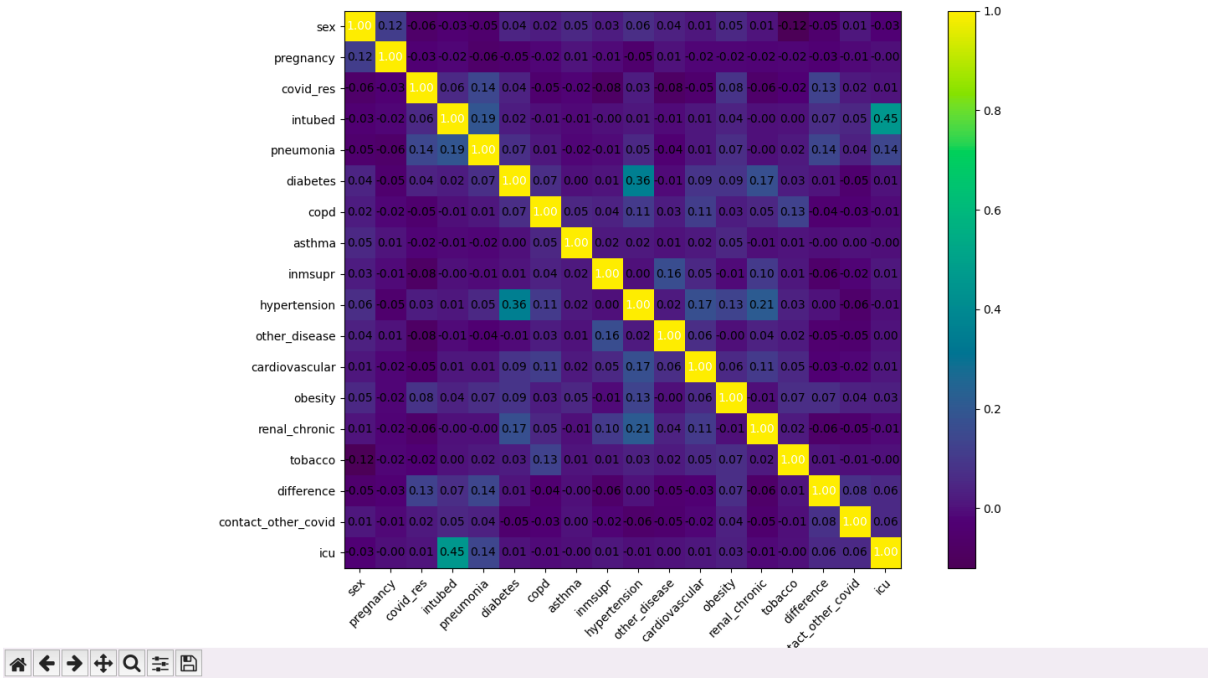
I spent a lot of time trying to analyze the data by looking at the columns in detail. I tried looking at the correlation matrix and use the features having a high correlation with the output icu (intubed,pneumonia), the magnitude of weight for these particular features was high as well which helped make my decision easier. Since there weren't many features related to icu, I logically went ahead and chose some other features to include in the model.I removed values in the 'icu' column which were not specified since they do not help us with identifying whether someone will end up in the icu or not. I immediately removed the id column since it is just a label and does not contribute to the model, I also did not consider the patient type since if a person is not hospitalized, they possibly can't be in the icu. I removed the sex column as well since there doesnt seem to be a relation between one sex having a higher chance of going to the icu than the other. I thought age could be an important factor since a person who is more aged is likely to have more health complications. For almost all other features, there was hardly any correlation with the icu, and even more, it did not sound logically right to include them since those features might cause the person to visit the hospital, but i doubt it would cause them to be admitted to the icu. I changed all the values which represented "no" to a 0 as well, just to be consistent with logistic regression.Other than that, I made use of feature engineering and created a new feature which would be the difference between the date that the person entered the hospital and the date that the person first got symptoms, I thought that if the person delayed coming to the hospital, they have a higher chance of going to the icu.



