Ell409

Assignment 1 Report

-Avadhesh Prasad

-2019MT60747

PART 1A-

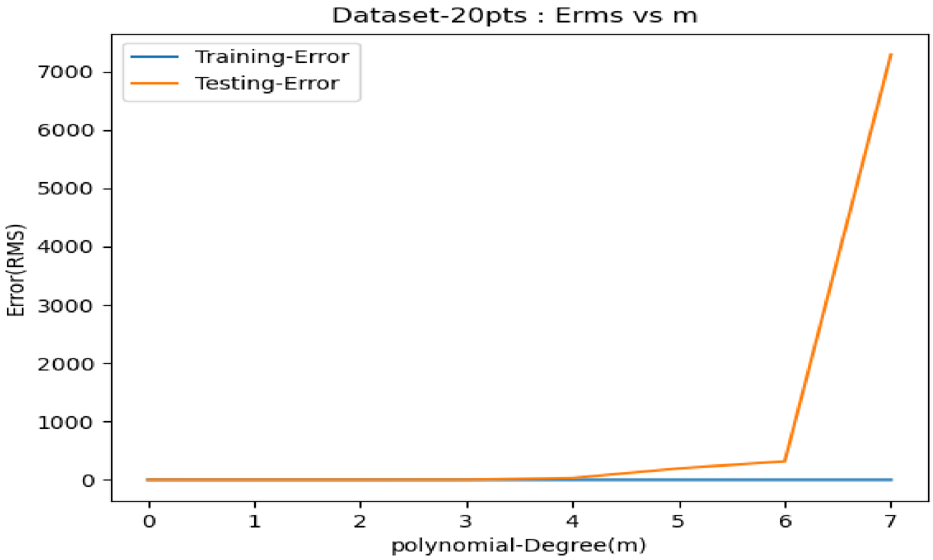
We have to carry out least-square regression to minimize error function for the given gaussian-noise data-set, here error function is given by

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2.$$

Case-1: 20 points

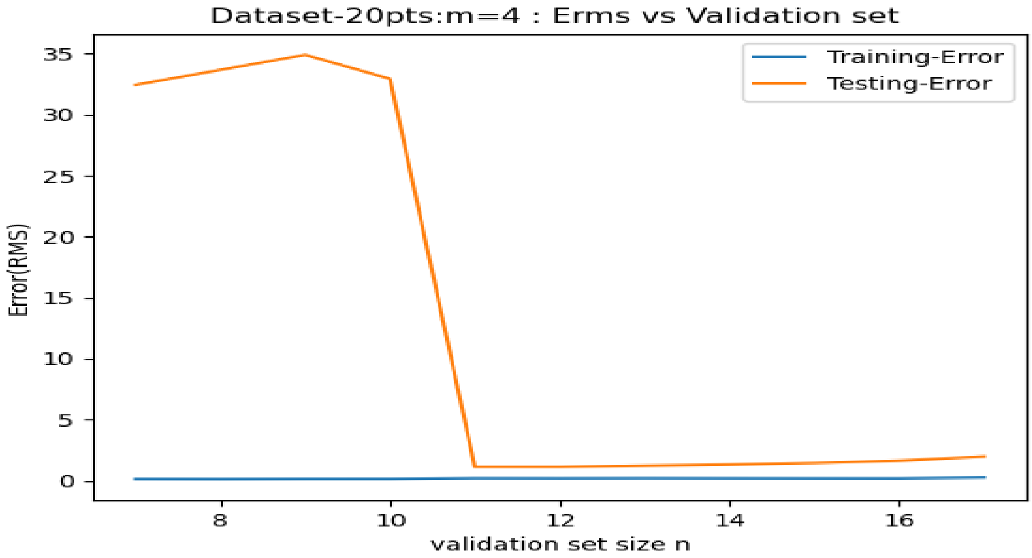
First, we do all the analysis for first 20 points

A) Relation between Erms vs m



This is the error curve for 20 points for Moore Penrose polynomial fitting. It is obtained using Pinv function. In this plot, we can see minima occur at m=4. Training error decreasing throughout and Testing error first decreases up to 4 which is underfitting case and then increasing after m=4 which is overfitting case. So polynomial degree should be 4 for best fit. This is expected as Erms vs polynomial curve is u-shape for testing error.

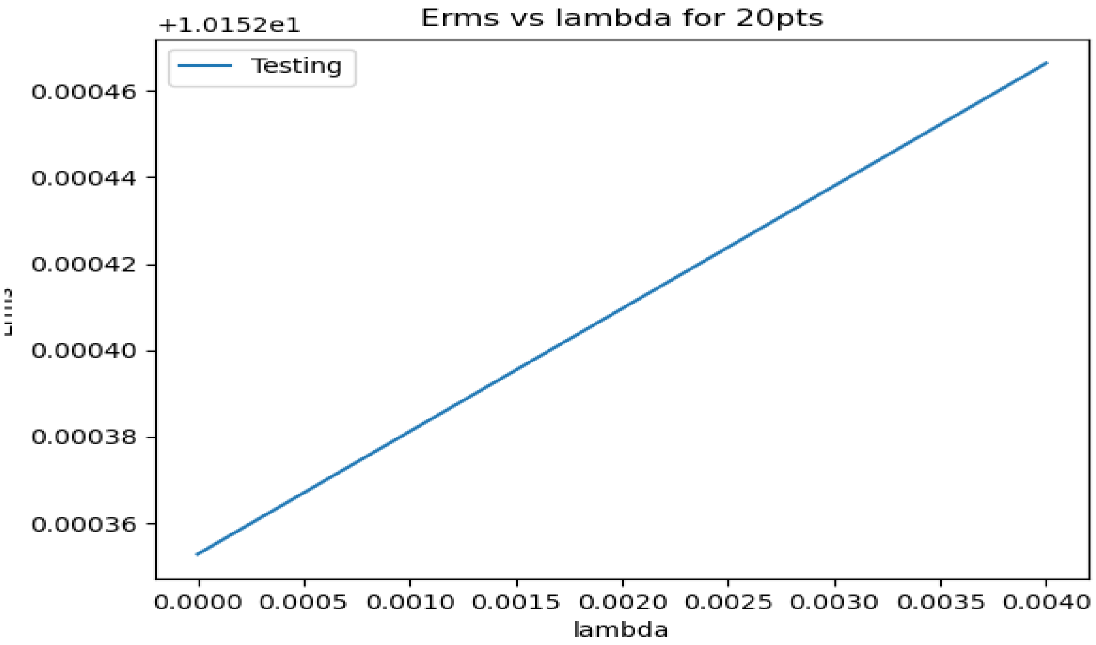
B)Train validation-set:



Here is the plot between Error vs validation set size. Here we clearly see that Erms is decreasing with increase in batch size for testing Error for remaining points up to size n=11 and after that it is increasing very slightly which is because we should have a validation set so that we can train data so that our function fit the curve but should have a testing set so that we test the data.

Here, for m=4 validation set size is 11.

C)Hyperparameter(lambda) tuning:



Now, above plot is for Erms vs lambda, here we see Erms is increasing with lambda for m=4 which means curve is best fitted and adding lambda is causing underfitting. Here, no case of overfitting is observed.

From the figure lambda is almost 0.

