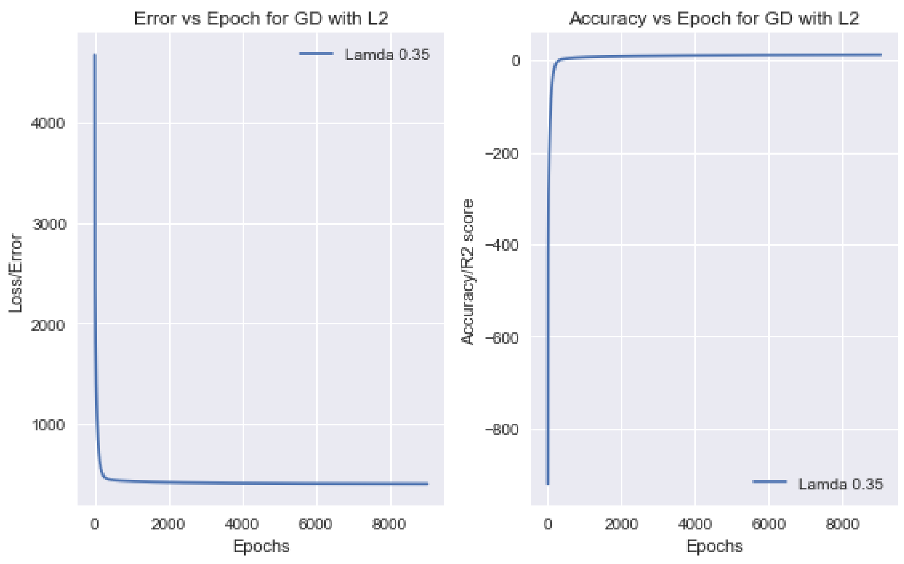


Figure 14: Error and Accuracy vs Epoch for Polynomial of Degree 8

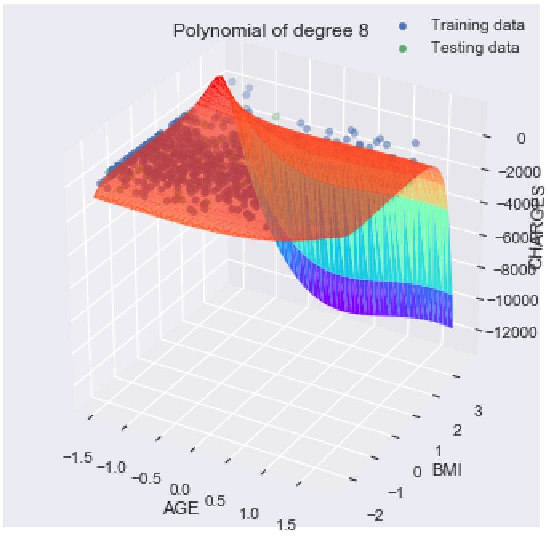


Figure 15: Surface Plot for Polynomial of Degree 8

4.4 Stochastic Gradient Descent Algorithm with Lasso Regularization

This is the basic Stochastic Gradient descent algorithm with Lasso regularization technique. The error function (loss function) that we used for the algorithm is sum of squares of errors plus the regularization term multiplied by sum of absolute weights. We performed this method for all 10 degree polynomial by varying the regularization parameters from 0 to 1.The initial weights are assigned random values from a normal distribution. 0.001 is used as the learning rate. On running our python code for this method on the given data data, we get the following results-

- Degree 3 polynomial performs the best with regularisation parameter as 0.05 and validation, testing, and training accuracy as 13.1840, 14.5879, 10.6330 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 3. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

