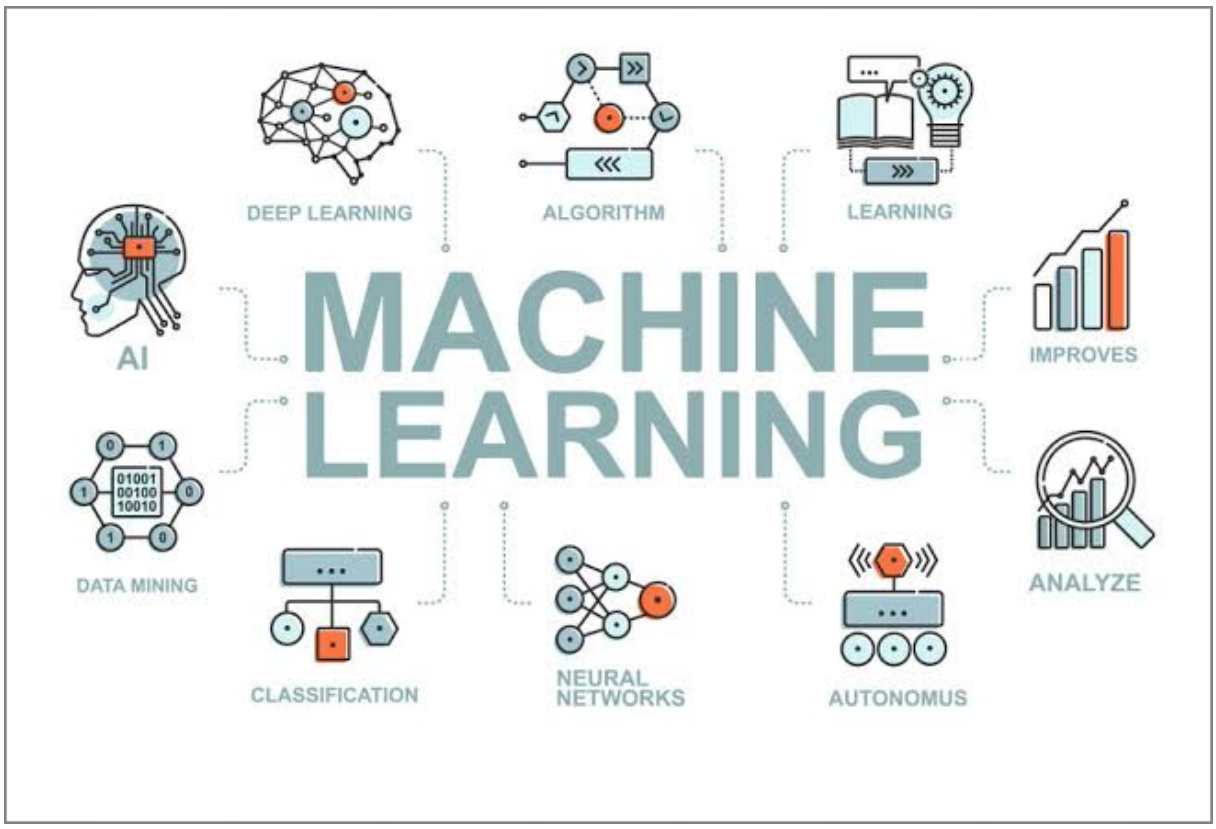
ML REPORT FOR WOC

31/03/2023



By Manav Jain

22JE0538

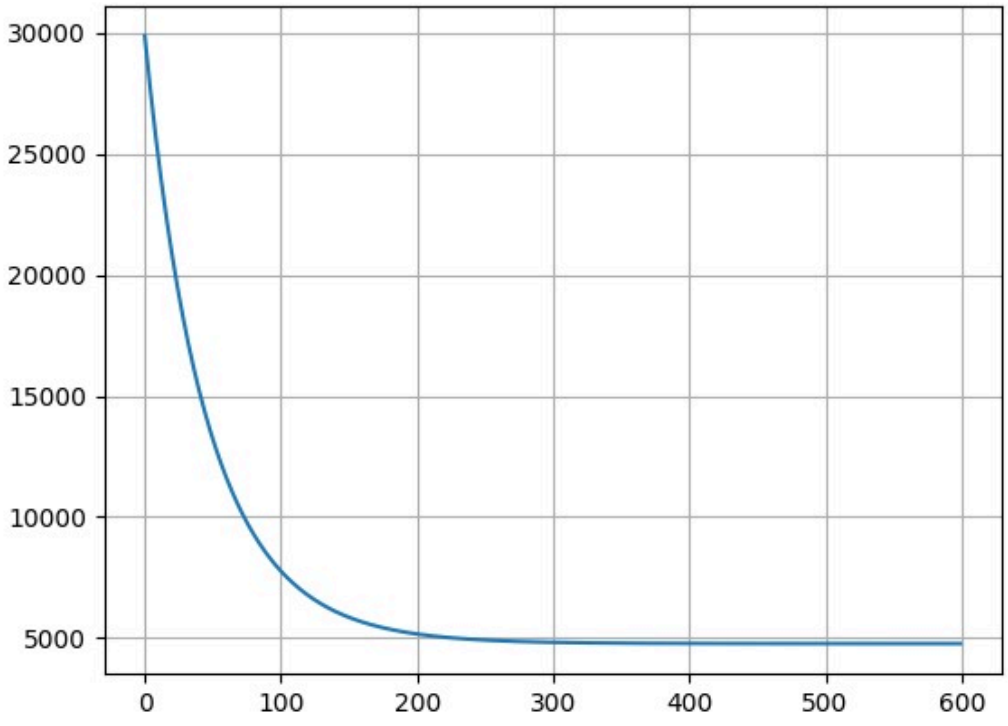
Progress Through The Weeks

Before the start of coding phase and provision of training and test datasets, I installed jupyter notebook and started getting familiar with its working. I started implementing basic algorithms like single variable linear regression. As soon as the datasets were made available, I started working with them creating primitive algorithms for linear and polynomial regression. Similarly, I created primitive algorithms for logistic regression and started watching machine learning specialisation course. Then, I attended the mid-evaluation, where under the guidance of my mentors, I realised there were many unvectorized spots in my code. I also learnt about r2 scores and accuracy measurement for classification. Over the next week, I vectorised my code and calculated r2 scores and accuracies for various algorithms. Vectorising my code reduced the runtime drastically. I also started working on KNN and continued watching machine learning specialisation course. While analysing my code, I realised that a lot of major factors of my code were common through all the algorithms. Then, I started implementing all 5

algorithms in the same notebook. Then, I created python file in which I defined a base class labelled as “base_class” in which I defined all common parts of all 5 algorithms and created different classes for each algorithm. Using concept of inheritance, I defined the base class as the parent class for all algorithm classes