1. Basic Implementation

1.1 Using Fixed number of Maximum Iterations (Maxit)

I have set the $maxit = 100$. For different values of alpha, the results are as follow

For alpha = 0.1

The gradient explodes and reaches an undesired value.

For alpha = 0.01

The gradient explodes and reaches an undesired value.

For alpha = 0.001

Dataset	MSE	MAE
Train	0.4820761997	0.5515330842
Validation	0.5785129074	0.5791165942

1.2 Stop when the relative decrease in the cost function on validation data drops below a Threshold value (Reltol)

I have set the threshold value to 1e-50.

Dataset	MSE	MAE
Train	0.6708609876	0.640970994
Validation	0.7004918027	0.6704655356

2. Ridge Regression

The loss function

$$J = \frac{1}{2N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})^2 + \lambda\sum_{j=1}^{n}\theta_j^2)$$

Thus we need to minimize J

We find the gradient of J
Taking gradient w.r.t θ_i

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}(\frac{1}{2N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})^2 + \lambda\sum_{j=1}^{n}\theta_j^2))$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{2N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})x_{ij} + \lambda 2\theta_j)$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{N}(\sum_{i=1}^{m}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})x_{ij} + \lambda\theta_j)$$

The gradients are exploding for alpha = 0.1 and 0.01. The tables below are for alpha = 0.001

2.1 Maxit

λ	Dataset	MSE	MAE
5	Train	0.4818566141	0.5514516287
5	Validation	0.5784335876	0.5790955687
25	Train	0.4809797819	0.5511235398
25	Validation	0.5781223292	0.5790110737

2.2 Reltol

λ	Dataset	MSE	MAE
5	Train	0.6708369788	0.6409701349
5	Validation	0.7004544474	0.6704489352
25	Train	0.6707435193	0.6409666807
25	Validation	0.7003074793	0.6703824869

Comparison with Basic implementation

As we can see from the two tables, ridge regression performed better than the basic implementation.
On the Maxit implementation, making $\lambda = 25$ made the MSE on validation drop from 0.5785 (in basic implementation) to 0.5781. Thus Ridge regression performs better.
All the MSE and MAE are lower in ridge regression by an order of 0.001(approx)

3 Using Scikit-Learn Library

λ	Dataset	MSE	MAE
5	Train	1.59E-29	3.10E-15
5	Validation	1.025033089	0.8322307522

We can see that Scikit-Learn library did a lot of overfitting. Since the training MSE and MAE are both almost 0 where as the validation MSE and MAE are significant. Thus this model didn't perform well since the validation error and training error should be close to each other and both should be low.
Therefore the basic linear regression model and the ridge regression models are better than scitkit learn model

4 Feature Selection

I have set $\alpha = 0.1$

4.1 SelectKBest

λ	Dataset	MSE	MAE
5	Train	1.184151069	0.9048716049
5	Validation	1.406317287	0.9953324699

Since we are dealing with lesser number of features(10 in this case) , the model didn't performed well as compared to previous implementations

4.2 SelectFromModel

λ	Dataset	MSE	MAE

λ	Dataset	MSE	MAE
5	Train	1.205193969	0.8449160389
5	Validation	1.67607792	1.046997379

5 Classification

Loss function

$$J(\theta) = -y\log(h_\theta(x)) - (1-y)log(1-h_\theta(x))$$

Where

$$h_{\theta_r}(x) = \frac{e^{\theta_r^T X}}{1+\sum_{p=1}^{8} e^{\theta_p^T X}}$$

Explanation

We will seperately find the error in the θ_i's individually and sum them all.

Loss function when the j^{th} dataset belongs to any class 1-8 is (i = class of the j^{th} dataset)

$$J_j(\theta) = -\log(h_{\theta_i}(x))$$

Else if it belongs to class 9, we take sum of all $h_{\theta_i}(x)$ for all integer $i \in [1,8]$ and calculate the final J_j as

$$J_j(\theta) = -log(1-\sum_{i=1}^{8} h_{\theta_i}(x))$$

Now to find the total error for all integers $j \in [1,N]$ we just take the sum over all $J_j(\theta)$

$$J(\theta) = \sum_{j=1}^{N} J_j(\theta)$$

If a dataset lies in class p_i (where $p_i \in \{1,2,3,....,9\}$ we can say x lies in p_i. Thus we can say that

$J = -(\sum_{x \in p_1} \log(\frac{e^{\theta_1^T x}}{1+\sum_{j=1}^{8} e^{\theta_j^T x}}) + \sum_{x \in p_2} \log(\frac{e^{\theta_2^T x}}{1+\sum_{j=1}^{8} e^{\theta_j^T x}}) + \ldots\ldots + \sum_{x \in p_9} \log(\frac{1}{1+\sum_{j=1}^{8} e^{\theta_j^T x}}))$
because $1 - \sum_{i=1}^{8} h_{\theta_i}(x) = \frac{1}{1+\sum_{j=1}^{8} e^{\theta_j^T x}}$

Simplifying ...
we get

$$J = (\sum_{x \in p_1} \log(1 + \sum_{j=1}^{8} e^{\theta_j^T x}) - \log(e^{\theta_1^T x})) + (\sum_{x \in p_1} \log(1 + \sum_{j=1}^{8} e^{\theta_j^T x}) - \log(e^{\theta_1^T x})) + \ldots\ldots + \sum_{x \in p_9} \log(1 + \sum_{j=1}^{8} e^{\theta_j^T x})$$

Now taking the differential of J with respect to θ_1

$$\frac{\partial J}{\partial \theta_1} = \sum_{x \in p_1} (\frac{e^{\theta_1^T x} x}{1 + \sum_{j=1}^{8} e^{\theta_j^T x}} - x) + \sum_{x \in p_2} (\frac{e^{\theta_1^T x} x}{1 + \sum_{j=1}^{8} e^{\theta_j^T x}} - 0) + \ldots\ldots + \sum_{x \in p_1} (\frac{e^{\theta_1^T x} x}{1 + \sum_{j=1}^{8} e^{\theta_j^T x}} - 0)$$

Simpifying

$$\frac{\partial J}{\partial \theta_1} = (\sum \frac{x e^{\theta_1^T x}}{1 + \sum_{j=1}^{8} e^{\theta_j^T x}} - \sum_{x \in 1} x)$$

$$\frac{\partial J}{\partial \theta_1} = (\sum x h_{\theta_r}(x) - \sum_{x \in 1} x)$$

I have implemented this equation

We can also see it as

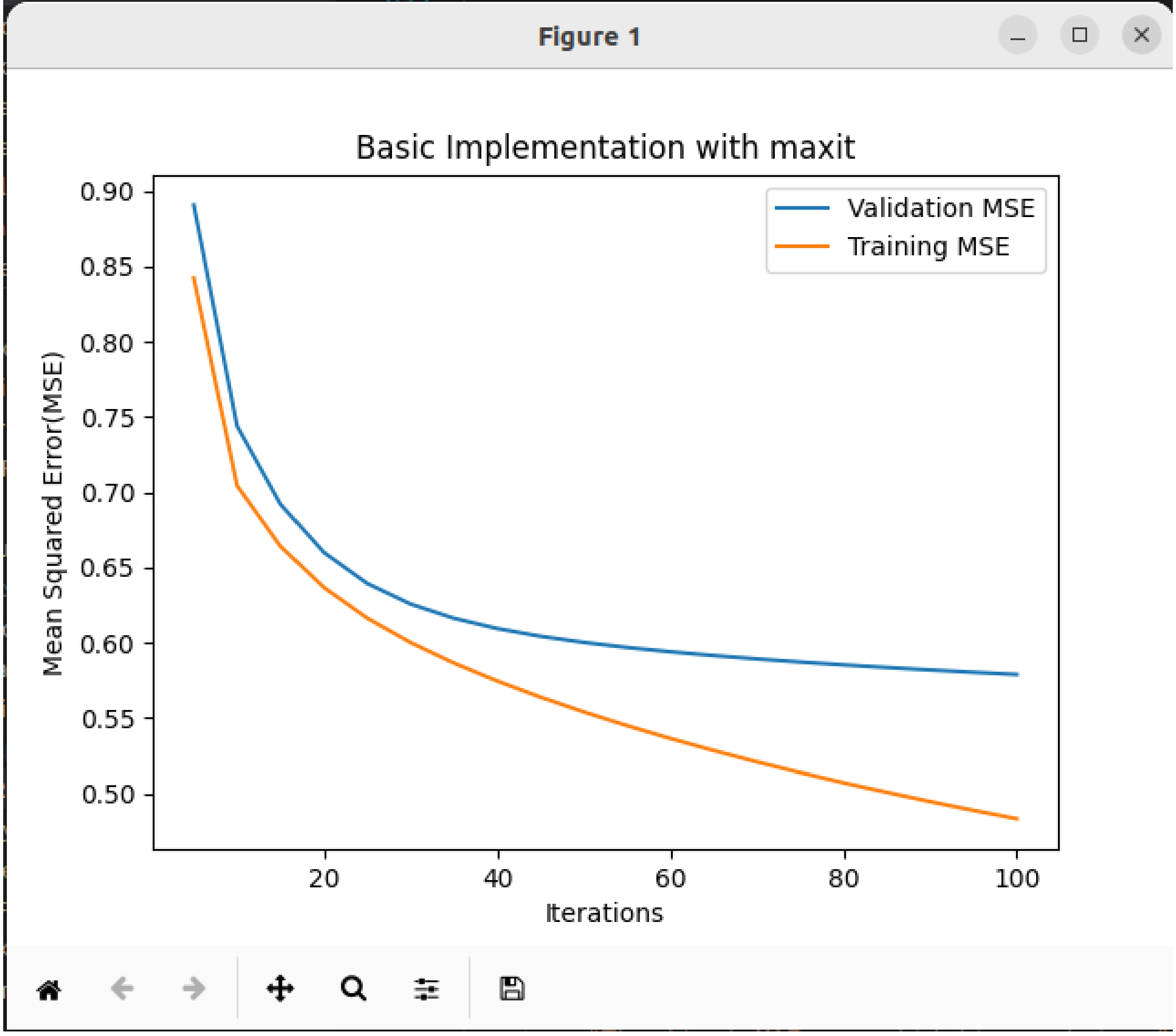
$$\sum x * (\text{predicted probability} - \text{belongs to that class? (1:0)})$$

6 Visualization

Basic implementation

For Maxit

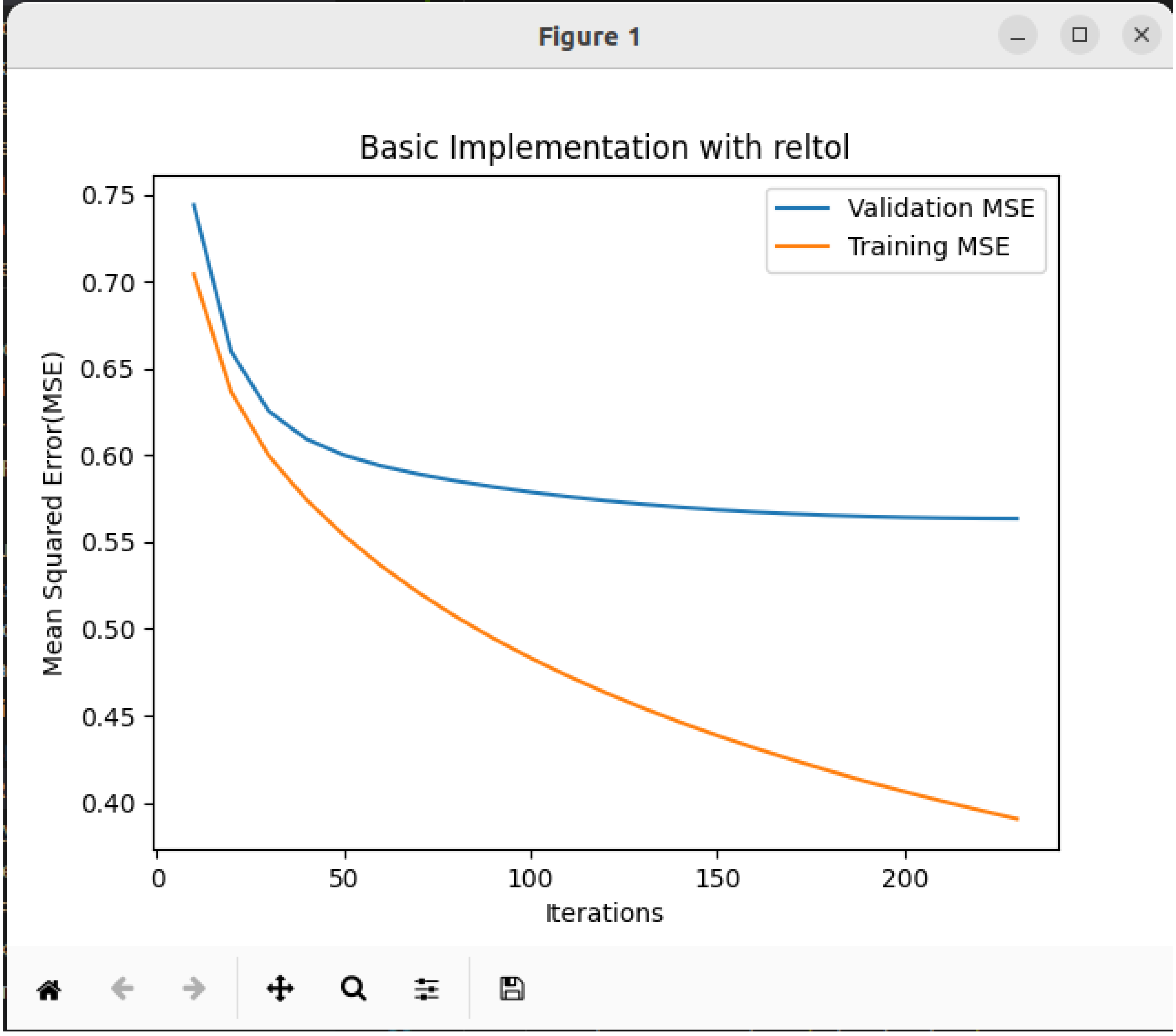
I have set the maximum iterations = 100 value of alpha = 0.001. (Since The MSE was exploding for alpha = 0.01 and 0.1)



Observation :

We can observe that the MAE and MSE fall significantly in the first few iterations and the relative decrease in the MSE in the latter stages is very low. Also the training MSE falls faster than the validation MSE

For Reltol



Since I have set a very small threshold value, it takes more number of iterations.

