redwine quality classification

Sina Tayebi

March 2024

1 Problem statement

In this project, we implemented a MLP(Multi Layer Perceptron) from scratch using Numpy only! We used this network to classify red wine quality(from 0 to 10). we trained and evaluate our model on redwineQT dataset

1.1 Dataset

columns:

1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol

Output variable (based on sensory data), quality (score between 0 and 10). it has no categorical feature and total number of rows are 1114. So we have a small dataset with small set of features.

2 Exploring data

2.1 Distributions

let us first explore the distribution of the features and also correlation between features and target.

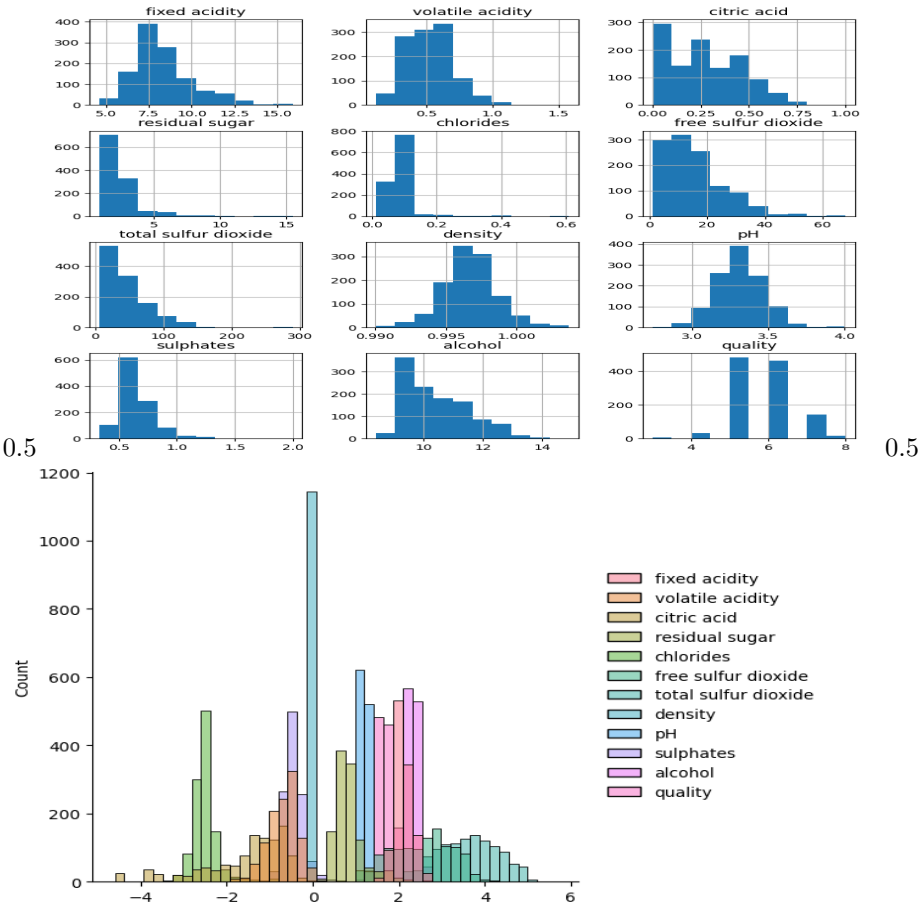

Figure 1: distribution of data

as we can see, the distributions are not normal and also the scales are different, so we can standardize the data before feeding into our network. the standardization technique is better to use with MLP, because it doesn't force the data to scale in a specific range and it may help the model to become more generalize.

2.2 Correlation

let see the correlation between features and the target.

