FoDS ASSIGNMENT - 1

Submitted To

Dr. NL Bhanu Murthy

Foundations of Data Science: CS F320

Submitted By:

Name	ID NO
AASHISH CHANDRA K	2021A7PS0467H
AASHUTOSH A V	2021A7PS0056H
TUSHAR BRIJESH CHENAN	2020A7PS0253H

Birla Institute of Technology and Science, Hyderabad Campus

Aim of the project

The aim of this project is to learn the mathematical derivations of regression, and test variations of it on select datasets using Python libraries like NumPy and Pandas. Implementing this project helped us learn the ideas of Regression, Polynomial Regression, and Regularization, and made us understand the importance of testing and plotting graphs and doing comparative analysis to find the best models.

Acknowledgment

We are grateful to have been given the opportunity to work on this project. We would especially like to thank our course professor Dr. N.L. Bhanu Murthy for giving us the chance to work on this wonderful assignment. The opportunity to work as a team, and collectively learn and enhance our Data Science skills has been enriching. Applying theoretical knowledge from class into practice and getting encouraging results has been motivating to say the least. We have learned implementation skills that will be useful for our future career growth.

Task - A -> Single Variate Polynomial Regression

Task:

Our task here was to:

1. Preprocess the data.
2. Build polynomial regression models with degrees ranging from 1 to 9.
3. Plot the appropriate graphs
4. Comparative analysis of the developed models.

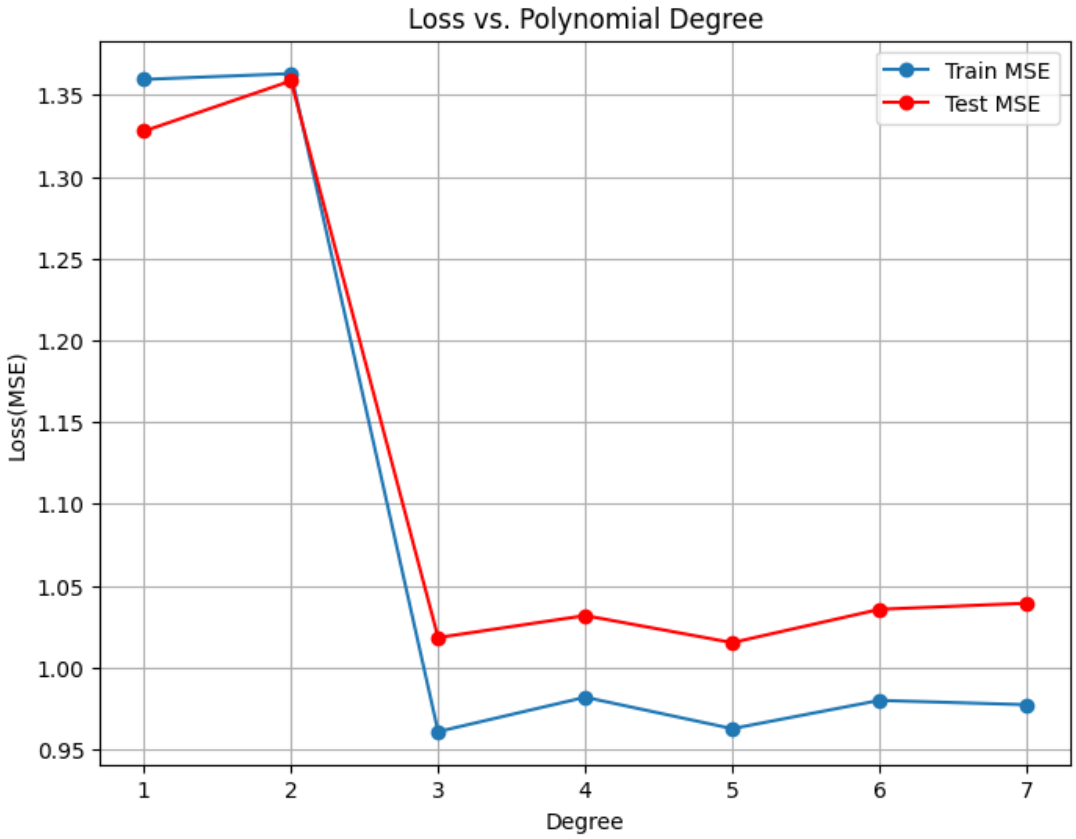
Methodology:

1. We approached the data-preprocessing in the required way, which was normalizing the data over its mean and standard deviation, removing any nan values (which weren't there), and manually implementing the 80% random train-test-split using numpy functions.
2. While building our polynomial regression models, we used the numpy function column stack to develop features X^i, for i ranging to the degree of the polynomial w.r.t our testing loop.
3. While training, we implemented a gradient descent algorithm over the models, which is the Batch Gradient Descent. We used an appropriate learning rate which gave us the best results.
4. Further, we documented our inferences from all the models trained with this descent algorithm.