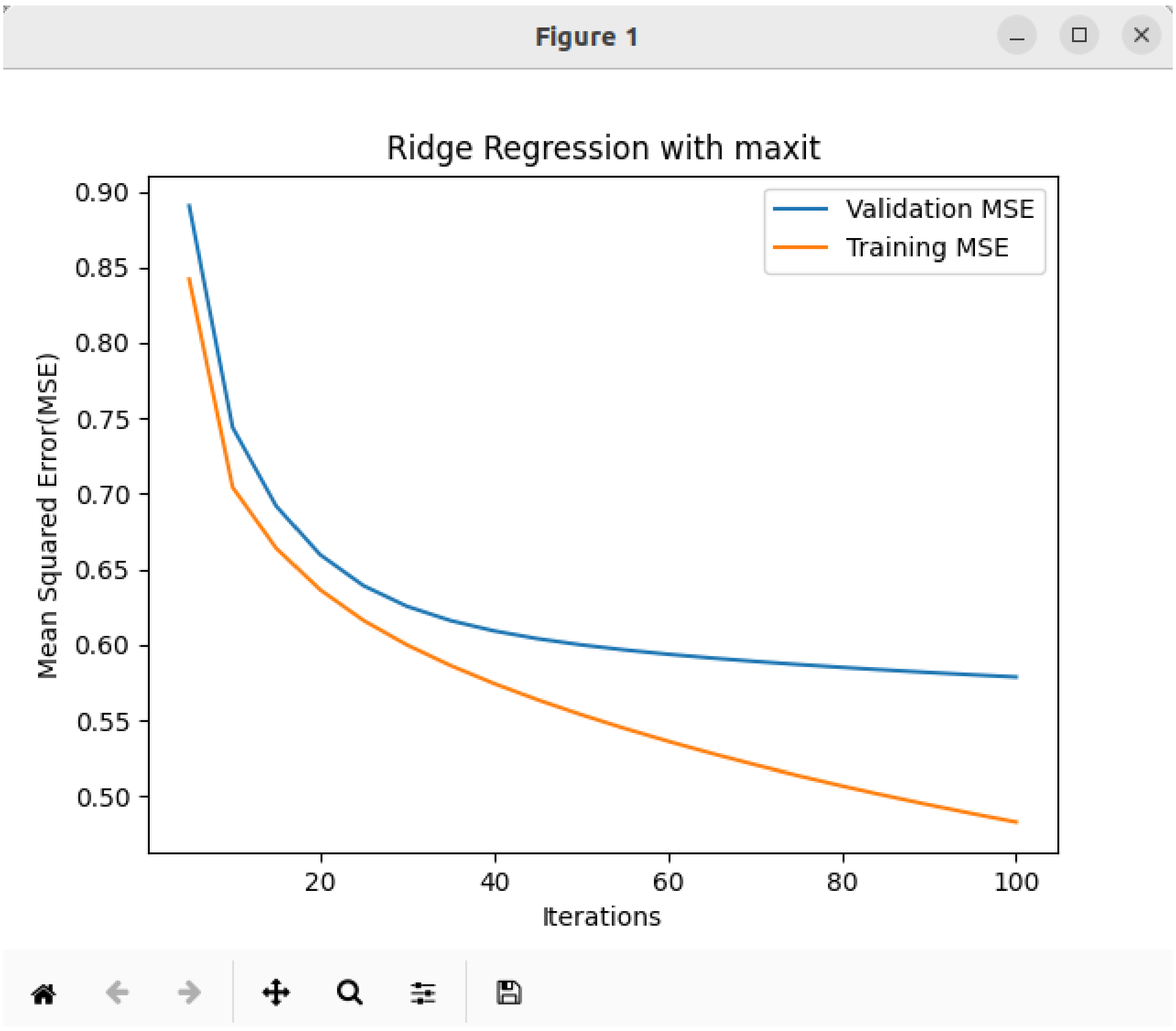
Observations:

We can observe that the MAE and MSE fall significantly in the first few iterations and the relative decrease in the MSE in the latter stages is very low. Also the training MSE falls faster than the validation MSE

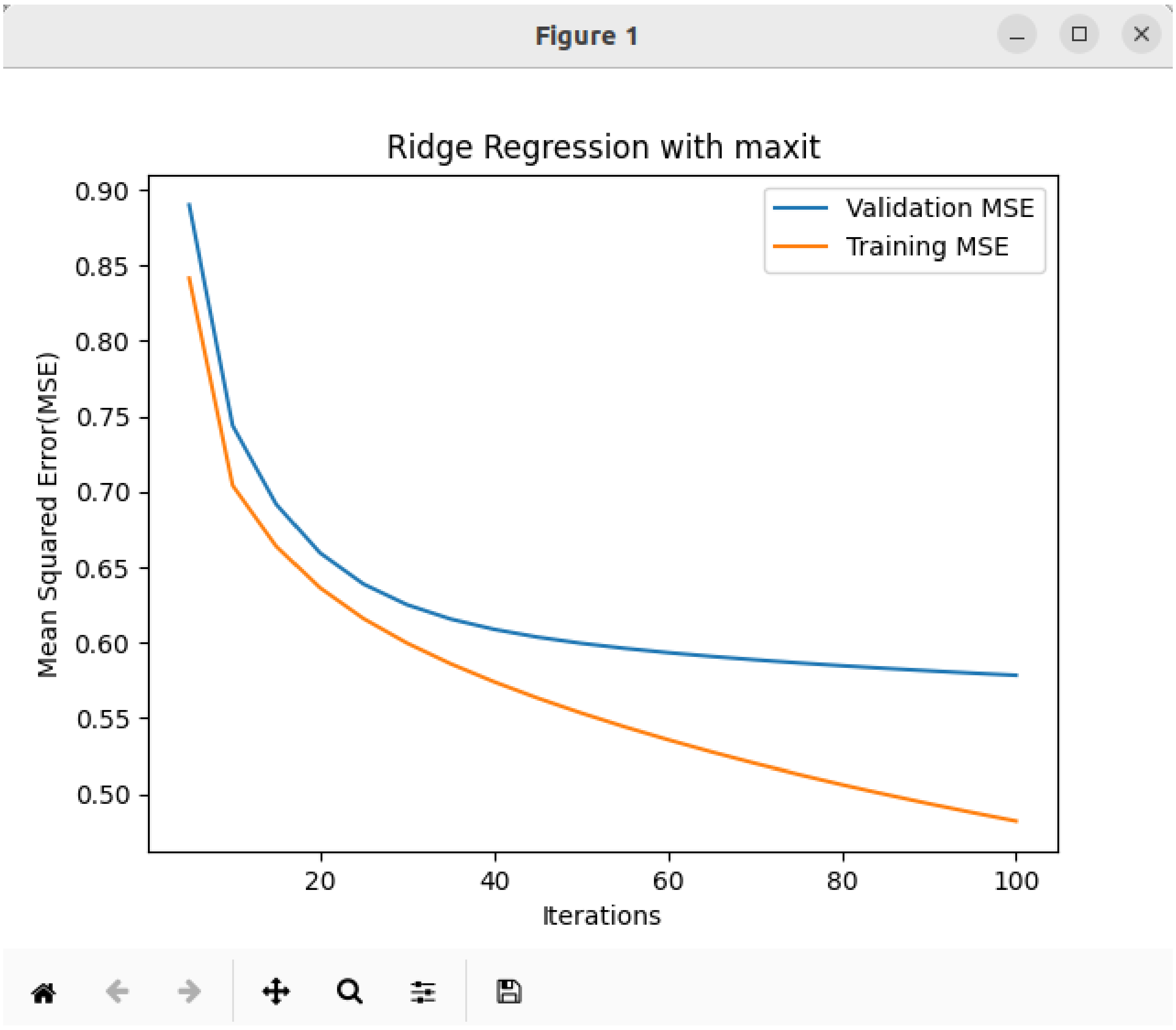
Ridge Regression

Maxit

With $\lambda = 5$



With λ = 25



Observation and comparison with basic implementation

We can see that the Validation MSE falls faster in *Ridge Regression* then *Linear Regression*
Since we have applied a penalty on the weights, it prohibits overfitting to some extent therefore it performs better than the basic implementation of the linear regression