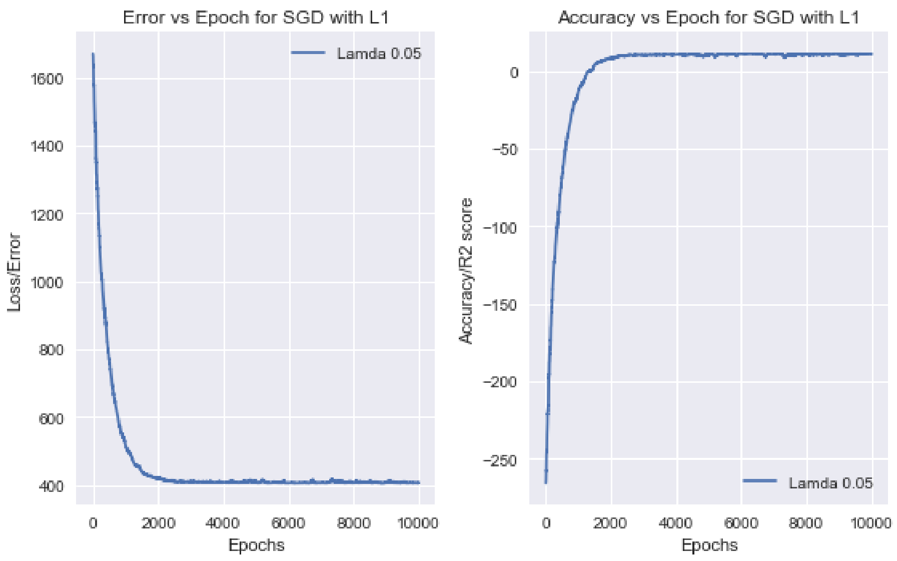


Figure 16: Error and Accuracy vs Epoch for Polynomial of Degree 3

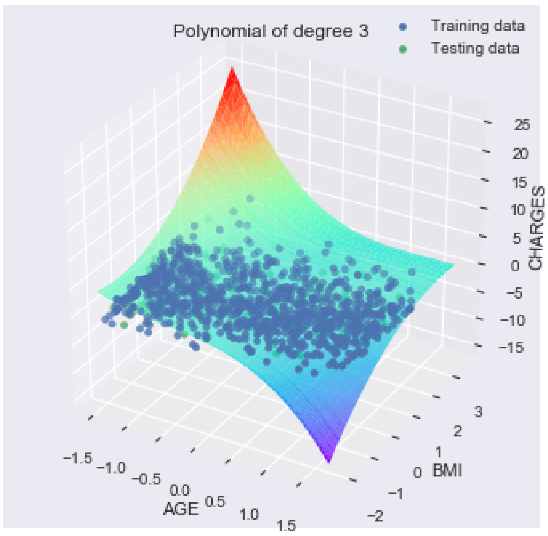


Figure 17: Surface Plot for Polynomial of Degree 3

- Degree 9 polynomial performs the worst with regularisation parameter as 0.05 and validation, testing, and training accuracy as -7.3314, 5.9049, -2.1856 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 9. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

