| **BITS PILANI HYDERABAD CAMPUS** |
| --- |
|  |
|  |
|  |
| **CONTENTS**  Introduction……………………………………………………………...3  Preprocessing…………………………………………………………….3  Gradient Descent……….………………………………………………...4  Error values without regularisation………………………………..5  Random Hyperparameters on which the model was tested on…….7  Optimal Hyperparameters obtained from Lasso Regularization…...8  Minimum Errors associated with Hyperparameters Obtained from Lasso Regularization………………………………………………..9  Optimal Hyperparameters obtained from Ridge Regularization……10  Minimum Errors associated with Hyperparameters Obtained from Lasso Regularization……………………………………………….10  Surface Plots for Gradient Descent…………………………………11  Stochastic Gradient Descent……………………………………………….15  Error values without regularisation…………………………………15  Random Hyperparameters on which the model was tested on……...17  Optimal Hyperparameters obtained from Lasso Regularization……17  Minimum Errors associated with Hyperparameters Obtained from Lasso Regularization………………………………………………..17  Optimal Hyperparameters obtained from Ridge Regularization…....19  Minimum Errors associated with Hyperparameters Obtained from Ridge Regularization………………………………………………..19  Conclusion…………………………………………………………………20  Points to ponder on………………………………………………………21 |

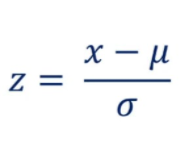
**INTRODUCTION**

In this assignment we try to predict the insurance of a person by fitting a polynomial regression model from degrees 1 to 10 using the sklearn **PolynomialFeatures** library by taking into account the features **Age** and the **BMI** of a person by -

1. Gradient Descent accompanied with Lasso and RIdge Regularization.
2. Stochastic Gradient Descent accompanied with Lasso and Ridge Regularization.

**PREPROCESSING**

* First we accept the dataset and convert it into a dataframe. We then drop the **‘Number of children’** feature and then transform the remaining 2 features into polynomial features of degrees 1 - 10 and append it to an array. We then calculate the mean and variance for all the polynomial features columns. These variables are then used to standardise each entry in each of these columns in the dataframe by using the formula



Where u is the mean and 𝞂 is the standard deviation to make sure all the features of the polynomial features array have a mean of 0 and a standard deviation of 1 to ensure good training of the model without any bias for any feature values which have a large value.

* After this, we randomly shuffle the dataset setting a constant random state value to ensure consistent results upon running the code multiple times.
* We then split the first 70 percent of the target attributes into train targets, the next 20 percent into validation target attributes and the last 10 percent of the data into test target attributes
* The training,validation,and test data for the input features was created while training the polynomial models.

**FIRST ALGORITHM-**  Gradient Descent -

* It is an iterative algorithm to minimize the loss function by updating the weights at each iteration. It makes small steps in the negative direction of slope at each point of the loss function until it reaches the global minimum considering all the training examples at a time in one iteration. The loss function we optimize is the sum of squares of errors given by -

**Loss = ½\*ΣNi=1(ypred - yreal)** where n is the number of training examples, ypred is the predicted output and yreal is the actual target attribute.

* We train the model using an optimal learning rate of 0.01.
* At first we initialize the weight vector with zero values of the size of the number of features for that respective polynomial.
* In each iteration, we modify the weight vector such that the error value converges to the lowest possible error which is repeated 10000 times.
* The weights are modified by using the formula -

w=w-1/n\*αdw

Where w=weight vector

α=learning rate

dw=gradient(partial derivative of Error function with respect to each

of the weight parameters)

We vectorized gradient descent and found dw by using -

dw= AB

Given that-

A= X\_train*T*

B= X\_train\*w-y\_train

Where X\_train contains values of the features of all the training data

points and y\_train contains the values for all the target attributes of

the training data.

* After all the iterations are completed, we return the weight vector as well as the error value between the predicted values and the actual values from the gradient descent function.
* The error metric we used to evaluate the test data are the root mean squared error(Only for the part without regularization) and the sum of squares of errors.
* Root means squared error is given by \_\_√**ΣNi=1(ŷi-yi)**\_\_
* The sum of squared errors are given by \_\_\_**ΣNi=1(ŷi-yi)**\_\_\_\_\_
* Here the **ŷi** is the predicted output and **yi** is the actual output.
* We implemented regularization by randomly initializing 10 values for the hyperparameter and we trained it for each model, compared the validation error and took the best hyperparameters for each degree.
* We then implemented Lasso regularization by adding to the dw term the sign of the respective weights (dw = AB + np.sign(w))
* We implemented ridge regularization by adding to the dw term 2\*w where w is the weight vector for that polynomial regression model (dw = dw + 2\*w).

