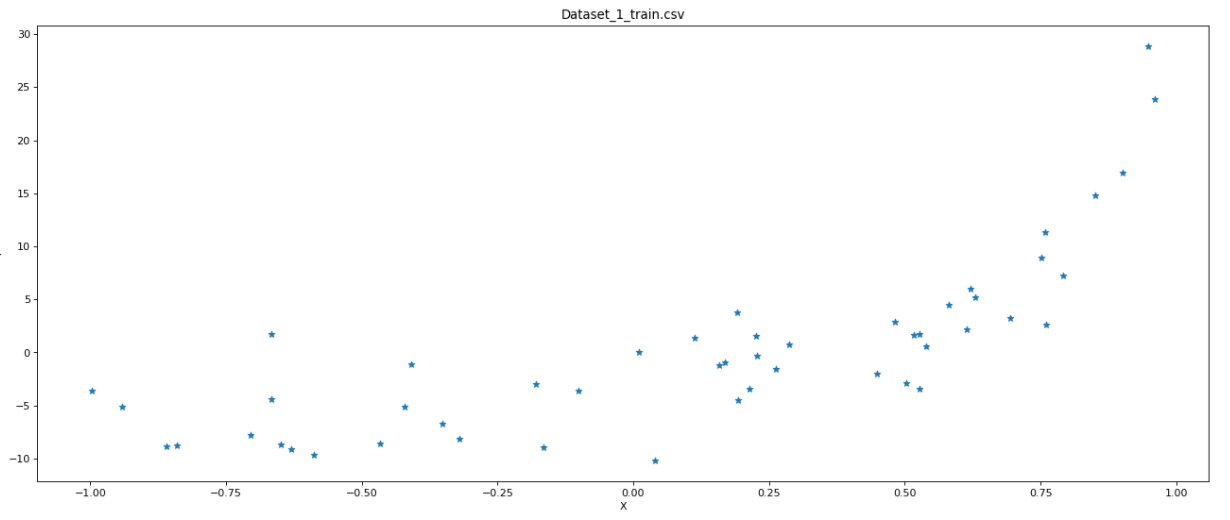
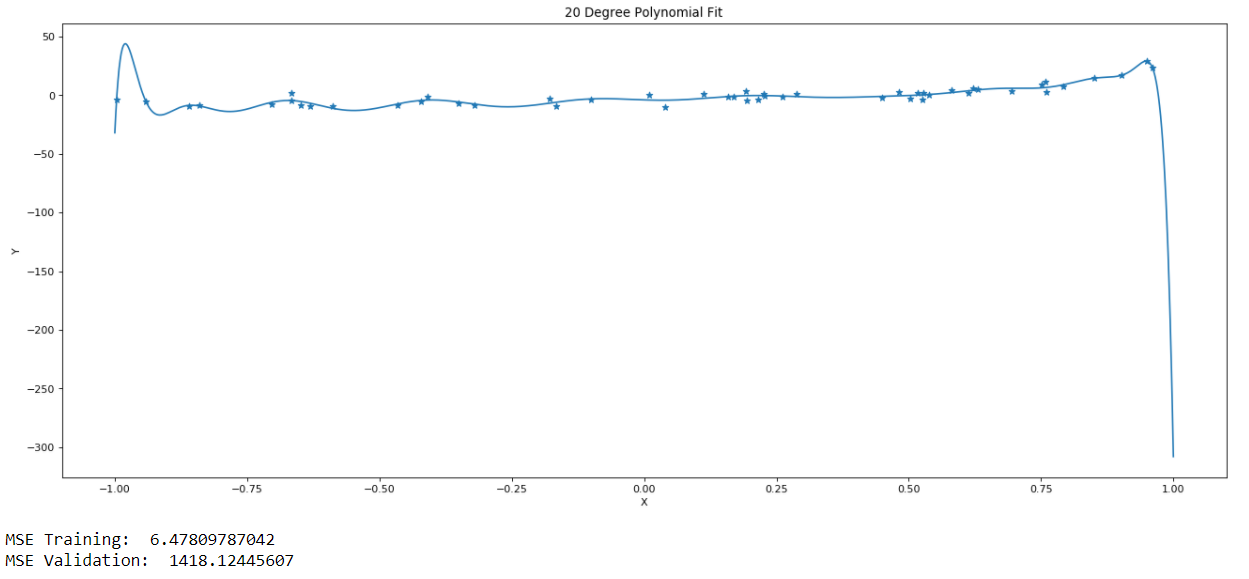
# Question 1

