BLOOD TRANSFUSION SYSTEM

A PROJECT REPORT

Submitted as a Jth Component for the course

B.TECH (I.T)

by

TEAM MEMBERS:

NAME	REG. NO.
AYUSH MALRA	16BIT0272
ROHIT BURA	16BIT0169
PRAKHAR RUNGTA	16BIT0160
HEMANT KUMAR SINGH	16BIT0227
APOORV GUPTA	16BIT0134

Under the Guidance of
VALARMATHI B

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering

Oct,2018

Contents:

ABSTRACT:

Blood Donation and Blood Transfusion Services play a vital role in saving people's lives. It is a known concern that the blood donation process usually consumes a lot of time and effort from both donors and medical staff. This issue is present, mainly due to the absence of a concrete and precise information system. Currently the blood transfusion sector lack efficient tools for information and knowledge management, and process automation. The exchange of knowledge isn't potential among the blood centres as they are not systematically regulated under the present legislation, because the weak operational structure of insertion authorities restricted their reaching into the system. If such a system is implemented it could have the ability to allow donors and blood donation centres to communicate efficiently and collaborate with each other to minimize the time and effort required for the blood donation process. Besides, most blood banks work in isolation and are not integrated with other blood donation centres, which affect and degrade the quality of overall blood donation and blood transfusion services. Blood transfusion is mostly the method of receiving blood or blood merchandise into one's circulation intravenously. Transfusions are used for various medical conditions to replace lost components of the blood. By this system we have come to know whether a person donated blood or not. The output is in binary form. 1 stands for donating blood and 0 is for not donating blood.

OBJECTIVE:

i. To teach the neural network to predict whether a blood donor gave blood previously based on characteristics that are given as input parameters.

Input parameters are:

R (Recency - months since last donation),
F (Frequency - total number of donation),
M (Monetary - total blood donated in c.c.),
T (Time - months since first donation)

PROPOSED METHODOLOGY:

i. The first thing we need is a data set.
ii. To deal with this classification problem, the data set needs to be normalized first.
iii. The type of neural network that will be used in this experiment is multi-layer perceptron with backpropagation.

LITERATURE REVIEW:

Blood transfusion is a life-saving therapy, and its provision and safety are often taken for granted in the industrialized world. The study of the factors that influence the behaviour of blood donors has been conducted extensively due to the significant impact of blood shortages to the survival of patients. Mostafa introduced five factors leading to the study of blood donor behaviour. These factors are (1) altruistic values, (2) perceived risks of blood donation, (3) blood donation knowledge, (4) attitudes toward blood donation, and (5) intention to donate blood. In Thailand, there was a study in various reasons of factors that stimulate the altruistic value in donating blood. People who have positive attitude are usually considerate of mankind and altruistic. Perception of donors about risks in blood donating is one of the main reasons affecting a decision-making process in blood donation because it influences donor to have negative feeling toward blood donation. Moreover, the fear of infection directly affects donors' intention to donate blood presuming that those who donate blood may likely be infected more than those who do not. On the other hand, those who donated blood in the past tend to have positive thought about blood donation and may donate more in the future because they understand processes of blood donors screening better than the first-time donors. Blood donors usually have much better positive attitude and good intention toward blood donation than those who have not donated blood before. The intention to donate blood is one of the most important factors that can be used to predict behaviours in donating blood. These aforementioned five factors are used in developing questionnaire to conduct individuals' opinions. This information is used for blood donor classification analysis in Thailand.

Efficacy of red blood cell transfusion in the critically ill: A systematic review of the literature:

- The literature survey is done to determine the association between red blood cell transfusion, mortality and morbidity in very high risk hospitalized patients.
- Red blood cell transfusions are very common in trauma, intensive care unit and surgical patients. But the haematocrit that should be maintained in every patient as the risks of further transfusion of the red blood cells outweigh the benefits remains unclear.
- Quite a few observational studies were analysed.
- The risks of red blood cell transfusion far outweigh the benefits, the risk was found to be neutral in two studies and the benefits outweighing the risks in a subgroup of another single study.
- In all the studies that were conducted blood transfusion was an independent risk factor for infection.
- Red blood cell transfusions increase the risk of having multi-organ dysfunction syndrome as well as acute respiratory distress syndrome.