ERROR VALUES ACHIEVED WITHOUT REGULARIZATION**-**

**For model with degree 1:**

**Train Sum of squares of error = 419.2640621027218**

**Validation Sum of squares of error = 124.38730683792616**

**test Sum of squares of error = 46.88233760063521**

**Train RMSE = 0.9465005972324431**

**Validation RMSE = 0.9652668678281606**

**test RMSE = 0.8333985538978576**

**For model with degree 2:**

**Train Sum of squares of error = 414.8128691031874**

**Validation Sum of squares of error = 126.95138068353819**

**test Sum of squares of error = 49.33647018097834**

**Train RMSE = 0.9414628428188337**

**Validation RMSE = 0.9751649459382771**

**test RMSE = 0.8549331373550959**

**For model with degree 3:**

**Train Sum of squares of error = 413.6702353655027**

**Validation Sum of squares of error = 128.39530699375433**

**test Sum of squares of error = 51.55529309320786**

**Train RMSE = 0.9401652829837939**

**Validation RMSE = 0.9806949572645677**

**test RMSE = 0.8739462911983641**

**For model with degree 4:**

**Train Sum of squares of error = 413.1166802369856**

**Validation Sum of squares of error = 128.25507973120182**

**test Sum of squares of error = 52.23305533880523**

**Train RMSE = 0.9395360286994471**

**Validation RMSE = 0.9801592767118812**

**test RMSE = 0.8796721219047321**

**For model with degree 5:**

**Train Sum of squares of error = 412.5224103674776**

**Validation Sum of squares of error = 128.00559707816032**

**test Sum of squares of error = 52.825894335681426**

**Train RMSE = 0.9388600224798421**

**Validation RMSE = 0.9792055064216318**

**test RMSE = 0.8846501240660599**

**For model with degree 6:**

**Train Sum of squares of error = 412.0676449311544**

**Validation Sum of squares of error = 127.916363995928**

**test Sum of squares of error = 53.27721619493005**

**Train RMSE = 0.9383423793014279**

**Validation RMSE = 0.9788641433853081**

**test RMSE = 0.8884211229910854**

**For model with degree 7:**

**Train Sum of squares of error = 411.7905077600999**

**Validation Sum of squares of error = 128.03560267872717**

**test Sum of squares of error = 53.44431512120502**

**Train RMSE = 0.9380267839196496**

**Validation RMSE = 0.9793202667459607**

**test RMSE = 0.8898132564899557**

**For model with degree 8:**

**Train Sum of squares of error = 411.58907669355676**

**Validation Sum of squares of error = 128.2433778986727**

**test Sum of squares of error = 53.43211576400711**

**Train RMSE = 0.9377973336967836**

**Validation RMSE = 0.9801145614407073**

**test RMSE = 0.8897116949931099**

**For model with degree 9:**

**Train Sum of squares of error = 411.3487831840955**

**Validation Sum of squares of error = 128.40841230306907**

**test Sum of squares of error = 53.36194157479265**

**Train RMSE = 0.9375235417945554**

**Validation RMSE = 0.9807450057558917**

**test RMSE = 0.8891272589396417**

**For model with degree 10:**

**Train Sum of squares of error = 411.047297643678**

**Validation Sum of squares of error = 128.48398600217809**

**test Sum of squares of error = 53.313885623929565**

**Train RMSE = 0.9371799141820745**

**Validation RMSE = 0.9810335679473645**

**test RMSE = 0.8887268098672022**

RANDOM HYPERPARAMETERS ON WHICH THE MODELS WERE TESTED ON -

[0.04427172, 0.08085184, 0.15858914, 0.15967957, 0.26061979,0.38222588, 0.39914724, 0.4466604 , 0.83815634, 0.96190113]