so that I can access all the common parts through base class. After this, I created a notebook in which I imported classes from python file to run my code. After this, I tried implementing neural network and figured out single layer neural network. I was also done with n-layer neural network and spent the rest of the time improving the runtime and accuracy of my code.

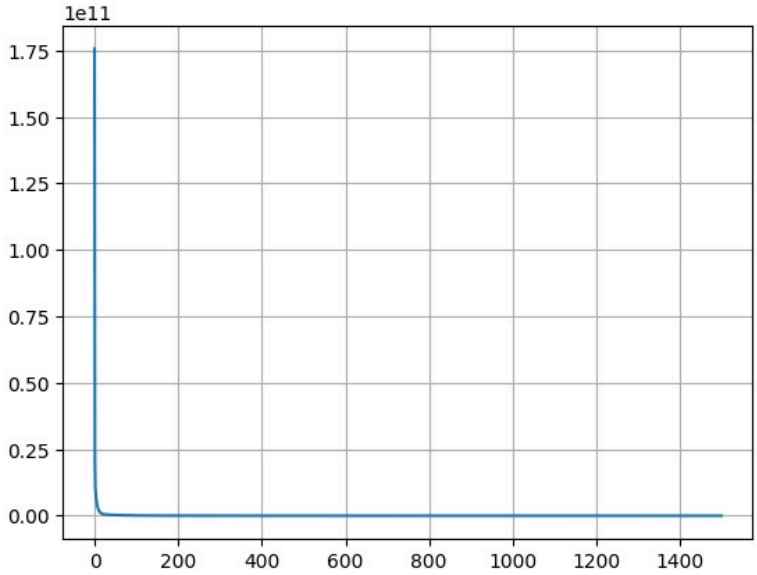
Linear Regression

For linear regression, I defined the algorithm class as "**linear_regression**" connected to "**base_class**" as its parent class. The code is running for 600 iterations with learning rate of 0.01. I splitted my data in the ratio of 80:20(40000:10000) for training and testing. I achieved r2 score of **0.841825173644053**. The reduced cost function after 600 iterations was **4774.883533808007**. The cost vs iterations graph that I obtained with 600 iterations and learning rate 0.01 is attached below.



Polynomial Regression

For polynomial regression, I defined the algorithm class as "**polynomial_regression**" connected to "**base_class**" as its parent class. The code is running for 1500 iterations with learning rate of 0.0009. I set the degree of the polynomial to be considered as an user-defined input. I splitted my data in the ratio of 80:20(40000:10000) for training and testing. I achieved r2 score of **0.99963567784632**2 for polynomial of degree 5. The reduced cost function after 1500 iterations at degree 5 was **949981.3327363413**. The cost vs iterations graph that I obtained with 1500 iterations and learning rate 0.01 is attached below.