Despite the inherent limitations in the analysis of the cohort studies, the analysis suggests that in adults, trauma, intensive care unit, and surgical patients the red blood cell transfusions are associated with an increase in morbidity and mortality and hence, the current transfusion practices may require a re-evaluation. The benefits and risks of

the red blood cells transfusion should be assessed in all the patients before the transfusion.

REQUIREMENTS SPECIFICATIONS:

Hardware Requirements:

	Hardware	Required Configuration
	Processor	Pentium –V or Higher
	RAM	512 MB (Minimum)
	Hard Disk	128MB or more
	Keyboard	Standard Windows Keyboard
	Mouse	Standard
	Monitor	SVGA or Higher

	Software	Version
S. No.		
1	Operating System	Windows 7 or Higher
2	Compiler	NetBeans (Java)
		Spyder (Python)
3	Text file (Dataset)	Notepad for Java Excel Csv for python

DATASET:

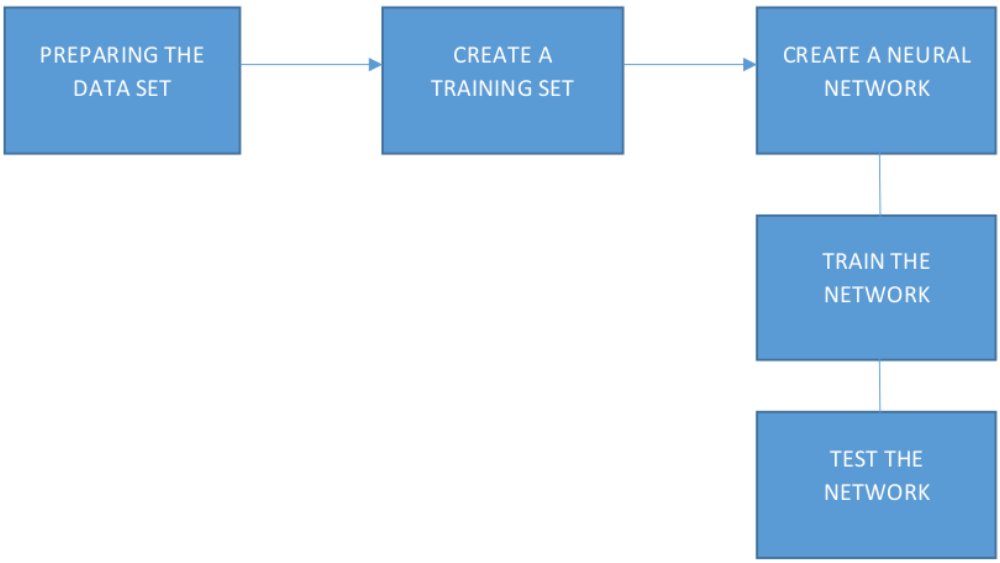
To demonstrate the RFMTC marketing model (a modified version of RFM), this study adopted the donor database of Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The center passes their blood transfusion service bus to one university in Hsin-Chu City to gather blood donated about every three months. To build a FRMTC model, we selected 748 donors at random from the donor database. These 748 donor data, each one included R (Recency - months since last

donation), F (Frequency - total number of donation), M (Monetary - total blood donated in c.c.), T (Time - months since first donation), and a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood).

Dataset Link:- https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

Data Set Characteristics:	Multivariate	**Number of Instances:**	748	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	5	**Date Donated**	2008-10-03
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	161657

BLOCK DIAGRAM:



Architecture:

A multilayer perceptron (MLP) is a class of feedforward artificial neural system. A MLP comprises of no less than three layers of hubs. Aside from the info hubs, every hub is a neuron that uses a nonlinear actuation work. MLP uses a regulated learning system called backpropagation for preparing. Its various layers and non-straight actuation recognize MLP from a direct perceptron. It can recognize information that isn't directly detachable.