OPTIMAL HYPERPARAMETERS OBTAINED FROM LASSO REGULARIZATION -

For degree 1 - lambda = **0.04427172**

For degree 2 - lambda = **0.96190113**

For degree 3 - lambda = **0.96190113**

For degree 4 - lambda = **0.96190113**

For degree 5 - lambda = **0.26061979**

For degree 6 - lambda = **0.83815634**

For degree 7 - lambda = **0.96190113**

For degree 8 - lambda = **0.96190113**

For degree 9 - lambda = **0.96190113**

For degree 10 - lambda = **0.96190113**

MINIMUM ERRORS ASSOCIATED WITH THE OPTIMAL HYPERPARAMETERS FOR LASSO REGULARIZATION -

**FOR MODEL WITH DEGREE 1 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 419.264063971483**

**Validation Sum of squares of error = 124.38804585034293**

**Test Sum of squares of error = 46.88337760209099**

**FOR MODEL WITH DEGREE 2 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 415.28549004157287**

**Validation Sum of squares of error = 126.45880273739816**

**Test Sum of squares of error = 48.758487551064825**

**FOR MODEL WITH DEGREE 3 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 413.7875041978176**

**Validation Sum of squares of error = 127.85934512408012**

**Test Sum of squares of error = 51.01549353789224**

**FOR MODEL WITH DEGREE 4 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 413.2089861523027**

**Validation Sum of squares of error = 128.11432612089018**

**Test Sum of squares of error = 52.0794543459987**

**FOR MODEL WITH DEGREE 5 AND LAMBDA 0.26061979:**

**Train Sum of squares of error = 412.5948238235635**

**Validation Sum of squares of error = 127.95355210881806**

**Test Sum of squares of error = 52.74749301550966**

**FOR MODEL WITH DEGREE 6 AND LAMBDA 0.83815634:**

**Train Sum of squares of error = 412.3292601860834**

**Validation Sum of squares of error = 127.80640391097361**

**Test Sum of squares of error = 53.04297141767573**

**FOR MODEL WITH DEGREE 7 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 412.0704199969147**

**Validation Sum of squares of error = 127.76931040232279**

**Test Sum of squares of error = 53.20285945616376**

**FOR MODEL WITH DEGREE 8 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.85117396260625**

**Validation Sum of squares of error = 127.84797265073502**

**Test Sum of squares of error = 53.2378706488529**

**FOR MODEL WITH DEGREE 9 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.65876707881966**

**Validation Sum of squares of error = 127.98159796966536**

**Test Sum of squares of error = 53.21677096719913**

**FOR MODEL WITH DEGREE 10 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.45327967210983**

**Validation Sum of squares of error = 128.13813920293893**

**Test Sum of squares of error = 53.16342713113508**

OPTIMAL HYPERPARAMETERS OBTAINED FROM RIDGE REGULARIZATION -