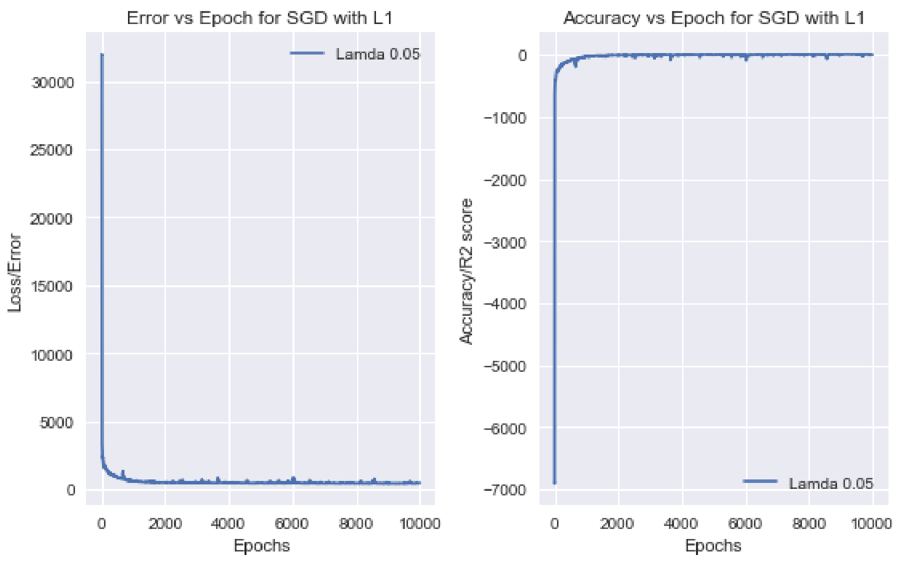


Figure 18: Error and Accuracy vs Epoch for Polynomial of Degree 9

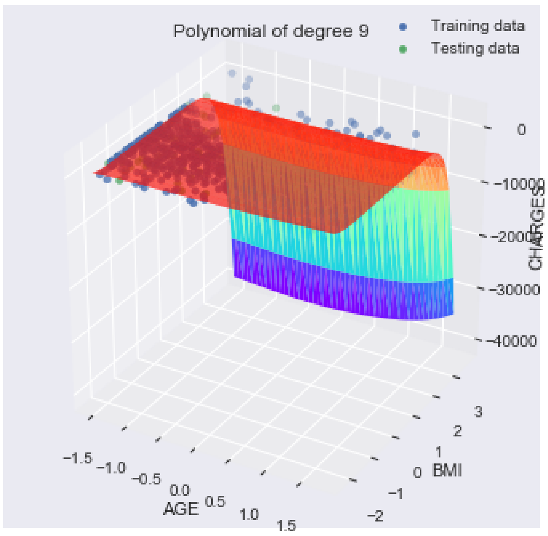


Figure 19: Surface Plot for Polynomial of Degree 9

4.5 Stochastic Gradient Descent Algorithm with Ridge Regularization

This is the basic Stochastic Gradient descent algorithm with Ridge regularization technique. The error function (loss function) that we used for the algorithm is sum of squares of errors plus the regularization term multiplied by sum of square of weights. We performed this method for all 10 degree polynomial by varying the regularization parameters from 0 to 1.The initial weights are assigned random values from a normal distribution. 0.001 is used as the learning rate. On running our python code for this method on the given data, we get the following results-

- Degree 9 polynomial performs the best with regularisation parameter as 0.75 and validation, testing, and training accuracy as 14.7718, 14.8118, 7.1959 respectively. For normal gradient descent, we observed that there is over-fitting. Here, we can see that regularization has reduced over-fitting and increased the validation and testing accuracy. The surface plots shows that there are very less points on the plot which means that over-fitting is reduced. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

