Neural Networks

Diabetes is influenced by a range of factors including genetics, lifestyle choices, environmental factors, and other health conditions. This complexity makes it a suitable candidate for analysis using neural networks, which are adept at handling multifaceted relationships within data.

Preprocessing

In order to train a neural network on the desired dataset, it is first important to preprocess the data. Just like on the previous methods, the data was balanced by selecting a desired number of observations for each class, and under sampling or over sampling each class (depending on if the original number of observations for that class was bigger or smaller than the established value). Afterwards, we used a min-max scaler to scale the data (this step is extra important on neural networks) and we split the data on to training, testing and validation datasets. Finaly, it was necessary to one-hot encode the categorical integer labels on to vectors (for example, the class 0 became [1, 0, 0].

Neural Network Parameters

Seeing as though our dataset has got a multiclass target, we decided to use a categorical cross-entropy loss function, combined with a SoftMax activation function on the output layer. We also used a dropout of 0.2.

In order to discover the parameters that best suited our model at hand, we decided to perform some hyperparameter tuning, using OpenCV's GridSearchCV. First, we searched through the following architectures:

First Layer	Second Layer
5	-
25	-
100	-
25	5

These values were chosen because we knew that the model probably wasn't too complex, but we decided to add at least one model with more than one layer, just to check if some non-linear representation was needed.

As to the rest of the parameters, these were the values we tried:

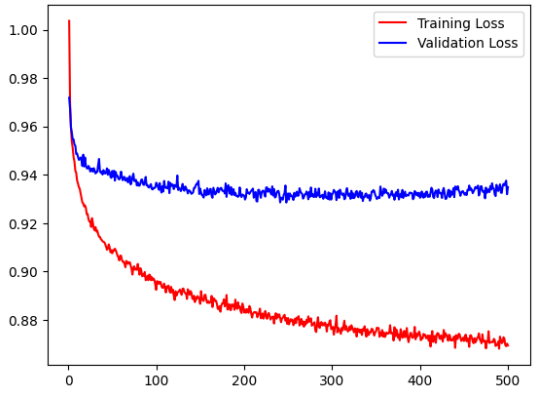
Parameter	First Candidate	Second Candidate	Third Candidate
Activation Function	ReLU	Sigmoid	Tanh
Learning Rate	$7e-3$	$1e-3$	$1e-4$
Batch Size	16	32	64

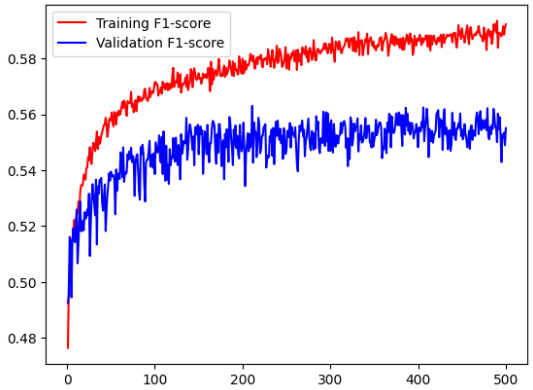
After running the code, we concluded that the best neural network within these parameters had one layer with 100 neurons, activated by a ReLU function, a learning rate of 0.001 and a batch size of 64. It produced the following validation metrics:

Validation Loss	Validation F1-score	Validation Sensitivity	Validation AUC	Validation Accuracy
0.9349	0.5550	0.3762	0.7406	0.5581

It is important to denote that, because our dataset is multiclass, the AUC is computed separately for each label and then averaged across labels. We decided to include the precision in the considered metrics because we are trying to predict the diagnose of a disease and, therefore, it makes sense to try to minimize the rate of false negatives as much as possible.

We also decide to plot the evolution of the F1-score and the loss along the epochs:





As we can see by the results shown, there is a point when our model starts overfitting, meaning that the validation metrics start to converge while the training metrics are still improving. This is why we have been and will be focusing on the validation metrics to determine what is our best model.

In addition, our neural network is not performing very well: the F1-score is not very good, suggesting that the model is not equally good at identifying classes. The AUC is decent, but it is the average of all three classes, so it doesn't disprove the fact that on of the classes could be performing worse. In order to investigate this, we decided to plot all three confusion matrixes:

```
Training Confusion Matrix Test Confusion Matrix Validation Confusion Matrix
[[4584 1071 1337]         [[632 164 219]        [[1214  374  405]
 [1355 3565 2086]          [228 467 354]         [ 429  849  667]
 [ 915 1335 4752]]         [137 298 561]]        [ 297  453 1252]]
```

As we can see, even though the dataset is balanced, the second value of the diagonal in each matrix is lowest. This makes sense, if we notice that the performance associated with the second class (prediabetic) is overall worse than the others. Therefore, we conclude that, for some reason, our model isn't very good at identifying whether people are prediabetic or not.

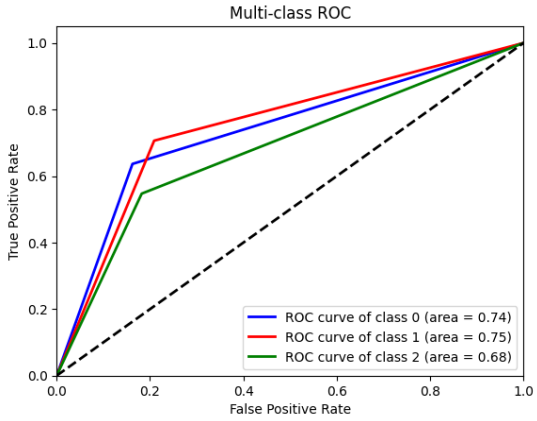
Improvements

We suspect this issue might be associated with the feature selection. To test this hypothesis and hopefully improve the model's performance, we decided to implement a lasso regularization on our neural network, since it can reduce the weight associated with certain features if it leads to a performance improvement. We obtained the following results:

Validation Loss	Validation F1-score	Validation Sensitivity	Validation AUC	Validation Accuracy
0.9613	0.5213	0.2977	0.7201	0.5256

Contrarily to what we expected, the results obtained were worse than before, so we decided to consider other methods. After an analysis of the graphs presented above regarding validation loss, we noticed that both values started to converge very early. If this only happened only with the validation loss, it could mean the model was overfitting and this logic wouldn't apply but because the training loss stops decreasing as well, we are led to believe that, for some reason, our neural network wasn't learning the data well and perhaps an increase in model complexity could help. So, we decided to try and create a neural network with four layers containing 150, 100, 50 and 25 neurons (respectively) and got the following results:

Validation Loss	Validation F1-score	Validation Sensitivity	Validation AUC	Validation Accuracy
0.8983	0.6243	0.5363	0.7944	0.6258



This was without a doubt the best model we managed to attain. Overall, the model has a reasonable ability to differentiate between classes (as indicated by the AUC), but it struggles with correctly labeling positive cases (low sensitivity) and overall precision and recall balance (moderate F1-score).

Unfortunately, even though we could improve our model a little bit, this method didn't perform as well as we expected it to do. There could be several reasons why your neural network is being outperformed by the other methods, in our opinion, one strong reason could be a high level of noise in the data that wouldn't affect other simpler models as much or problem suitability, seeing as though some problems are just more suited to methods like random forests.