Polynomial degrees and lambda value -

For degree 1 - lambda = **0.04427172**

For degree 2 - lambda = **0.96190113**

For degree 3 - lambda = **0.96190113**

For degree 4 - lambda = **0.96190113**

For degree 5 - lambda = **0.96190113**

For degree 6 - lambda = **0.96190113**

For degree 7 - lambda = **0.96190113**

For degree 8 - lambda = **0.96190113**

For degree 9 - lambda = **0.96190113**

For degree 10 - lambda = **0.96190113**

MINIMUM ERRORS ASSOCIATED WITH THE OPTIMAL HYPERPARAMETERS FOR RIDGE REGULARIZATION -

**FOR MODEL WITH DEGREE 1 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 419.2640624715913**

**Validation Sum of squares of error = 124.38751382538473**

**Test Sum of squares of error = 46.88275438950511**

**FOR MODEL WITH DEGREE 2 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 414.99547729904634**

**Validation Sum of squares of error = 126.75493260250643**

**Test Sum of squares of error = 49.14210680275751**

**FOR MODEL WITH DEGREE 3 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 413.7006911781194**

**Validation Sum of squares of error = 128.20891944075902**

**Test Sum of squares of error = 51.31478448926672**

**FOR MODEL WITH DEGREE 4 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 413.1407015613187**

**Validation Sum of squares of error = 128.18552428952216**

**Test Sum of squares of error = 52.09603338457427**

**FOR MODEL WITH DEGREE 5 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 412.5572254720738**

**Validation Sum of squares of error = 127.98435195953248**

**Test Sum of squares of error = 52.71861847471179**

**FOR MODEL WITH DEGREE 6 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 412.1000635710099**

**Validation Sum of squares of error = 127.90210125138421**

**Test Sum of squares of error = 53.18606316415034**

**FOR MODEL WITH DEGREE 7 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.8150023069548**

**Validation Sum of squares of error = 128.00712021848213**

**Test Sum of squares of error = 53.38226657927083**

**FOR MODEL WITH DEGREE 8 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.6124621728334**

**Validation Sum of squares of error = 128.20204075017583**

**Test Sum of squares of error = 53.39717935324592**

**FOR MODEL WITH DEGREE 9 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 411.3829303324228**

**Validation Sum of squares of error = 128.36596447244608**

**Test Sum of squares of error = 53.34399912992714**

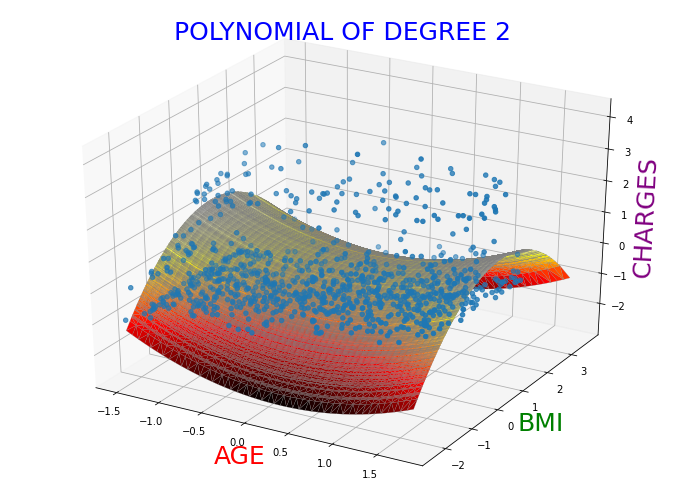
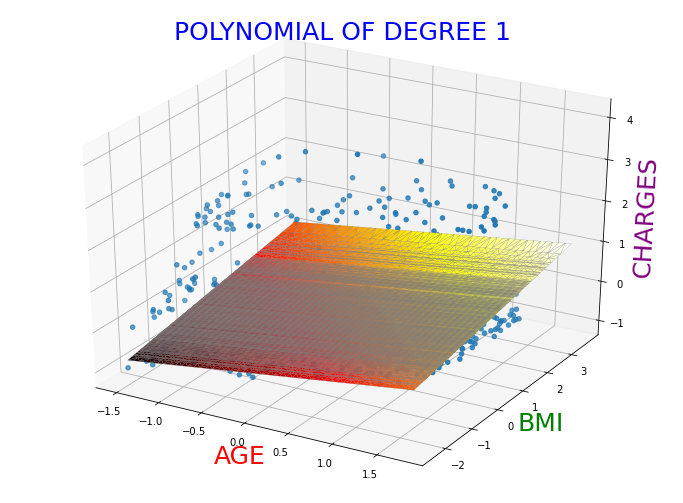
**FOR MODEL WITH DEGREE 10 AND LAMBDA 0.96190113:**

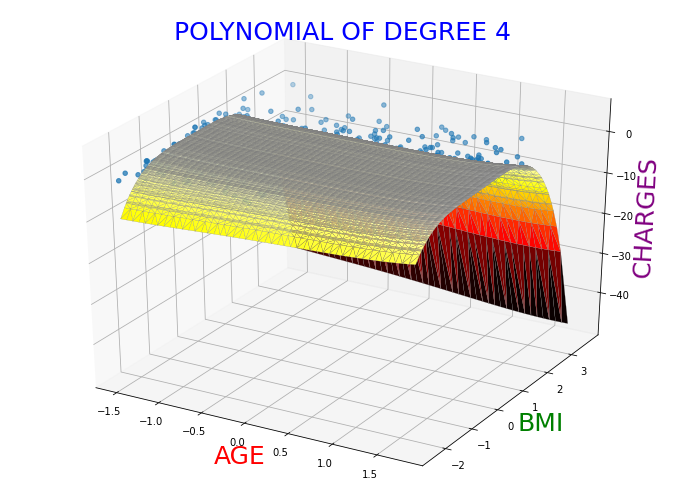
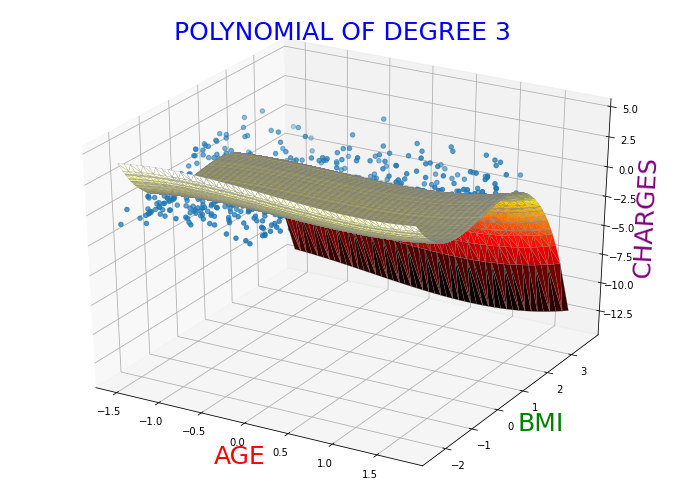
**Train Sum of squares of error = 411.0968957044615**