## 1.1

**Visualizing the Dataset**



**Visualizing the 20-Degree Polynomial Fit**

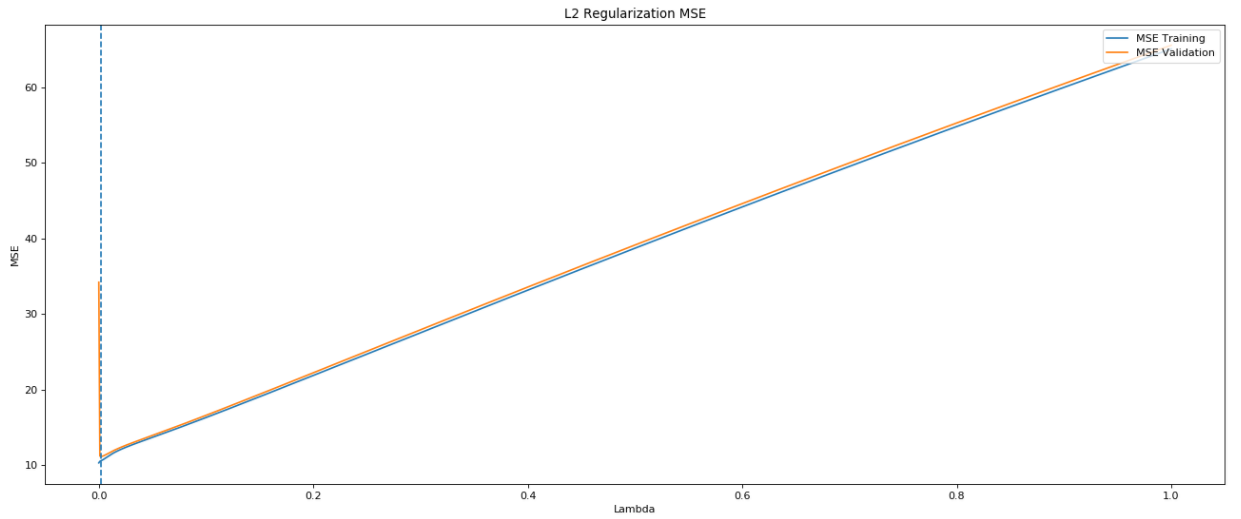
****

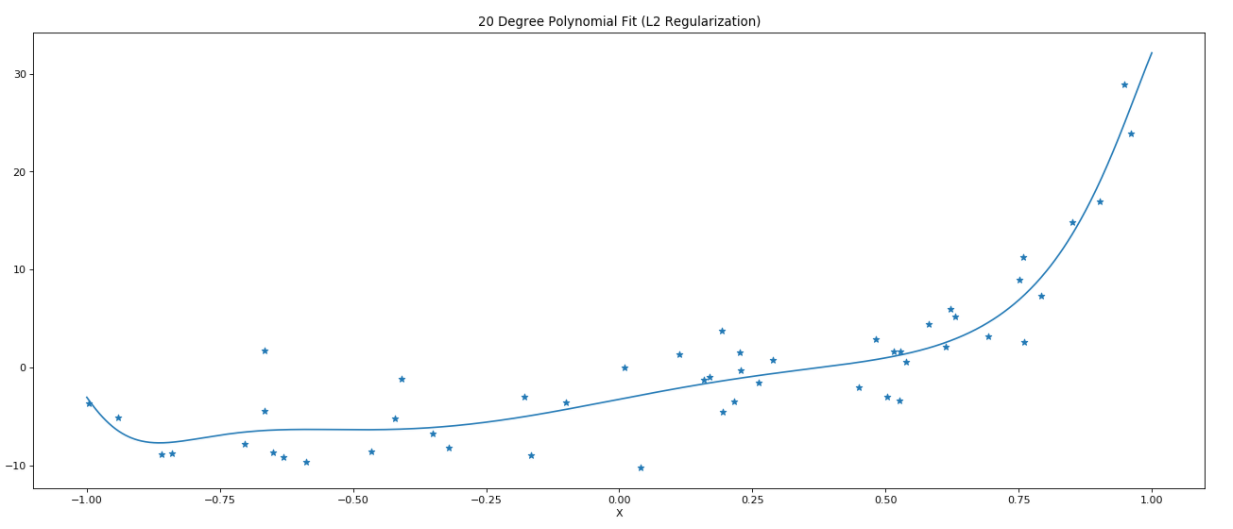
**MSE Training = 6.47**

**MSE Validation = 1418**

The quality of the fit looks fine if our target vector is between -0.8 and 0.8. However, there is clear overfitting. For any target Y greater than 0.8, the predicted value will be off significantly. We need to add regularization to the fit.