Reltol

$\lambda = 5$

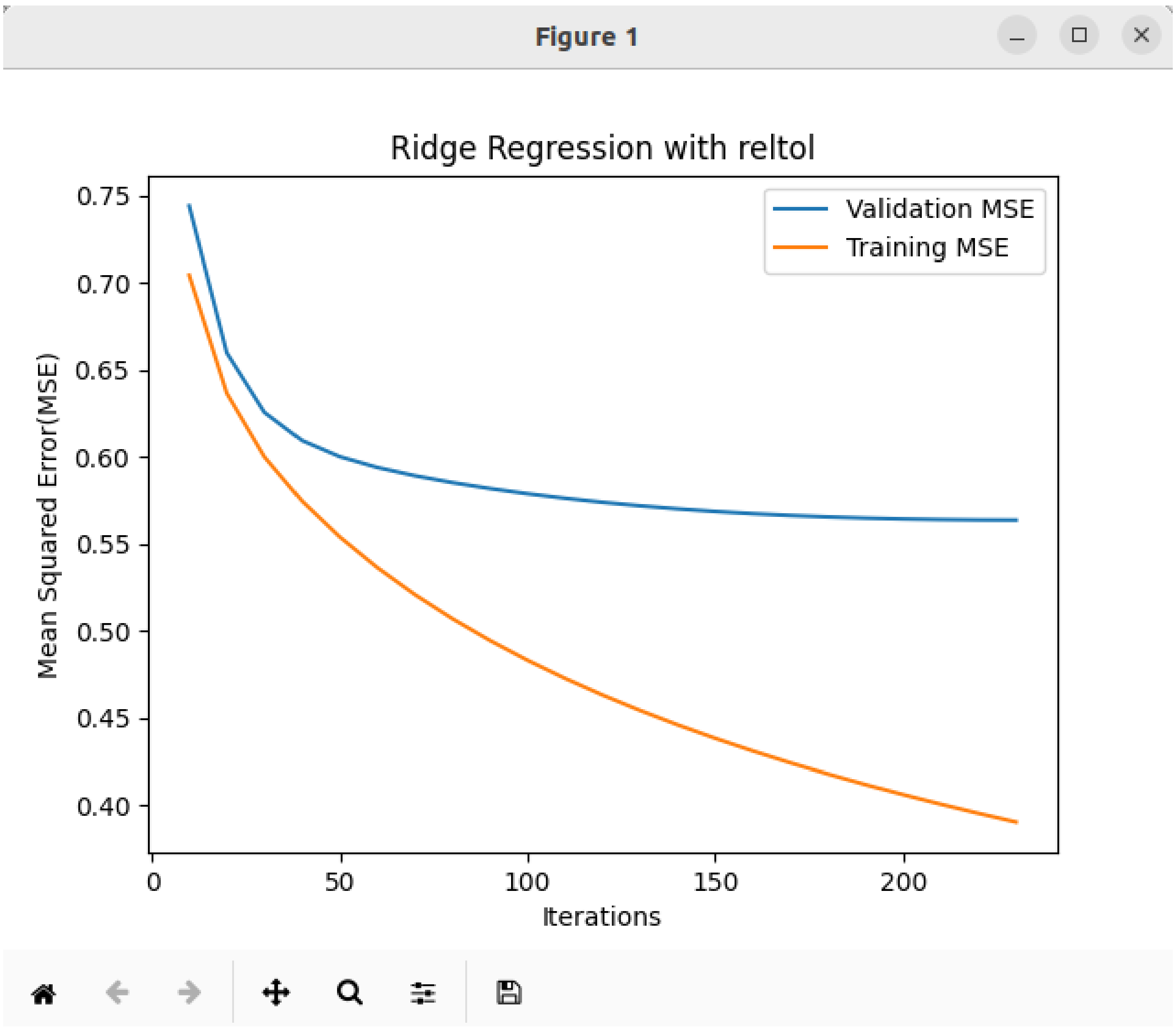
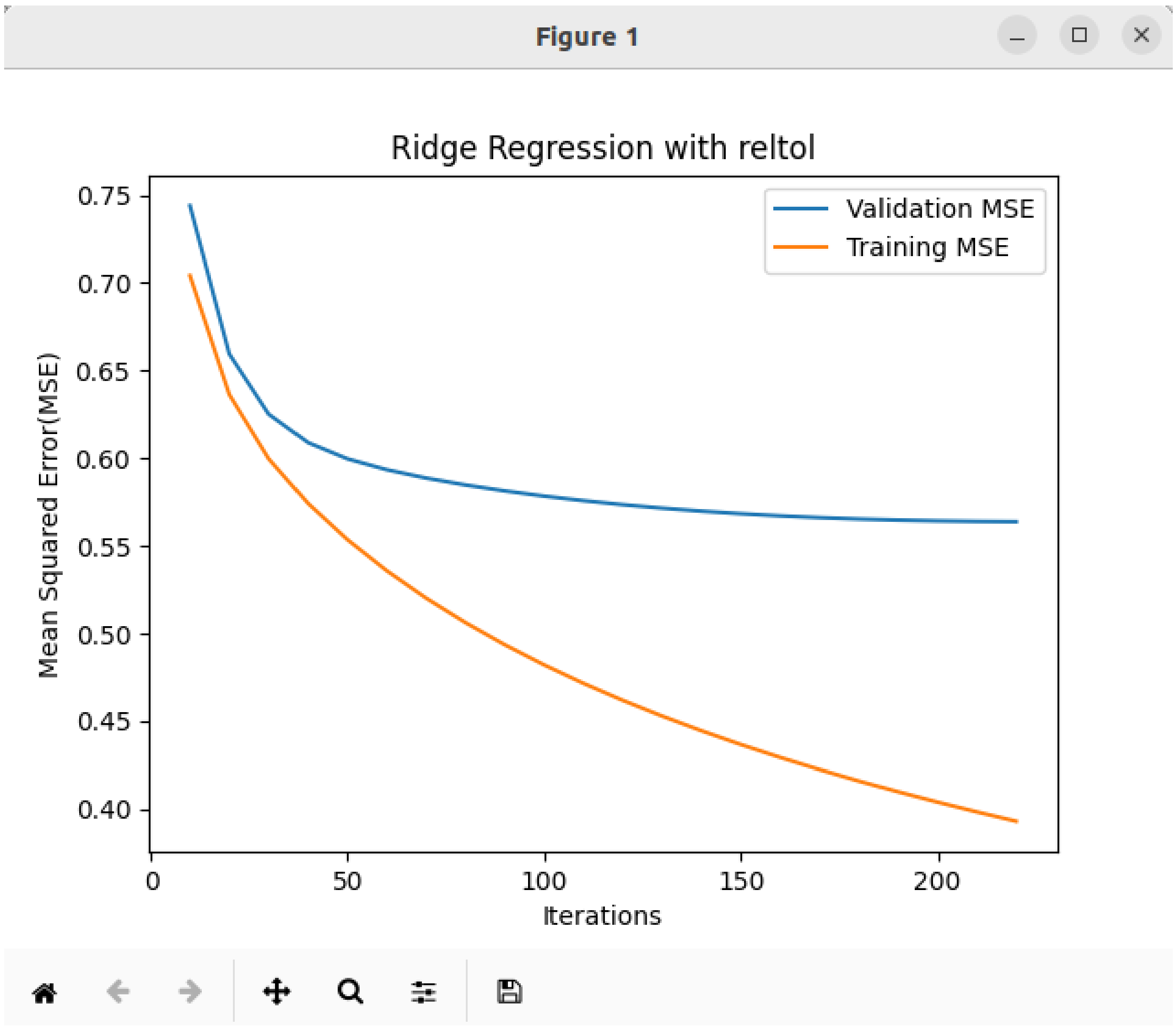


Figure 1
Ridge Regression with reltol
Validation MSE
Training MSE
Mean Squared Error(MSE)
Iterations
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
50
100
150
200

λ = 25

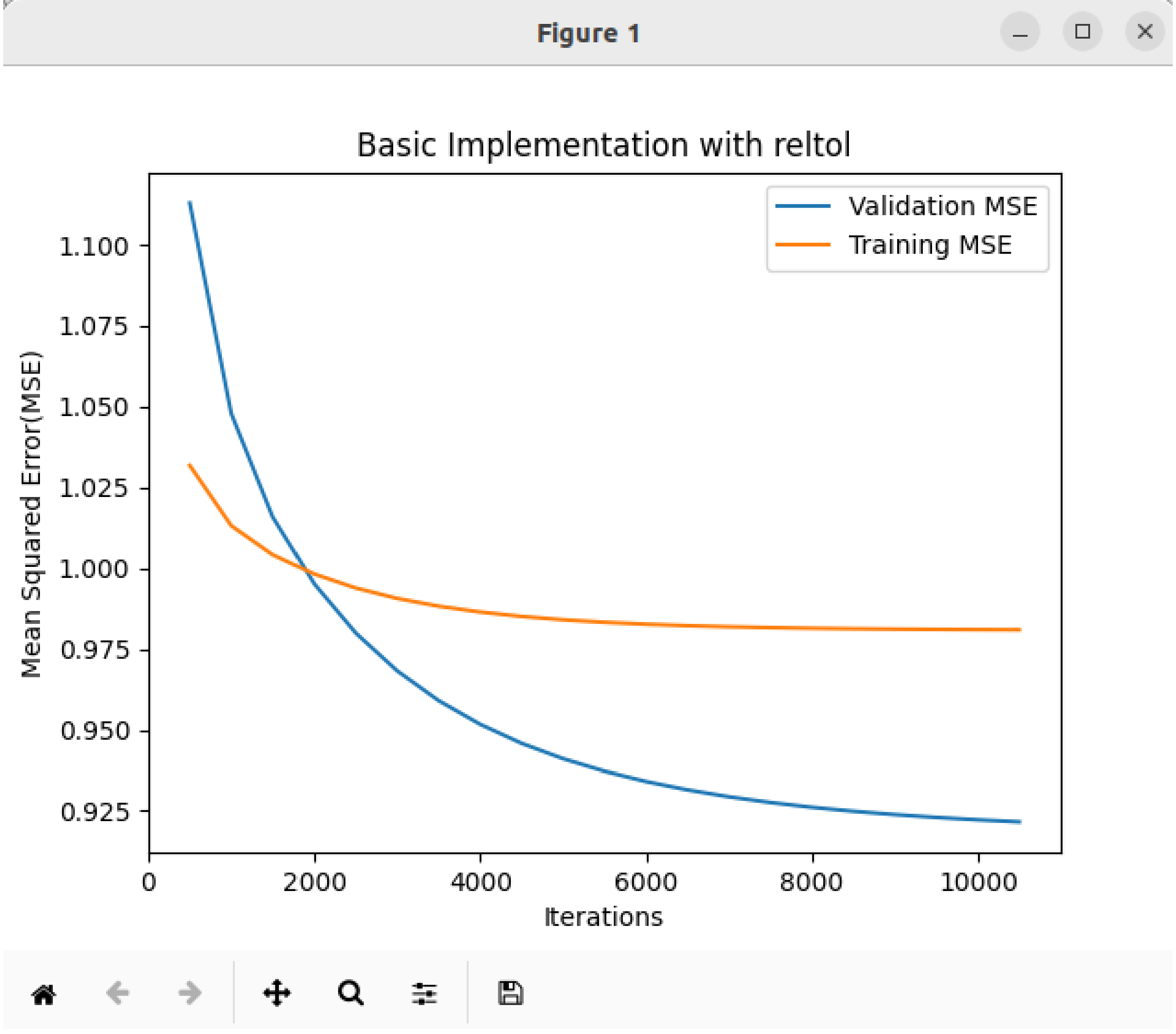


Observation

I have set the threshold limit to 1e-10
We can see that the error in validation set decreases less significantly after around 50 iterations
Also comparing it with the Basic implementation we can see that the MSE in Ridge Regression is lesser.

Feature Selection

Selectkbest



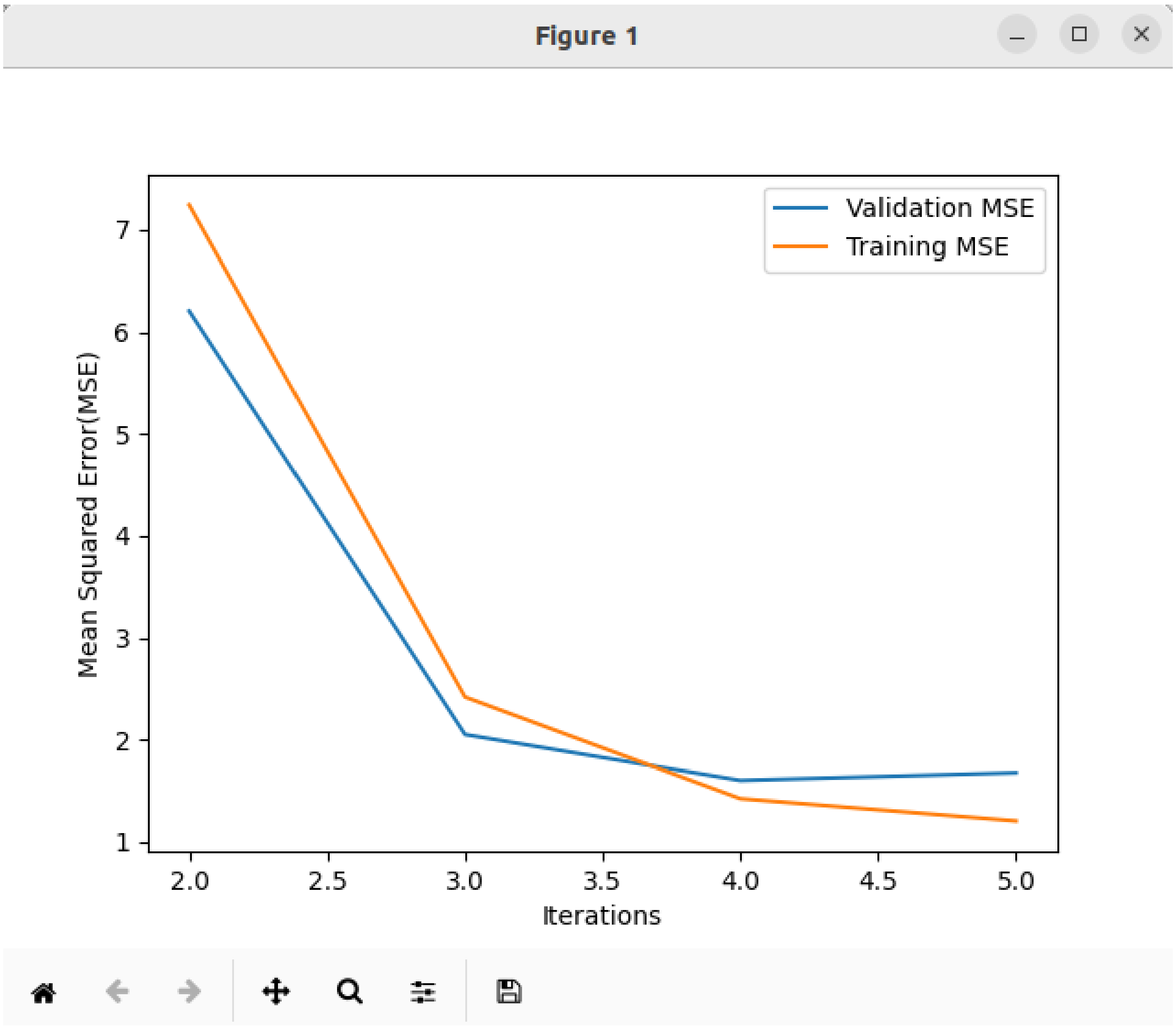
observation

We can see that since the model didn't do any overfitting for all the features, the validation MSE and the train MSE are significantly close to each other. But on the other hand their MSE is higher than The MSE found in the rigde or basic model. Therefore we need to select an optimal value for k to train it on k most significant features and get the best results