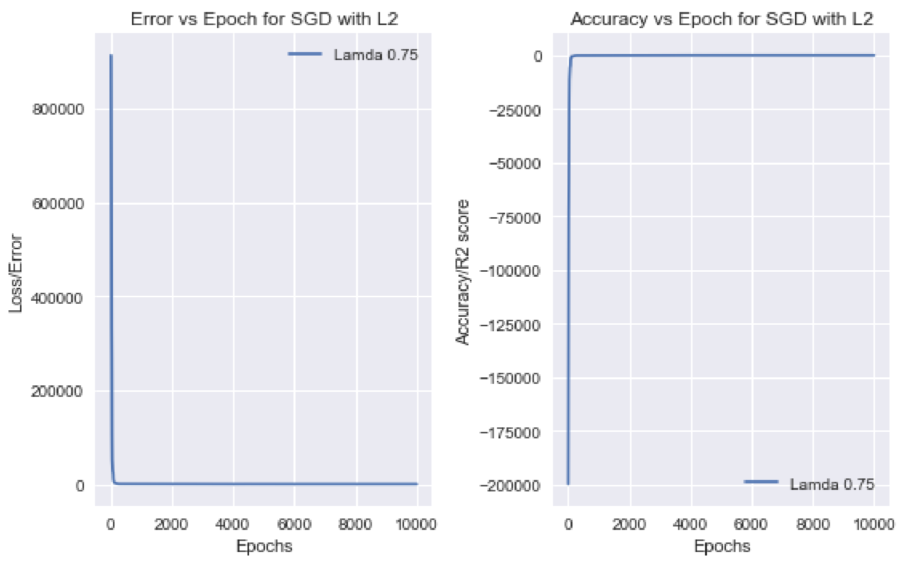


Figure 20: Error and Accuracy vs Epoch for Polynomial of Degree 9

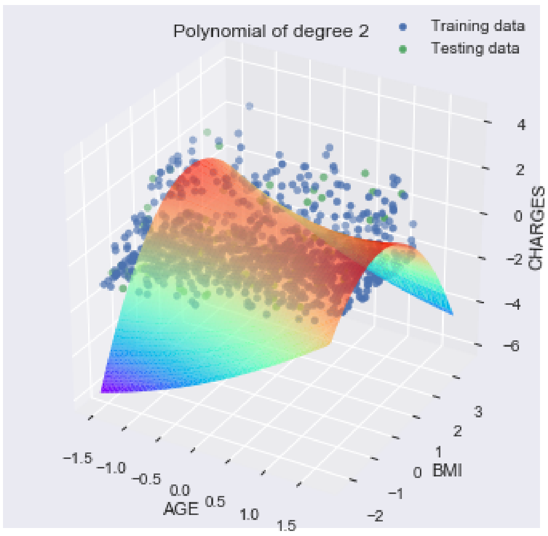


Figure 21: Surface Plot for Polynomial of Degree 9

- Degree 10 polynomial performs the worst with regularisation parameter as 0.45 and validation, testing, and training accuracy as 0.4212, 15.7277, 7.5110 respectively. We have made surface plot, error vs epoch and Accuracy vs epoch graph for degree 10. From the surface plot, we can infer that over-fitting is reduced because number of points on the surface plots are very less. From the error vs epoch graph and the accuracy vs epoch graph, we infer that after some iterations, the error value and the accuracy value is saturated and does not change much.

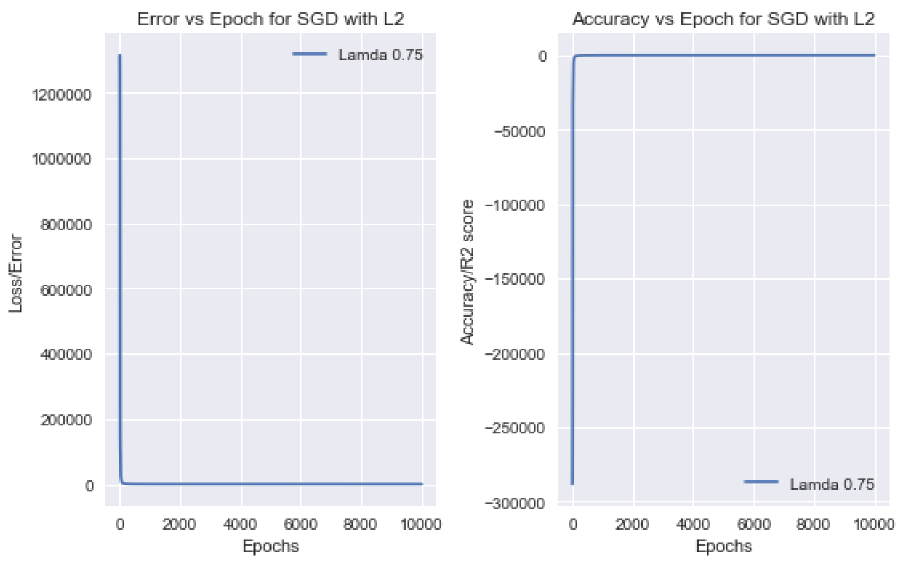


Figure 22: Error and Accuracy vs Epoch for Polynomial of Degree 10

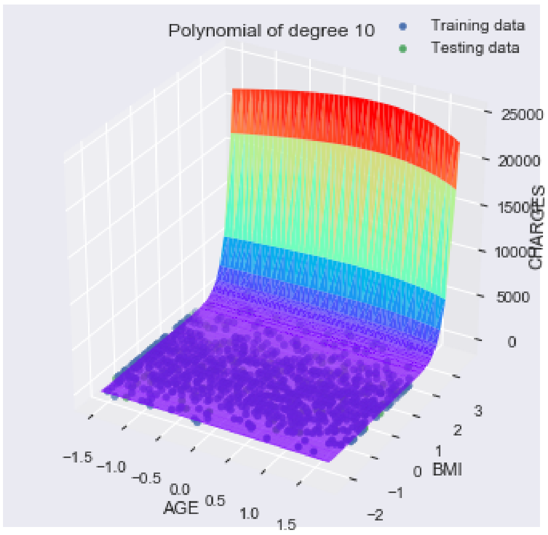
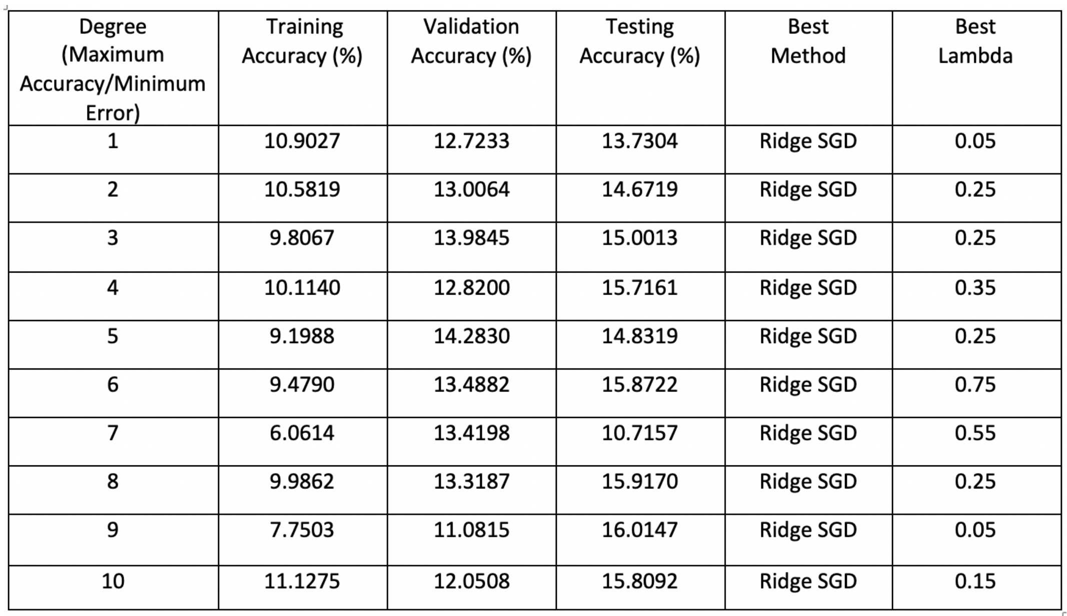