## 1.2





The best L2 regularization occurs at **Lambda = 0.002003**. At this point, the minimum MSE for training is **10.57**, and the minimum MSE for validation is **11.16**, and the minimum MSE for testing is **12.84.**

## 1.3

It looks like the fit with regularization is a 6th degree polynomial. I am inferring this from the number of highs and lows of the graph. The fit has 2 local maximum, and 3 global minimum. This would mean that in theory, this function would cross the X-axis 5 times, and therefore be a 6th degree polynomial.

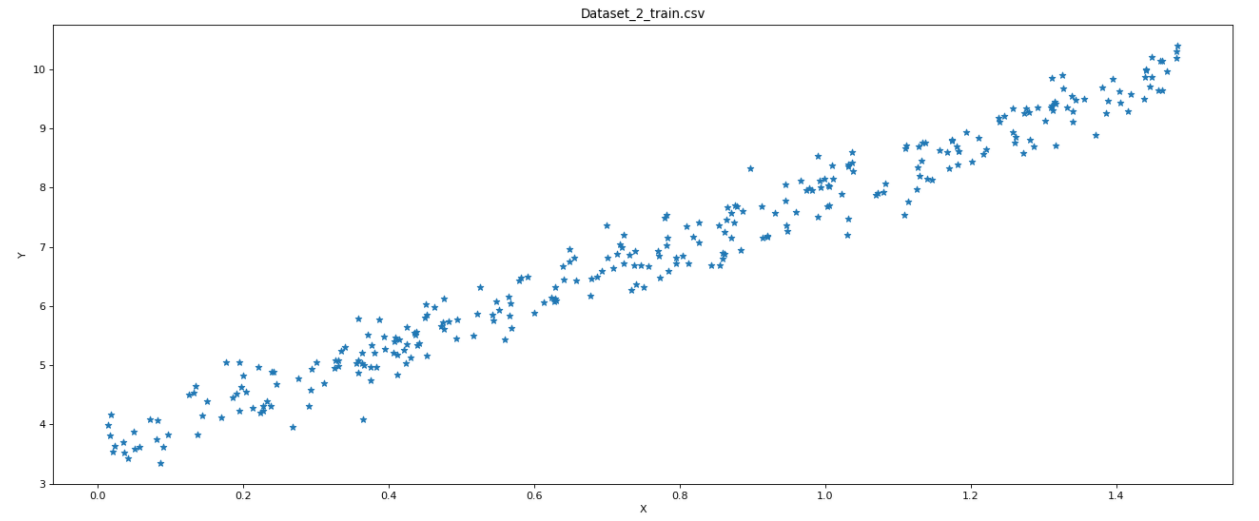
Shown below is an example of a 6th degree polynomial.