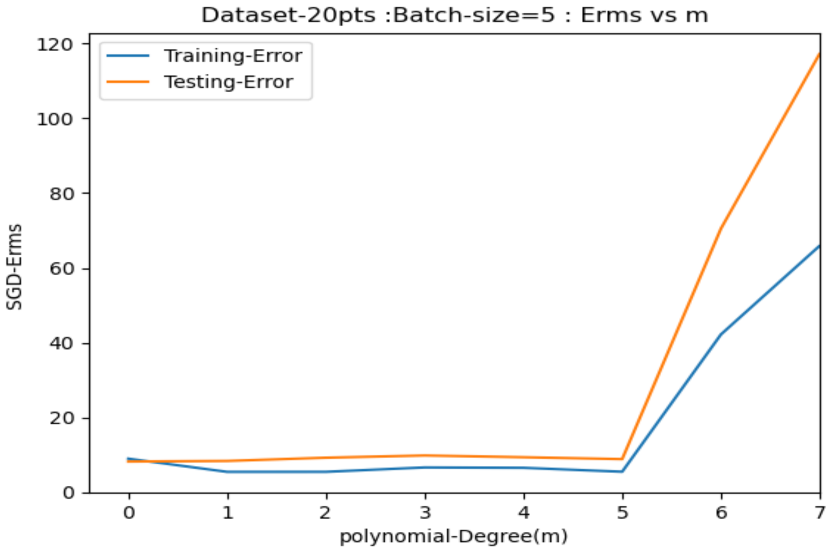
So, overall, we have m=4, lambda=0, validation-set size=11

So, overall polynomial will be (from the code):

$Y=13.89775815 - 0.07816197x - 0.41333523x^2 + 1.2520562x^3 - 0.42046043x^4$

Now, we will do stochastic gradient analysis:

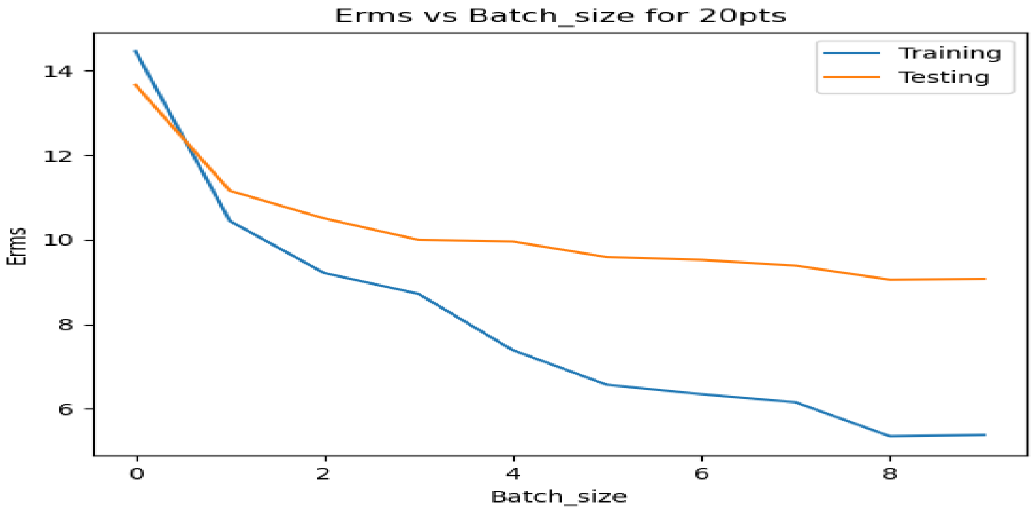
D) Erms vs polynomial degree m



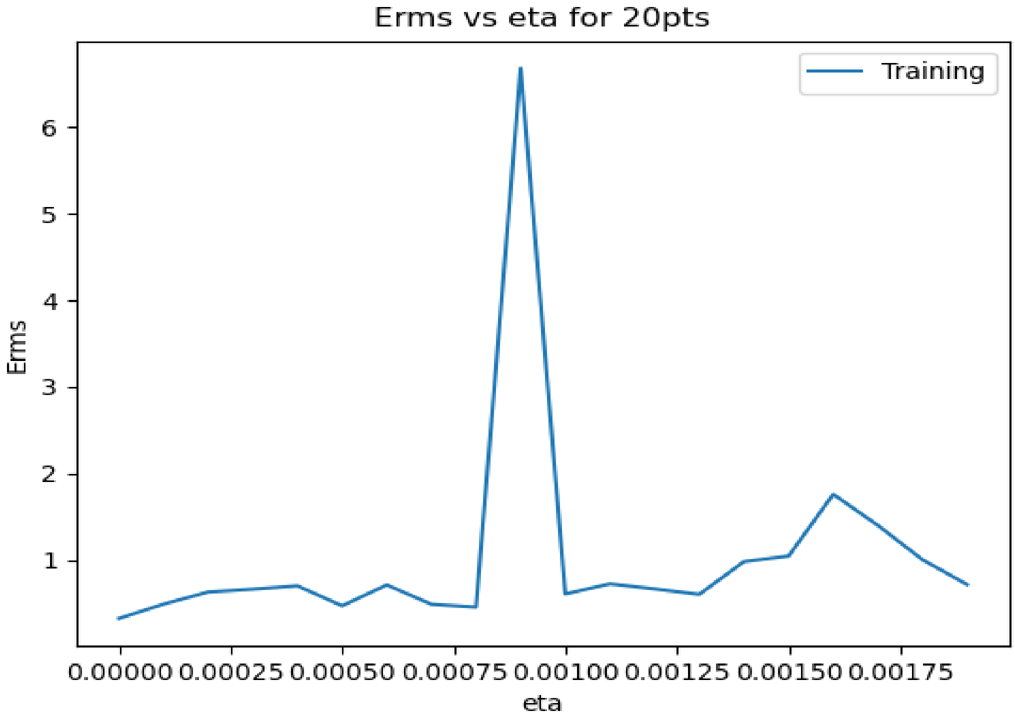
Here, is stochastic gradient descent part for 20 points. We see here also first testing error is decreasing (underfitting case) then increasing (overfitting case) after m=5.

So, here m=5

E)Erms vs Batch_size



Here, in stochastic gradient descent error is decreasing throughout for batch size. It is because taking more points to train will reduce the error. So, here best batch size is full batch size.

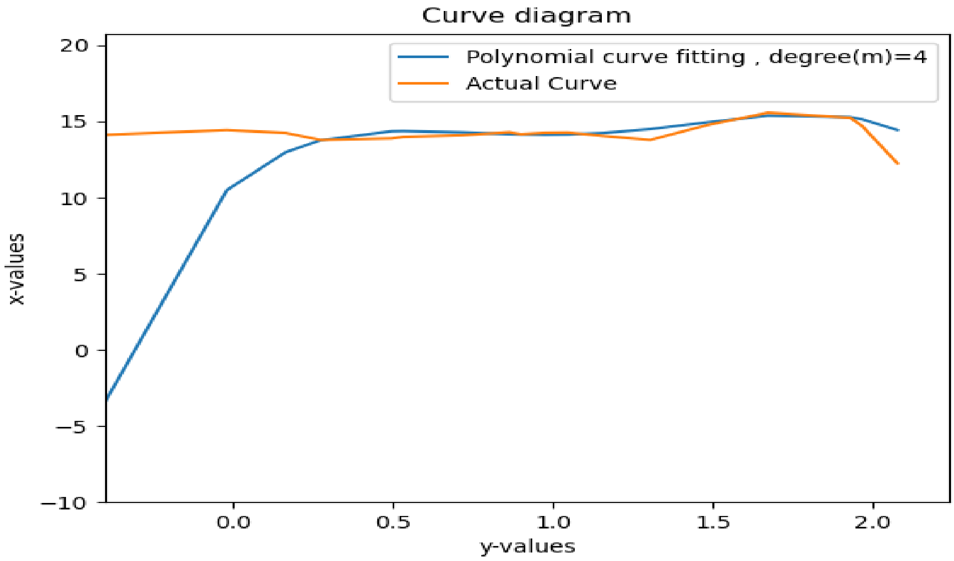


This is the Erms vs eta curve, it does not have a fix relationship with Erms but we can see we have minimum Erms for eta around 0.005