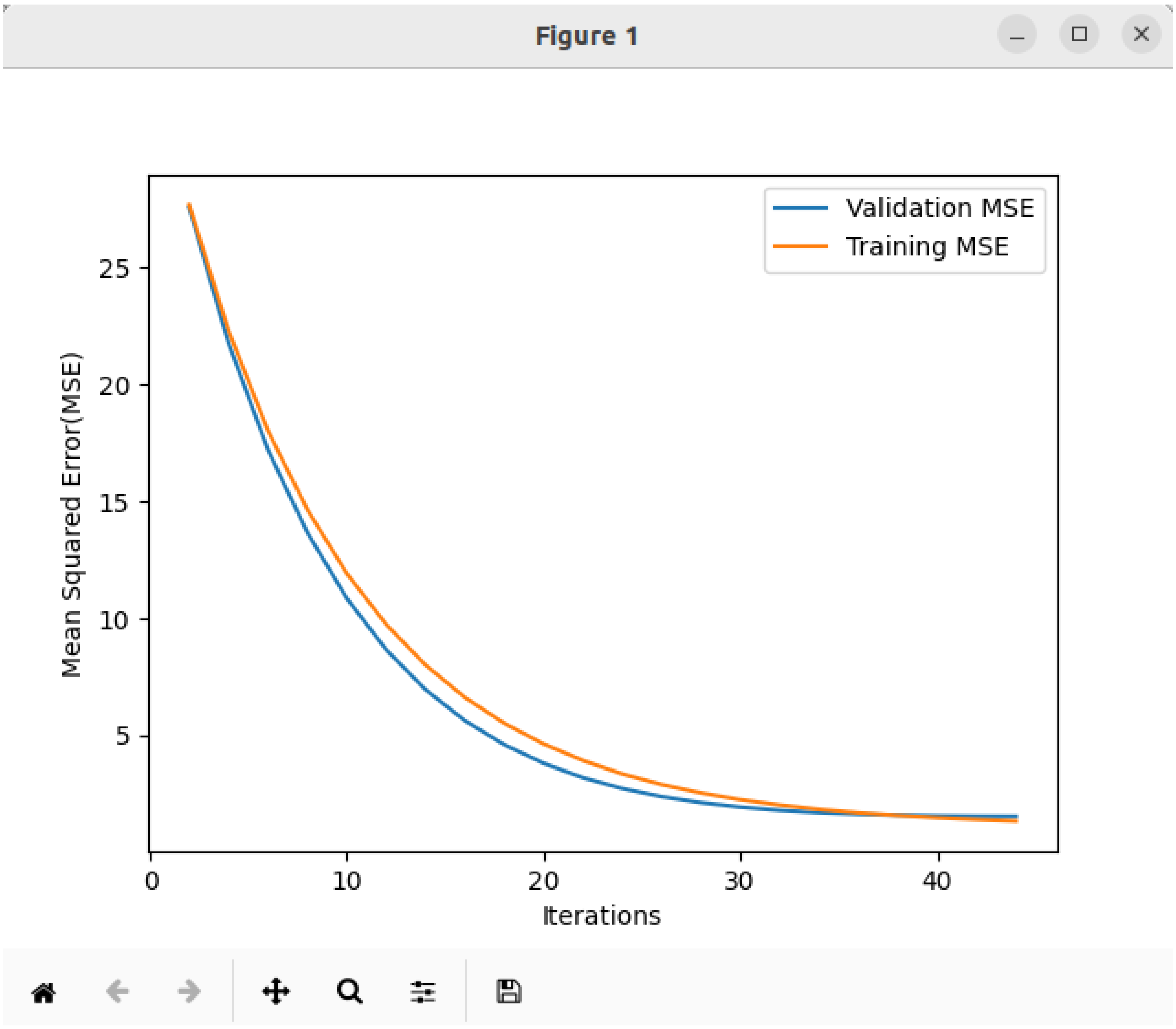
Another observation we made is that the validation mse falls faster than the training mse

SelectFromModel

for alpha = 0.1 I was getting the following graph ,



to smooth out the graph I made alpha = 0.01



We can see that the minima is reached faster using selectfrommodel than selectkbest. But both algorithms performs worse than the linear and ridge model because the number of training features are limited.

Normalization

maxit

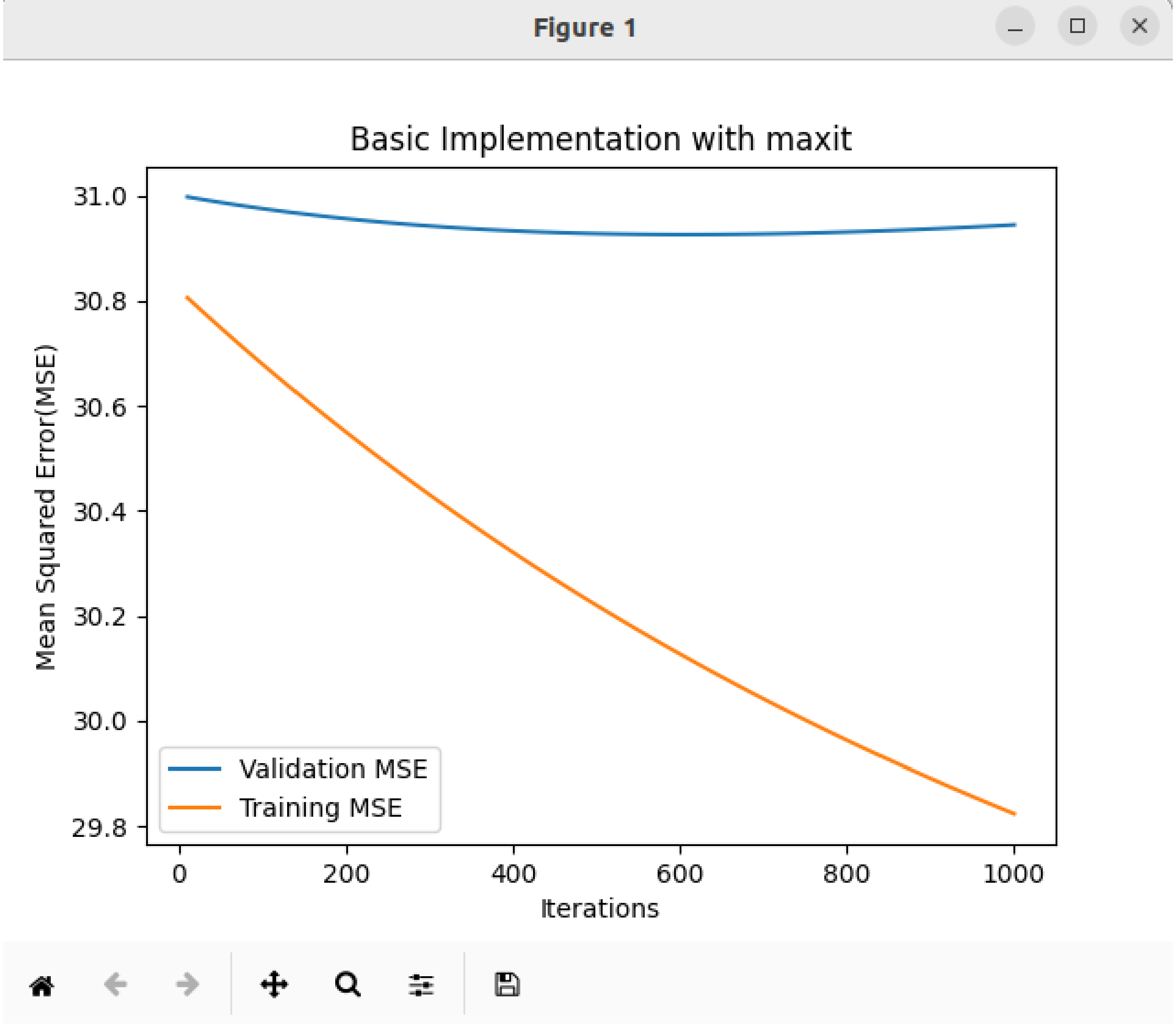
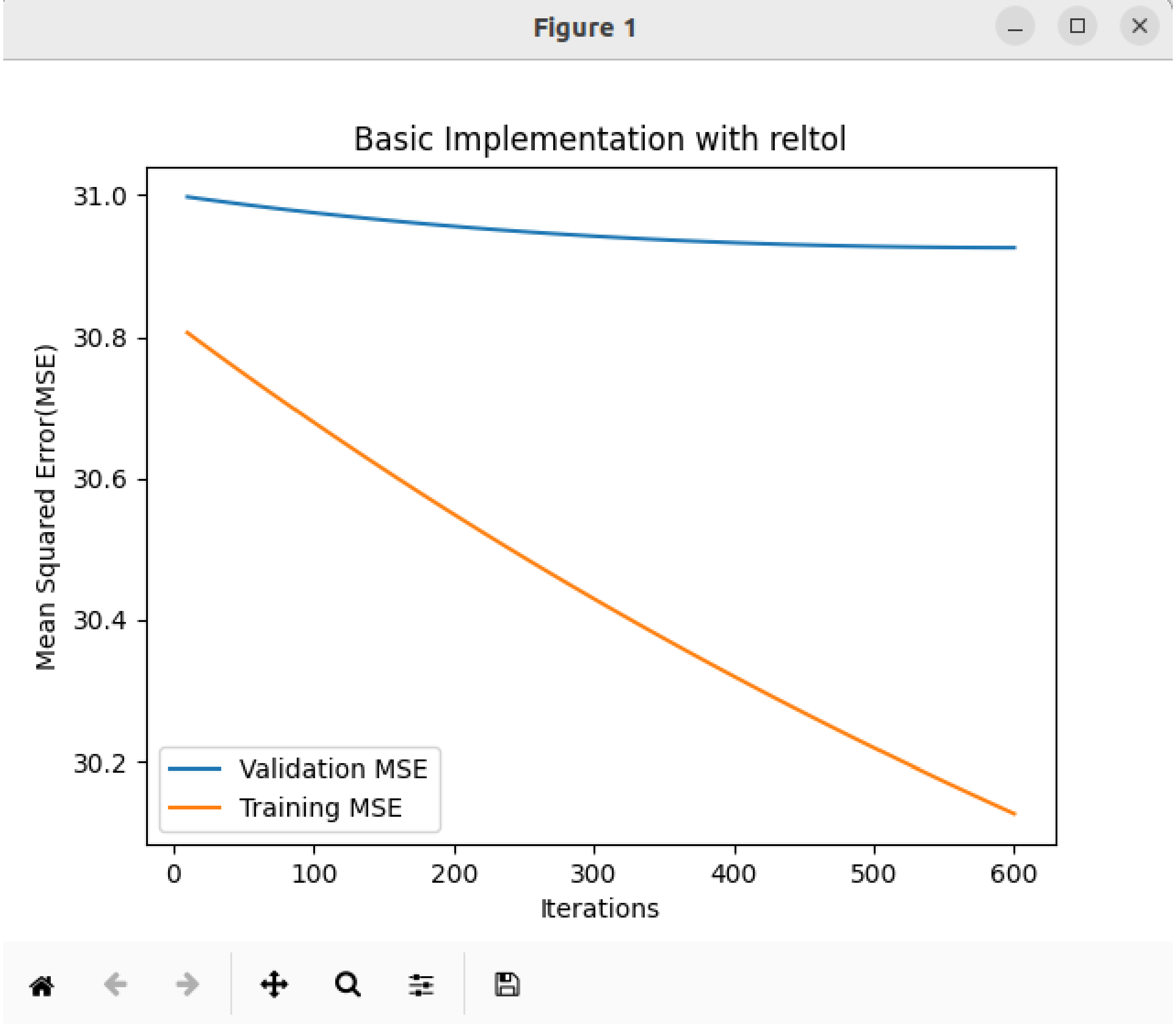
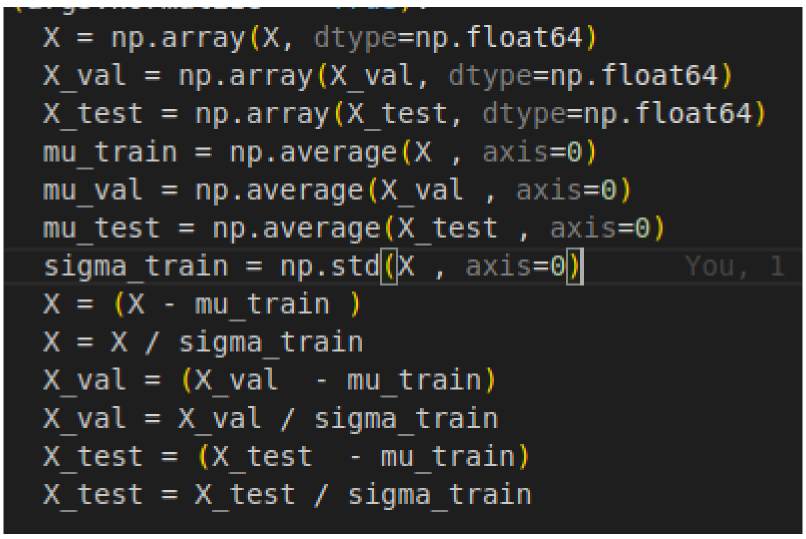


Figure 1
Basic Implementation with maxit
Mean Squared Error(MSE)
Iterations
31.0
30.8
30.6
30.4
30.2
30.0
29.8
0
200
400
600
800
1000
Validation MSE
Training MSE

