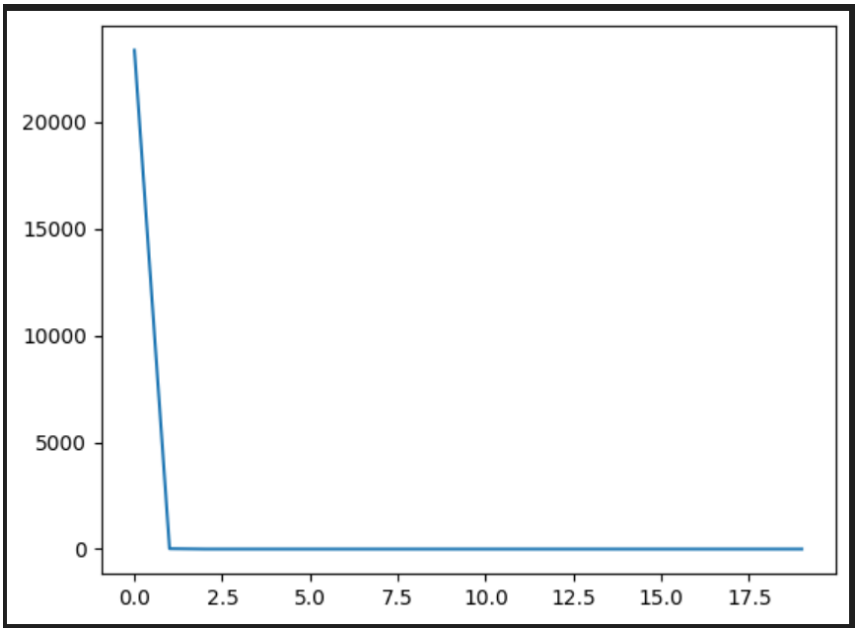
WOC REPORT

Machine Learning

❖ **Linear regression**

1. I made simple linear regression model with the help of for loops and numpy.
2. I used Mean Squared Error as the cost function.
3. Then I added mini-batch feature.
4. Other than that it also had a function that can normalize the data.
5. I also normalized the data so that it will be easier to implement gradient descent.
6. In mid evaluations mentors told me to vectorize the algorithm after which it takes 0.8 seconds to run for 100 iterations before vectorization it used to take 5 min.
7. While vectorizing linear regression I got many errors due to shape of vectors which I solved by printing the shape of the arrays.
8. I divided my training data in 9:1 ratio for training set and cross validation set.
9. I decided hyperparameters by experimenting I used different values of learning rate ranging from 0.01 to 1.5.I tried different values of iteration ranging from 1000 to 15.
10. In the end I decided to keep the value of alpha 1 and number of iterations 5.Which takes only 0.0 seconds takes to run.

11. Cost on training set was 0.0050498683692287O7 and 0.005058363804088417 on cross validation set.



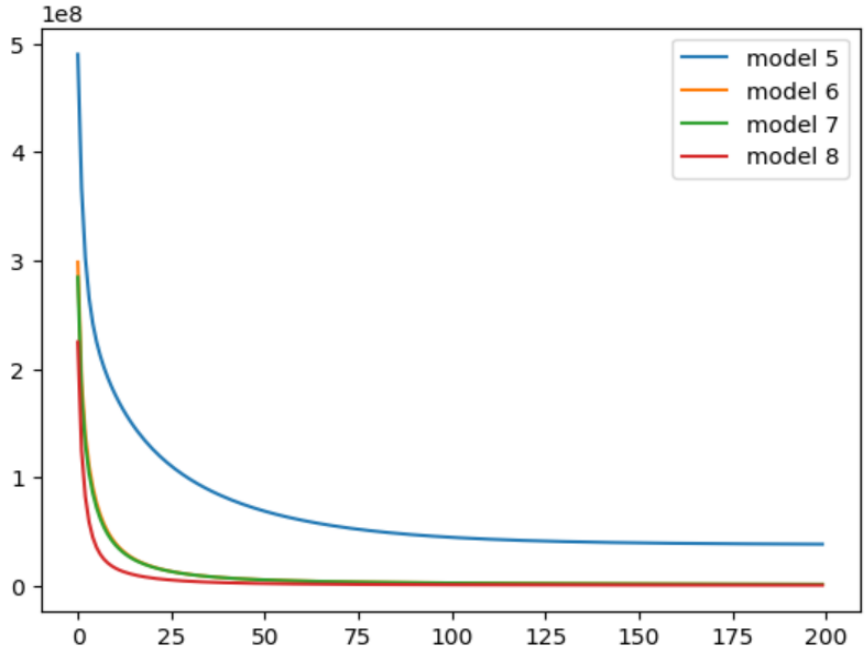
Learning Curve of Linear regression for 20 iterations