Figure 23: Surface Plot for Polynomial of Degree 10

5 Result Visualisation

5.1 Comparison of Models with Regularization

We have tabulated the results of basic Gradient and Stochastic Gradient Descent algorithms with regularization below for all degrees from 1 to 10. The table below shows the best model for each degree i.e. minimum training error, validation error and testing error for each degree. The initial weights are assigned random values from a normal distribution for both GD and SGD. 0.00001 is used as the learning rate for GD, whereas 0.001 is used for SGD.



Degree (Maximum Accuracy/Minimum Error)	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Best Method	Best Lambda
1	10.9027	12.7233	13.7304	Ridge SGD	0.05
2	10.5819	13.0064	14.6719	Ridge SGD	0.25
3	9.8067	13.9845	15.0013	Ridge SGD	0.25
4	10.1140	12.8200	15.7161	Ridge SGD	0.35
5	9.1988	14.2830	14.8319	Ridge SGD	0.25
6	9.4790	13.4882	15.8722	Ridge SGD	0.75
7	6.0614	13.4198	10.7157	Ridge SGD	0.55
8	9.9862	13.3187	15.9170	Ridge SGD	0.25
9	7.7503	11.0815	16.0147	Ridge SGD	0.05
10	11.1275	12.0508	15.8092	Ridge SGD	0.15

Figure 24: Comparison of Models with Regularization having minimum training, testing and validation error

The surface plot for the best Degree 1 polynomial with the parameters mentioned above is plotted below. We can see that there is no over-fitting since very few points lie on the surface plots and degree is also small.

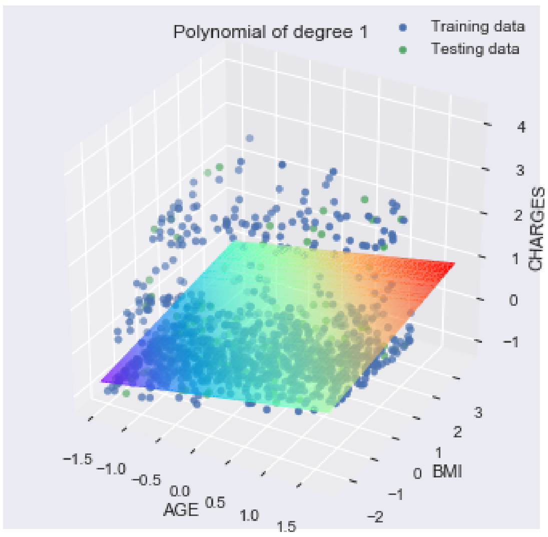


Figure 25: Best Degree 1 Polynomial

The surface plot for the best Degree 2 polynomial with the parameters mentioned above is plotted below. We can see that there is no over-fitting since very few points lie on the surface plots and degree is also small.