Reltol



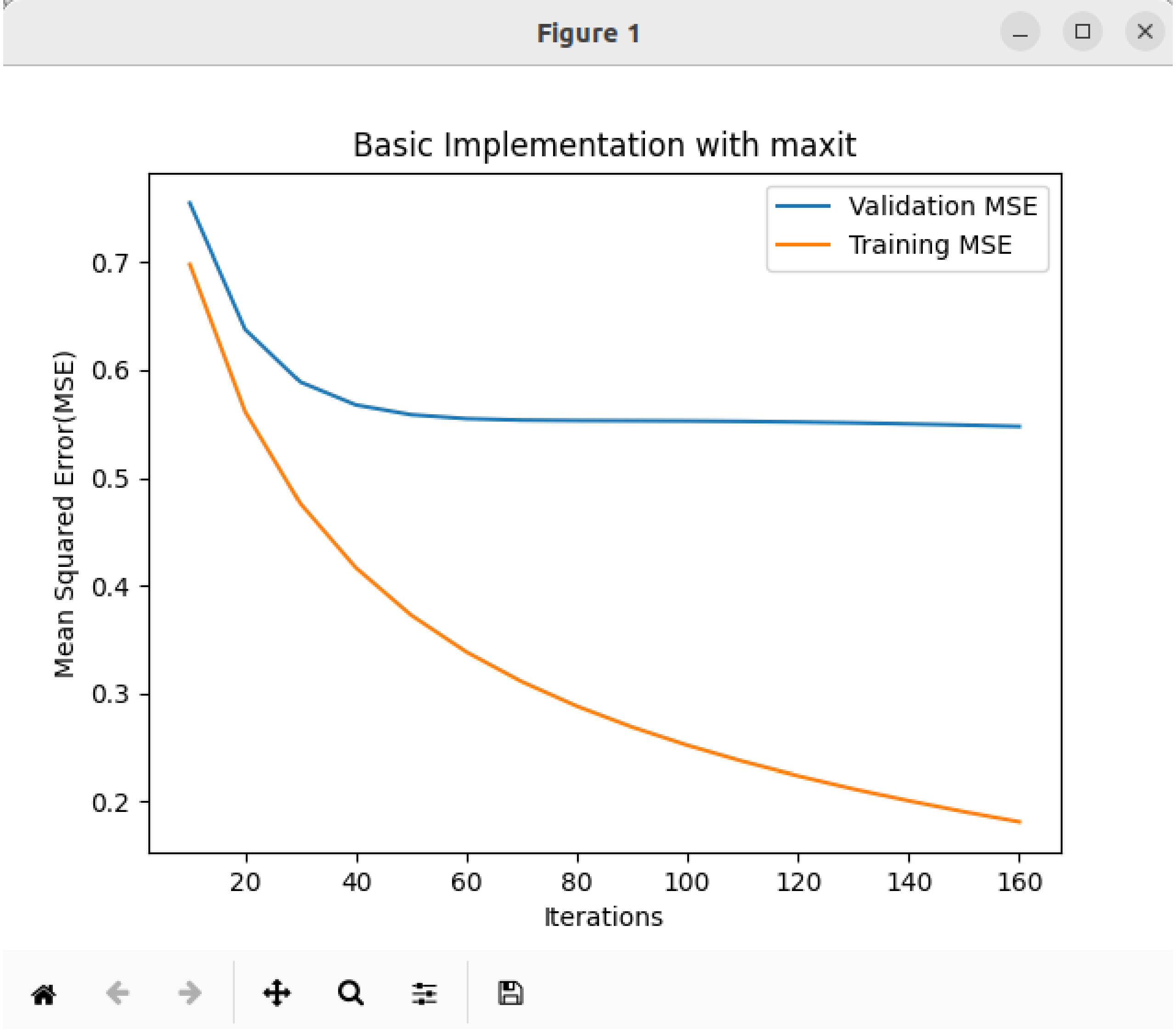
As we can see, normalizing the data made the results poorer
Below are the normalizing equations in my code



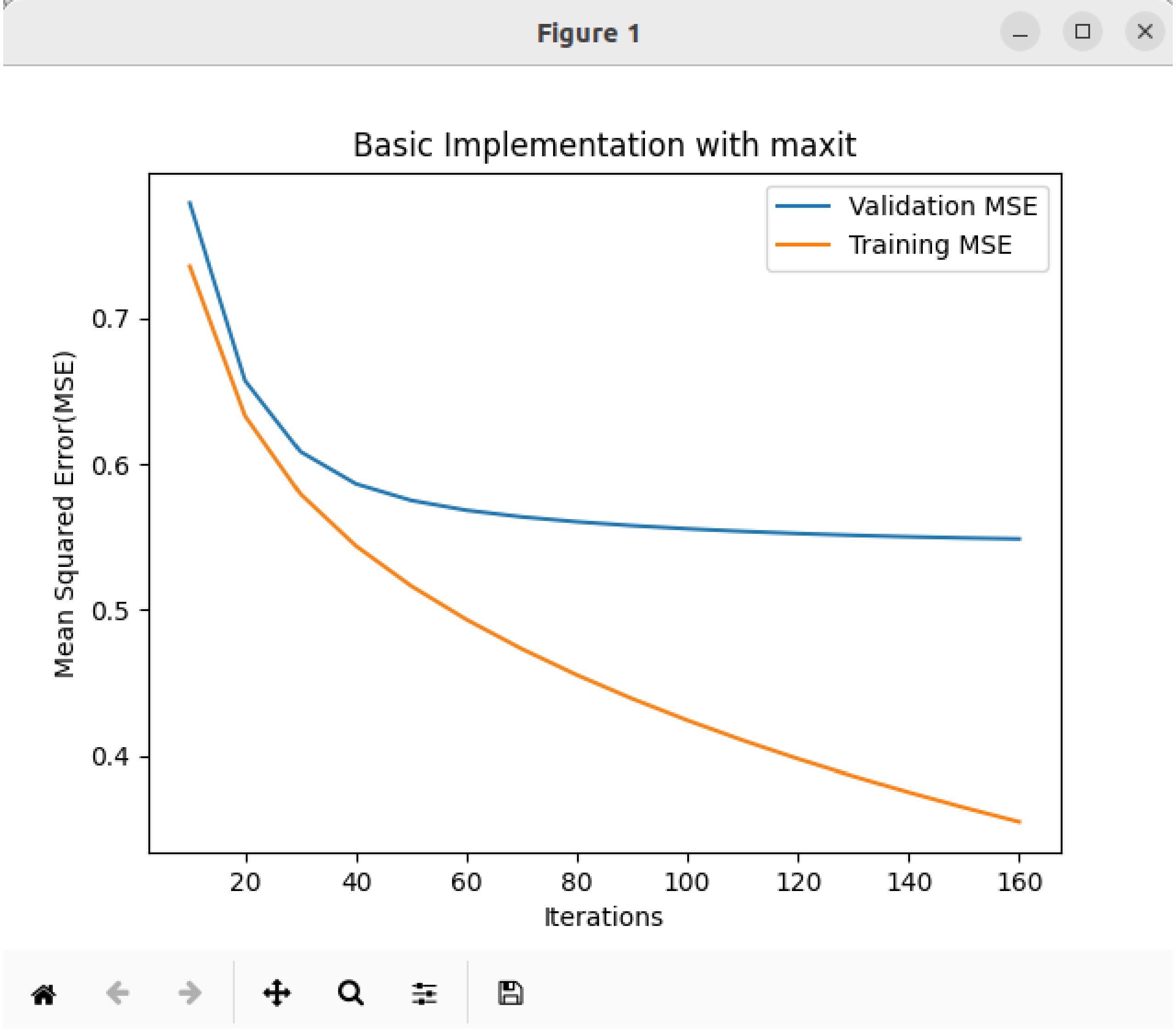
```
  X = np.array(X, dtype=np.float64)
  X_val = np.array(X_val, dtype=np.float64)
  X_test = np.array(X_test, dtype=np.float64)
  mu_train = np.average(X , axis=0)
  mu_val = np.average(X_val , axis=0)
  mu_test = np.average(X_test , axis=0)
  sigma_train = np.std(X , axis=0)        You, 1
  X = (X - mu_train )
  X = X / sigma_train
  X_val = (X_val  - mu_train)
  X_val = X_val / sigma_train
  X_test = (X_test  - mu_train)
  X_test = X_test / sigma_train
```