A MLP is a system of straightforward neurons called perceptron. The perceptron figures a solitary yield from different genuine esteemed contributions by framing a straight blend as indicated by its info weights and after that perhaps putting the yield through some nonlinear initiation work. Numerically this can be composed as

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

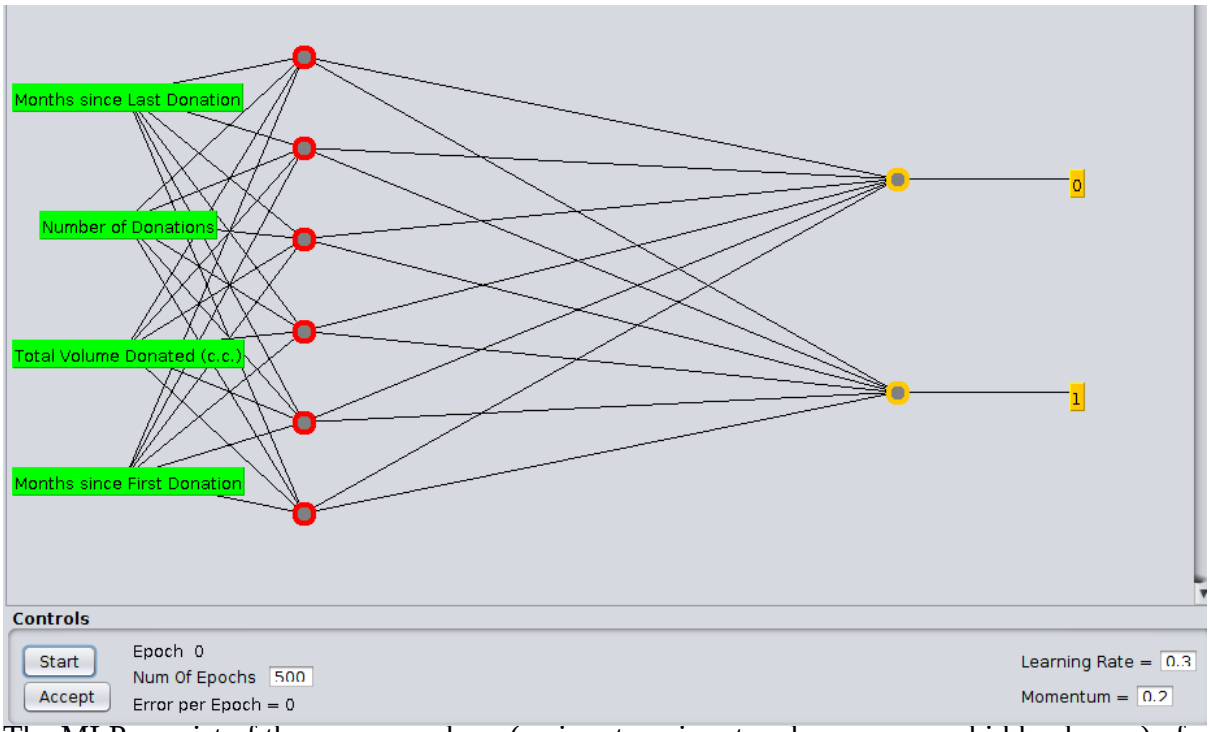
where $\mathbf{w}$ denotes the vector of weights, $\mathbf{x}$ is the vector of inputs, b is the bias and φ is the activation function.

A solitary perceptron isn't exceptionally valuable on account of its constrained mapping capacity. Regardless of what enactment work is utilized, the perceptron is just ready to speak to a situated edge like capacity. The perceptron can, be that as it may, be utilized as building pieces of a bigger, substantially more handy structure. A regular multilayer perceptron (MLP) arrange comprises of an arrangement of source hubs shaping the information layer, at least one shrouded layers of calculation hubs, and a yield layer of hubs. The info flag proliferates through the system layer-by-layer.