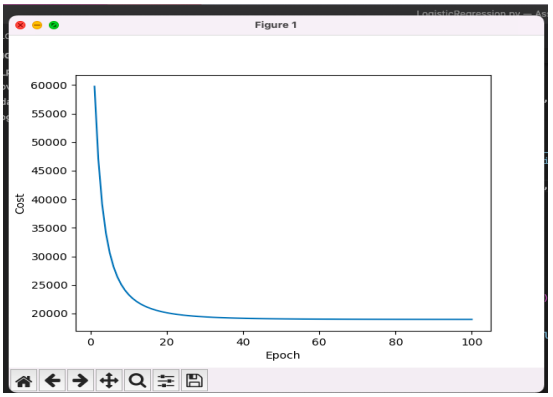
In regards to the other 4 elements of training, the unknown target function would be something that can perfectly classify whether a person will be admitted into the icu or not based on the data provided. We made use of a logistic regression learning algorithm to feed our data into, this provided us with a set of candidate formulas, out of which we choose the final hypothesis one, which in our case happens to be when the training finishes and the weight values associated with the features are the most optimal for the data set. This final hypothesis gives the best approximation in my case of whether a patient will be admitted into the icu or not.

I implemented all the scalers(min_max,standard,robust,max abs) by hand. I was expecting min max to give the best output since there aren't many outliers in our data however, the difference in output for all of the scalers was not that much.This is probably since almost all features are already between 0 and 1, it's just the age and the difference feature which i engineered that are not, and these two features don't have a big impact on our model since the correlation with the icu is not big.I went ahead and used the standard scaler since it had the lowest cost associated with it and worked the best in my case, i'm assuming it's because the data is normally distributed.

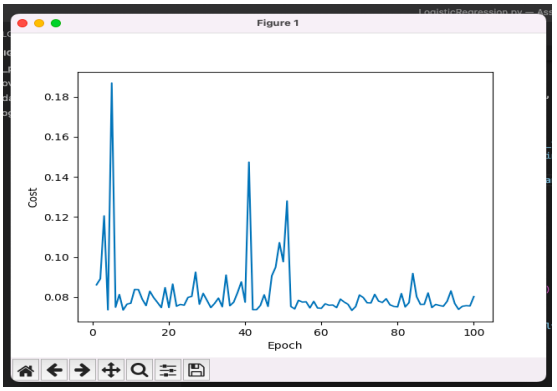
To determine the best learning rate, I made us of the metrics provided by k-cross validation and tried running the model with different learning rates and chose the one which gave the best values for the precision,accuracy,recall, and f1 score. For the batch gd, I found this learning rate to be 0.00001 which gave me an accuracy of 0.920. I did the same thing for stochastic gd and got a learning rate of about 0.1. I tried playing around with different learning rates for batch gd, I found that using higher values started giving me nan values for the cost which means that the data is overfitting.

Comparing both stochastic and batch gradient descent, I decided to go with batch gradient descent. I was expecting stochastic gd to give me better results since the dataset is huge, however, batch gd converges faster and the cost graph straightens at around 30 epochs according to the graph and the printed cost values associated with it. The batch process took about 2 seconds where as the stochastic method on 100 epochs took around 8 seconds. The stochastic GD had a relatively lower cost compared to batch, the difference was too big and I was not sure what to make of it.

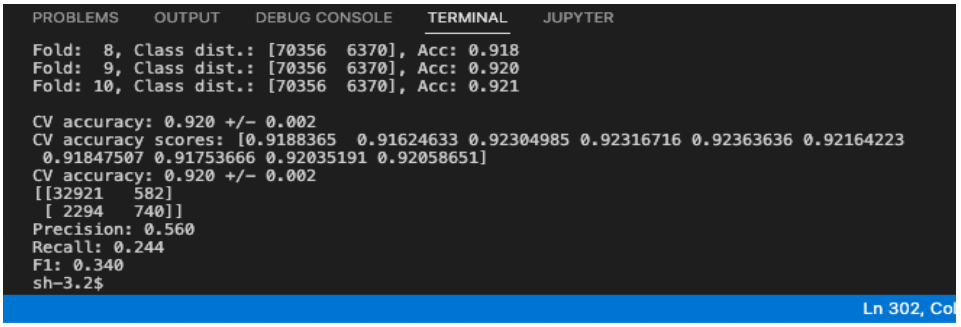
Batch GD



Stochastic GD



I felt that the model that I produced has several errors. I wasn't sure what to make of the accuracy and precision metrics since I wasn't getting a reasonable value, I spent a lot of time trying to play around with it but it just wasn't working as it should have. For the confusion matrix, the top left column was quite large compared to the other rows, which is a good sign.



Overall, I’m very disappointed with the way this assignment turned out. I’m going to try my best to make up for it in the upcoming assignments. The past couple of weeks have been very busy/stressful and I wasn’t able to focus as much as I would have liked.