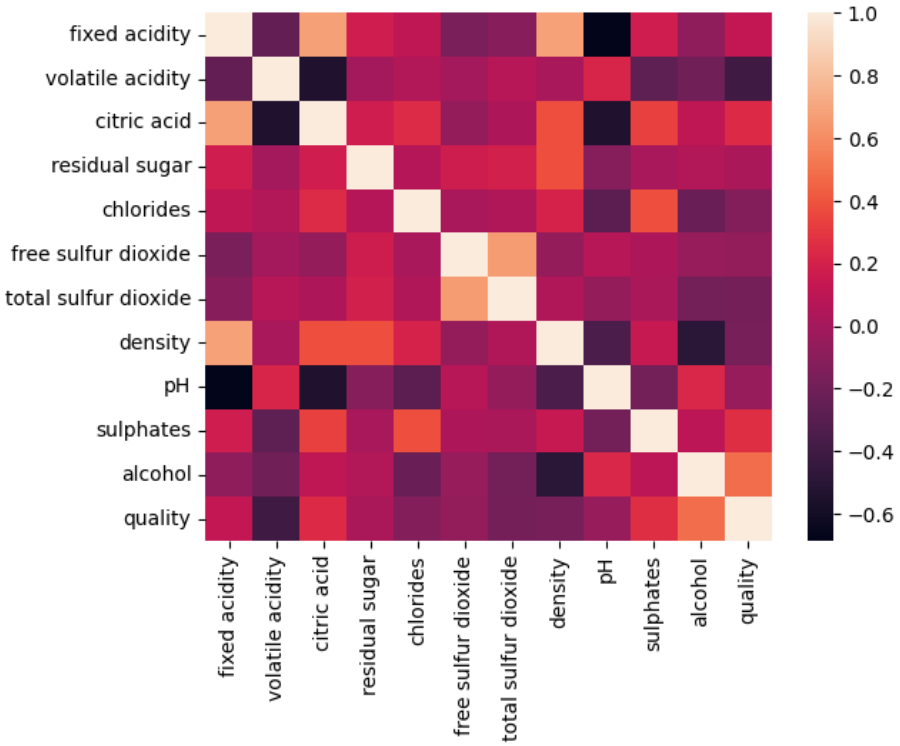

Figure 2: correlation heatmap

as we can see, some of features are not correlated, so applying dimension reduction techniques or a appropriate feature engineering(feature selection) could help in improving model performance.

2.3 Output classes

Let us see if the target feature is imbalance or not.

as we can see in the [figure 3], the classes are imbalance, and it may cause the model to not perform well on test and validation data, so the balancing techniques like oversampling and SMOTE, or assigning small weight to rare classes in loss function could improve the model performance.

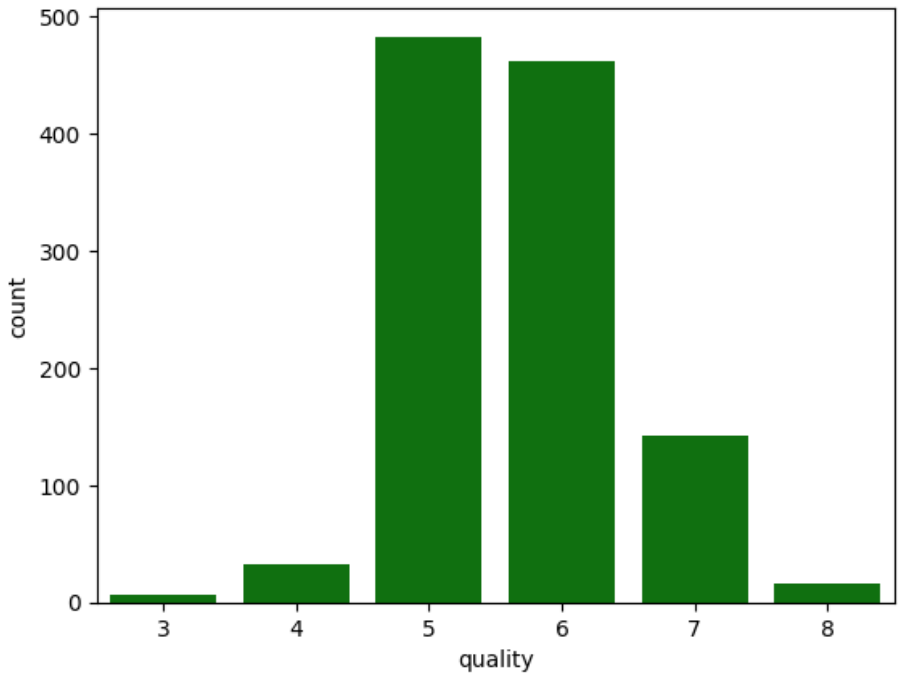

Figure 3: barplot of target labels

3 Design network architecture

We build MLP from scratch using numpy, now we need to design an architecture to classify the wine qualities, so we design a network consist of two hidden layers with 64 neuron each and ReLU as activation function and He weight initialization which is appropriate for using with ReLU. Since we have 11 input feature and 9 class as output(0 to 8), we have 11 input neurons and 9 output neurons, at last we apply a softmax to calculate the probability of an instance belongs to which class and evaluate the performance using Cross-entropy loss and gradient descent algorithm for optimization.

