**22CSCI33H**

**Artificial Neural Networks**

****

**Phase 2**

Rawan Mohamed 193529 **[SOM]**

Rahma Abdulla 196192 **[MLP]**

Aissatou Aida Oumarou 197346 **[RBF]**

Table of Contents

[Introduction 3](#_Toc121770202)

[Dataset 3](#_Toc121770203)

[MLP 3](#_Toc121770204)

[RBF 5](#_Toc121770205)

[SOM 6](#_Toc121770206)

[Analysis 7](#_Toc121770207)

[Comparison 8](#_Toc121770208)

[References: 8](#_Toc121770209)

# Introduction

We will be using three different neural networks to perform classification on a dataset and compare the results. For each network, we’ll also be trying out different variations to get the best results possible. The networks we’ll be comparing are MLP, RBF, and SOM. MLP and RBF have been proven to produce good results when it comes to classification, however, SOM is a supervised learning algorithm, and it performs quite differently than other neural networks, the results may be interesting to observe how such an algorithm can compare to classic one, and also how differently RBF and MLP perform on the same data.

# Dataset

The dataset we’ll be using is the JM1 dataset from NASA, it is a binary classification dataset for classifying the defects of software. It contains 22 features describing the properties of the code of the software and its complexities. The target is binary, either defective or not. We preprocessed the data separately before using it on any network as it had few null rows which we removed, and it had class imbalance which we handled by using the upsampling method and produced a new dataset that has balanced class ratios. We then used the new dataset to work on each of the networks and try to obtain the best results.

# MLP

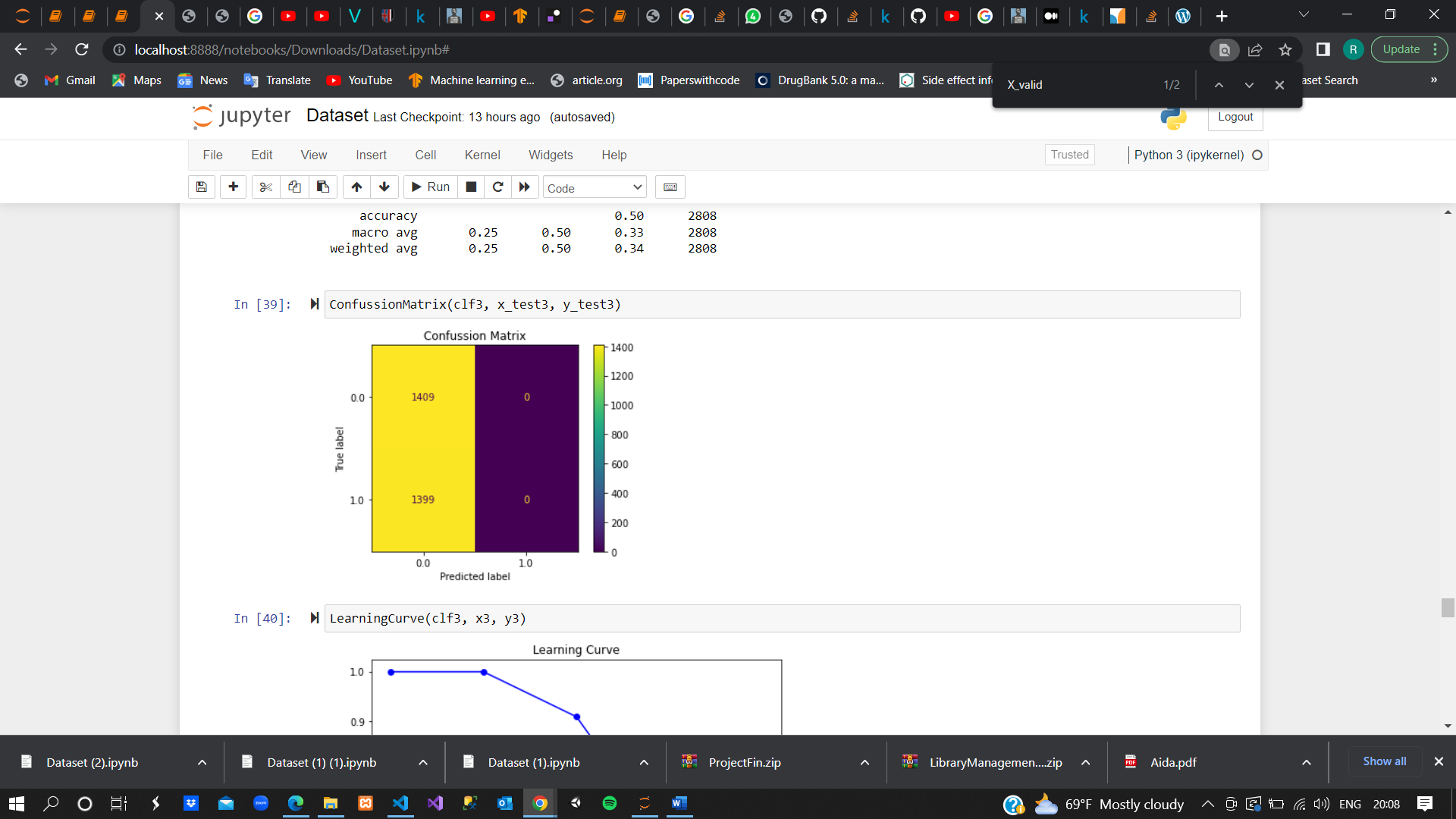
The most popular and widely used kind of neural network is the multilayer perceptron, the signals are propagated in a unidirectional form; from the input to the output without having any loops. The output of each layer is the input for the next layer so that the output of a layer affects the next layer but not itself [2]. A feedforward network that has one input, one output layers and one or more hidden layer(s), each neuron must include non-linear activation function, each layer must work with the same activation function [1]. In the data training the MLP uses the backpropagation algorithm, the MLP has a lot of applications one of the most popular applications is the classification [2].