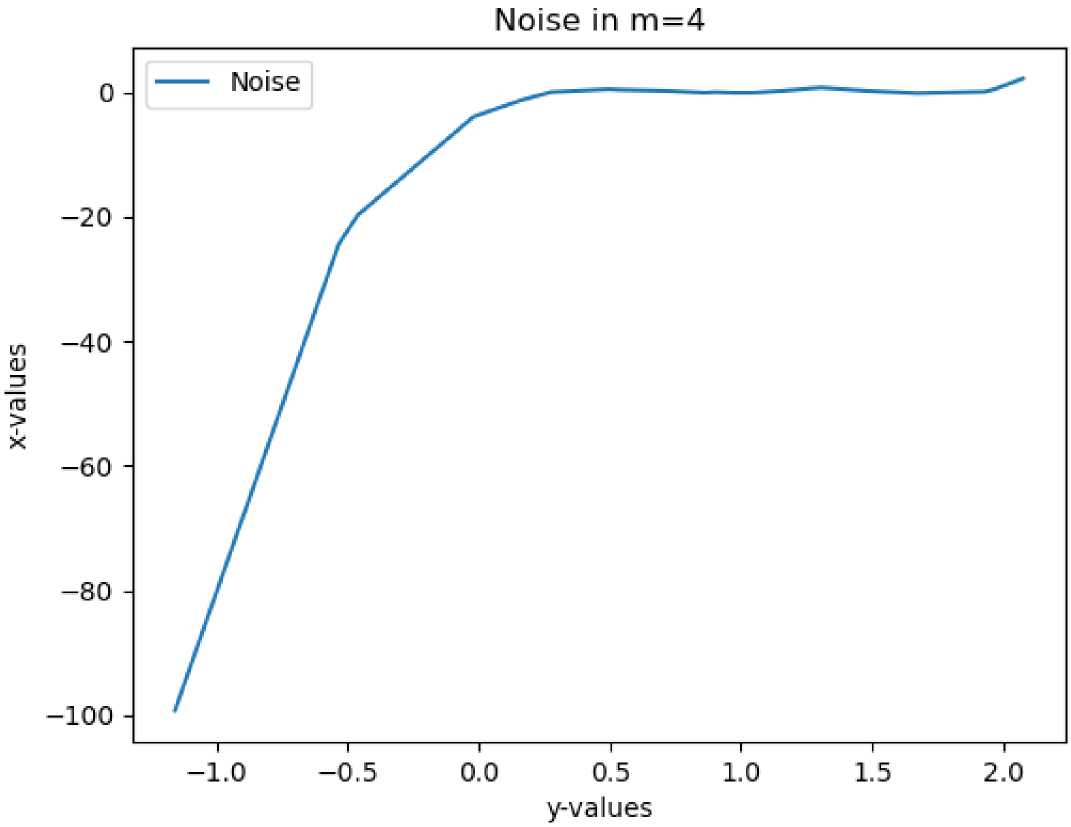
So, overall, we have m=5, lambda=0, eta=0.005, batch size=20

So, overall polynomial will be (from the code) :

$Y= 11.26426+1.45652204x+2.004553x^2 +0.16841548x^3 -0.151842x^4 -0.23165x^5$



Now here is the actual curve vs fitting polynomial for m=4. We saw here that curve rights side of origin does not match exactly, it is because there is only 1 to 2 points which does not affect error so much as we have cumulated all points in one side.



This is the noise curve for m=4, at values less than 0 we see high noise it is because we have very less points i.e. 20 points and we have very less negative points, so error is approx. 0 near most of the points.

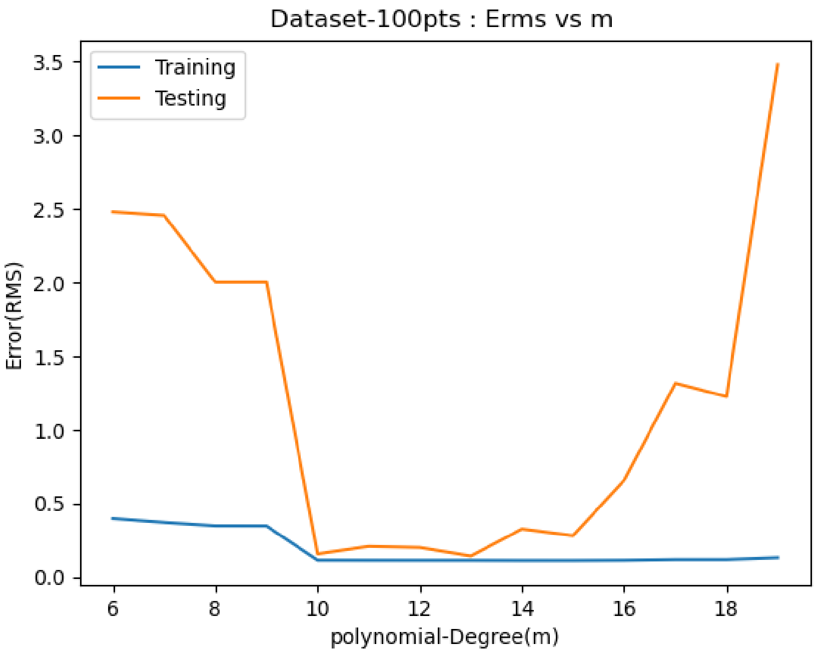
mean= -7.218383654620844

variance= 490.5348778310099

As points is very less, variance is that much high.

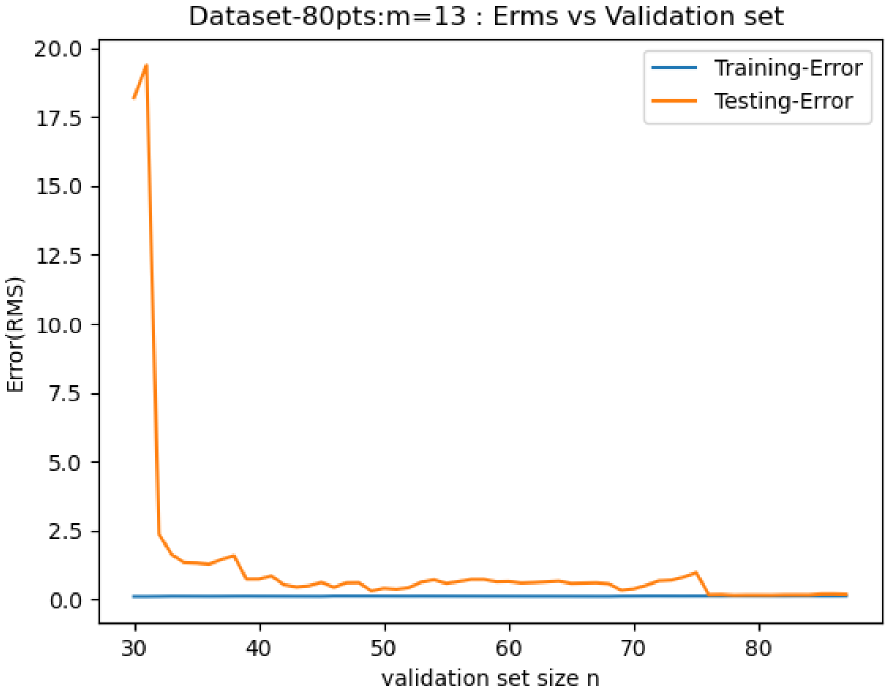
Case2 : For 100 points

A) Erms vs m



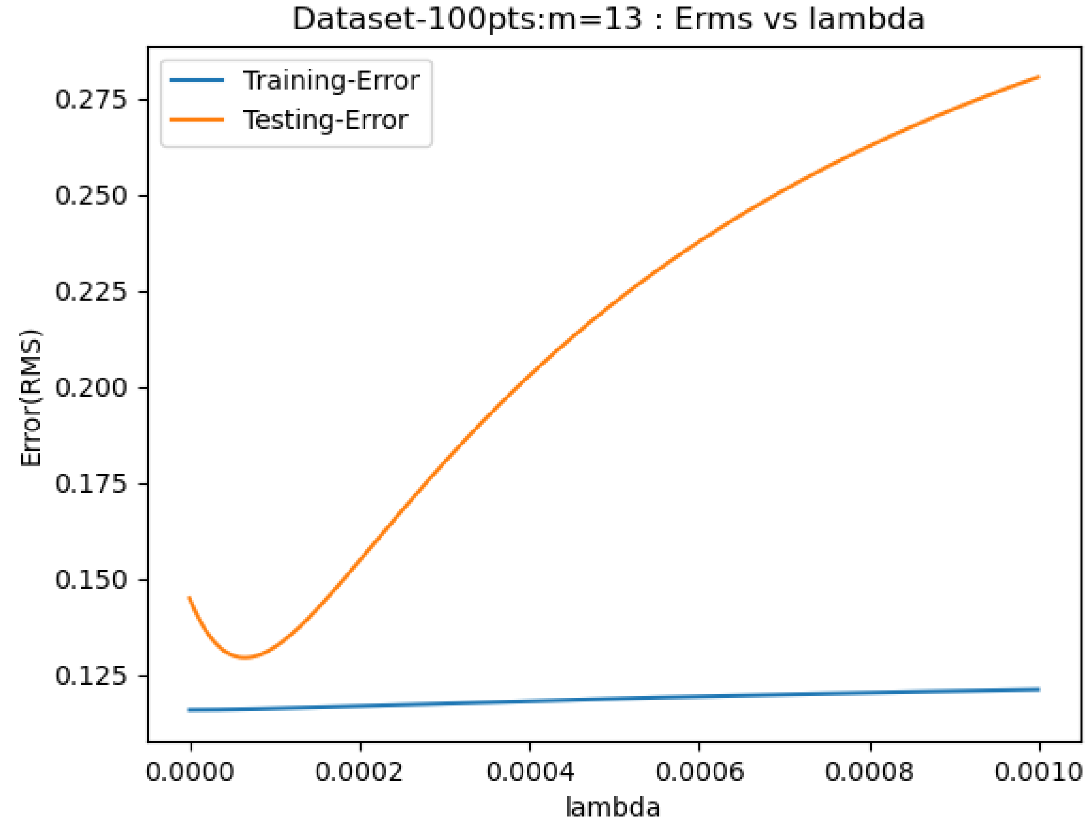
This is the error curve for 100 points for Moore Penrose polynomial fitting. We can see minima occur at m=13. Training error decreasing throughout and Testing error first decreases up to 13 which is underfitting case and then increasing after m=13 which is overfitting case.