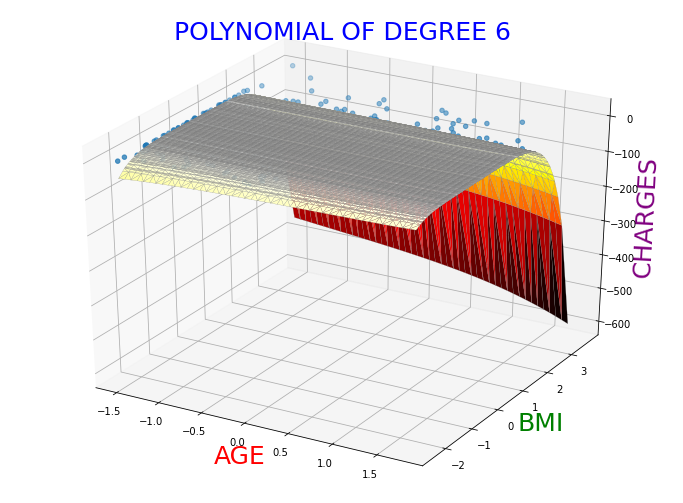
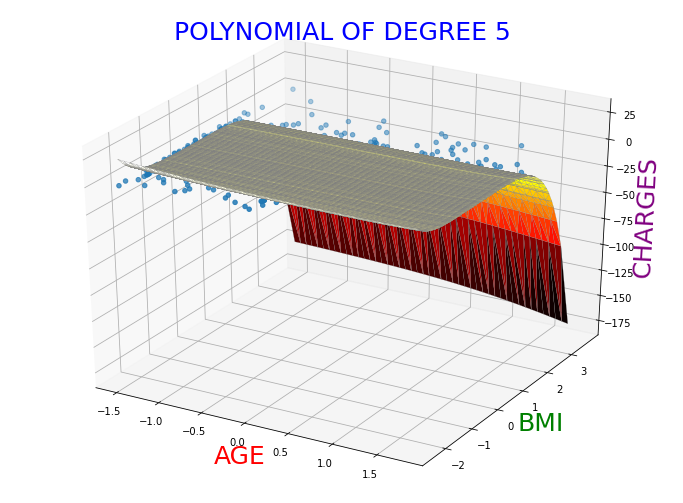
**Validation Sum of squares of error = 128.4487029315497**

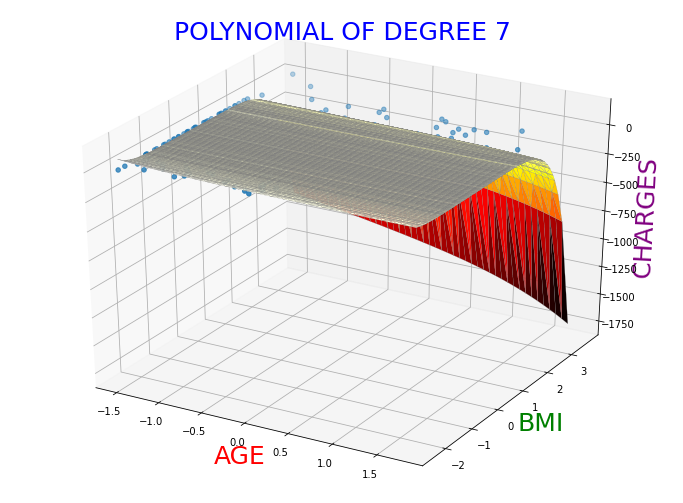
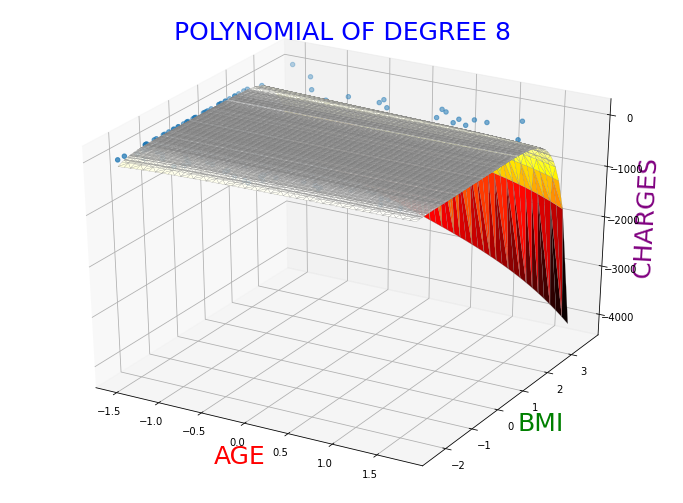
**Test Sum of squares of error = 53.3019347062286**

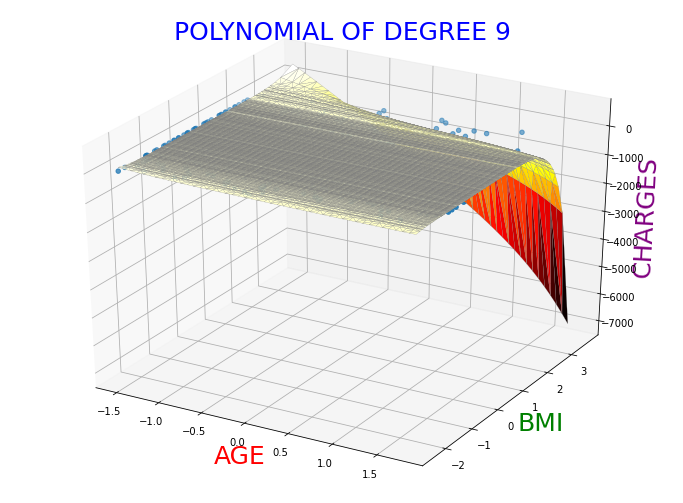
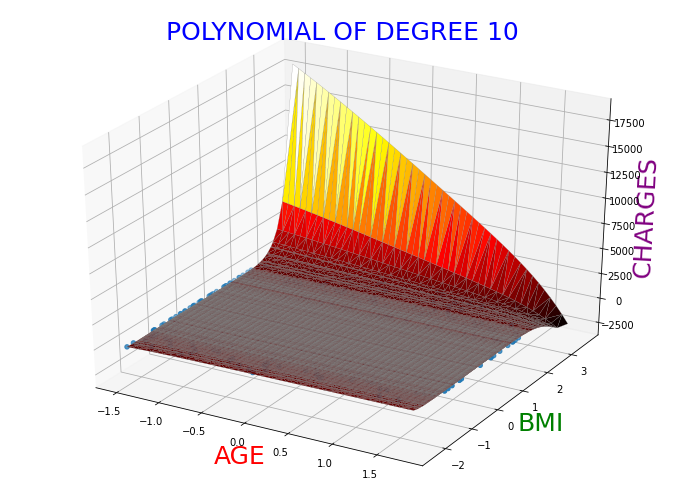
**SURFACE PLOTS FOR GRADIENT DESCENT -**

