Summary

ASSESSMENT 1

For this test, a binary classification task is described. Use Python and any open-source library to classify if the patient deceased using the HeartFailure.csv provided, containing thirteen clinical features from multiple patients. Provide a justification for all the choicesmade.

Dataset Description

- **Age: age of the patient (years)**
- **Anaemia: decrease of red blood cells or hemoglobin (Boolean)**
- **Blood pressure: measures artery blood pressure (mm Hg)**
- **Creatine phosphokinase (CPK): level of the CPK enzyme in the blood (mcg/L)**
- **Diabetes: if the patient has diabetes**
- **Ejection fraction: percentage of blood leaving the heart at each contraction(percentage)**
- **Platelets: platelets in the blood (kiloplatelets/mL)**
- **Sex: male or female**
- **Serum creatinine: level of serum creatinine in the blood (mg/dL)**
- **Serum sodium: level of serum sodium in the blood (mEg/L)**
- **Smoking: if the patient smokes or not**
- **time: follow-up period (days)**
- **[target] Death event: if the patient deceased during the follow-up period**

EXERCISE 1

Perform exploratory data analysis to analyze the dataset. Provide useful visualizations and conclusions from your findings.

EXERCISE 2

Perform the classification task and evaluate the performance of your models. How different parameters affect the performance of your model? What about feature importance?

EXERCISE 3

Create a simple Neural Network from scratch to perform the classification task using only
NumPy for building the model and the computational graph.

SOLUTION

Assessment 1

Exercise 1

Exploratory data analysis is performed in this exercise. First we load and check the HeatFailure.csv file. We explore the features of the dataset by printing:

1. The dataset.
2. The description of the dataset.
3. The shape of our dataset.
4. The column names of the dataset.
5. The number of unique values, of each feature column.
6. The unique values for sex and smoking feature.
7. The number of null values in our dataset.

My first observation was that we have 4 answers for ‘sex’ column and 6 for ‘smoking’ column. Some of the answers were wrong and needed to be replaced by the correct grammatical answers,to be the same with the other unique answers of the feature column. Also, I replaced with number 1 the answer ‘yes’ for the people who smoke and with number zero for the people who are not smoking. I did the same for ‘Female’ and ‘Male’ on the sex column. This replacement was done, because I wanted to make the string values into float values and make them suitable to fit the model. Next we need to clean the outliers of a dataset, which is defined as a value that is more than 3 standard deviations from the mean. Removing outliers from a pandas.DataFrame removes any rows in the DataFrame which contain an outlier. Outlier calculations are performed separately for each column with the help of “scipy.stats.zscore()” . Then I visualize the data with 5 pairplots, for more clear observation. My observations:

1. We can observe that age and anaemia doesn't affect each other.

2. People with creatinine_phosphokinase more than 1000 have probability to not die in our dataset.
3. The CPK(creatinine_phosphokinase) normal range for a male is between 39 – 308 U/L, while in females the CPK normal range is between 26 – 192 U/L and we can observe that people with normal CPK die.
4. A normal heart's ejection fraction may be between 50 and 70 percent and we can observe that only less than 5% of the sample has a normal percent number. Ejection fraction looks to affect the people from our dataset and takes the decision of their life.
5. People with more than 4 serum_creatinine die.
6. People with time more than 250 don't die.
7. We observe that people with normal ejection fraction is very difficult to die from anaemia.

After the pairplots and the observations above, I focused on 'ejection_fraction' and it's importance on our dataset. Next a heatmap, a relplot, a distplot and a catplot are visualized and make clear that 'ejection_fraction' affects the 'DEATH_EVENT' in our dataset.