B)Erms vs Validation set



Here is the plot between Error vs validation set size Here we clearly see that Erms is decreasing with increase in batch size for testing Error for remaining points upto size n=80 and after that it is increasing(almost constant) very slightly which is because we should have a high training data points so that model overlaps much better.

B) Hyperparameter Tuning



Here, we get the same curve as we get in class i.e U shape. Before minima, it is overfitting case and after minima it is underfitting case.

Here, minima occurs approximately at approximately in range lambda= $5*10^{-5}$.

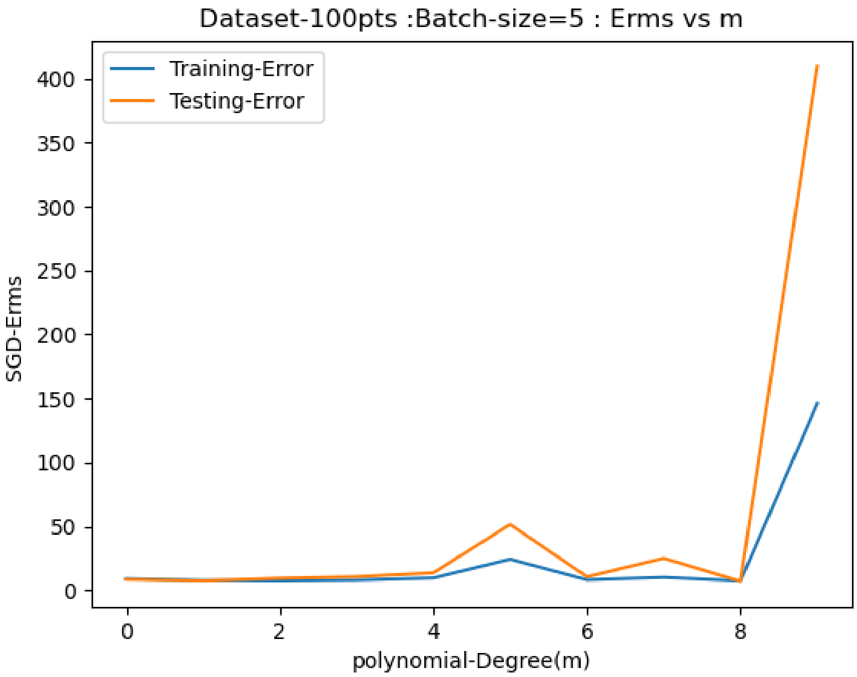
So, overall we have m=13, lambda=10^{-5}, validation-set size=80

So, overall polynomial will be (from the code) :

$$Y=14.21635946-0.5975059x-3.4050552x^{2}+3.47303x^{3}+12.08224191x^{4} -9.89886553x^{5} -11.33295862x^{6} + 9.80500604x^{7}+1.97234854x^{8} -2.22996488x^{9}+0.2002153x^{10} -0.27747251 x^{11}+0.2027221 x^{12} -0.03048975 x^{13}$$

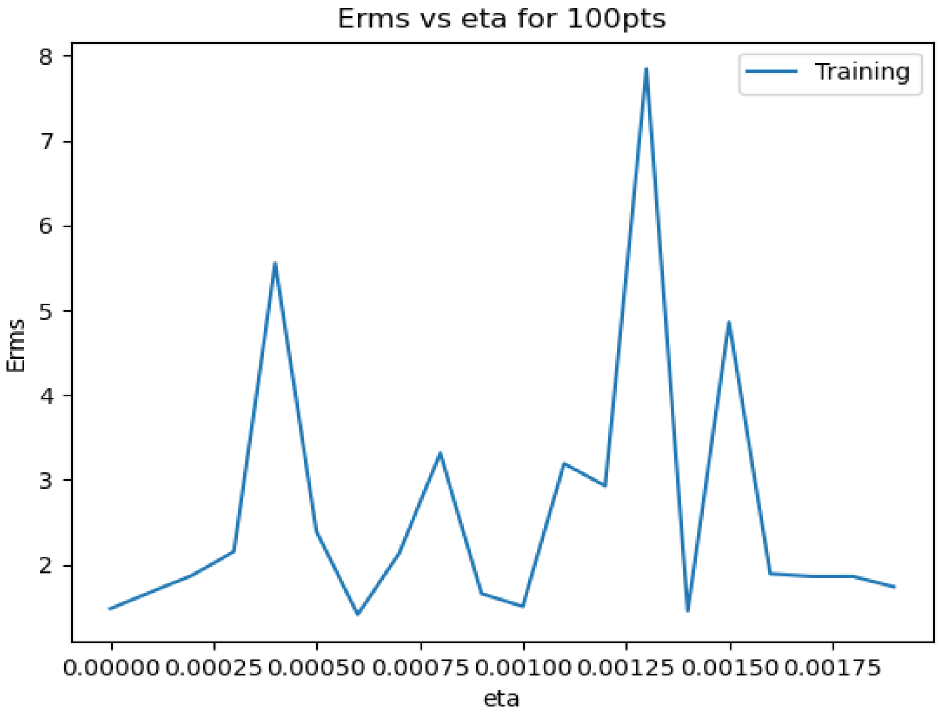
Now, we will do stochastic gradient analysis:

A) <u>Erms vs m</u>

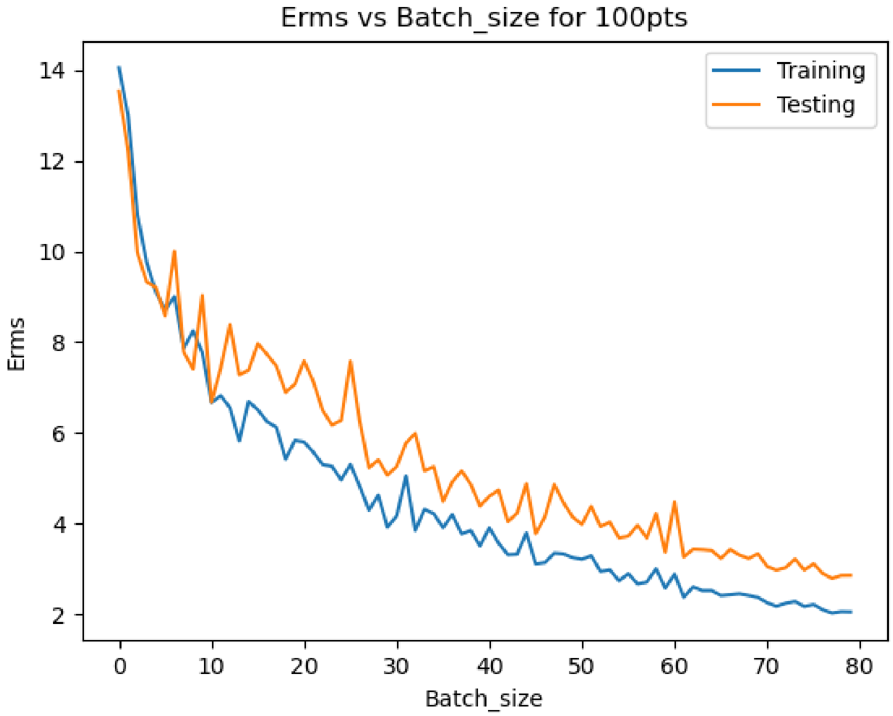


Here, is stochastic gradient descent part for 100 points. We see here also first testing error is decreasing (underfitting case) then increasing(overfitting case) after m=8