3.1 Hyperparameter tuning

We have tested different learning rates for the network to find the appropriate learning rate, check the figure below. as we can see, the higher values of learning rate, causes the loss reduction curve become less smooth, and very small learning rate slows down the learning process and causes the model to stock in local minima. so the best learning rate is 0.001.

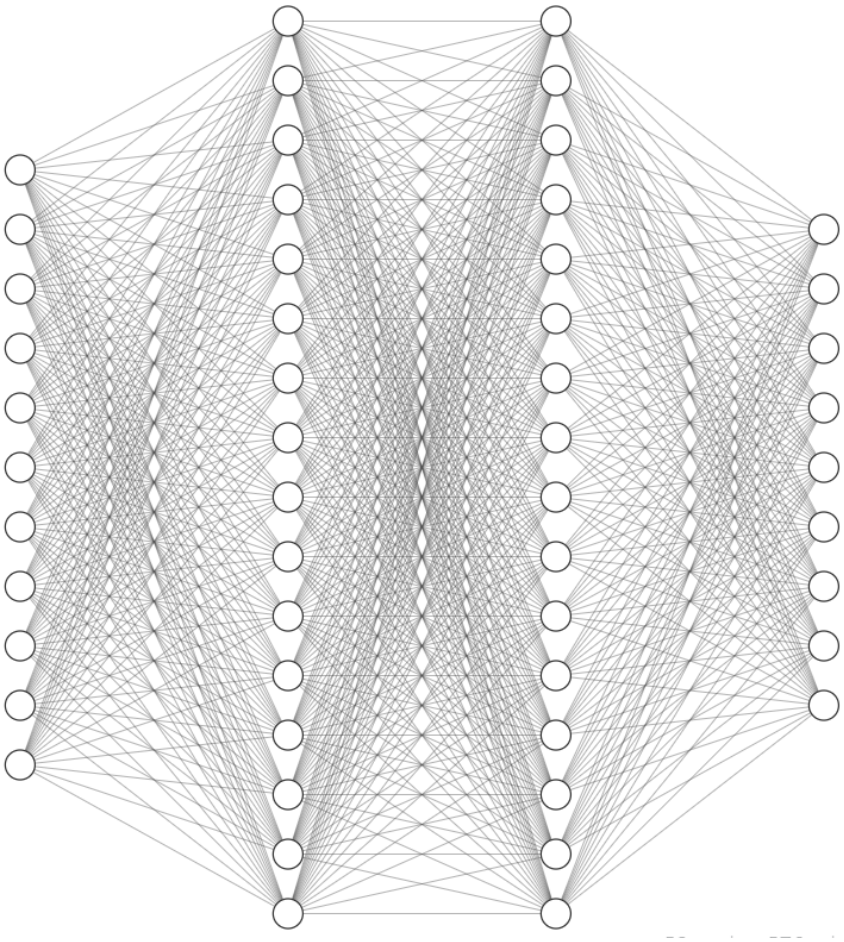

Figure 4: Network architecture

3.2 Network performance analysis

We trained our network over 1000 epochs and batch size 8, check the train and validation accuracy plot and loss reduction plot.

as we can see in the accuracy plot, the model overfited, this is because of imbalance labels, and the amount of data, so with increasing the amount of data and some balancing techniques, we can improve the performance.

we can see the test result in [Table 1].

3.3 Imbalance learning techniques

We perform SMOTE techniques to balance our classes, and trained our model with same architecture in previous section.

Table 1: Classification Metrics for Classes 3, 4, 5, 6, 7, and 8

Class	Precision	Recall	F1-score
3	0	0	0
4	0	0	0
5	0.68	0.61	0.65
6	0.52	0.62	0.56
7	0.50	0.43	0.46
8	0	0	0
accuracy	0.58	0.56	0.58

as we can see in the [Figure 8] the result is not improved, but model become more generalized and the percision, recall and f1-score of class 4 is not 0 anymore.

Table 2: Classification Metrics for Classes 3, 4, 5, 6, 7, and 8 for SMOTE

Class	Precision	Recall	F1-score
3	0	0	0
4	0.07	0.33	0.12
5	0.69	0.44	0.54
6	0.55	0.48	0.51
7	0.45	0.62	0.53
8	0	0	0
accuracy	0.58	0.47	0.51

4 Conclusion

As we saw in previous sections, the complexity of our network was enough to solve the problem, but model have become overfited on data, the main problem is lack of data, as we saw, the network only learned the labels like 5, 6, 7, and other labels are unknown! so by increasing the data, it could lead to a better performance on the validation and test.