Results and Observations:

We plotted the Train and Test Errors vs Polynomial Degree.
The best Degree for Train Error was Degree 3 and for Test Error was Degree 5.
Based on the input dataset, this makes sense as the dataset looks seems to have been taken from an odd-degree polynomial. (Explains higher loss for Even degrees)
Also, we can observe the train errors are a little less than the test errors showing some signs of overfitting.

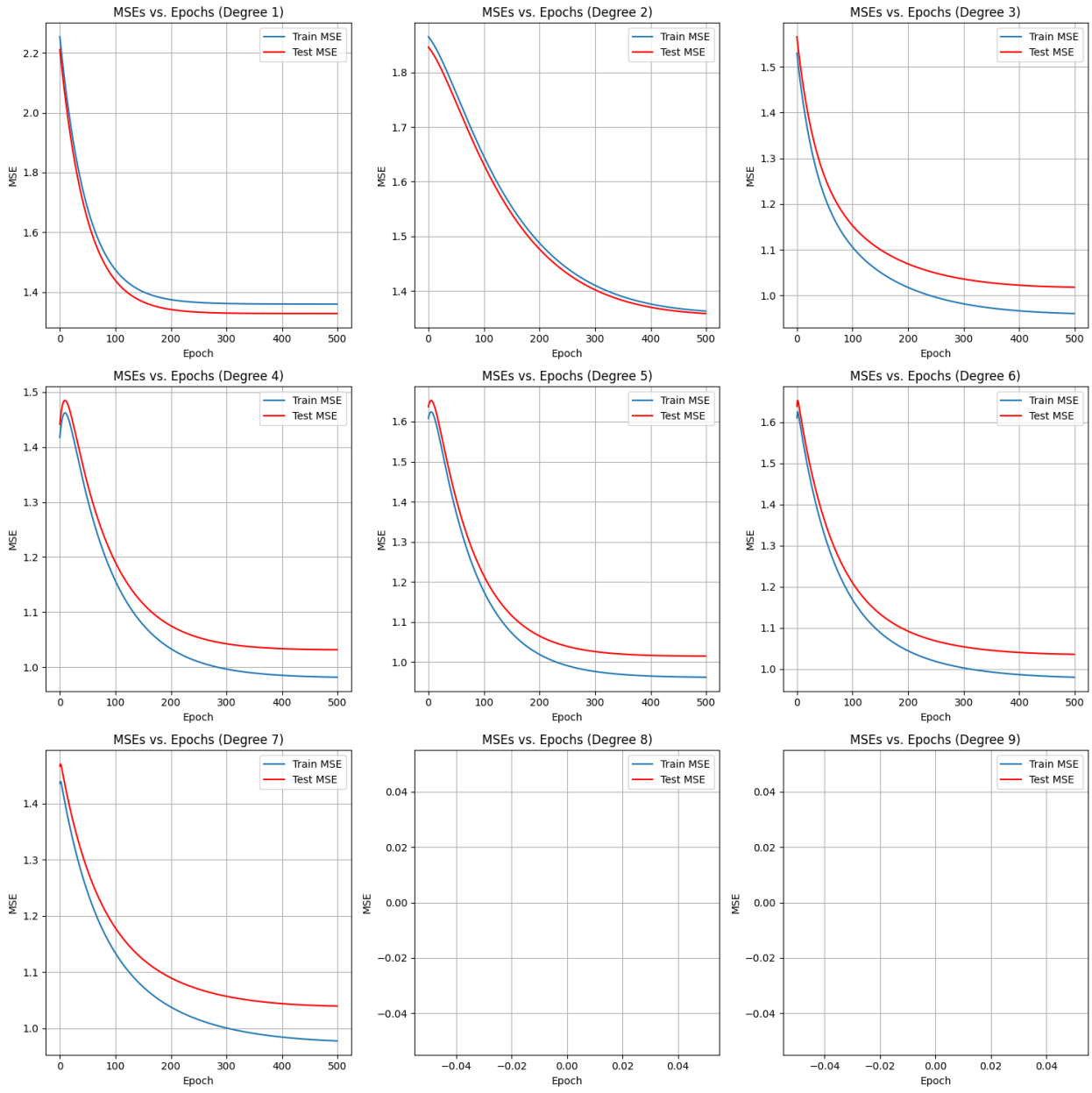
One other observation we made is that, because we aren't doing any Regularization the weights are growing to too high values for Degrees 8 and 9 and we were getting INF and NAN as Errors, which is why it isn't displayed correctly in the plot.



These are the Epoch vs Train-Test Loss for each degree.