****

****

****

** **

** **

As we can see from the surface plots above, the higher degree surface plots have much sharper peaks and try to fit the training data better and hence the training error decreases. However, these models don't generalize very well on new unseen and as a result of this our test and validation errors go up.

**SECOND ALGORITHM-**  Stochastic Gradient Descent -

* It is an iterative algorithm to minimize the loss function by updating the weights at each iteration. It makes small steps in the negative direction of slope at each point of the loss function until it reaches the global minimum considering only one training example in one iteration. The loss function we optimize is the sum of squares of errors given by

**Loss = ½\*(ypred - yreal)** in a single iteration.

* We train the model using an optimal learning rate of **0.0001**.
* We take individual data points sequentially in a loop and update the weights with respect to that training example.
* At first we initialize the weight vector with zero values of the size of the number of features for that respective polynomial.

The weights are modified using -

w=w-αdw

Where w=weight vector

α=learning rate

dw=gradient(partial derivative of Error function with respect to each of the weight parameters)

The Vectorized implementation of the gradient is given by:-

* dw= AB

Given that-

A= X\_train*\_ jT*

* B= X*j*w-y\_trainj

Where X\_trainjis the matrix consisting of the values of all the features of the data point picked in the jth iteration and y\_train is the target variable of the data point picked in the jth iteration from the training set.

* We implemented regularization by randomly initializing 10 random uniform values for the hyperparameters and trained it for each model, compared the validation error amongst all the models and took the optimal hyperparameters for each degree.
* We then implemented lasso regularization by adding to the dw term the sign of the respective weight (dw = AB + np.sign(w)) one training example at a time
* We implemented ridge regularization by adding to the dw term 2\*w where w is the weight vector for that polynomial regression model (dw = dw + 2\*w) one training example at a time.