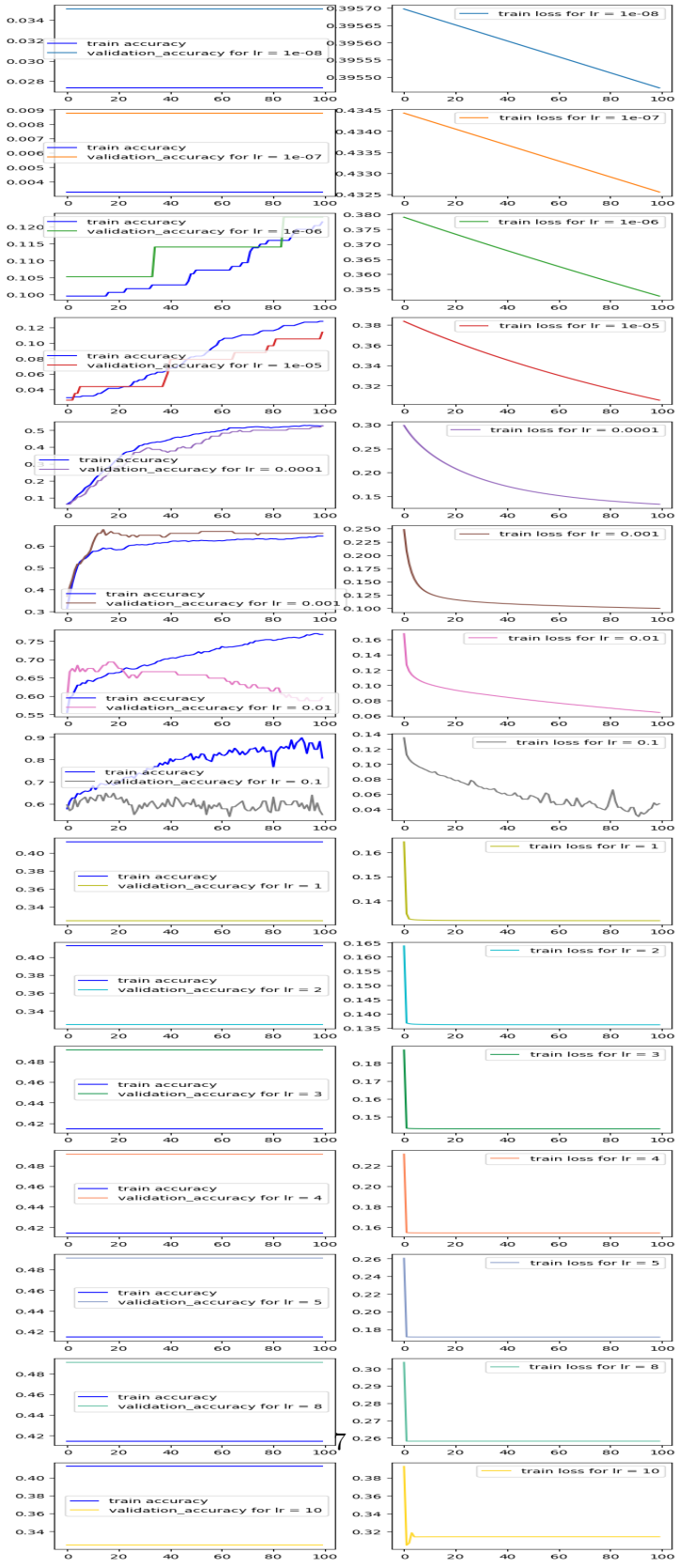

Figure 5: Learning rates

0.49

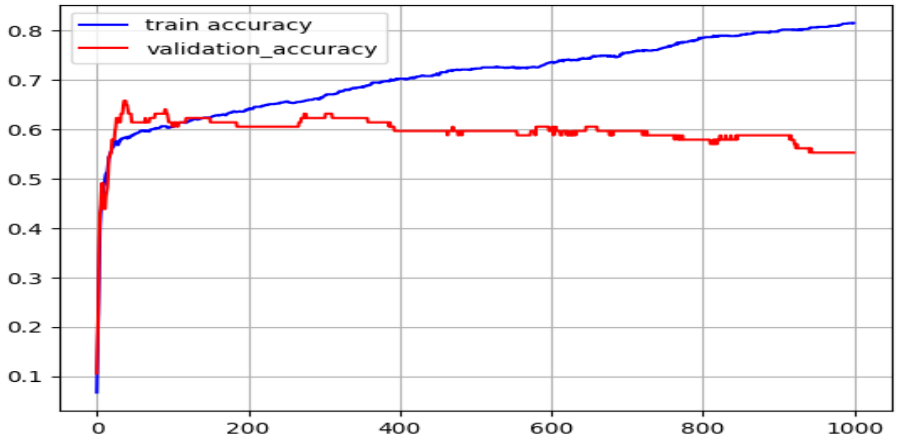

Figure 6: train and validation accuracy

0.49

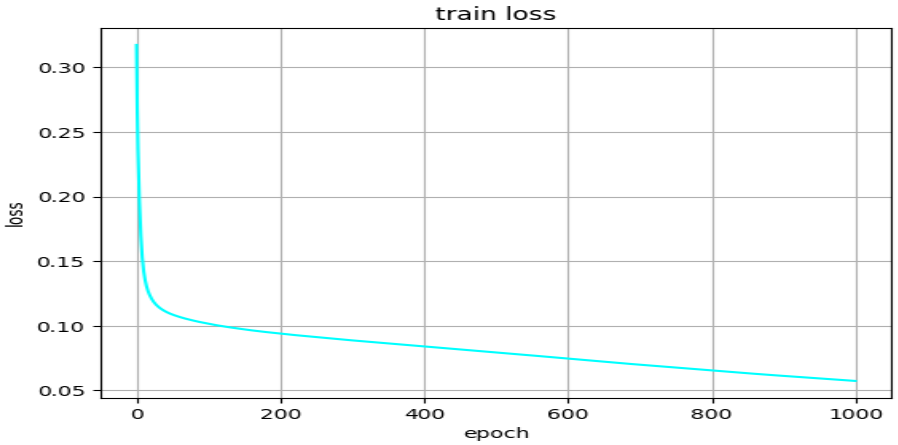