As depicted, degrees 8 and 9 don't have a plot because we are getting INF and NAN as the losses.

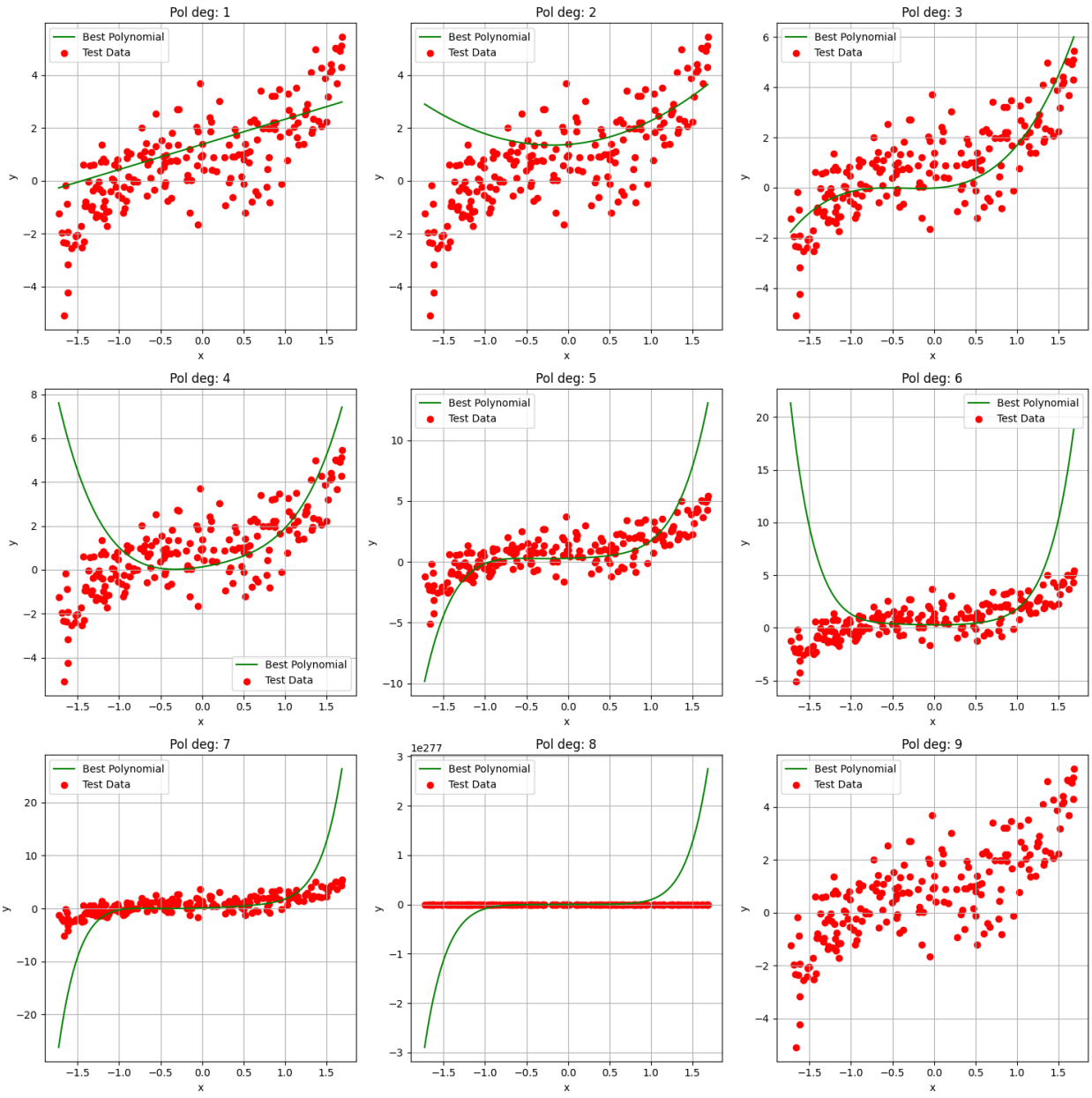
Else, it can be observed that the Gradient Descent is working properly since both training and testing loss are decreasing over time.
And the Test Loss is more than the Train Loss throughout the epochs, this is because we do worse for unseen data.