ERROR VALUES ACHIEVED WITHOUT REGULARIZATION**-**

**For model with degree 1:**

**Train Sum of squares of error = 425.65689985471136**

**Validation Sum of squares of error = 126.97220376496401**

**test Sum of squares of error = 49.60652870654878**

**Train RMSE = 0.9536893049037746**

**Validation RMSE = 0.975244917917968**

**test RMSE = 0.8572698153984633**

**For model with degree 2:**

**Train Sum of squares of error = 418.819575340185**

**Validation Sum of squares of error = 125.10001313404291**

**test Sum of squares of error = 47.63475096180487**

**Train RMSE = 0.9459987434039026**

**Validation RMSE = 0.9680282795805798**

**test RMSE = 0.8400595301816182**

**For model with degree 3:**

**Train Sum of squares of error = 418.34243900761726**

**Validation Sum of squares of error = 125.79667411021684**

**test Sum of squares of error = 48.07216240906202**

**Train RMSE = 0.9454597296784388**

**Validation RMSE = 0.9707199309513499**

**test RMSE = 0.8439076867987136**

**For model with degree 4:**

**Train Sum of squares of error = 418.1566172030626**

**Validation Sum of squares of error = 126.67776671042091**

**test Sum of squares of error = 48.928882054493045**

**Train RMSE = 0.9452497264245514**

**Validation RMSE = 0.9741135092570119**

**test RMSE = 0.8513943426716144**

**For model with degree 5:**

**Train Sum of squares of error = 417.8546979804996**

**Validation Sum of squares of error = 127.44315252251724**

**test Sum of squares of error = 49.88575110275148**

**Train RMSE = 0.9449084181875783**

**Validation RMSE = 0.9770518696435271**

**test RMSE = 0.8596791055302013**

**For model with degree 6:**

**Train Sum of squares of error = 417.25890761499295**

**Validation Sum of squares of error = 127.99561169357364**

**test Sum of squares of error = 50.82317093464941**

**Train RMSE = 0.9442345378269101**

**Validation RMSE = 0.9791673130360075**

**test RMSE = 0.8677187710880235**

**For model with degree 7:**

**Train Sum of squares of error = 416.4228000579276**

**Validation Sum of squares of error = 128.36370497757062**

**test Sum of squares of error = 51.661932472487265**

**Train RMSE = 0.9432880301456626**

**Validation RMSE = 0.9805742603096425**

**test RMSE = 0.8748496799767186**

**For model with degree 8:**

**Train Sum of squares of error = 415.60040389363917**

**Validation Sum of squares of error = 128.6724666238139**

**test Sum of squares of error = 52.36969851232614**

**Train RMSE = 0.9423561168978448**

**Validation RMSE = 0.9817528718073488**

**test RMSE = 0.8808219941439897**

**For model with degree 9:**

**Train Sum of squares of error = 415.09755383517756**

**Validation Sum of squares of error = 129.0607894093513**

**test Sum of squares of error = 52.956452454910846**

**Train RMSE = 0.941785848862628**

**Validation RMSE = 0.9832331799793826**

**test RMSE = 0.8857426468049563**

**For model with degree 10:**

**Train Sum of squares of error = 415.10988407367165**

**Validation Sum of squares of error = 129.60688463792627**

**test Sum of squares of error = 53.456938969256555**

**Train RMSE = 0.9417998363672345**

**Validation RMSE = 0.9853111626158905**

**test RMSE = 0.8899183397348284**

RANDOM HYPERPARAMETERS ON WHICH THE MODELS WERE TESTED ON -

[0.04427172, 0.08085184, 0.15858914, 0.15967957, 0.26061979,0.38222588, 0.39914724, 0.4466604 , 0.83815634, 0.96190113]