Figure 7: train loss

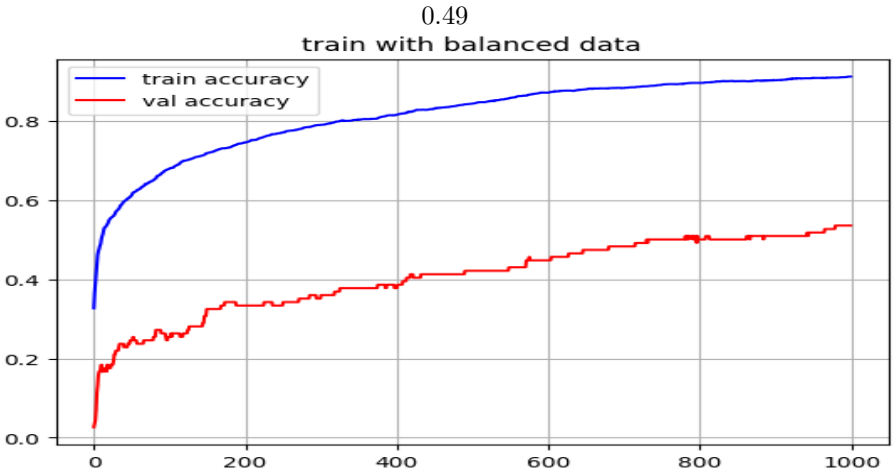

Figure 8: train and validation accuracy

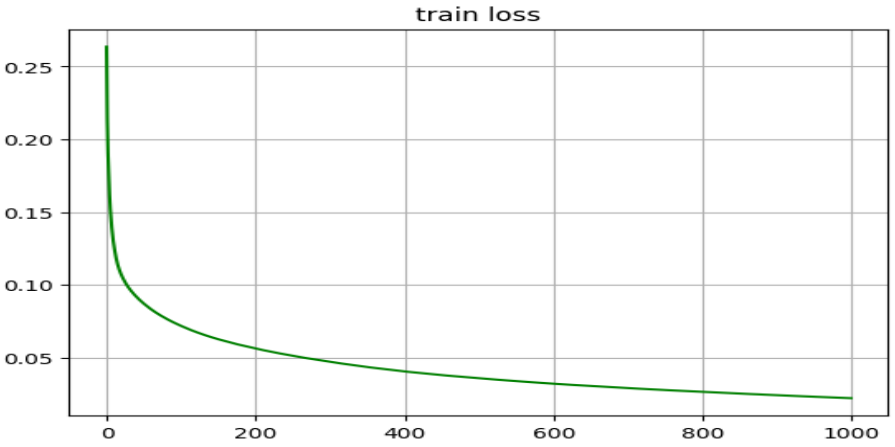

Figure 9: train loss