# Question 2

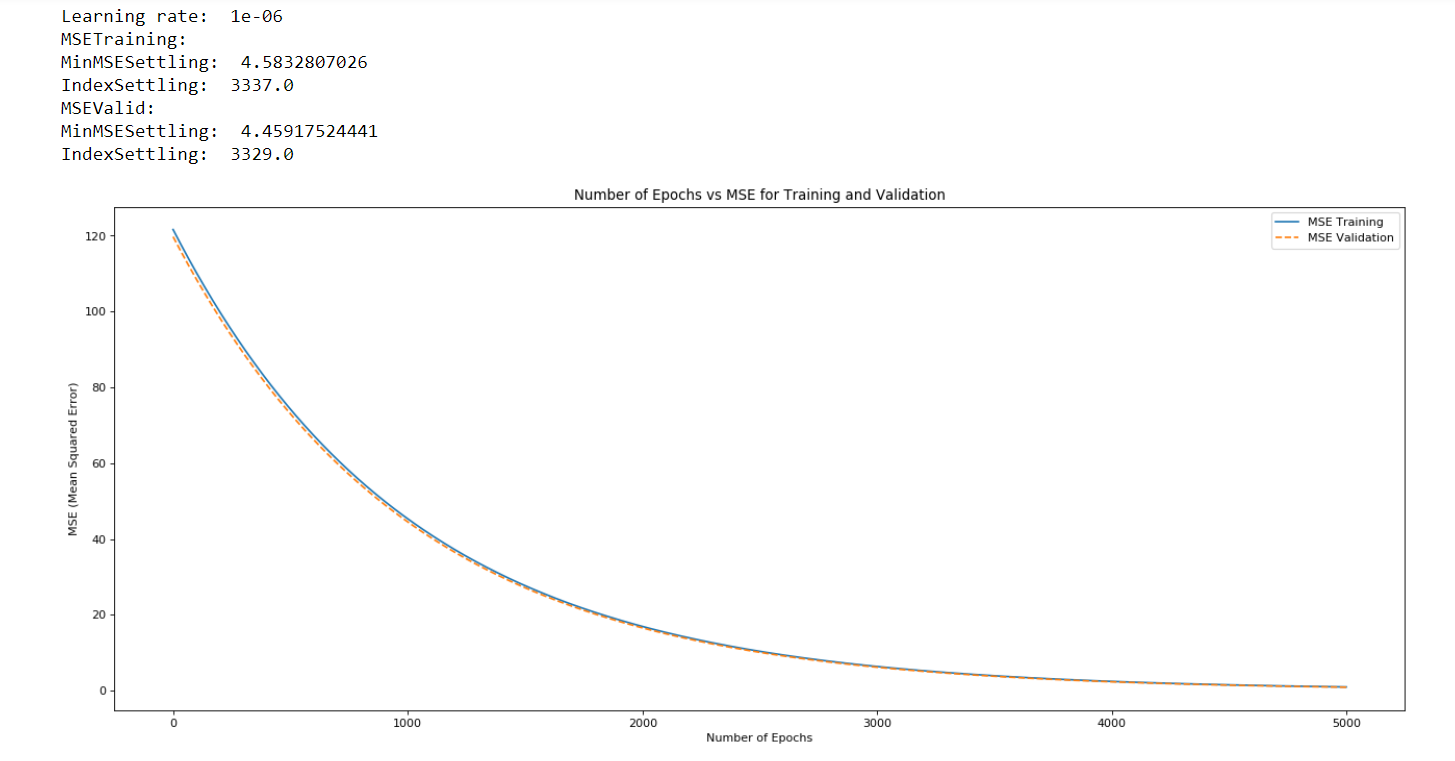
## 2.1

**Visualizing the Data**

****

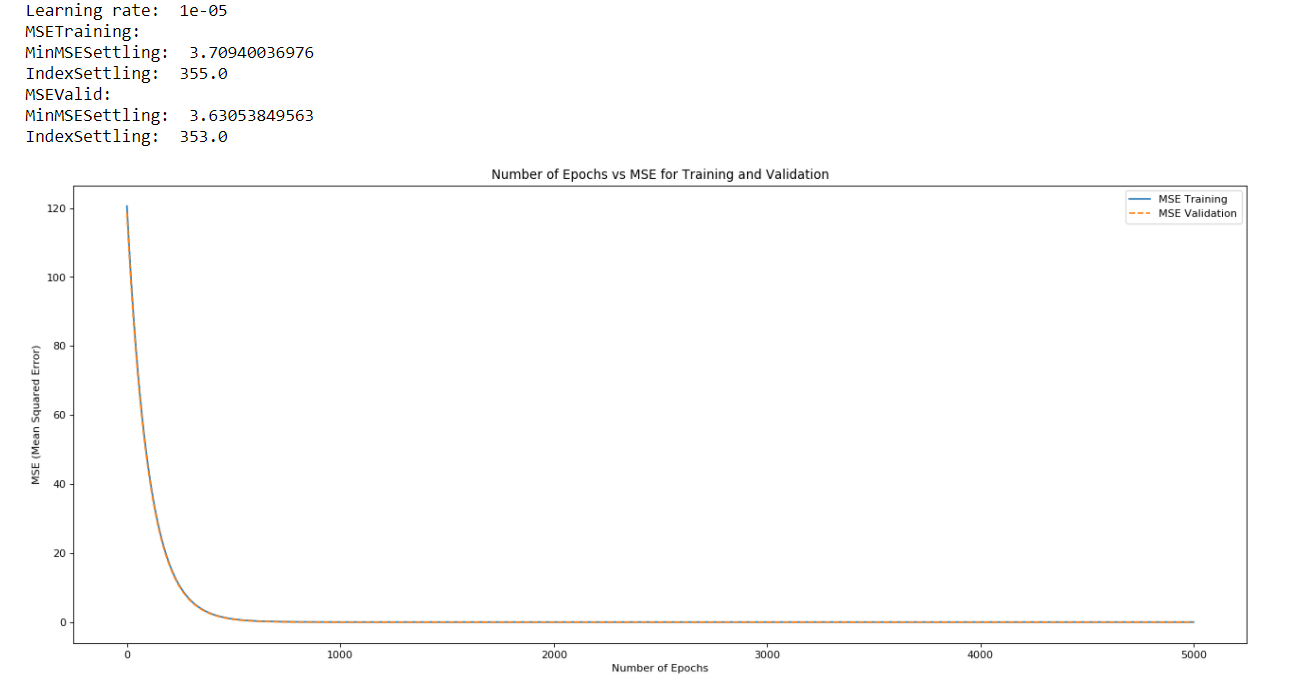
For this question, I was running 5k epochs. I declare that my MSE (therefore weights as well) have converged when the MSE has completed 5 half-lives. This corresponds to 97% of the final value.