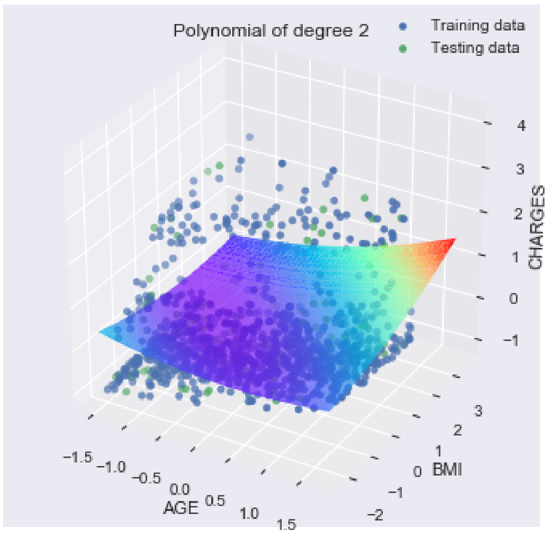


Figure 26: Best Degree 2 Polynomial

The surface plot for the best Degree 3 polynomial with the parameters mentioned above is plotted below. We can see that there is no over-fitting since very few points lie on the surface plots and degree is also small.

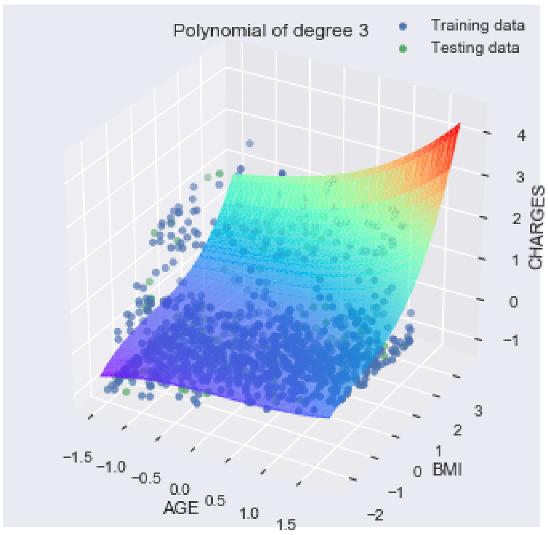


Figure 27: Best Degree 3 Polynomial

The surface plot for the best Degree 4 polynomial with the parameters mentioned above is plotted below.We can see that there is no over-fitting since very few points lie on the surface plots and degree is not so small.

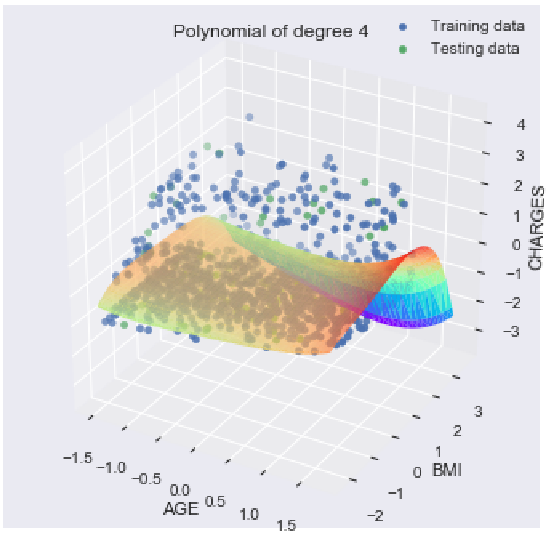


Figure 28: Best Degree 4 Polynomial

The surface plot for the best Degree 5 polynomial with the parameters mentioned above is plotted below. We can see that there is no over-fitting since very few points lie on the surface plots. We can see that we are increasing the number of degrees

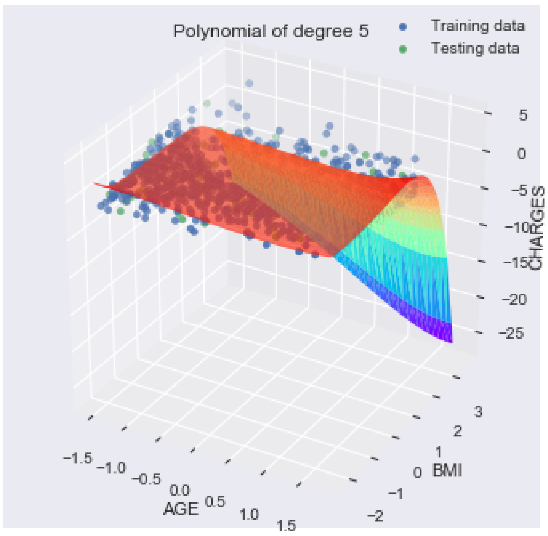


Figure 29: Best Degree 5 Polynomial

The surface plot for the best Degree 6 polynomial with the parameters mentioned above is plotted below. We can see that there is no over-fitting since very few points lie on the surface plots. We can see that there is no over-fitting since very few points lie on the surface plots.