Splitting the data

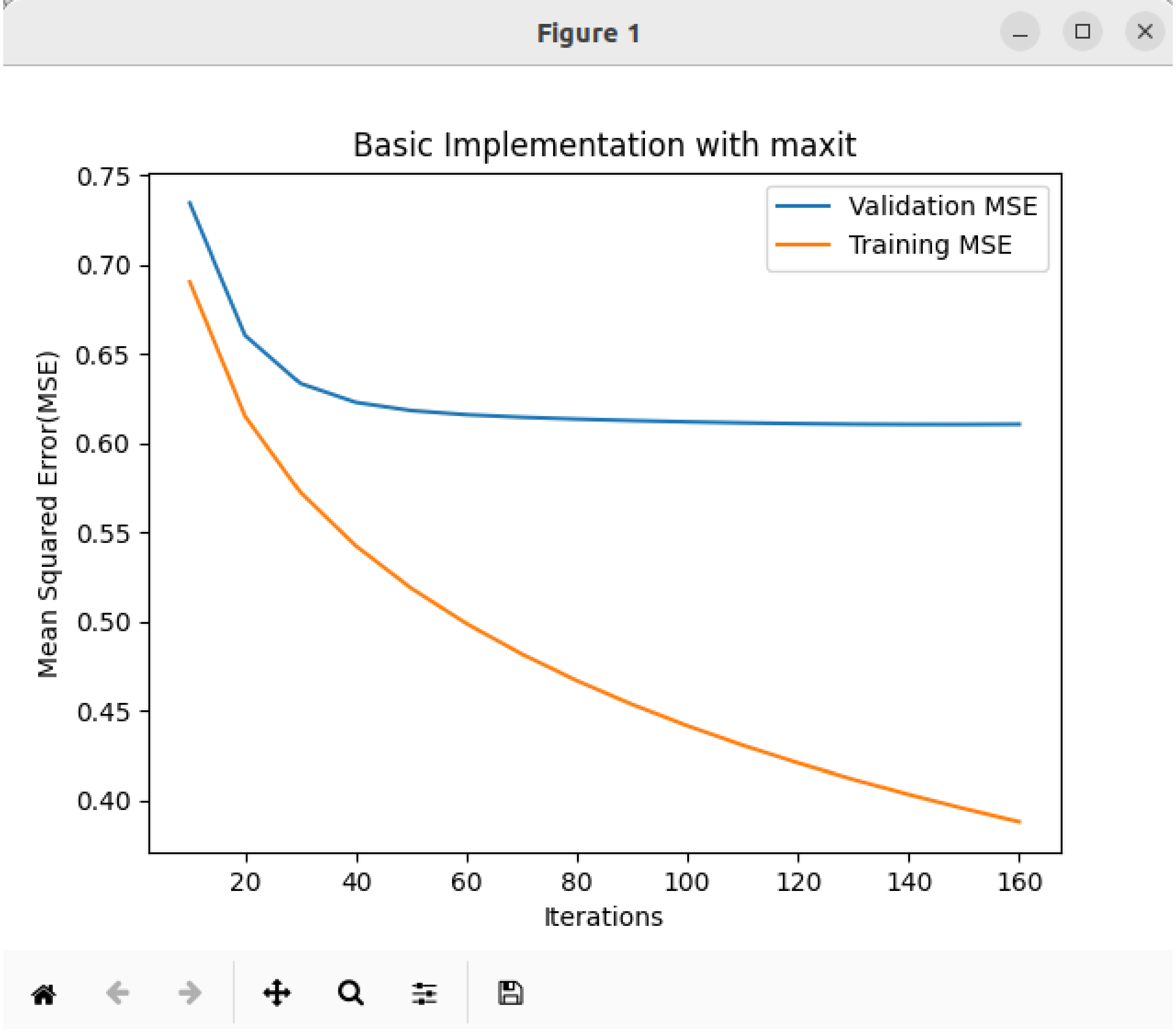
splitting by 1/4



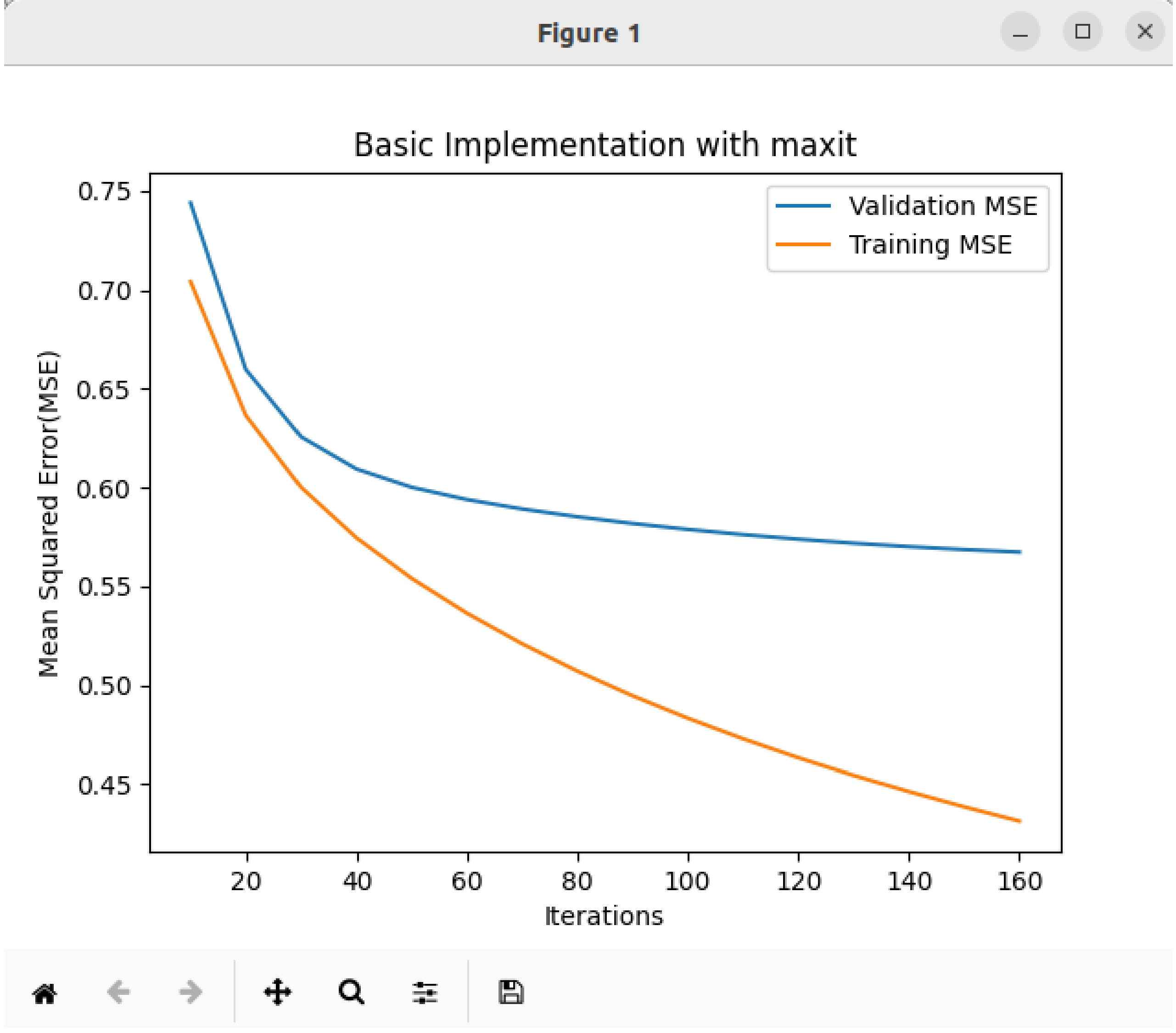
Splitting by 1/2



Splitting by 3/4



Splitting by 4/4



Observations

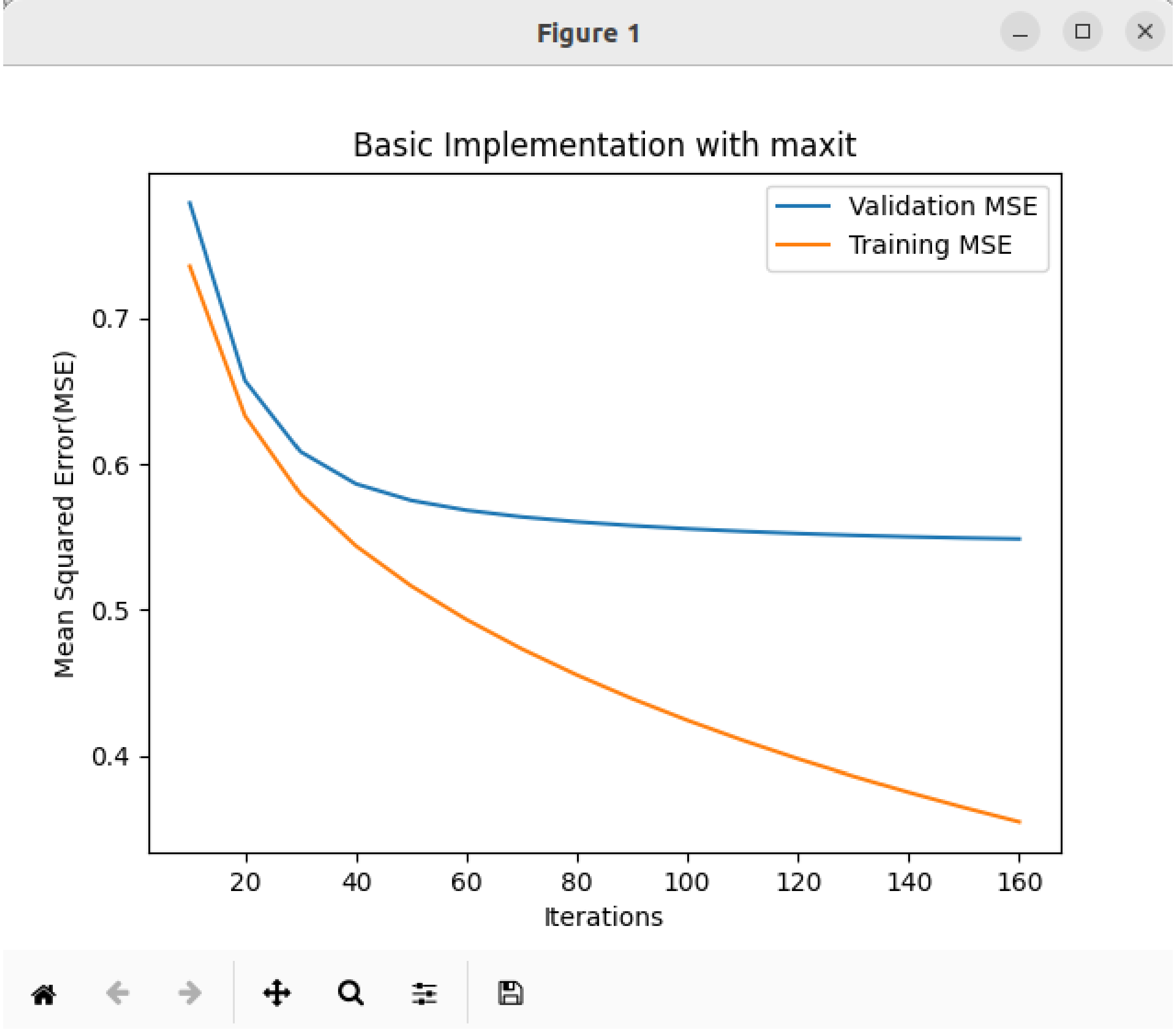
The validation error increases as we decrease the size of the training data. Since there is lesser data to train therefore the model performs poorer

Interestingly train data of size 1/4th the original performs better than train data of size 3/4 on validation MSE