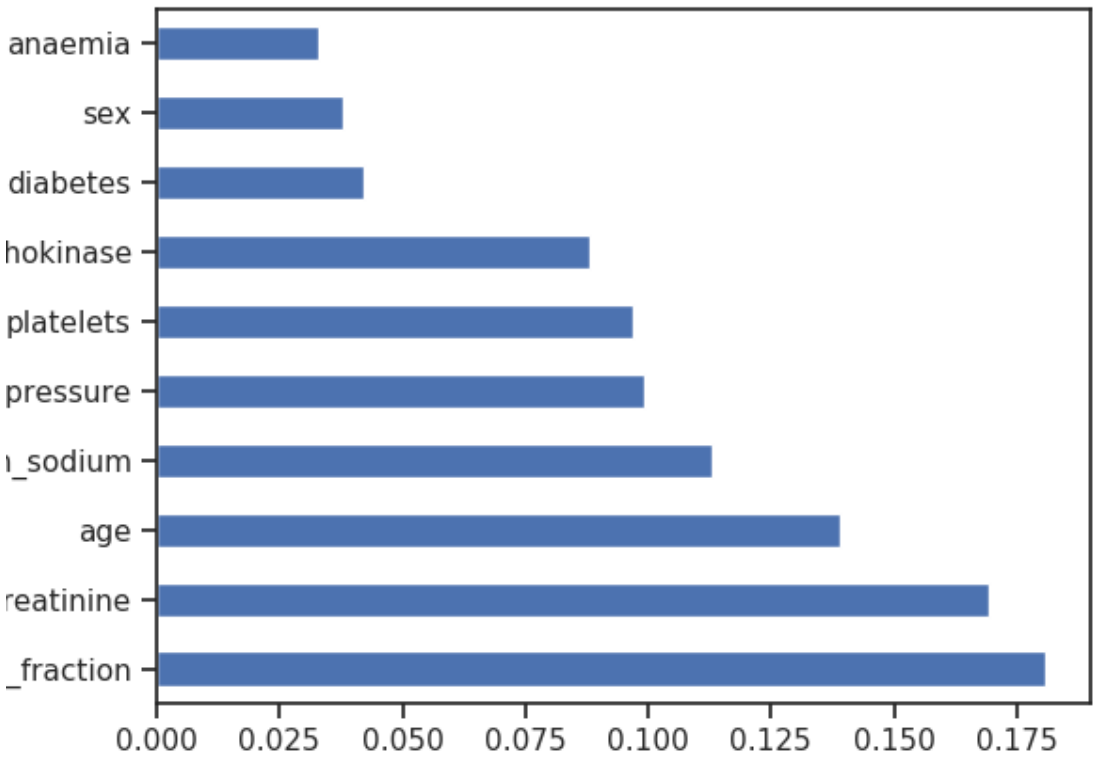
Exercise 2

In this part I will use 5 machine learning classifiers to evaluate the performance of our model:

1. Logistic Regression
2. Linear SVC
3. Decision Tree Classifier
4. Random Forest Classifier
5. Gaussian Naive Bayes

I use 'DEATH_EVENT' as a target column for parameter 'y' and the rest of our features for 'X' parameter. The 'y' parameter balanced data set is visualized. Then I define a dictionary with the performance metrics we need such as accuracy, precision, recall and f1-score for each classifier.
Next we have the models evaluation function with 'X' and 'y' as parameters and we perform cross-validation to each machine learning classifier. At the end, I create a dataframe with the model performance metrics scores and it's printed on the terminal screen as a result. The conclusion is that, when we run our code with different target

parameters, we observe that our model has better performance with other machine learning classifiers, depending on the target parameter each time.
Next, I use the ‘Extra Tree Classifier’, to extract the top 10 features of our dataset. The result is visualized and now we can see clearly that “eject_fraction” is the most important feature in our dataset.



Exercise 3

Our target column is again the “DEATH_EVENT”, I split the data into train and test sets for ‘X’ and ‘y_label’ . Then the shape of the datasets, which will be fed in our neural network are printed on the terminal screen. I created the class “NeuralNet”, which represents the simple neural network from scratch using only Numpy and matplotlib to visualize the results. The skeleton body consists of 13 functions, which represent the whole part of our neural network.The neural network above will have one hidden layer and a final output layer. The input layer will have 12 nodes because we have 12 features, excluding the target.

The hidden layer can accept any number of nodes, but I will start with 8, and the final layer, which makes the predictions, will have 1 node.
Below are represented each function separately:

1. During initialization, I created some variables to hold intermediate calculations. (__init__)
2. Initialize the weights and biases as random numbers. These weights are initialized from a uniform random distribution and saved to a dictionary called “params”. (init_weights)
3. The activation function I used is ReLU. ReLU (Rectified Linear Unit) is a simple function that compares a value with zero. That is, it will return the value passed to it if it is greater than zero, otherwise it will return zero. (relu)
4. The “dRelu” function, which makes values into zero if “x<=0” and values into one if “x>0”. (dRelu)
5. The sigmoid function takes a real number and squashes it to a value between 0 and 1. In other words, it outputs a probability score for every real number. (sigmoid)
6. I compare the value, and if it is zero, I replace it with an extremely small value (0.00000001). (eta)
7. The choice of the loss function is dependent on the task and for classification problems, we use cross-entropy loss. (entropy_loss)
8. In forward propagation I perform all the dot products and addition using the weights and biases I initialized earlier, calculate the loss by calling the entropy_loss function, save the calculated parameters and finally return the predicted values and the loss. These values will be used during backpropagation. (forward_propagation)
9. In the backpropagation function, first, I create a function to calculate the derivatives of the ReLU, then I calculate and save the derivative of every parameter with respect to the loss function. (back_propagation)
10. The “fit” function takes 2 parameters: X(input dataset) and y (labels). First, it saves the train and target to the class variable and then initializes the weights and biases by calling the init_weights function. Then, it loops through the specified number of iterations, performs forward and backpropagation, and saves the loss.

11. Predict function passes the data through the forward propagation layer and computes the prediction using the saved weights and biases.
12. The “acc” function calculates the accuracy between the predicted values and the truth labels. (acc)
13. The “Plot_loss” function which plots the loss curve. (plot_loss)

Next, I train the model with the “fit” function and after that I make my predictions with the “predict” function for “Xtrain” and “Xtest” sets separately. I made the same procedure with different parameters such as learning_rate and iterations. The accuracy for each procedure is printed on the terminal screen.We observe that train accuracy was increased with the second architecture.