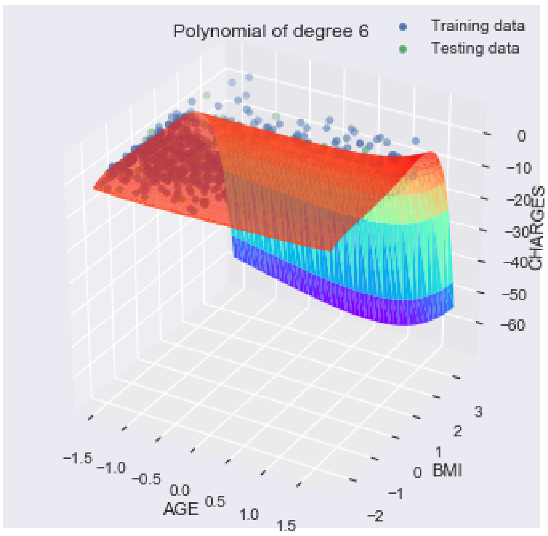


Figure 30: Best Degree 6 Polynomial

The surface plot for the best Degree 7 polynomial with the parameters mentioned above is plotted below.

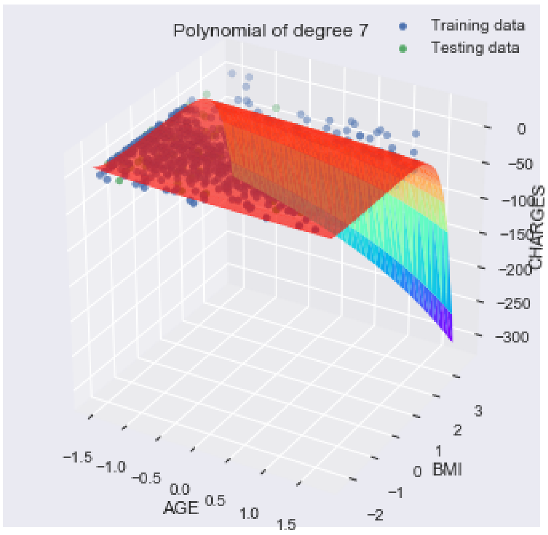


Figure 31: Best Degree 7 Polynomial

The surface plot for the best Degree 8 polynomial with the parameters mentioned above is plotted below. We can see that there is some what over-fitting since few points lie on the surface plots. We can see that as degree is increasing over-fitting is increasing.

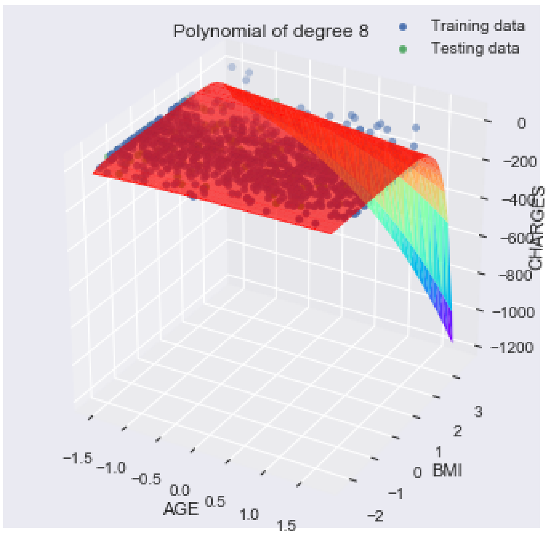


Figure 32: Best Degree 8 Polynomial

The surface plot for the best Degree 9 polynomial with the parameters mentioned above is plotted below. We can see that over-fitting has increased.