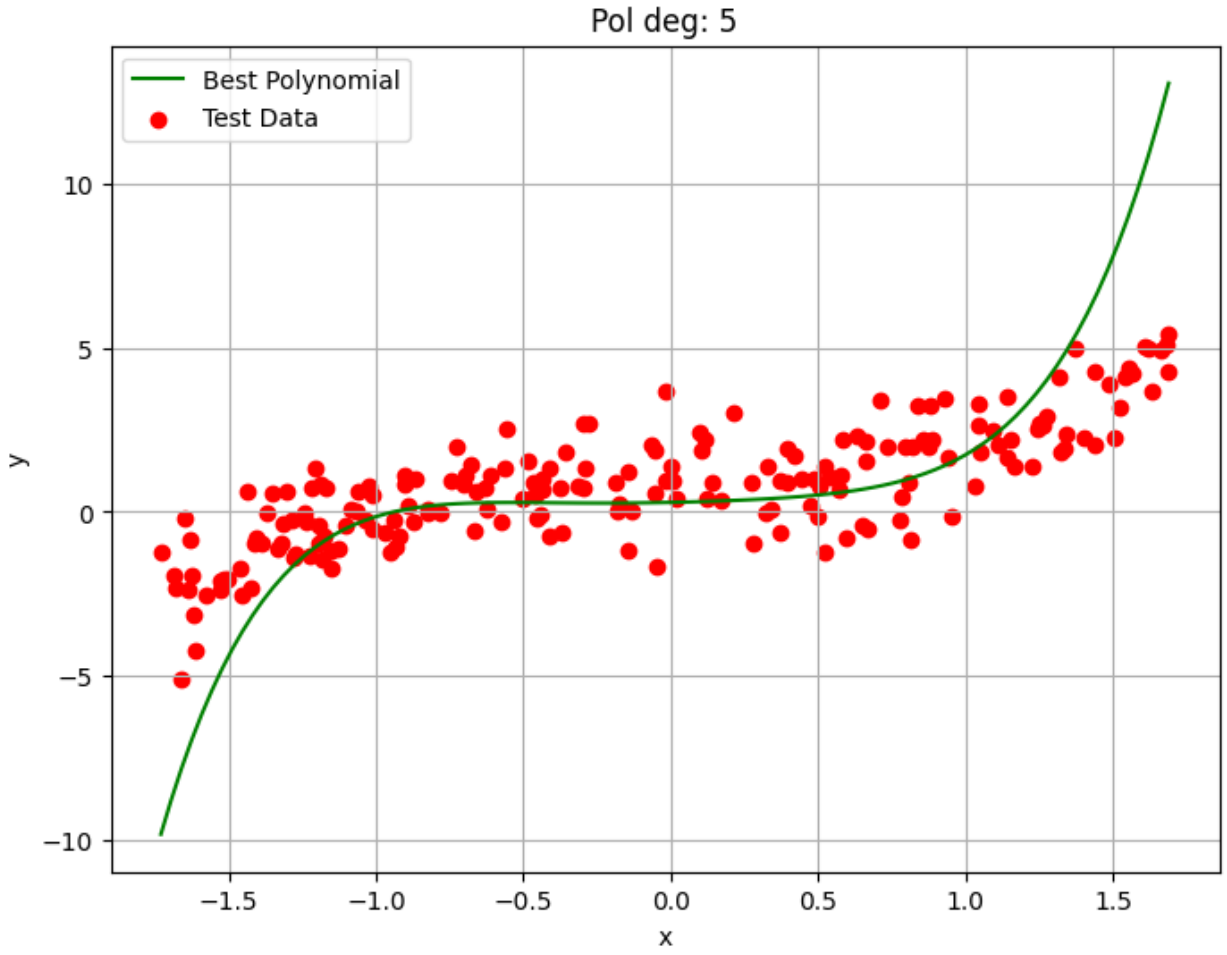
These were the polynomials we obtained for each degree.
The green curve denotes our polynomial, and the red points are the testing data.

As depicted Degrees 3 and 5 fit the shape of the data the best and thus have less loss. On the other hand, degrees 1,2,4,6 cannot fit this shape properly so they have higher losses.

Degree 8 had extremely high weights so the Y value exploded quickly and degree 9 was NAN



Finally, this was the best polynomial that we have arrived at, with degree = 5.



Conclusion:

- Overall, from our observations of the scatterplots generated, and the inferences from the output train and test errors, we have concluded that the polynomial with **degree 5** is the best-fit polynomial for this dataset.
- While the polynomial of **degree 3** might have the best performance on training data, its inability to bend and fit the data in comparison to **degree 5** is the reason why **degree 5** outperforms **degree 3** in testing error.
- We observed that the error values for **degrees 8** and **9** are approaching infinity, and NaN respectively, which makes us conclude that the calculations involved in fitting a high-degree polynomial are numerically unstable, and the higher-order terms lead to overflow/underflow issues.
- The following chart illustrates all our observations made from coding this experiment:

Comparative Analysis:

Degree VS Error

Degree of Polynomial	Train Error	Test Error
1	1.359	1.327
2	1.363	1.358
3	0.961	1.018
4	0.982	1.031
5	0.962	1.015
6	0.979	1.035
7	0.977	1.039
8	inf	inf
9	nan	nan

Our inferences we made in the conclusion section for task 1 are validated based on the data from these charts.