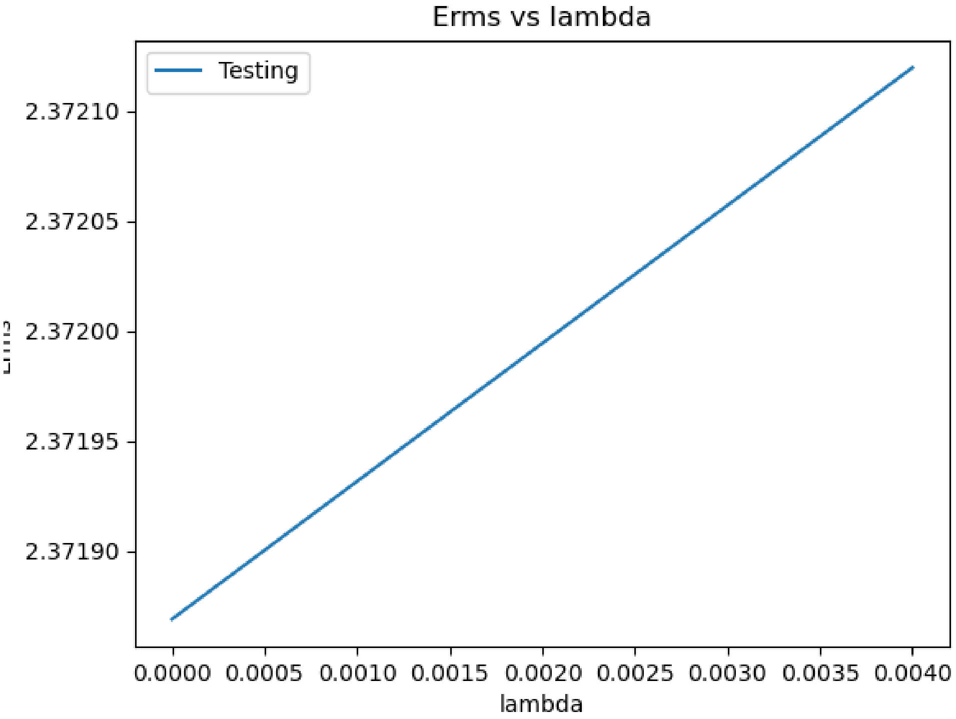
B)**<u>Hyperparameter Tuning</u>**



This is the Erms vs eta curve, it does not have a fix relationship with Erms but we can see we have minimum Erms for eta around 0.0055

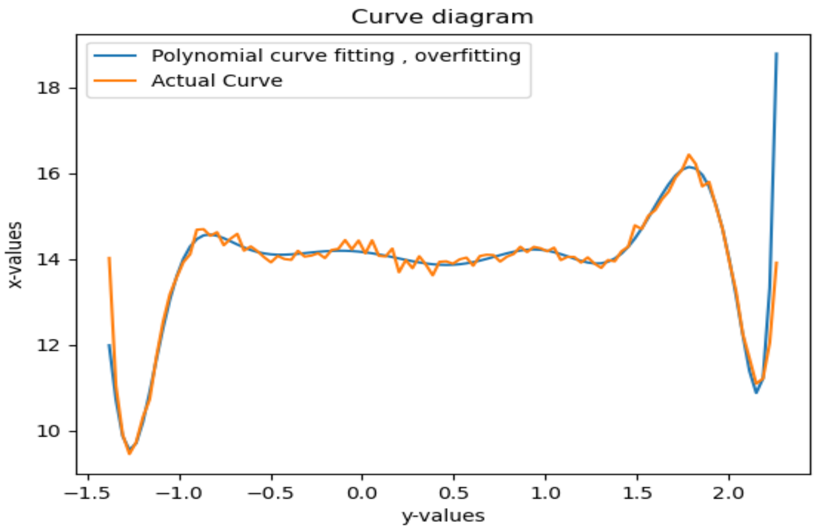


Here, in stochastic gradient descent error is decreasing throughout for batch size. It is because taking more points to train will reduce the error. So, here best batch_size is full batch_size.

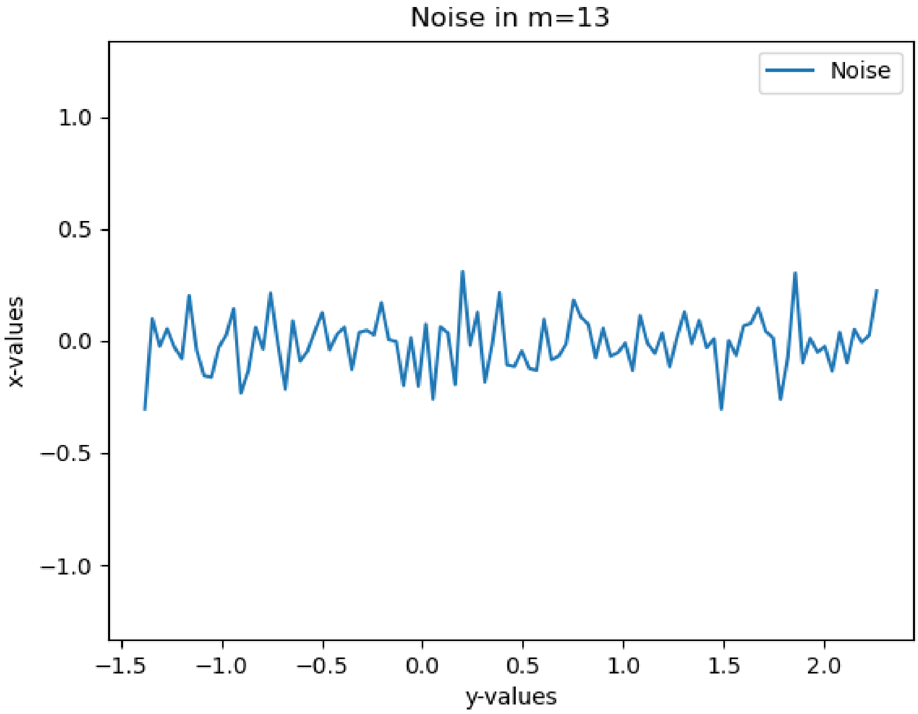


Now, above plot is for Erms vs lambda, here we see Erms is increasing with lambda for m=4 which means curve is best fitted and adding lambda is causing underfitting.

So, Here best lambda is 0.



This is the polynomial curve fitted vs actual data-set, here we can clearly see for high number of data points, curve matches exactly for m=13 case.



This is noise curve, we see that noise is very less here, it is because curve matches exactly.