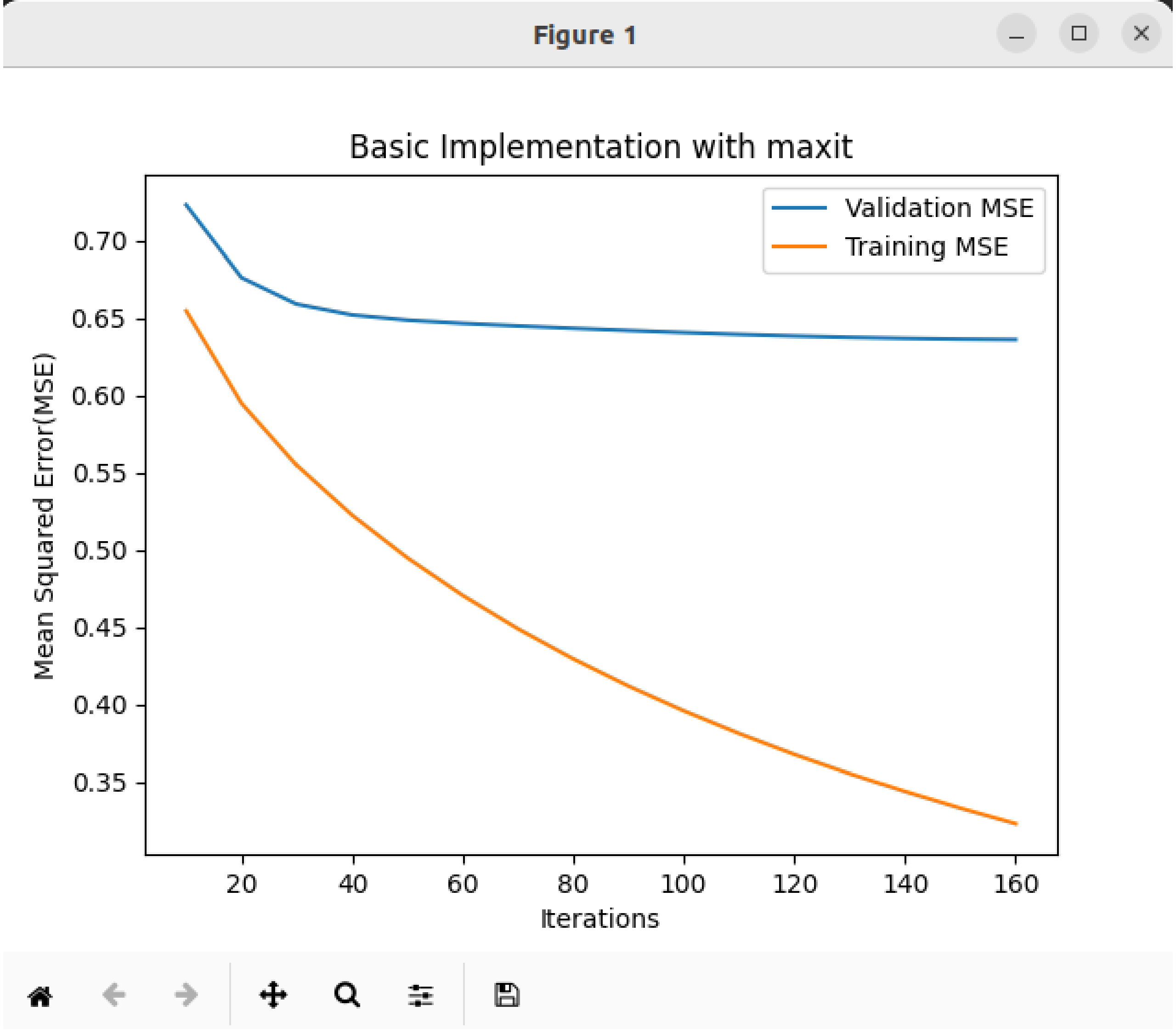
Dividing and training

Training and testing on basic implementation

Using the first half of the dataset



Using the other half of the dataset

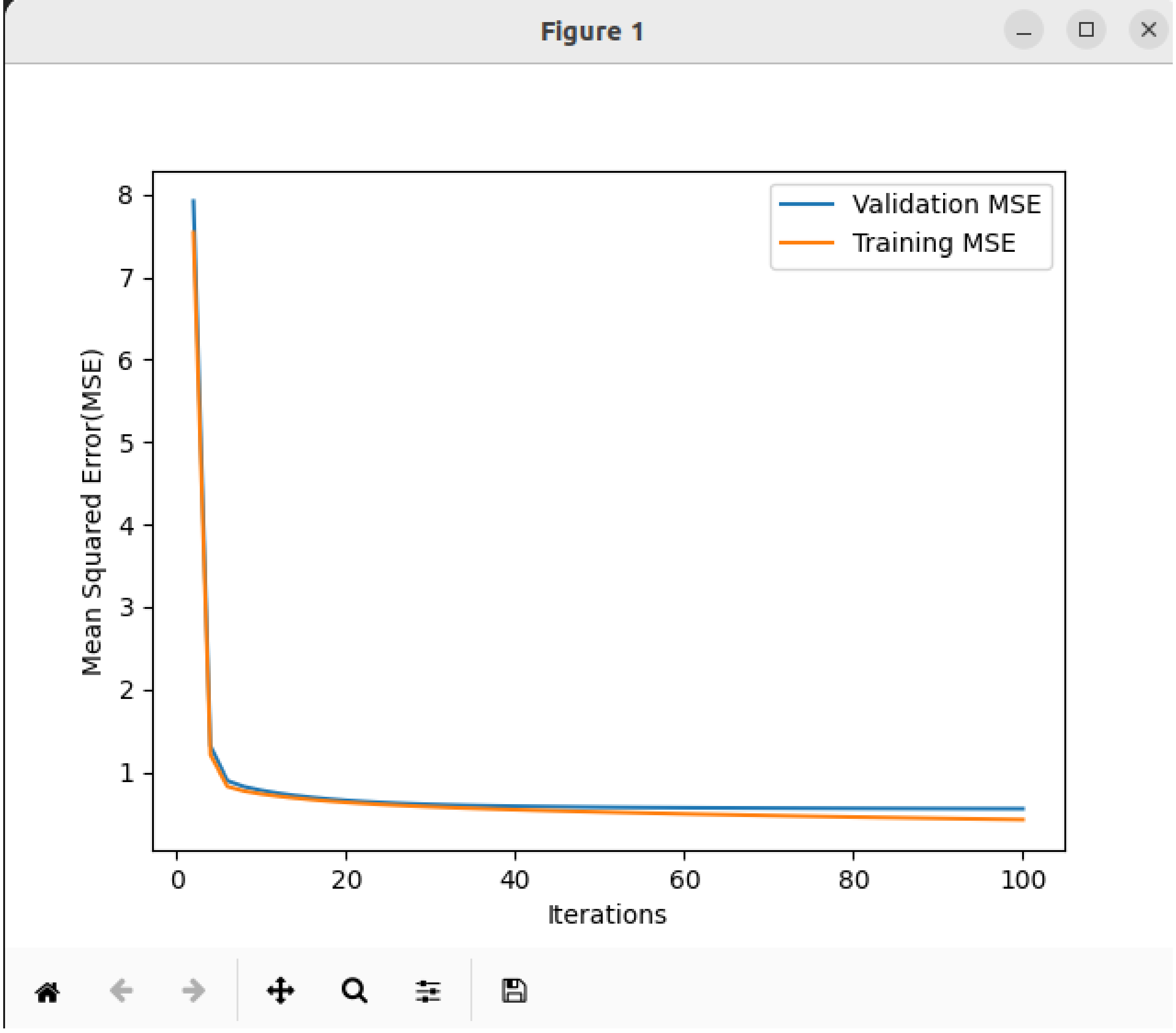


Average difference in MSE is 0.2513007186667529

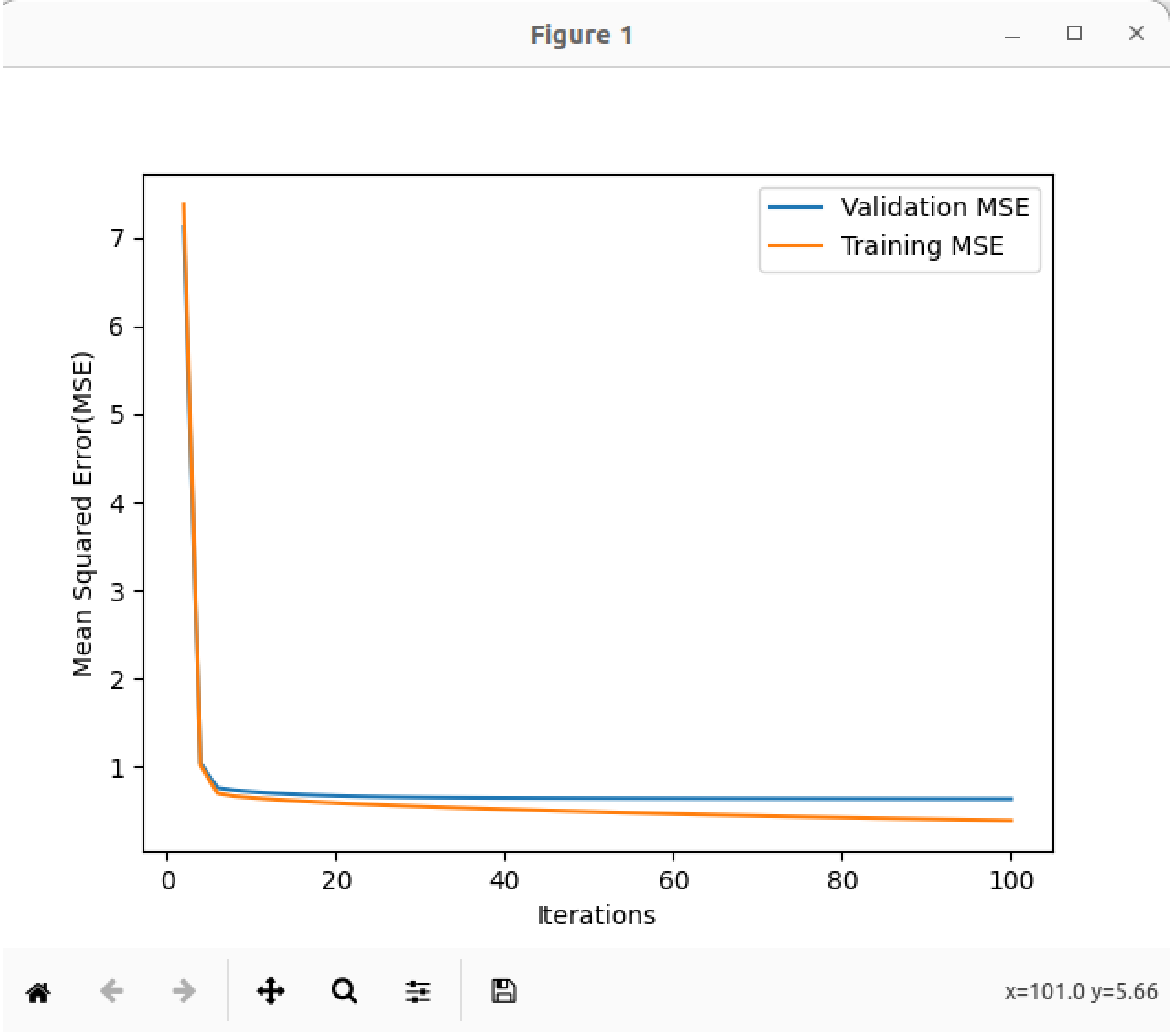
As we can notice that Both the datasets performed well and the reported values are also close to each other for both

Training and testing for ridge regression

Using the first half of the dataset



Using the second half of the dataset

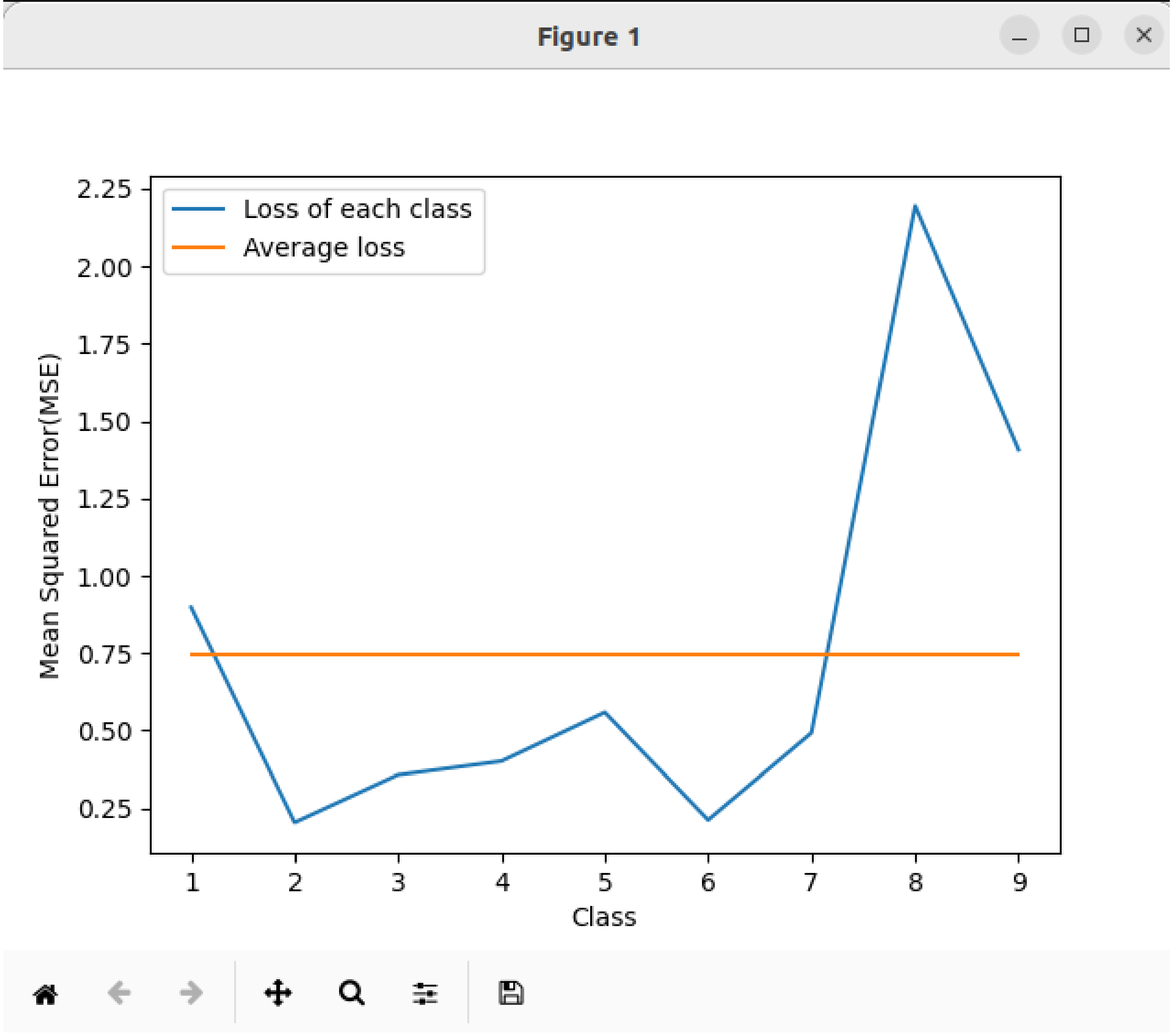


Average difference in MSE is 0.20751640770106067

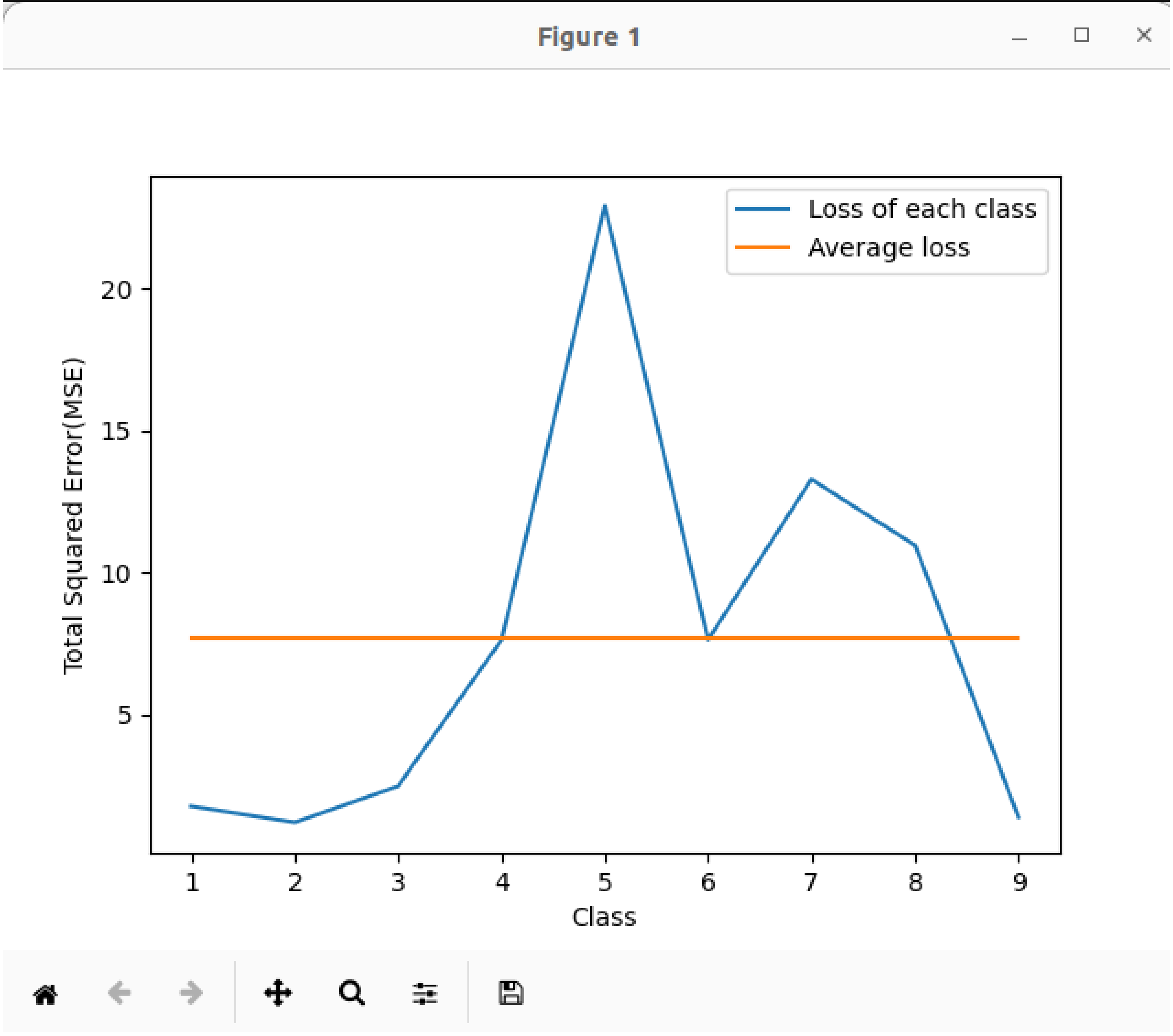
We can see that the ridge regressioin is more precise than basic implementation as the results of both the data sets (splitted) are closer