The computations performed by such a feedforward network with a single hidden layer with nonlinear activation functions and a linear output layer can be written mathematically as

$$\mathbf{x} = \mathbf{f}(\mathbf{s}) = \mathbf{B}\boldsymbol{\varphi}(\mathbf{A}\mathbf{s} + \mathbf{a}) + \mathbf{b}$$

where $\mathbf{s}$ is a vector of inputs and $\mathbf{x}$ a vector of outputs. $\mathbf{A}$ is the matrix of weights of the first layer, $\mathbf{a}$ is the bias vector of the first layer. $\mathbf{B}$ and $\mathbf{b}$ are, respectively, the weight matrix and the bias vector of the second layer. The function $\boldsymbol{\varphi}$ denotes an elementwise nonlinearity.



The MLP consist of three or more layer(an input, an input nad one or more hidden layers) of non-linearly activating nodes making it a deep neural network. Since MLPs are fully connected, each layer in one node connects with a certain weight to every node in the following layer.

Implementation:

#Multilayer Perceptron-16BIT0227(HEMANT KUAMR SINGH)

Java Code

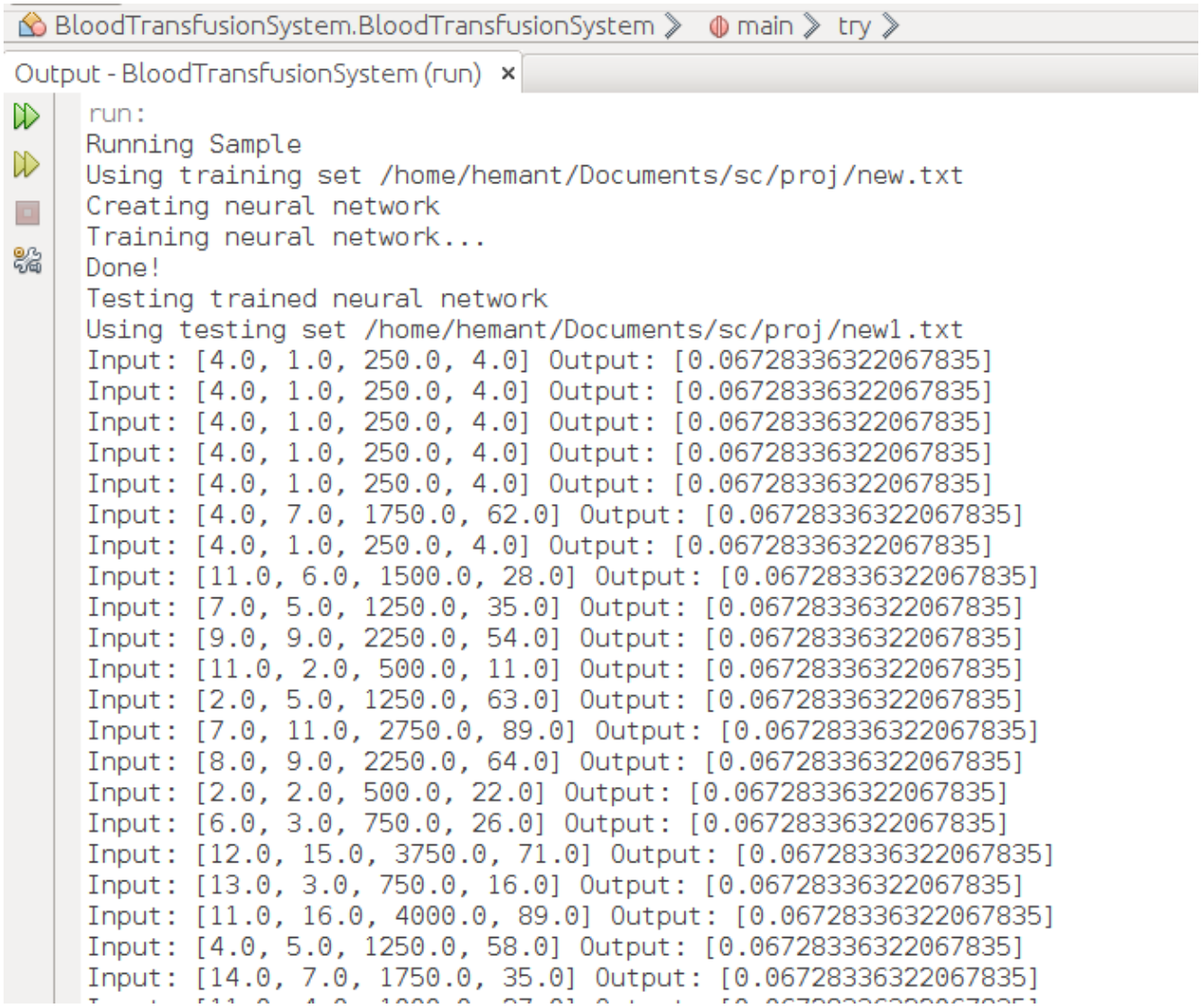
BloodTransfusionSystem.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BloodTransfusionSystem;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Arrays;
import org.neuroph.core.NeuralNetwork;
import org.neuroph.core.data.DataSet;
import org.neuroph.core.data.DataSetRow;
import org.neuroph.nnet.MultiLayerPerceptron;
import org.neuroph.nnet.learning.MomentumBackpropagation;
import org.neuroph.util.TrainingSetImport;
import org.neuroph.util.TransferFunctionType;
/**
 *
 * @author hemant
 */
```