Here,

mean= -0.009498575807364205

variance= 0.014841690312379117

COMPARISION FOR N=20 VS N=100 CASES

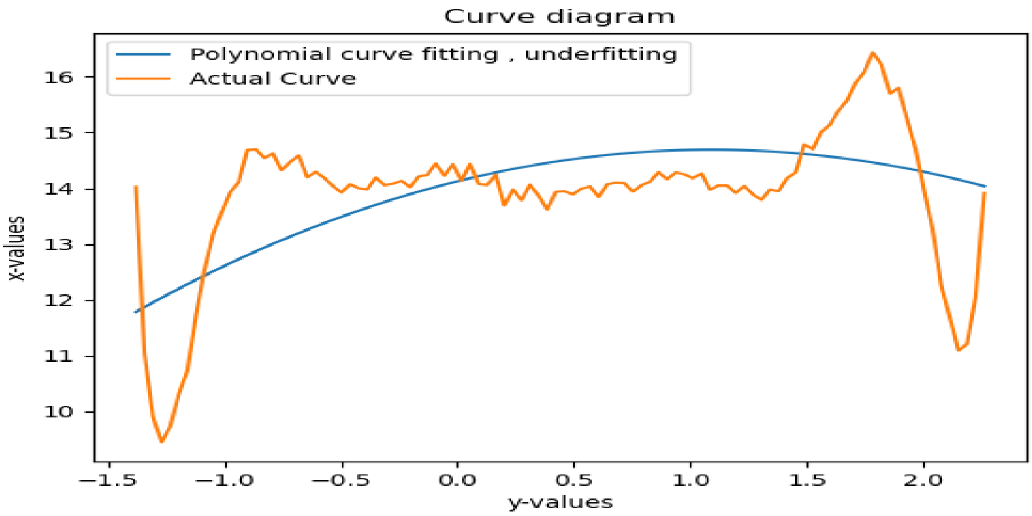
Here, we can see N=20 have points which is very less, so bestfit polynomial will be of lesser degree than N=100 cases, so it will be less flexible. So more Erms because we know more data will lead to better result.

OVERALL CONCLUSION AND POLYNOMIAL:

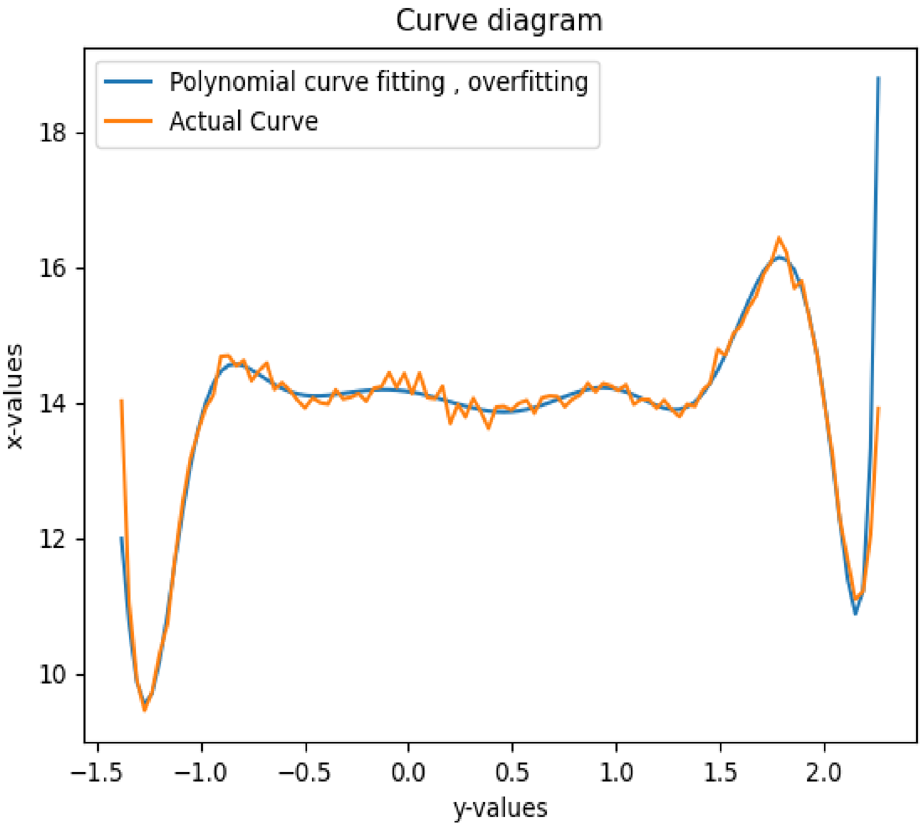
Hence, overall polynomial should be taken of higher data set i.e. 100 points and final polynomial will be and here moore penrose is preferred due to incompetent fitting and moore penrose is relatively faster.

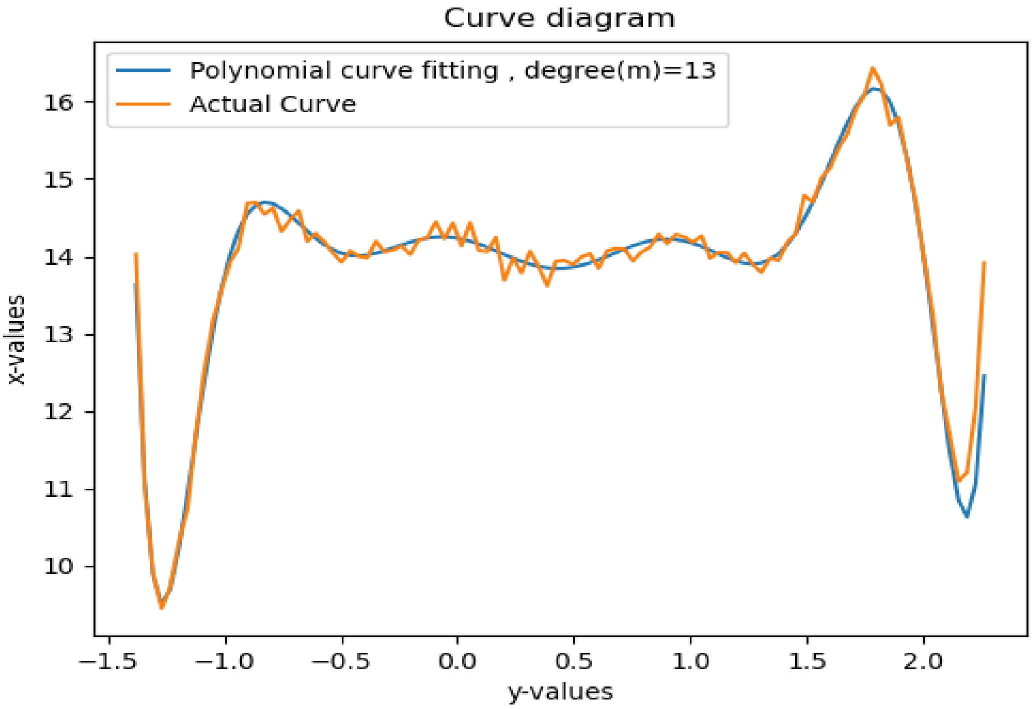
$Y=14.21635946-0.5975059x-3.4050552x^{2}+3.47303x^{3}+12.08224191x^{4}$
$-9.89886553x^{5} -11.33295862x^{6} + 9.80500604x^{7}+1.97234854x^{8}$
$-2.22996488x^{9}+0.2002153x^{10} -0.27747251\ x^{11}+0.2027221\ x^{12}$
$-0.03048975\ x^{13}$

Case of underfitting:-



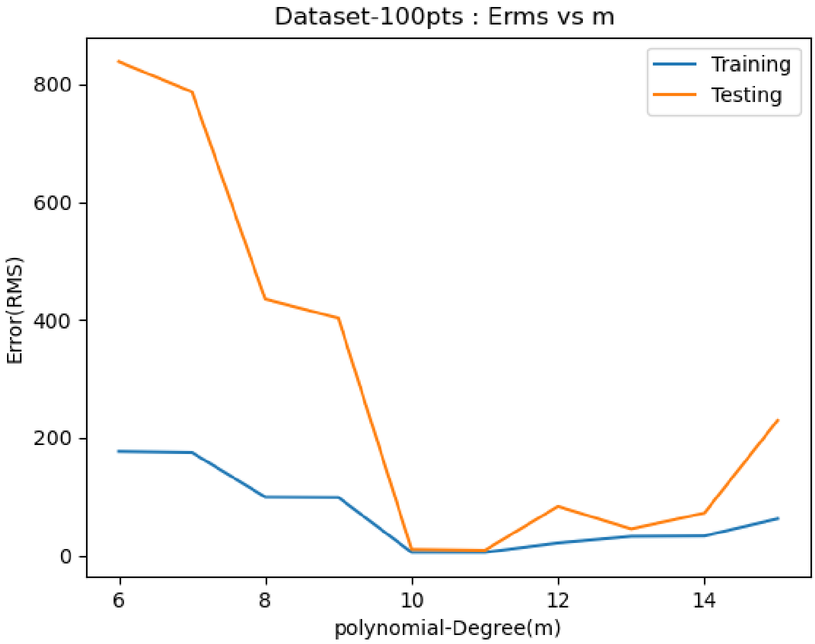
Here, we can see the case of underfitting clearly.