**Learning rate = 1e-06**



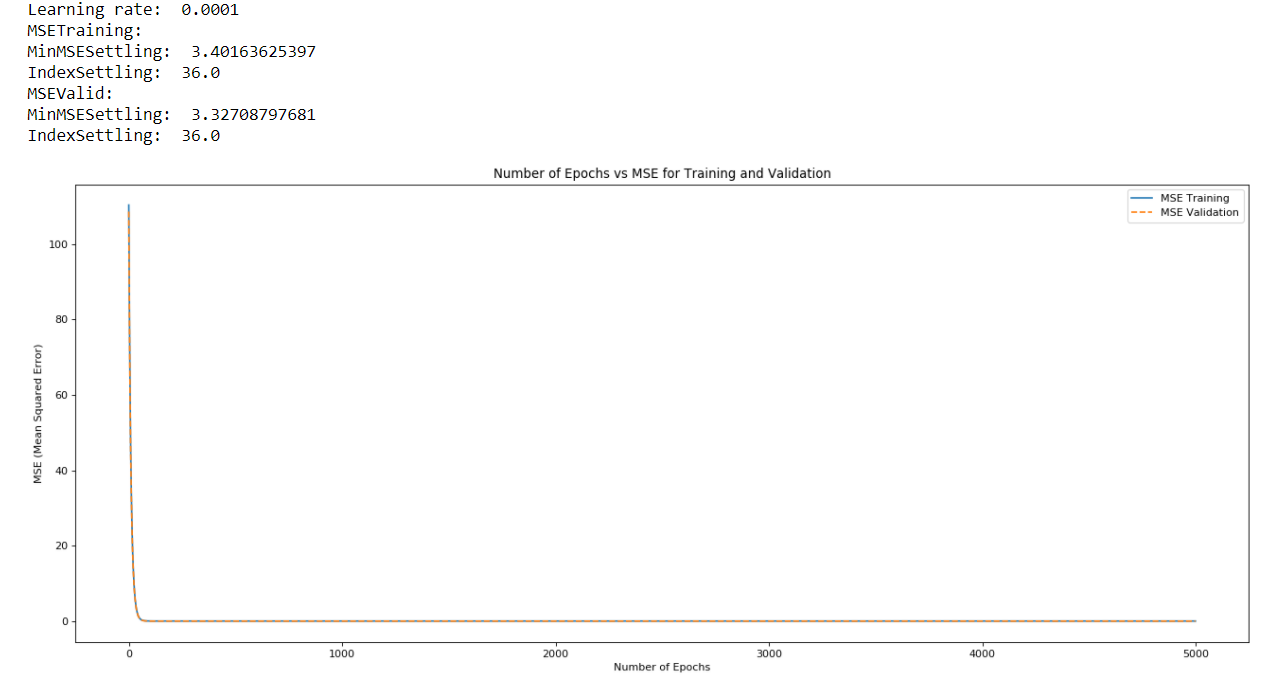
MSE converges at epoch number **3337**. The settling training MSE is **4.58**, and the settling validation MSE is **4.46**.