Logistic Regression

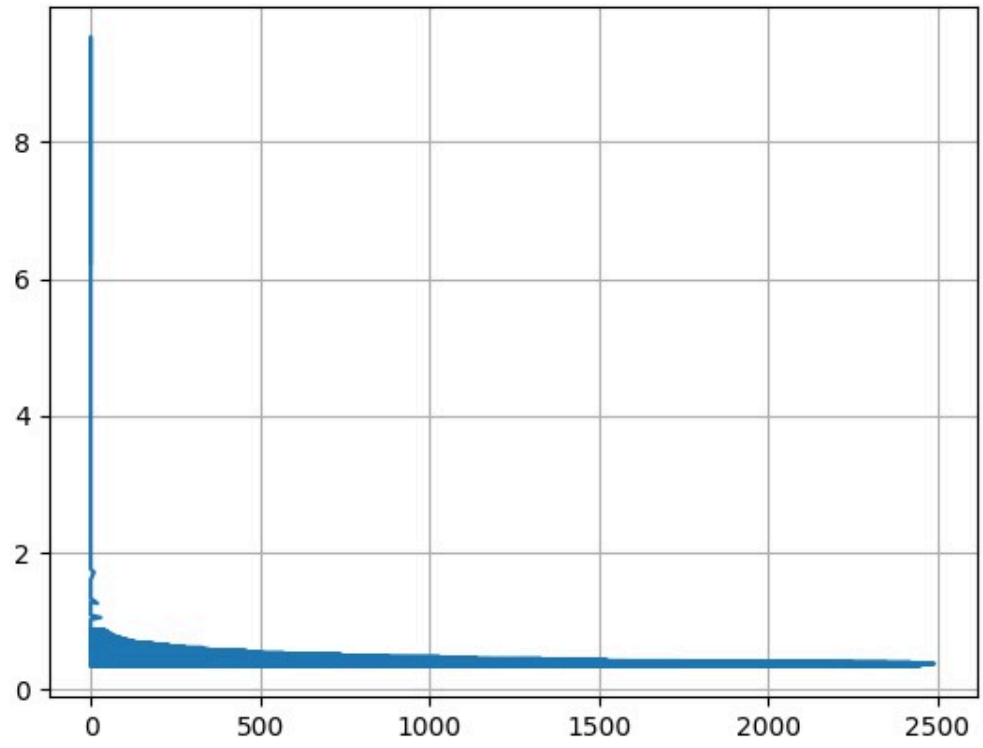
For logistic regression, I defined the algorithm class as "**logistic_regression**" connected to "**base_class**" as its parent class. The code is running for 250 iterations each label with learning rate of 0.099. I splitted my data in the ratio of 80:20(24000:6000) for training and testing. I achieved **79.08333333333333%** accuracy using **one v/s all data classification**.

Knn

For KNN, I defined the algorithm class as "**KNN**" connected to "**base_class**" as its parent class. I have taken the value of k to be a user-defined input. I splitted my data in the ratio of 80:20(24000:6000) for training and testing. I achieved **84.0%** accuracy for value of k set as 5.

Neural Network

For single layer neural network,I defined the algorithm class as "**single_layer_neural_network**" connected to "**logistic_regression**" as its parent class. The purpose of doing so was that now I could access functions defined in both "**logistic_regression**" and "**base_class**". This was necessary as there were certain functions defined in "**logistic_regression**" that were needed to be accessed in neural network. I splitted my data in the ratio of 80:20(24000:6000) for training and testing. I achieved **83.28333333333333%**. The cost v/s iterations graph is attached below.



This accuracy was achieved after running 2500 iterations at a learning rate of 0.0003. The single layer was of **28 neurons of sigmoid activation**.

For n-layer neural network, I defined the algorithm class as "**n_neural_network**" connected to "**single_layer_neural_network**" as its parent class. With this, I was able to access functions from "**single_layer_neural_network**", "**logistic_regression**" and "**base_class**". I splitted my data in the ratio of 80:20(24000:6000) for training and testing. I achieved **38.2%** accuracy. This accuracy was achieved after running 200 iterations at a learning rate of 0.09. There were two hidden layers in which the first hidden layer was of **112 neurons of tanh activation** and the second hidden layer was of **28 neurons of tanh activation**.

About Me

Name: M Manav Jain

College: Indian Institute Of Technology (Indian School Of Mines),Dhanbad

Place: Chennai, Tamil Nadu

Discipline: Bachelor Of Technology in Electrical Engineering

Year of study: 2022-2026

Contact Number: 9962599992

Email Address: 22je0538@iitism.ac.in

LinkedIn profile: www.linkedin.com/in/manav-jain-05a711255