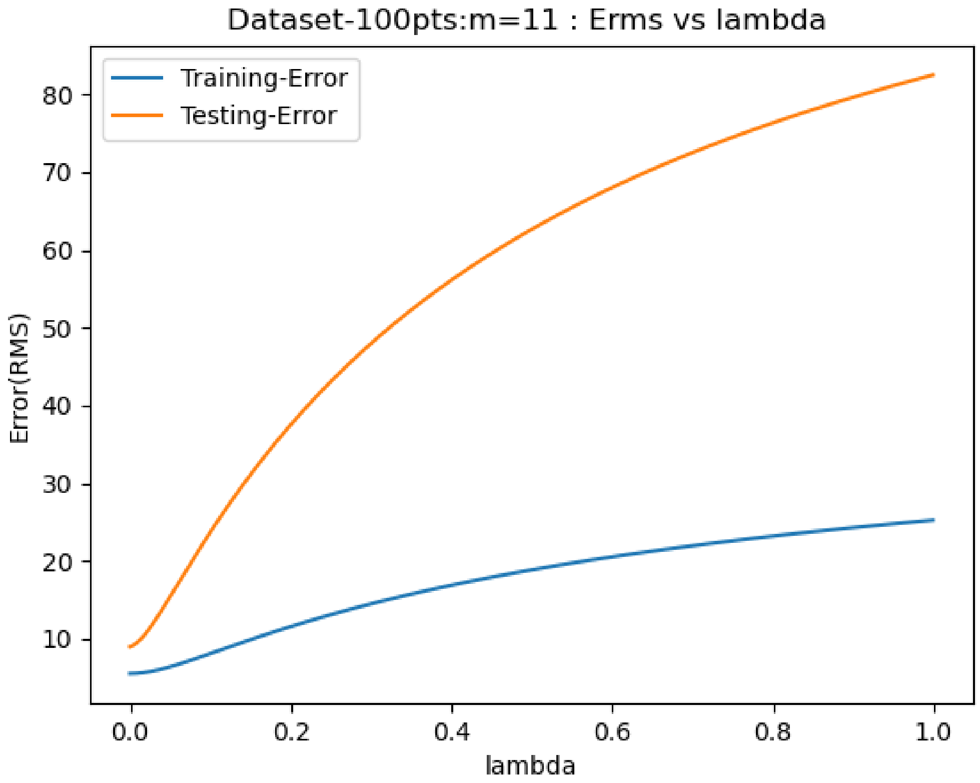


Above 2 curves, above is best fit and below is also fitted curve.

PART 1 B



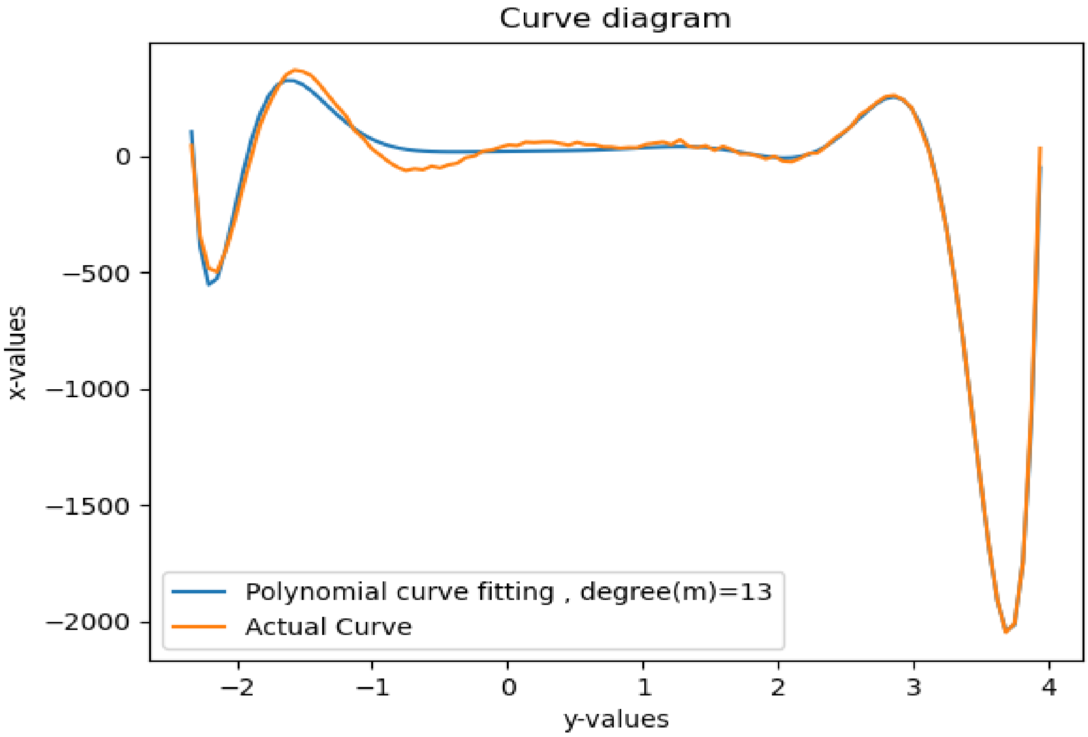
This is the error curve for 100 points for moore-penrose polynomial fitting. We can see minima occur at m=11. Training error decreasing throughout and Testing error first decreases upto 11 which is underfitting case and then increasing after m=4 which is overfitting case.



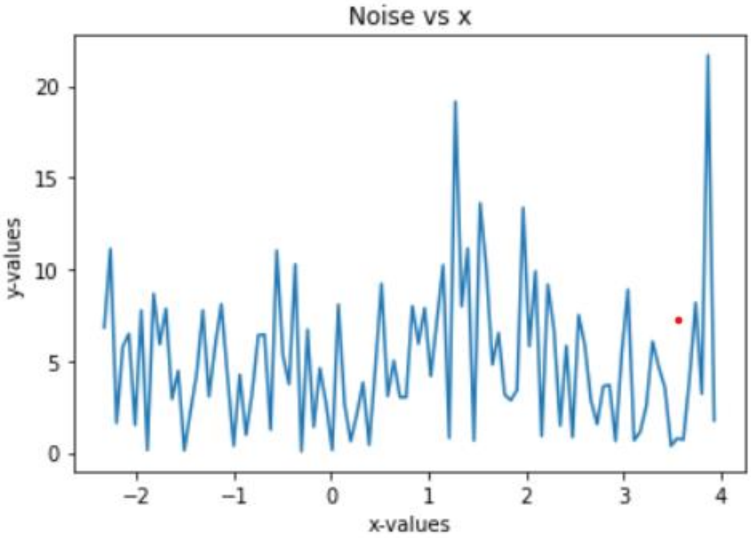
Now, above plot is for Erms vs lambda, here we see Erms is increasing with lambda for m=4 which means curve is best fitted and adding lambda is causing underfitting.

From the figure lambda is almost 0.

Now,we draw actual curve, noise plot and try to identify the polynomial.



This is the fitted polynomial for m=13.