**Learning rate = 1e-05**



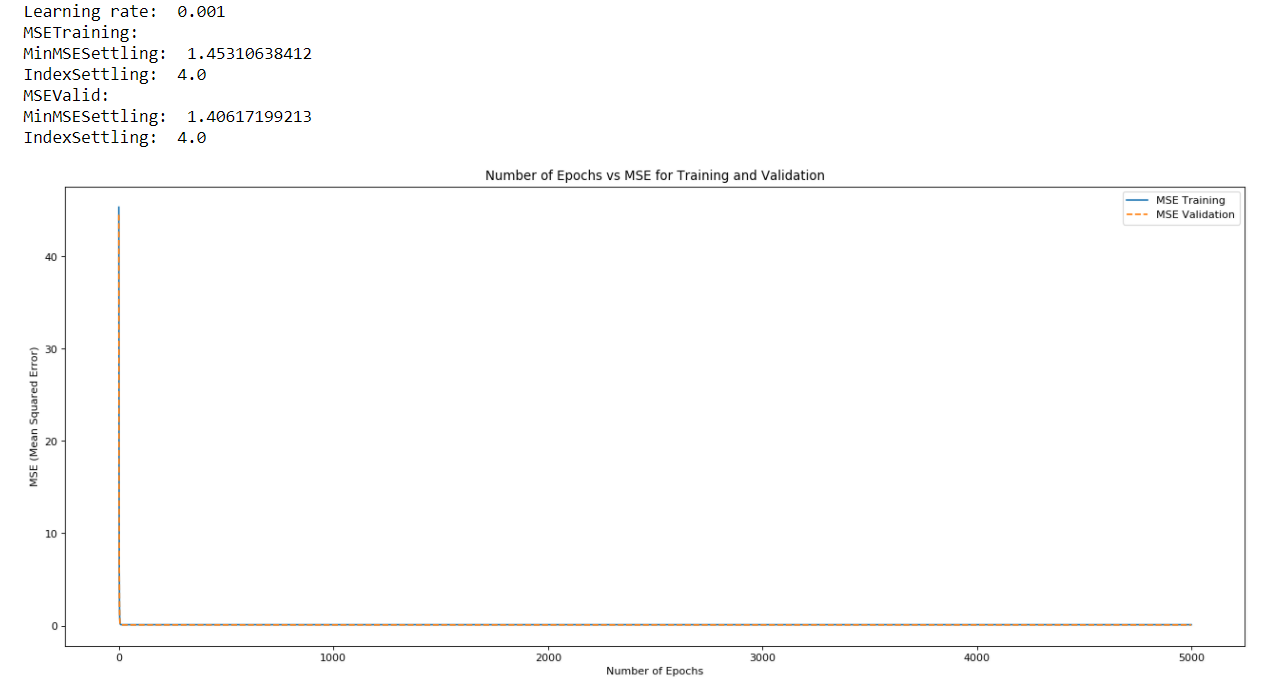
MSE converges at epoch number **335**. The settling training MSE is **3.71**, and the settling validation MSE is **3.63**.

**Learning rate = 1e-04**



MSE converges at epoch number **36**. The settling training MSE is **3.4**, and the settling validation MSE is **3.33**.

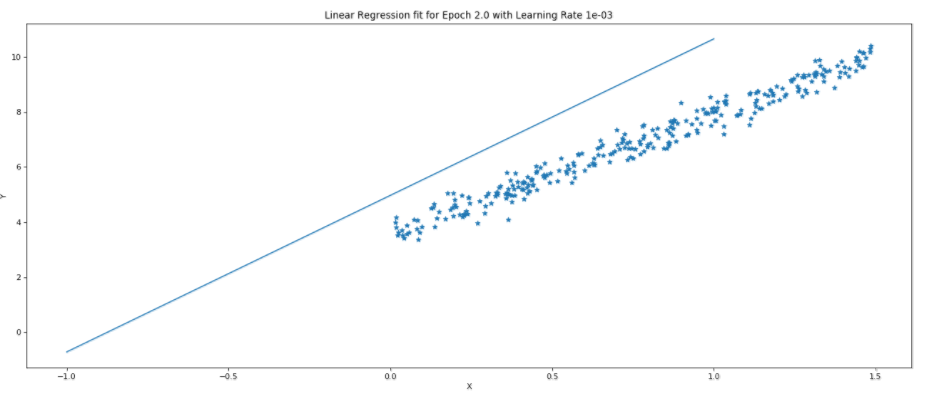
**Learning rate = 1e-03**



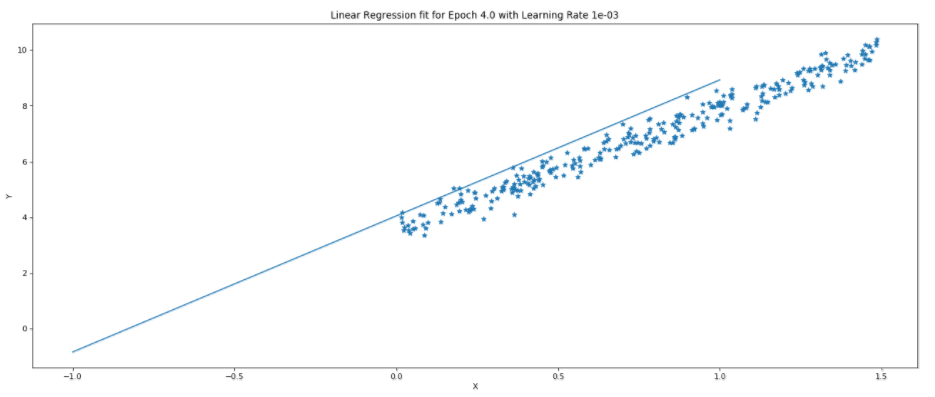
MSE converges at epoch number **4**. The settling training MSE is **1.45**, and the settling validation MSE is **1.406**.