Task - B -> Polynomial Regression & Regularization

Task:

Our task here was to:

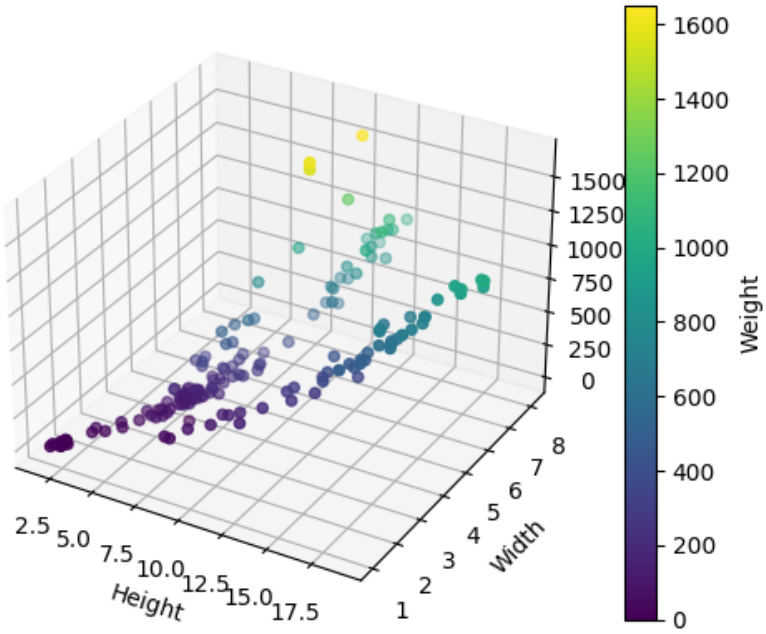
1. Preprocess the data.
2. Build polynomial regression models with degrees ranging from 1 to 9.
3. Plot the appropriate graphs
4. Comparative analysis of the developed models.

Methodology:

1. We did the required 80% train-test split on the normalized data.
2. Further, the approach to build the polynomials was slightly different here, owing to the fact that the data has 2 features instead of just 1, like the task from the previous dataset.
3. The polynomial we created for each polynomial was of the form:
   a. $\sum_i \sum_j x_1^i \cdot x_2^j$ where $i + j \leq D$ and $D \in [1, 9]$.
   b. Here x_1 represents the first feature and x_2 represents the second feature.
4. We added the regularization term to the loss function, and experimented with various exponents of the norm of the weight vector, and lambda, the coefficient of the regularization term.
5. Next, we trained the models using both batch-gradient descent and stochastic gradient descent algorithms,
6. Lastly, we inferred the results from the plots of the errors, and the polynomial functions themselves.

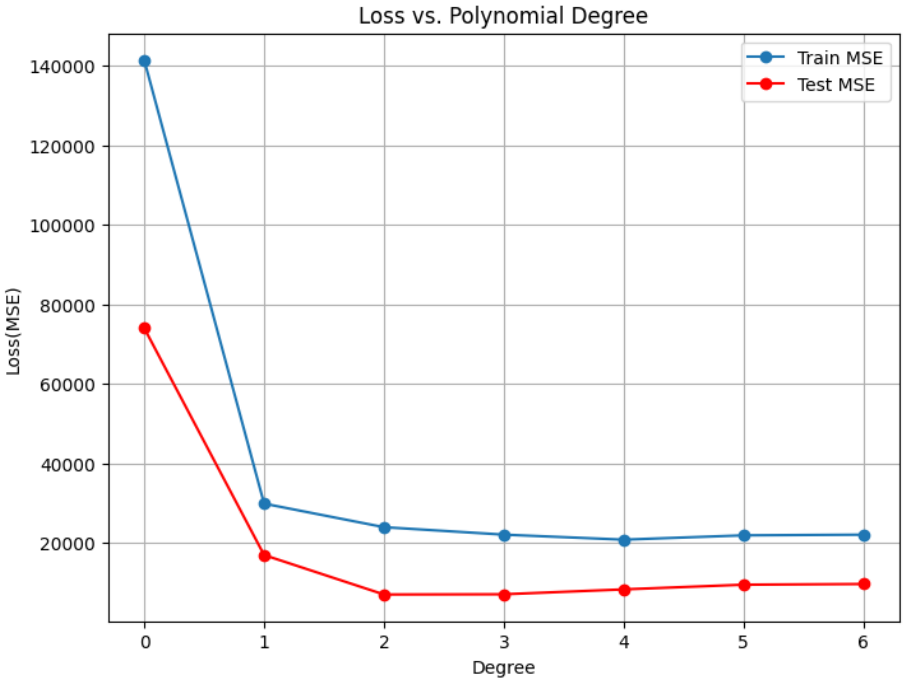
Results and Observations:

Visualizing the data, the weight of the fish appears to be proportional to its weight and height.