Graph of individual classes

average of MSE of each class with the mean of MSE of the data



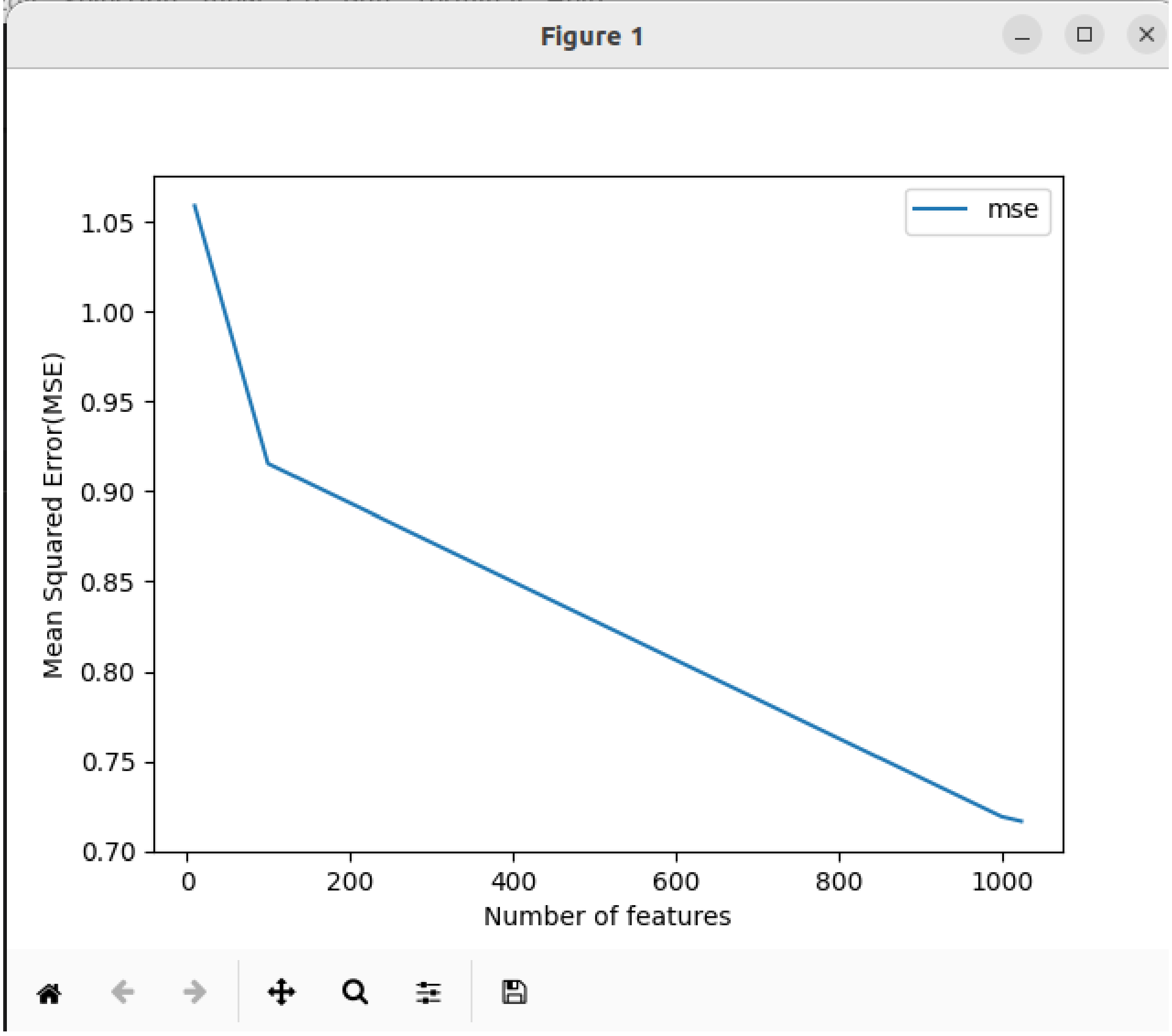
Sum of MSE of individual classes with the mean of the sum of the mse of entire data



We can see that the classes having higher sum of total mse have lower mean value and vice versa. This may be becaus the number of observations made with Y having those values with higher MSE sum is more Therefore we end up with having a higher accuracy for those classes

Feature selection and different number of features

Since selectkbest performed better, I will be using that



Observation - As the number of features are increased, the accuracy of the model on validation set increases

Generalization Analysis

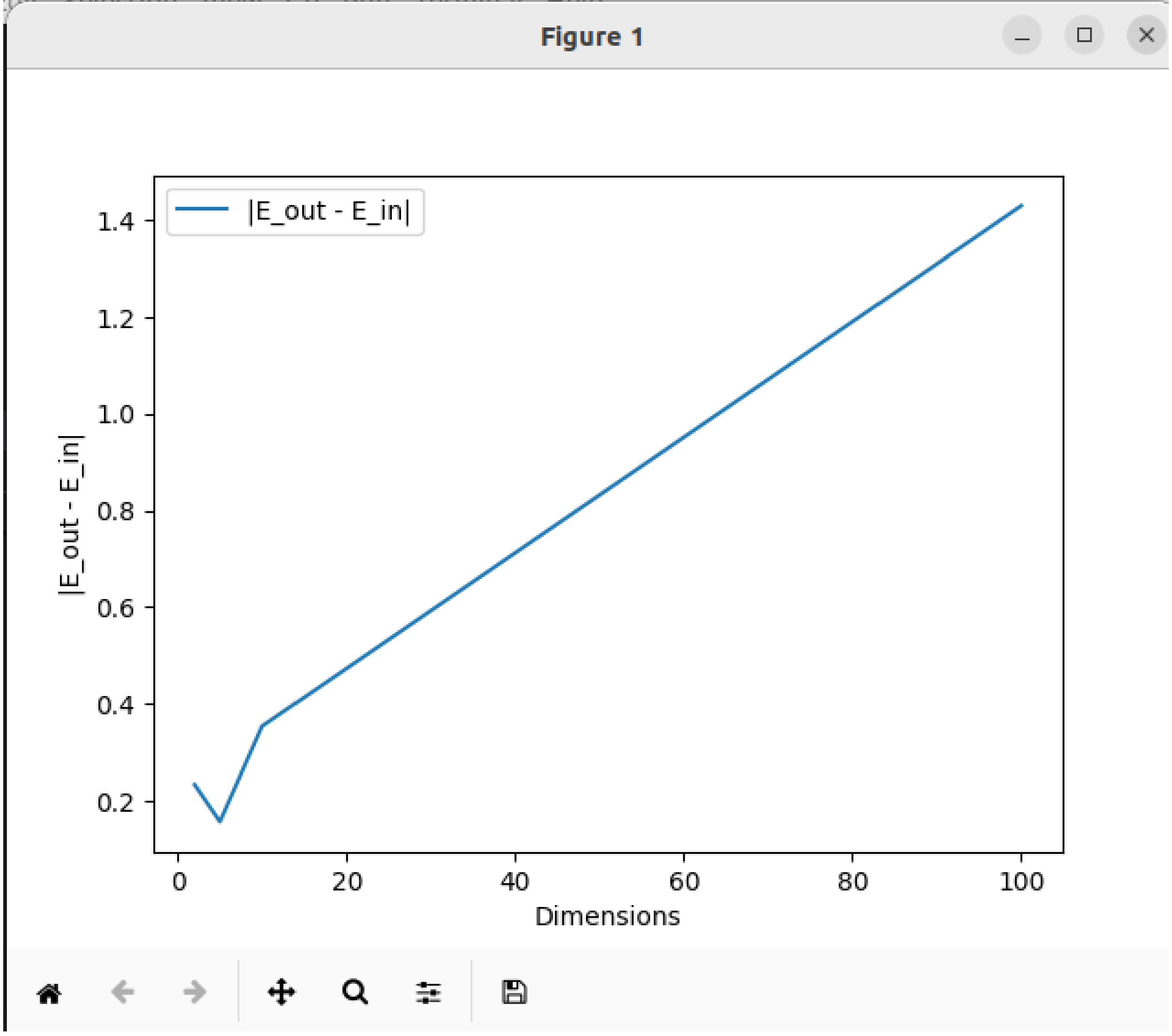


Figure 1
|E_out - E_in|
1.4
1.2
1.0
0.8
0.6
0.4
0.2
|E_out - E_in|
0
20
40
60
80
100
Dimensions

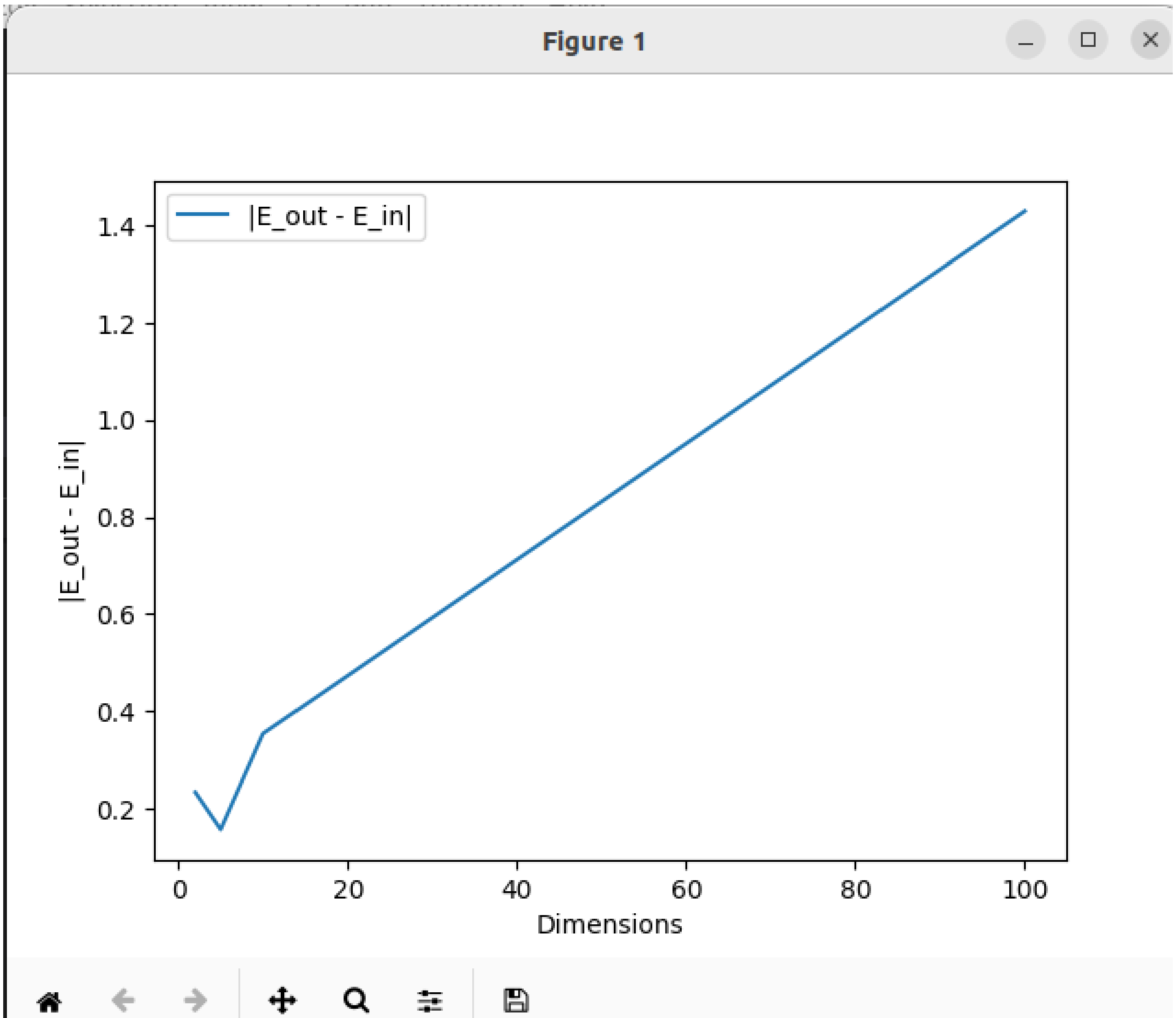


Figure 1
|E_out - E_in|
1.4
1.2
1.0
0.8
0.6
0.4
0.2
|E_out - E_in|
0
20
40
60
80
100
Dimensions

As we can see that the |E_out - E_in| value increases propotional to the square root of the dimenstions(approximately)

Bonus

Loss function

$$J(\theta) = -y\log(h_\theta(x)) - (1-y)log(1-h_\theta(x))$$

Where

$$h_{\theta_r}(x) = \frac{e^{\theta_r^T X}}{1+\sum_{p=1}^{9} e^{\theta_p^T X}}$$

It will be same as before only the sum will be on p = 1 till 9

The update equation will be

$$\frac{\partial J}{\partial \theta_1} = (\sum x h_{\theta_r}(x) - \sum_{x \in 1} x)$$

And there will be 9 such weights