It seems as if the best learning rate is 1e-03. Any higher than this and the function converges in less than 1 epoch, which is dangerous as it could diverge. Here are 5 epochs for how the fit evolves.

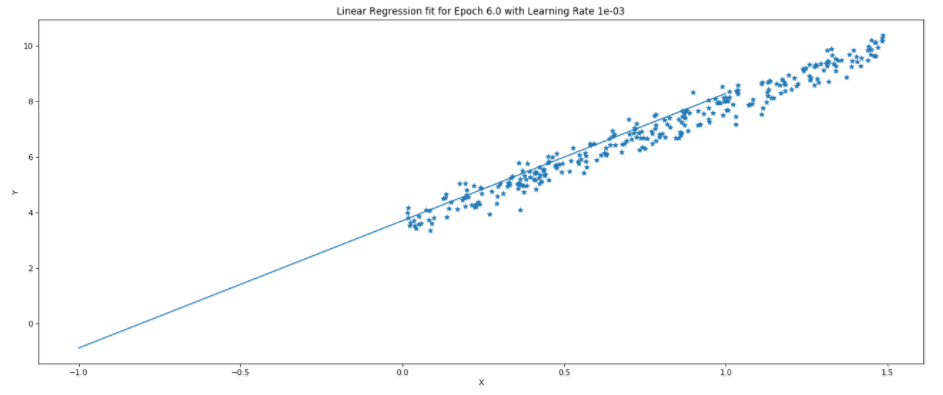
**Epoch 2**



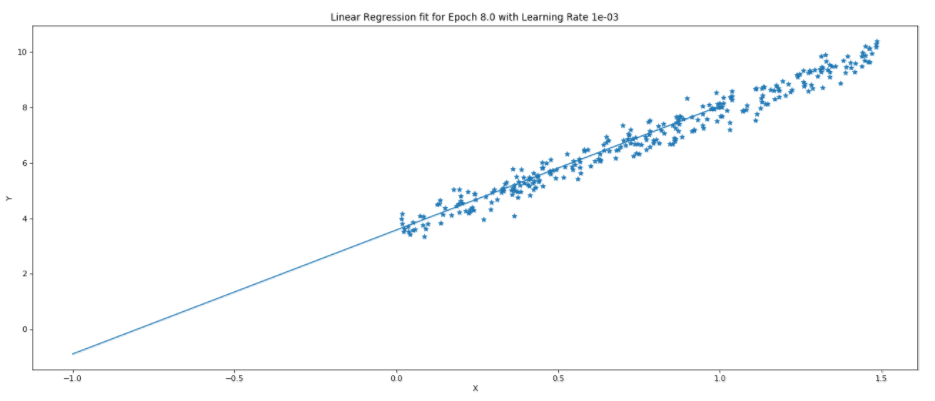
**Epoch 4**



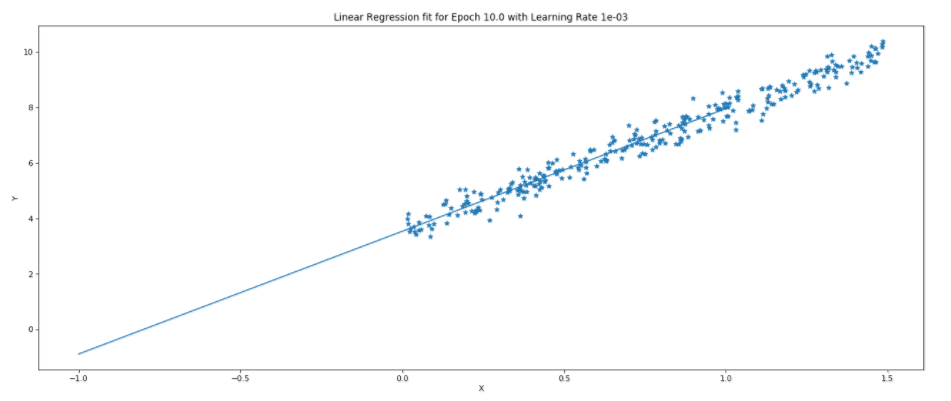
**Epoch 6**

****

**Epoch 8**

****

**Epoch 10**

****

As is clear from the visualizations above, the fit evolves quite rapidly for Learning rate = 1e-03. After 6 epochs we are very close to the final value, and after 8 we have practically found the best fit possible.

# Question 3

## 3.1

Files are saved as follows upon being shuffled:

Dataset\_CandC-train0.csv

Dataset\_CandC-test0.csv

The number varies from 0 to 4 for a total of 5 datasets.

The NaN values are filled within a uniform distribution of 1 standard deviation away from the mean for each column. That way the new values do not bias the data.