This is the Losses vs Polynomial Degree curve (Non-Regularized)

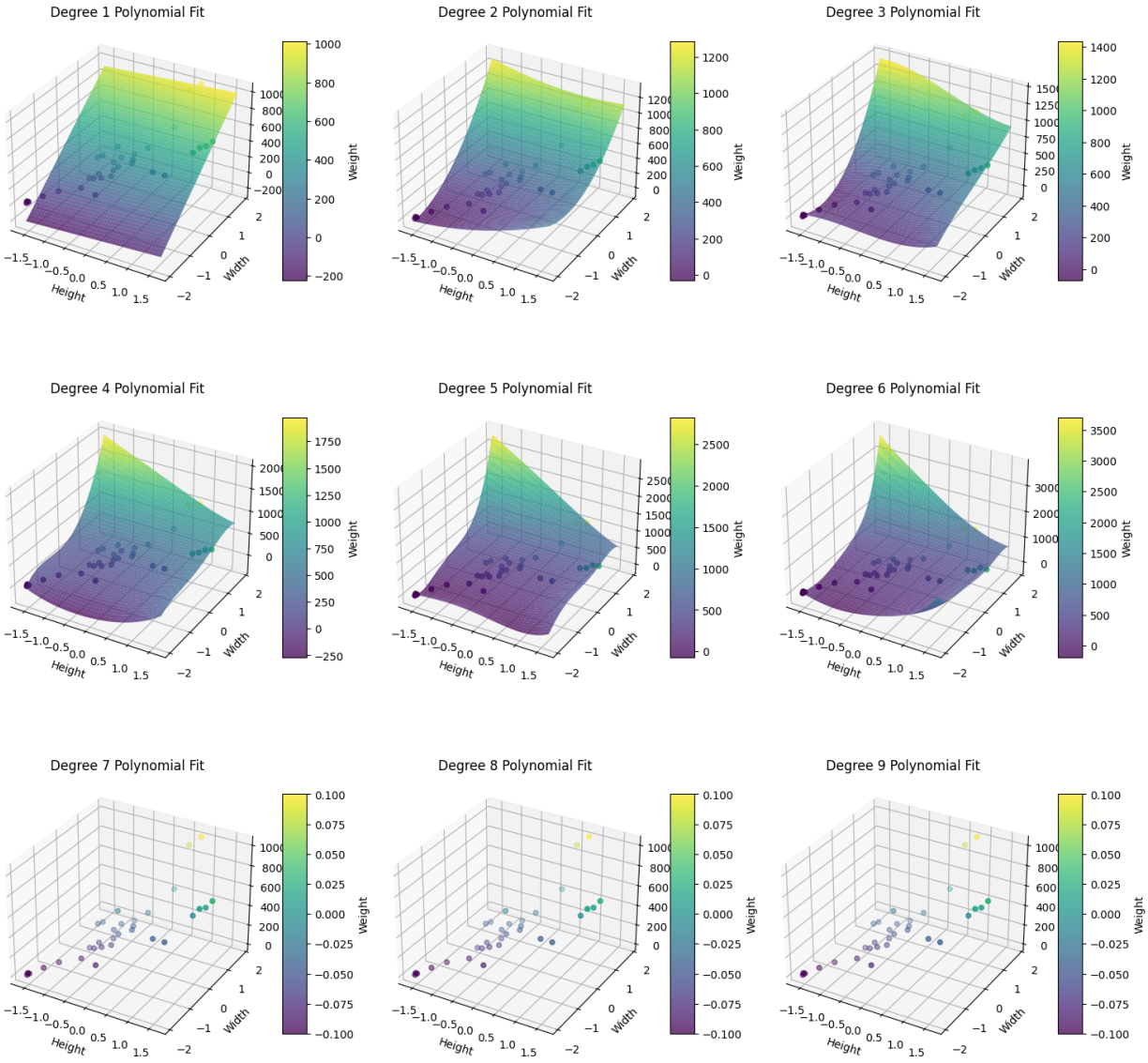
Degree 2 is the most ideal because for the reduced epochs we have used, higher degrees were underfitting. Degree 0 (only Bias) was the worst.



We trained 9 different models for each degree as mentioned above, and these were their plots. No regularization has been done here.

As seen, Degree 1 is linear, degrees 7,8,9 have not been shown because they exploded and became NAN. The reason for this is that the inputs are very small but the target is very large, causing the weights to grow to exponentially high values to try and fit the data.