❖ **Polynomial Regression**

1. My first approach for this problem was a simple program which can calculate nth power of a feature like x_1^2 , x_2^2 etc.
2. I used Mean Squared Error as cost function.
3. I also made other functions like one to normalize the data.
4. I also normalized the data so that it will be easier to implement gradient descent.
5. I also added regularization to avoid overfitting.
6. With this approach I got very high cost of magnitude 10^{58}. At that time I was not aware of the R2 score. So I didn't calculated it but after seeing the cost I had realized that I need to improve my algorithm.
7. I improved my algorithm by adding additional features like $x_1^2x_2^3$ etc.. With this model the cost was still very high but R2 score was 0.82 for degree 6.
8. In mid-evaluations mentors told me to that I can improve my R2 score by just using two for loops. After applying their

approach my R2 score for degree 6 increased to 0.9999968804120626 and cost was 2288.309887410971 for learning rate 0.28 and iterations 1000 which was low in comparison to earlier approaches.

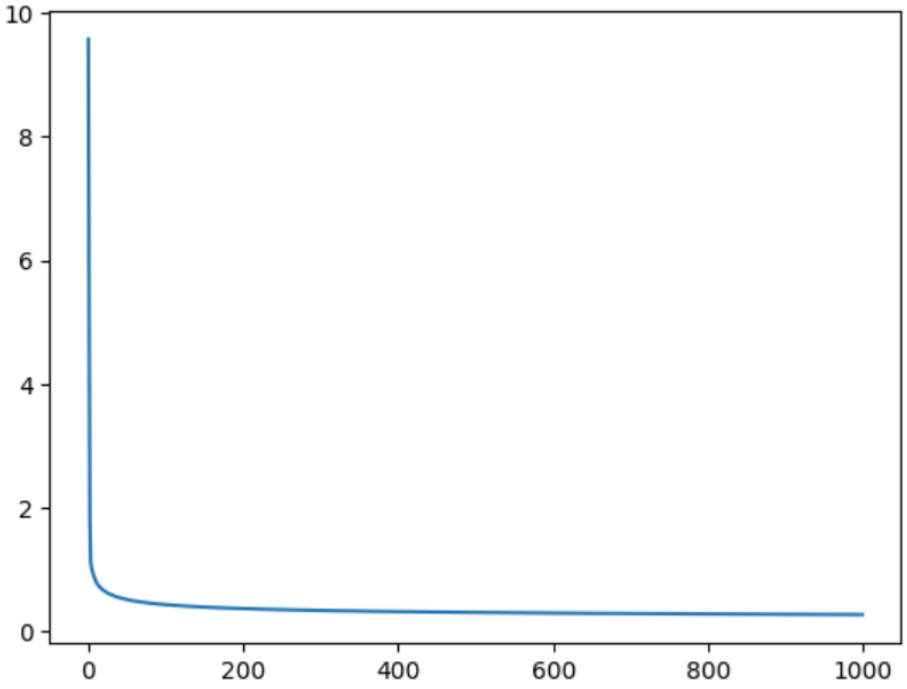
9. After mid evaluations I also vectorized my code which made it fast.
10. For hyperparameters I experimented with different values of learning rate ranging from 0.01 to 1, identations ranging from 100 to 1000, and regularization ranging from 0 to 2.
11. After experimenting I choose the degree value 6, learning rate 0.13, identation 400 and regularization value 0.
12. I choose the degree 6 because there was a jump of 0.95 to 0.99 from degree 5 to 6 in R2 score.
13. When I was experimenting with regularization parameter I noticed that results on cross validation set were better when I took regularization parameter 0.