I started by splitting the data to x and y then to x\_train3, x\_test3, y\_train3, y\_test3, then I implemented 2 functions that will help me in plotting the confusion matrix and the learning curve easily, then started implementing my 3 MLP models.

I tried 3 different MLP models the first one with relu activation function, 5 hidden layers with number of neurons = 6, the second MLP model with 4 hidden layers with number of neurons= 5, 10, 15 and 20 and tanh activation function and finally MLP model with 8 hidden layers with number of neurons = 20, 20, 20, 20, 20, 20, 20 and 20 and logistic as the activation function

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The confusion matrix of the 3 models shows that the first model had 993 true positives, 881 true negative, 416 false positive and 518 false negative, the second model showed 1343 true positives, 156 true negatives, 66 false positive and 1243 false negative, finally the third model showed 1409 true positives, 0 true negatives, 1399 false positive and 0 false negative. So the first model is the best model.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The learning curve of the 3 models also shows that the first model is the best model

A screenshot of a computer

Description automatically generated

Also, when comparing the accuracies, the first model had the highest accuracy of 0.69318937

# RBF

The RBF network is composed of three layers (input, hidden, output) with the hidden layer consisting of the activation function together with the Gaussian function. The algorithm was done in 5 stages:

* Defining the number of hidden neurons
* Positioning these centers using K-means
* Calculating ɕ (sigma, activation function) = d/√2M in which d= maximum distance between two hidden neurons and M is the number of hidden neurons
* Calculate actions of RBF: output of Gaussian function  with x= input
* Training of Gaussian function output

The accuracy was calculated using different methods as shown below.

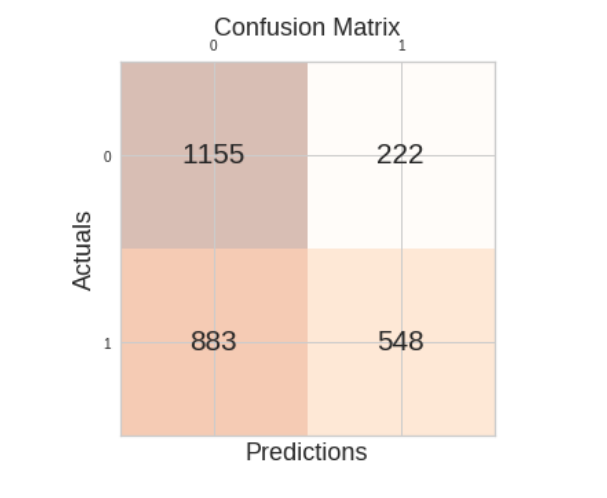


Figure 1. Confusion Matrix

548 of the testing datasets were classified as TP (true positive => classified correctly as positive), 1155 as TN (true negative => classified correctly as negative), 222 as FN (false negative => incorrectly classified as negative as they are positive), and 883 false positives.

The learning curve of high bias can be seen below.

Chart, line chart

Description automatically generated

Figure 2. The learning curve of K means clustering used for RBF

# SOM

After regular preprocessing like splitting and normalizing the data, a function to add classification properties to the self-organizing map was created, it classifies each instance into the class label of the winning neuron. After that, the first SOM classifier was made with a map size of 100x100, a learning rate of 0.7, and a triangle neighborhood function. This was trained for 10000 iterations and resulted in a quantization error of **0.273** and a topographic error of **0.261**. It was also evaluated using classification metrics, like accuracy, f-score, precision, recall, and more as shown below.

We can see it performed classification quite well and resulted in both good accuracy and f-score.

Chart, line chart

Description automatically generated

Figure 3. SOM classifier one results

The second SOM was made with the same map size of 100x100, and the same learning curve, however, this time a gaussian neighborhood function was used instead. This resulted in a quantization error of **0.466** and a topologic error of **0.0500**. Below is its classification results.

Chart, line chart

Description automatically generated

Figure 4. SOM classifier two results

## Analysis

It is observed that the classifier done with a triangle neighborhood function has better results when it comes to quantization error and classification metrics like accuracy, f-score, etc. However, the one using a gaussian neighborhood got a much better topographical error. This tells us that both functions provide different properties to their SOM. The triangle function gives an accurate representation of the data and is better at classifying it. However, the gaussian function, although the most commonly used one, didn’t really perform well on this data, but instead constructed a map that does a great job at preserving the topology of the data, hence every low topographical error.

# Comparison

|  |  |  |
| --- | --- | --- |
| **Best Model** | **Accuracy** | **F-Score** |
| **MLP** | 0.69318937 | 0.67 |
| **RBF** | 0.6064814814814815 | 0.498 |
| **SOM** | 0.770085 | 0.7696 |

# References:

[1] S. HAYKIN, “Multilayer Perceptron/ 4.1,” in *Neural networks and learning machines*, 3rd ed., vol. 10, Upper Saddle River ; Boston ; Columbus etc.: Pearson Education, 2009, pp. 123–124.

[2] V. E. Balas and N. E Mastorakis, “(PDF) Multilayer Perceptron and Neural Networks - ResearchGate,” *ReseacrchGate*, 01-Jul-2009. [Online]. Available: https://www.researchgate.net/publication/228340819\_Multilayer\_perceptron\_and\_neural\_networks. [Accessed: 20-Oct-2022].