Other degrees seem to have captured some complex topology and have good losses, They seem very complicated and awkward in nature.



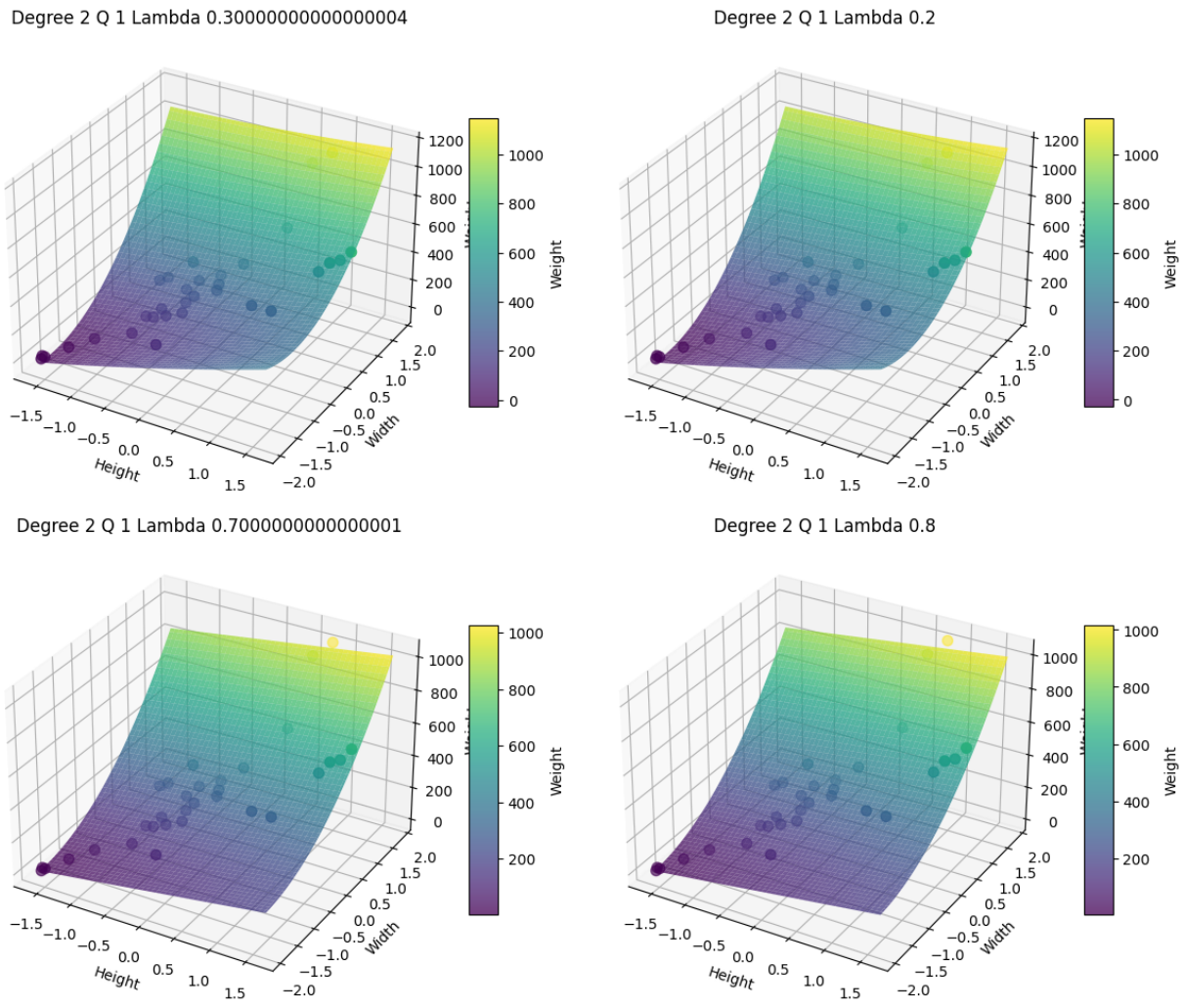
Further, as we have mentioned, we tried multiple Q values and Lambda values for the best degree 2, and have gotten the best 4 plots

The observations from these plots are, Q = 1 seems to have done better.

In that also Lambda = 0.2, 0.3 and 0.8, 0.9 seem to have done well.

When lambda is less we can see regularization is less and the curve is a little awkward.

But for the higher lambda counterparts, the surface is smoother and more planar like.