Learning Curve of Polynomial Regression for Degree 5,6,7,8

Logistic Regression

1. Firstly I created simple logistic regression with the help of for loops and Numpy.
2. It had functions for one hot encoding, for normalizing data with z-score, sigmoid, cost function , gradient descent and pred for making predections.
3. I also made it vectorized as mentors told me in mid evaluations.
4. I used Binary-Cross Entropy Cost as cost functions.
5. In my first approach I simply trained ten different models for ten different classes and then I made predictions using those ten different models. After which I concatenated them in a single array and made final predictions for each examples.
6. It took 4 min to run with alpha value 0.25 and iterations 100 and had accuracy of 95.82%.
7. After I made my Neural Network I realized that in logistic regression instead of training each model individually I can train them simultaneously.
8. It took only 2 minutes to run with alpha value 2 and iterations 1000 and had accuracy of 96.56% for training set of 25000.
9. I removed all the columns from the dataset which were always Zero and then It took 1min and 37 sec to run for 1000 iterations for training set of 25000.



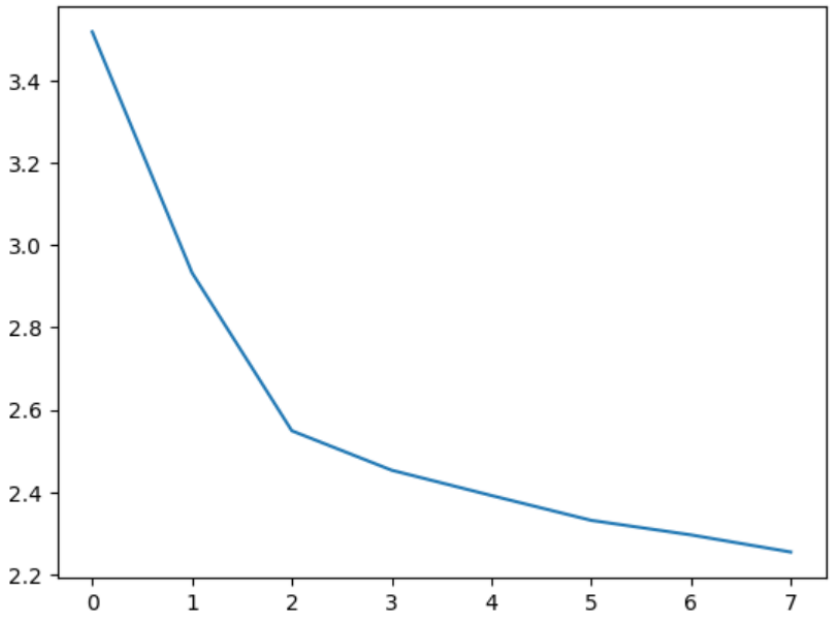
Learning Curve of Logistic Regression for 1000 Iterations

❖ **KNN**

1. I initially made simple KNN model with the help of for loops and Numpy.
2. It had only two main functions one for computing distance and one for running algorithm.
3. It used the concept of distance between two points and finding K closest points and giving the answer by calculating mode of these points.
4. It used took 75 min to run for 25000 dataset and 5000 queries as it was not vectorized.
5. After mid evaluations I vectorized my code for only one example and it still used to take 25 min.
6. I was unable to think at that time how to vectorize the code fully so I started to think in a different direction.
7. One of the approaches that I found out was related to finding the directions of two vectors using dot product and calculating the cos of the angle between them and comparing them. The more closer the values are to 1 the more the vectors are alike and more the values are closer to -1 more they are different.
8. After implementing its accuracy was coming to be 98.8% for k=1. For any other values of K accuracy just kept on getting worse.
9. After vectorization it took 4 minutes to run for 5000 queries with 25000 dataset.

❖ K-Means

1. I made initially a simple K-means model with the help of for loops and Numpy.
2. It had functions to calculate mean of centroids, closest centroids for each point, cost function etc.
3. Initially it was very slow as I was using distance as a parameter and I was not able to fully vectorize it.
4. So I used a different approach that approach was based on calculating the cosine of the angle between two vectors using dot product.
5. The closer the value is to the 1 the more they are alike and more the values are closer to the -1 more they are different.
6. It after implementing this approach my K-means got faster.
7. I decided the value of K by elbow method and after seeing the graph I have decided that the value of K is 3.
8. As we can see there is a drastic drop on 2(which is 3 in 1 based indexing).