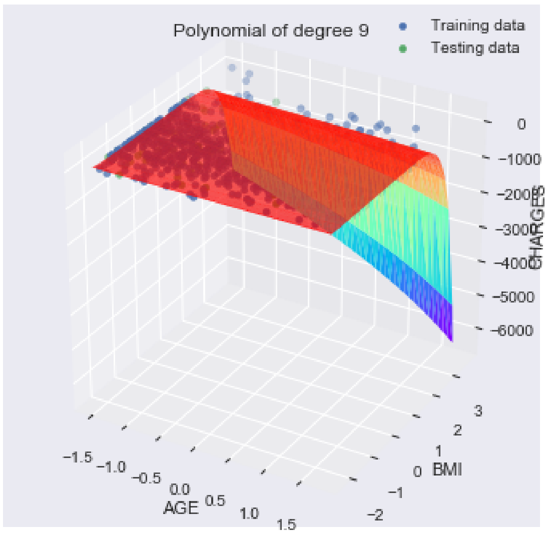


Figure 33: Best Degree 9 Polynomial

The surface plot for the best Degree 10 polynomial with the parameters mentioned above is plotted below. We can see that over-fitting has increased from degree 1 to degree 10.

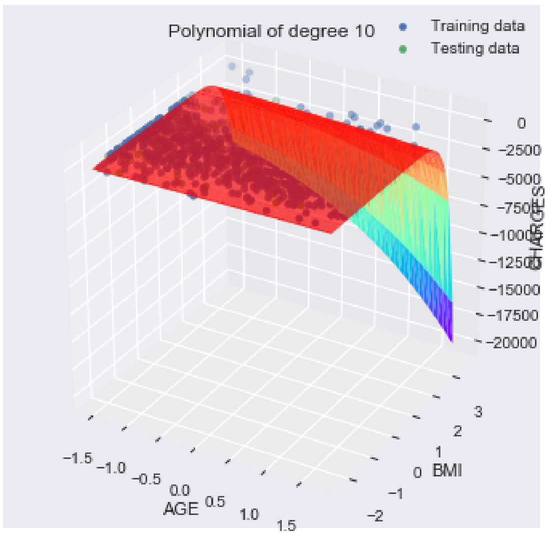


Figure 34: Best Degree 10 Polynomial

5.2 Inference

From the above results, we can conclude that all of the above optimization models are good for approximating the linear regression model. For the given data, the model built using 9 degree polynomial and Ridge SGD with lamda 0.75, has the minimum RMSE/maximum Accuracy for the validation data. On the other hand, the model built using 9 degree polynomial and Lasso SGD with lamda 0.05, has the maximum RMSE/minimum Accuracy for the validation data.

6 Discussion

After doing this project we can draw the following conclusions-

- As Polynomials of higher degree are used, the flexibility in the model increases (by increasing the polynomial degree), the training error continually decreases due to increased flexibility. However, the error on the testing set only decreases as we add flexibility up to a certain point. As the flexibility increases beyond this point, the testing error increases because the model has memorized the training data and the noise. This is because it is now trained to very specific data points and cannot make predictions for new data. Both overfitting and underfitting can be realized and removed by understanding the importance of model evaluation and optimization using cross-validation.

- The global minima exists in case of polynomial regression as well. Any polynomial regression equation can be solved to obtain the final weights by minimizing the error by differentiating w.r.t the weights. In case of polynomial regression, before a certain degree of polynomial, the training error is high while after that degree, the testing error is higher.Hence the error is minimum only for a certain degree of polynomial.

- Ridge regularization had slightly better results as compared to lasso regularization. Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (when only a few predictors actually influence the response). Ridge works well if there are many large parameters of about the same value (when most predictors impact the response).

- Larger the regularization parameter Lambda, more penalty is assigned to larger weights for features, hence, the extent of overfitting is inversely related to the value of regularization parameter. This regularization parameter thus, helps to tackle overfitting. For a high lambda value, high penalties are assigned to the weights for the features. As lambda is increased, the regression model starts to underfit the data and slowly the weights of some features become insignificant (tend to zero). For a really high lambda ($>$100), the regression line is almost parallel to the x-axis since only theta zero significantly contributes to the equation. Testing and training errors are extremely high in such a case.

- Regularization is necessary when we have a large number of features but limited training instances. When there are low numbers of training samples, our regression model tries to fit perfectly to these data points and thus the coefficients of the features also become large. When the same weights are used to make predictions on testing data, variance is very high. Hence, a regularization term is introduced to penalize the large parameters that lead to overfitting.

- If we are provided with D original features and are asked to generate new matured features of degree N, we will have (D+N)C(N) where D is the number of the original features, N is the degree of the polynomial.

- Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data. If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data. The degree of the polynomial and the regularization parameters need to be tuned to arrive at the optimal bias-variance trade-off.