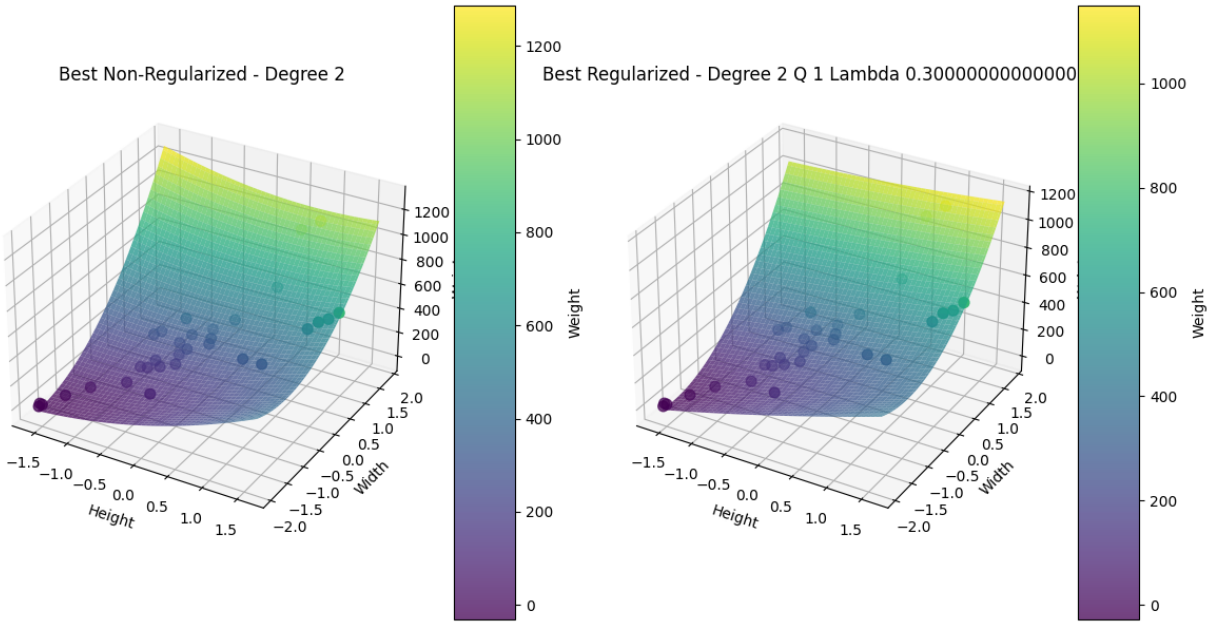
Finally comparing the best Regularized Model to the best Non-Regularized Model,

Best Non-Regularized Model | Test MSE : 7066.506532395979

Best Regularized Model | Test MSE : 4884.094866481239

Even though the target values are very high in this case, smoothing out the curve and reducing the weight seems to have combated overfitting and it more naturally fits the

data. This surprised us, since we assumed regularization will make loss worse



Conclusion

- We see that without any regularization, the polynomial which did best in the testing data was **degree 2**.
- While the higher degree polynomials did better in training data, their surface plots take awkward topological shapes indicative of the polynomial trying to bend to fit the training data, showing that higher degree polynomials are overfitting the training data. This is also why they perform poorer in testing data.
- We have taken the best **degree 2** polynomial and applied regularization to it. Of the various regularization error functions, we were supposed to experiment with $L_{\frac{1}{2}}$, L_1, L_2 and L_4 functions. Here, Q denotes the power to which we are regularizing and λ is the coefficient of regularization, which means that the generalized error function is:
    - $$MSE + \lambda(\sum_{i=1}^{f}(w_i)^Q)$$

- For this function, we have established that $Q = 1$, and therefore, L_1 regularization gives us the best results, for various values of λ.
- As we vary lambda in the case of L_1 regularization, we see that for higher values of λ, the w_i values are constrained to take smaller values and therefore the surface plot is more smooth and less steep.
- For lower values of λ, the w_i values are allowed to take bigger numerical values and therefore the surface plot is allowed to twist/turn and bend more with a higher slope.

- Lastly, the overall MSE for degree **2** polynomial was **lesser** when we employ the technique of L_1 regularization than when we use no regularization.

Comparative Analysis:

Degree VS Error (Unregularized)

Degree of Polynomial	**Train Error**	**Test Error**
0	141466.88	74066.96
1	29918.76	16995.25
2	23990.07	7066.50
3	22127.94	7116.04
4	20852.15	8350.13
5	21944.24	9529.22
6	22105.38	9702.13
7	nan	nan
8	nan	nan
9	nan	nan

The above chart depicts our basis for conclusion on the best degree polynomial without Regularisation

Errors for our best degree after Regularisation (degree=2):

Q	Lambda	Batch	Stochastic	Error
1	0.3	1	0	4884.09
1	0.2	1	0	4959.37
1	0.7	0	1	5101.44
1	0.8	0	1	5191.19

This chart depicts our observations on regularizing the best degree polynomial. We have shown the 4 best observations from the conditions on regularization out of the 88 possible models we've trained.