**I. Pen-and-paper**

1. **a)   
   Activation Function:** **Loss Function:**

**Forward Propagation:**

**  
Backward Propagation:  
1.** Deltas:

****

**2.** Bias and Weight derivatives:



**3.** Update Bias and Weights:  
**4.** Forward Propagation with the new Bias and Weights:  


**b)** Mostly equal to 1a), however the difference is that the activation function of the **output layer** is softmax instead of hyperbolic tangent and the loss function is cross-entropy instead of squared error.

**Activation Function:** **Loss Function:**

**Forward Propagation:**

 **Backward Propagation:  
1.** Deltas:

We can see that is the only delta different from 1a), due to the fact that the others layers have the same activation function as before, they are the same

**2.** Bias and Weight derivatives:

  
**3.** Update Bias and Weights:  
**4.** Forward Propagation with the new Bias and Weights:  


**II. Programming and critical analysis**

1. Chart, treemap chart

   Description automatically generatedChart, treemap chart

   Description automatically generated  
   We can observe that the predictions with no early stopping are closer to the real values, which can be because:
   1. Early stopping made the algorithm stop in a local minimum, not allowing it to fully learn;
   2. Not all the data was used to train, which can diminish the quality of the model, especially if the distribution of the data wasn’t random.
2. Chart, box and whisker chart

   Description automatically generated  
   Four strategies that would minimize the error are:
   1. A good choice for the regularization value;
   2. Using early stopping that could reduce overfitting;
   3. Choosing a different algorithm that may be better at fitting the training data and predict better results, too complex may give overfitting, but too simple may be unable to adapt to the data;
   4. Choosing a different variant of the given data, it’s said to be the variant 8nm, but can be others that better adjust the given model.

**III. APPENDIX**

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.neural\_network import MLPClassifier, MLPRegressor

from sklearn.model\_selection import KFold, cross\_val\_predict

import matplotlib.pyplot as plt

from matplotlib import pylab

from scipy.io import arff

import numpy as np

Kfol = KFold(n\_splits=5, random\_state=0, shuffle=True)

#\*2

file = open("breast.w.arff", "r")

data, meta = arff.loadarff(file)

input = data[meta.names()[:-1]].tolist()

output = data["Class"].tolist()

for earlyStop in [False, True]:

    classifier = MLPClassifier(hidden\_layer\_sizes=(3,2), activation='relu', alpha=0.2, max\_iter=2000, early\_stopping=earlyStop)

    prevision = cross\_val\_predict(classifier, input, output, cv=Kfol)

    conf\_mat = confusion\_matrix(output, prevision)

    disp  = ConfusionMatrixDisplay(conf\_mat, display\_labels=['Benign','Malignant'])

    disp.plot()

    if earlyStop:

        plt.title("Early Stopping")

    else:

        plt.title("No Early Stopping")

plt.show()

#\*3

file.close()

file = open("kin8nm.arff", "r")

data, meta = arff.loadarff(file)

input = data[meta.names()[:-1]].tolist()

output = data["y"].tolist()

for alpha, graph in zip([0, 0.2], [1,2]):

    classifier = MLPRegressor(hidden\_layer\_sizes=(3,2), activation='relu', alpha=alpha, max\_iter=2000)

    classifier.fit(input, output)

    prevision = cross\_val\_predict(classifier, input, output, cv=Kfol)

    residues = np.subtract(output, prevision)

    plt.subplot(1, 2, graph)

    plt.boxplot(residues)

    if graph == 1:

        plt.title("No Regularization")

    else:

        plt.title("With Regularization")

plt.show()

**END**