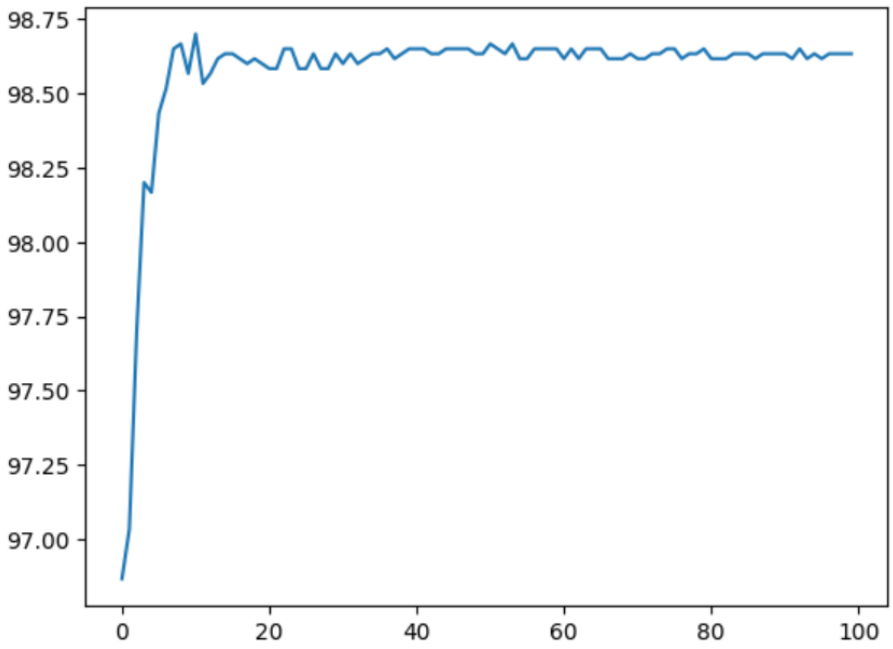


Cost on Y-axis and value of (K-1) on X-axis

❖ Neural Network

1. For making Neural Network I studied Backpropagation and took reference of coding from a book That I Found on Google.
2. The name of the Book is **Neural Network and Deep Learning** by **Michael Nielsen**.
3. I studied Backpropagation from the book then I started implementing Neural Network.
4. I made my first Neural Network using Sigmoid activation function and Binary Categorical Cost Function.
5. When I ran my model first time it didn't run properly due to shape's of arrays not matching.
6. It took me some time to realize what was the problem.
7. But even after fixing the previous problem My model was not learning anything.
8. After when I printed every element I realized that values e^{-Z} was coming out to be zero as values of zero were very large.
9. To encounter that problem I rescaled all the value in the given Dataset by 255 making all the features in the value of 0 and 1.
10. Doing only this much took me 2 weeks.
11. I Vectorized my whole code by my own after mid evaluations.
12. In mid evaluations Mentors also told me to apply softmax activation in last layer and Relu activation in hidden layers.
13. It took me some time to figure out the equations of Backpropagation for last layer.
14. After figuring out that applying Relu activation on hidden layer was easy.

15. Initially I was Initializing my weights from zeroes but I got to know that if I will use some other techniques to initialize my weights I can train my model faster.
16. I learned about Xavier's Initialization and He's Initialization.
17. I have used He's initialization as Xavier's initialization is better for activations in which mean is zero like Sigmoid and Tanh.
18. For neural architecture I tried different combinations like [784,40,20,10],[784,32,10],[784,1000,10] and [784,30,10].
19. I got best results on architecture [784,30,10] with an accuracy of 98.6% on cross validation set.
20. I also tried different values of Learning parameter ranging from 0.01 to 1 I decided on the value of 0.25 for 100 epochs and it take 2 minutes and 21 seconds to run.
21. For regularization parameter I tried different values of parameter ranging from 5 to 0. I got best value when regularization parameter 0.
22. I also tried different value of batches ranging from 5 to full training dataset but I got best value on the mini batch size 10.
23. I noticed that in training dataset there are many columns which are always zero.
24. After making some functions I got to know there were 94 columns which were always zero. So, I removed those features. Now my Neural network takes 1 min and 47 seconds which was earlier 2 minutes and 21 seconds for 100 epochs and dataset of 25000.



Graph of Accuracy after every epoch