This is the noise we have got in case of non gaussian,

mean= -0.15427889267061232

variance= 40.595842229958606

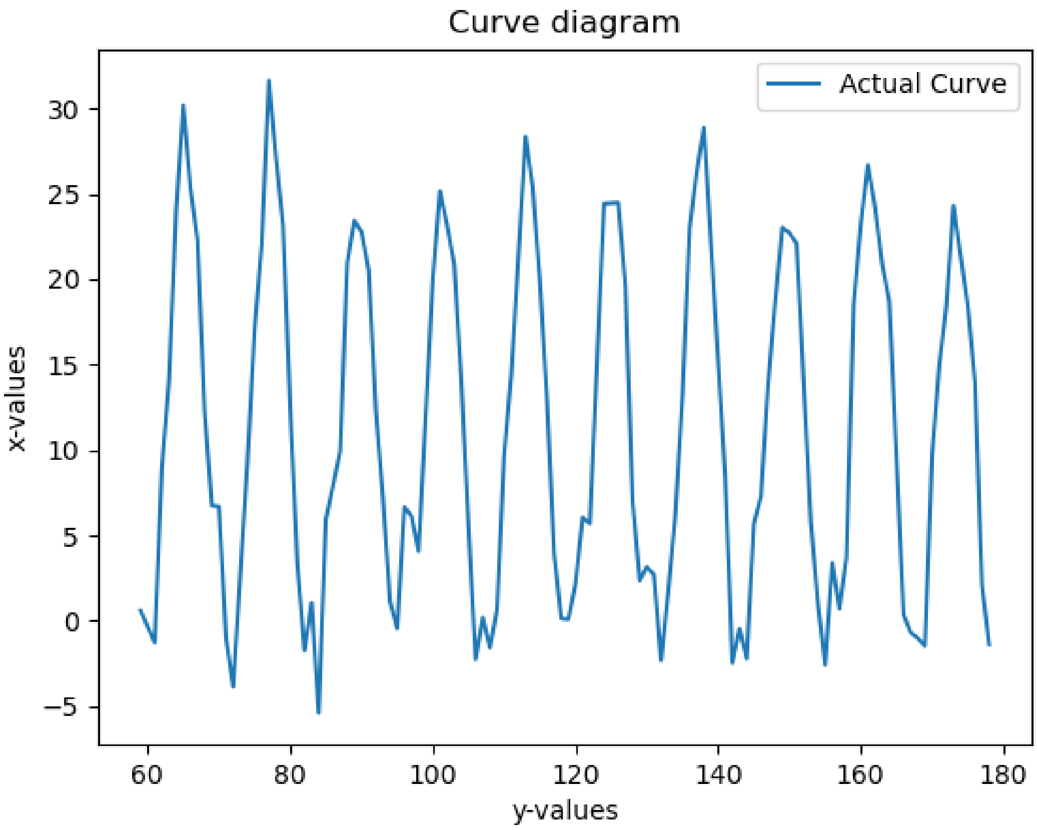
Clearly, it is not uniform and non-gaussian distribution. It seem to be sinusoidal.

PART 2:

For estimating we first merge month and year

So our data set x will be x=2*m+y.

For estimating the related model, we first see the actual curve.



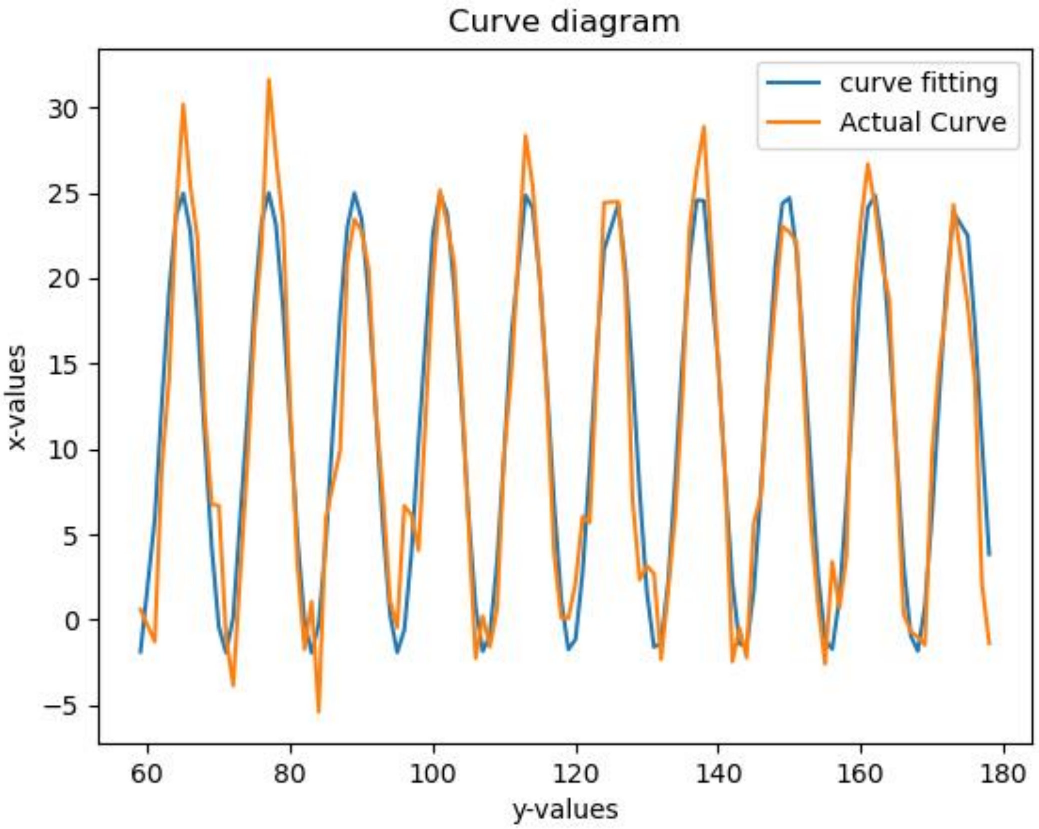
This is the actual curve got by 110 data points. Clearly it seems to be periodic like sinusoidal. So we will use feature vector as:

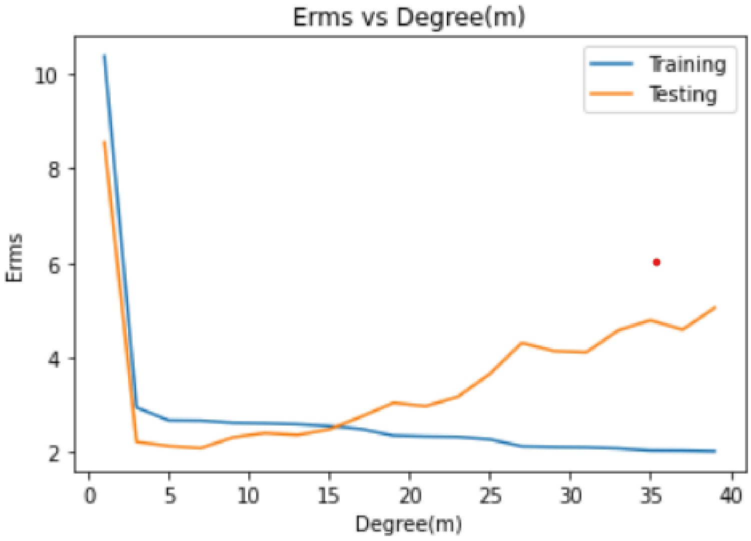
Phi_j = sin(jx) for j odd

Phi_j =cos(jx) for j even

Phi_0= 1 for j=0

Now, I will draw fitted curve for this feature vector for design matrix of size 80 X 3



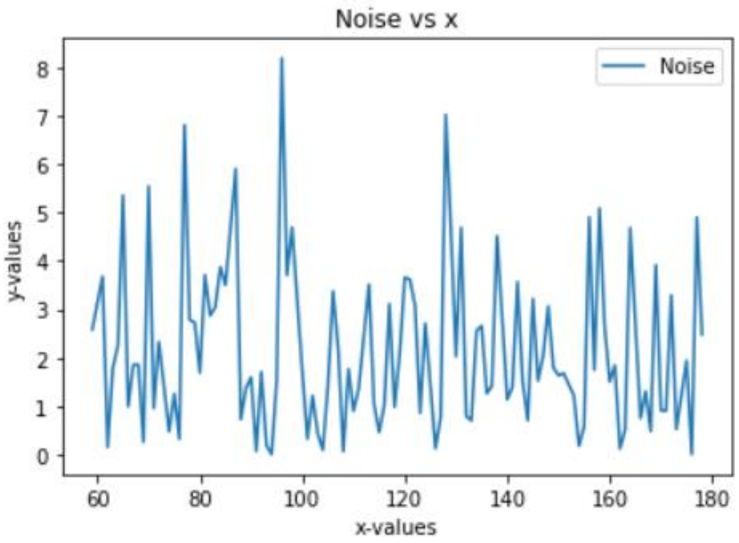


Here, clearly minima is at m=3. So we will find noise at m=3

So feature vector will we [1 sinx cosx]

Here x is in radian

```
Here y=11.38519427 +4.40500429sinx -12.95273667cosx
```



```
mean= 0.029581148979819803
variance= 5.484398118690715
```

End--