OPTIMAL HYPERPARAMETERS OBTAINED FROM LASSO REGULARIZATION -

For degree 1 - lambda = **0.04427172**

For degree 2 - lambda = **0.04427172**

For degree 3 - lambda = **0.04427172**

For degree 4 - lambda = **0.04427172**

For degree 5 - lambda = **0.96190113**

For degree 6 - lambda = **0.96190113**

For degree 7 - lambda = **0.96190113**

For degree 8 - lambda = **0.96190113**

For degree 9 - lambda = **0.96190113**

For degree 10 - lambda = **0.96190113**

MINIMUM ERRORS ASSOCIATED WITH THE OPTIMAL HYPERPARAMETERS FOR LASSO REGULARIZATION -

**FOR MODEL WITH DEGREE 1 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 425.66031136666356**

**Validation Sum of squares of error = 126.97351588721727**

**Test Sum of squares of error = 49.60760620095283**

**FOR MODEL WITH DEGREE 2 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 418.82040423284667**

**Validation Sum of squares of error = 125.10079505330874**

**Test Sum of squares of error = 47.63574097972983**

**FOR MODEL WITH DEGREE 3 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 418.3429037602601**

**Validation Sum of squares of error = 125.79737989328497**

**Test Sum of squares of error = 48.07327082236042**

**FOR MODEL WITH DEGREE 4 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 418.1562791695922**

**Validation Sum of squares of error = 126.67825152497747**

**Test Sum of squares of error = 48.92907674631692**

**FOR MODEL WITH DEGREE 5 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 417.9027426074643**

**Validation Sum of squares of error = 127.43979945942274**

**Test Sum of squares of error = 49.8803212287976**

**FOR MODEL WITH DEGREE 6 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 417.3456604991282**

**Validation Sum of squares of error = 127.99021798494438**

**Test Sum of squares of error = 50.80216270817387**

**FOR MODEL WITH DEGREE 7 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 416.54480488275806**

**Validation Sum of squares of error = 128.35110706543168**

**Test Sum of squares of error = 51.62036902375443**

**FOR MODEL WITH DEGREE 8 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 415.74130694028275**

**Validation Sum of squares of error = 128.65833405336696**

**Test Sum of squares of error = 52.33205147656648**

**FOR MODEL WITH DEGREE 9 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 415.2325370177284**

**Validation Sum of squares of error = 129.03643422645223**

**Test Sum of squares of error = 52.92279833711092**

**FOR MODEL WITH DEGREE 10 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 415.21453613761093**

**Validation Sum of squares of error = 129.56948600328138**

**Test Sum of squares of error = 53.43477101069985**

OPTIMAL HYPERPARAMETERS OBTAINED FROM RIDGE REGULARIZATION -

Polynomial degrees and lambda value -

For degree 1 - lambda = **0.04427172**

For degree 2 - lambda = **0.04427172**

For degree 3 - lambda = **0.04427172**

For degree 4 - lambda = **0.96190113**

For degree 5 - lambda = **0.96190113**

For degree 6 - lambda = **0.04427172**

For degree 7 - lambda = **0.04427172**

For degree 8 - lambda = **0.04427172**

For degree 9 - lambda = **0.04427172**

For degree 10 - lambda = **0.96190113**

MINIMUM ERRORS ASSOCIATED WITH THE OPTIMAL HYPERPARAMETERS FOR RIDGE REGULARIZATION -

**FOR MODEL WITH DEGREE 1 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 425.65689985471136**

**Validation Sum of squares of error = 126.97220376496401**

**Test Sum of squares of error = 49.60652870654878**

**FOR MODEL WITH DEGREE 2 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 418.819575340185**

**Validation Sum of squares of error = 125.10001313404291**

**Test Sum of squares of error = 47.63475096180487**

**FOR MODEL WITH DEGREE 3 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 418.34243900761726**

**Validation Sum of squares of error = 125.79667411021684**

**Test Sum of squares of error = 48.07216240906202**

**FOR MODEL WITH DEGREE 4 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 418.1566172030626**

**Validation Sum of squares of error = 126.67776671042091**

**Test Sum of squares of error = 48.928882054493045**

**FOR MODEL WITH DEGREE 5 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 417.8546979804996**

**Validation Sum of squares of error = 127.44315252251724**

**Test Sum of squares of error = 49.88575110275148**

**FOR MODEL WITH DEGREE 6 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 417.25890761499295**

**Validation Sum of squares of error = 127.99561169357364**

**Test Sum of squares of error = 50.82317093464941**

**FOR MODEL WITH DEGREE 7 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 416.4228000579276**

**Validation Sum of squares of error = 128.36370497757062**

**Test Sum of squares of error = 51.661932472487265**

**FOR MODEL WITH DEGREE 8 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 415.60040389363917**

**Validation Sum of squares of error = 128.6724666238139**

**Test Sum of squares of error = 52.36969851232614**

**FOR MODEL WITH DEGREE 9 AND LAMBDA 0.04427172:**

**Train Sum of squares of error = 415.09755383517756**

**Validation Sum of squares of error = 129.0607894093513**

**Test Sum of squares of error = 52.956452454910846**

**FOR MODEL WITH DEGREE 10 AND LAMBDA 0.96190113:**

**Train Sum of squares of error = 415.10988407367165**

**Validation Sum of squares of error = 129.60688463792627**

**Test Sum of squares of error = 53.456938969256555**

**CONCLUSION-**

When no regularization was used, the training errors decreased and

the testing and validation errors went up as the polynomial degree

increased indicating overfitting. To combat this, we used lasso and

ridge regularization in the gradient descent and Stochastic gradient descent approaches. We found that lasso regularization gave us slightly better results on the validation and test data because we standardized all the features values including the target and hence there was not a very big problem of the weights exploding as all the feature values and the target variables too small values.

**POINTS TO PONDER ON -**

* In general when higher degree polynomials are used for smaller dataset sizes, the higher degree polynomials tend to overfit the training data cause the weights are given freedom to grow how much ever they want to. As a result of this, the training error decreases but the testing and validation error increases as the model is not able to generalize well to new unseen data.
* Yes a single global minimum exists for polynomial regression as well as the error function(Sum of squares of errors) we try to optimize is convex as well as the maximum degree of any particular weight value is 2(as we find the error function with respect to the weights) and is similar to the weights of a linear regression problem and holds the same for any degree polynomial.
* Lasso Regularization curbed overfitting better in our case. Lasso regularization tends to dowell if there are a small number of significant parameters and the others are close to zero. Ridge regularizationworkswell if there are many large parameters of about the same value.
* The regularization parameter plays a very important role in the training process and weights. Depending on the regularization parameter we take, we decide whether we give more emphasis to optimize the sum of squares of errors function or to control the growth of the weights. If we take a very large regularization parameter(say>2), then we put more emphasis to control the growth of the weights. If we take a very small regularization parameter, then we put more emphasis on optimizing the error function.
* Yes regularization is very necessary when you have a large number of features and a limited number of training instances as in this case as the model tends to overfit the training data. As a result of this, the model cannot generalize well to new unseen data and we get very high error values on this unseen data. But if we regularize the weights, then we can control the growth of these weights and curb overfitting to an extent.
* If you are provided with D original features and are asked to generate new matured features of degree N, how many such new matured features will you be able to generate? Answer in terms of N and D. Number of features for (D,N)= (D+N)C(N) where D is the number of the original features, N is the degree of the polynomial.
* Here the bias and variance refer to the training and testing errors respectively. A model with a high bias and a high variance tends to underfit on the data. A model with a low bias and a high variance tends to overfit the data. An optimal model will have a low bias and a low variance. We should tune our hyperparameters such that we get the optimal bias and variance such that the model does not overfit too much and the model should not underfit either. So in general for models, the variance tends to decrease while the bias increases and hence we should choose the model with the most optimal bias and variance which gives us the best results which is known as the bias-variance tradeoff.