## 3.2



This is the graph of MSE vs number of epochs for training set and test sets “Dataset\_CandC-train0.csv” and “Dataset\_CandC-test0.csv”. Even at 10 epochs, the program took 20 minutes to run. Hence, it would be impossible to run 5k epochs without a much powerful computer. The MSE is shown to be decreasing, and if given enough time to run, should decrease to at least 3 digit numbers.

The current parameters learnt are shown in a 123 x 1 vector:

[-2.2110612359479471e+58, 5.6131592685503863e+56, -3.4520358447900248e+57, -1.1995627289040644e+58, 7.8404641934841262e+57, -3.1862670015588582e+57, 5.3046831932668113e+56, -1.7560878298645836e+57, 4.2455869735876663e+58, 1.0181548236524958e+58, 1.5085534267857655e+58, 9.4866670603062665e+56, 4.1988914651045971e+58, -1.0857582171932625e+58, -3.9918481033716562e+56, -7.3284934721806389e+57, -4.576934162734759e+58, -1.4989103870662064e+58, 8.1158466507651524e+57, 9.1920774984549106e+57, 1.0032800479405474e+58, 1.0679189390351476e+58, 1.035184468221502e+58, -1.2765181499109856e+58, 1.8714601624276659e+57, -2.6398016341672946e+57, 2.713087838677778e+57, 1.0559535594926575e+57, -8.62709558339133e+56, -4.6127163818292844e+57, -6.5188491607692707e+57, -1.6146126588174351e+58, 3.8441179308404464e+57, 3.0452635581857659e+57, 9.722977226151969e+57, 6.4019019043490177e+57, 3.9084673404870646e+57, 8.2165549687273536e+57, -4.3955727520185461e+57, -1.0164419718080775e+58, 2.550992102530322e+58, -2.1312440830845314e+58, -8.3655775996817217e+57, 2.055107167846005e+58, -1.6198254636947755e+57, -2.1376056429454059e+58, -1.2863569333446943e+58, -2.6584092752103644e+58, -1.8526603735979748e+58, 2.7048068418073749e+58, -8.6423329185622029e+56, -1.8504807885151898e+57, 3.6371107769090111e+56, -1.7254165829122914e+56, -1.9704834171068009e+58, 1.6788797475116458e+58, -1.1689363370116805e+58, 7.5687839287945561e+56, 1.2105704511567027e+57, 5.2553905699674465e+57, -5.3349363624178694e+57, -1.2854254335171776e+58, 3.6535949903341504e+57, 1.0067327517541021e+58, -1.0151537205886126e+58, 1.5692443080440949e+58, -1.0889284485046146e+58, 1.1581806796410007e+58, -7.7853185983945528e+56, -1.1375266643505856e+58, 9.4550269487405841e+57, -5.2238646706500588e+57, -1.3457610001748452e+57, 1.3230466847153691e+58, 1.5247559941837622e+58, 4.574399685462755e+57, -1.3892474104314147e+58, -1.0537670120744126e+58, -1.1892193475346963e+58, -1.4149899516881874e+57, -3.9392695655521756e+57, -2.1672657891451978e+57, -2.4059764412002491e+57, 2.3486153687092678e+58, -5.8357843421788917e+57, 1.3543712969063585e+58, -7.5595061918852325e+56, -4.761936352792218e+58, -3.7178184801292963e+57, -1.8528471494273816e+58, 7.3306189391616139e+55, -6.1999071790232347e+55, 1.3438431638448256e+57, -4.0162274613828509e+56, -1.3480838896962651e+58, 2.9698039461088188e+58, 2.0258888264312145e+58, 1.7742565987191e+57, 6.6801144193462943e+55, 1.6817832655404575e+58, -9.5516165719123502e+57, -4.0589512409756921e+57, 2.1676524597001373e+57, -5.4298446912148068e+57, 8.1222131074559226e+56, 1.7315938014179858e+58, -2.5902864915995224e+58, -1.9577058472748907e+58, -1.595645295392898e+56, -8.9148194029302596e+56, 1.5422953766826657e+58, -8.1289493576770244e+55, 5.4758748648029424e+57, 5.5239575671703243e+56, 2.8404169991175577e+56, 1.6184513971993542e+57, -3.2335509230867466e+57, 4.982338123282793e+57, -2.8769202496394164e+57, 5.421826121982496e+57, -1.3911624873886753e+57, -2.4397088182992268e+56, 3.0705750365132001e+57, 10]

Note: Above can be seen as the values for [w0, w1, w2, …, w123]

Each weight is separated by a comma. The original weights array was initialized with every weight being equal to 10.