```
public class BloodTransfusionSystem {
   public static void main(String[] args){
      String trainingSetFileName = "/home/hemant/Documents/sc/proj/new1.txt";
      String testingSetFileName = "/home/hemant/Documents/sc/proj/new.txt";
         int inputsCount = 4;
         int outputsCount = 1;
         System.out.println("Running Sample");
         System.out.println("Using training set " + trainingSetFileName);
         // create training set
         DataSet trainingSet = null;
      DataSet testingSet = null;
         try
         { trainingSet = TrainingSetImport.importFromFile(trainingSetFileName, inputsCount,
         outputsCount, ",");
         }
         catch (FileNotFoundException ex)
         {
         System.out.println("File not found!");
         }
         catch (IOException | NumberFormatException ex)
         {
         System.out.println("Error reading file or bad number format!");
         }
         // create multi layer perceptron
         System.out.println("Creating neural network");
         MultiLayerPerceptron neuralNet=new
MultiLayerPerceptron(TransferFunctionType.SIGMOID, 4, 6, 1);

         // set learning parametars
         MomentumBackpropagation
learningRule=(MomentumBackpropagation)neuralNet.getLearningRule();

         learningRule.setLearningRate(0.2);
         learningRule.setMomentum(0.3);
         // learn the training set
         System.out.println("Training neural network...");
         neuralNet.learn(trainingSet);
         System.out.println("Done!");
         // test perceptron
         System.out.println("Testing trained neural network");
      try
         { testingSet = TrainingSetImport.importFromFile(testingSetFileName, inputsCount,
         outputsCount, ",");
         }
         catch (FileNotFoundException ex)
         {
         System.out.println("File not found!");
         }
```

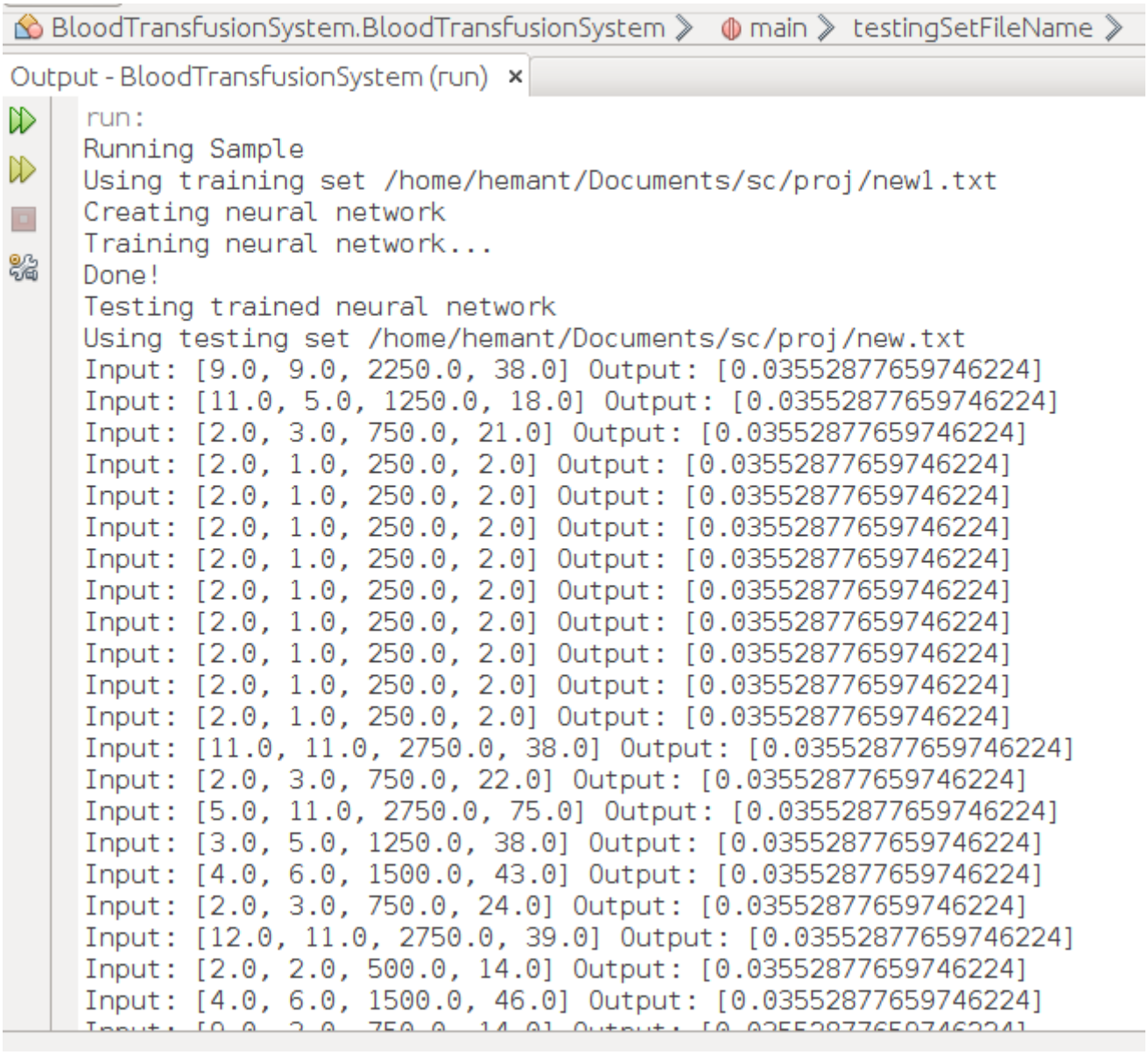
```
            catch (IOException | NumberFormatException ex)
            {
            System.out.println("Error reading file or bad number format!");
            }
        System.out.println("Using testing set " + testingSetFileName);
            testBloodTranfusionSystem(neuralNet, testingSet);
}
    public static void testBloodTranfusionSystem(NeuralNetwork nnet, DataSet dset) {
    for (DataSetRow trainingElement : dset.getRows())
        {
        nnet.setInput(trainingElement.getInput());
        nnet.calculate();
        double[] networkOutput = nnet.getOutput();
        System.out.print("Input: " + Arrays.toString(trainingElement.getInput()));
        System.out.println(" Output: " + Arrays.toString(networkOutput));
        }
    }
}
```

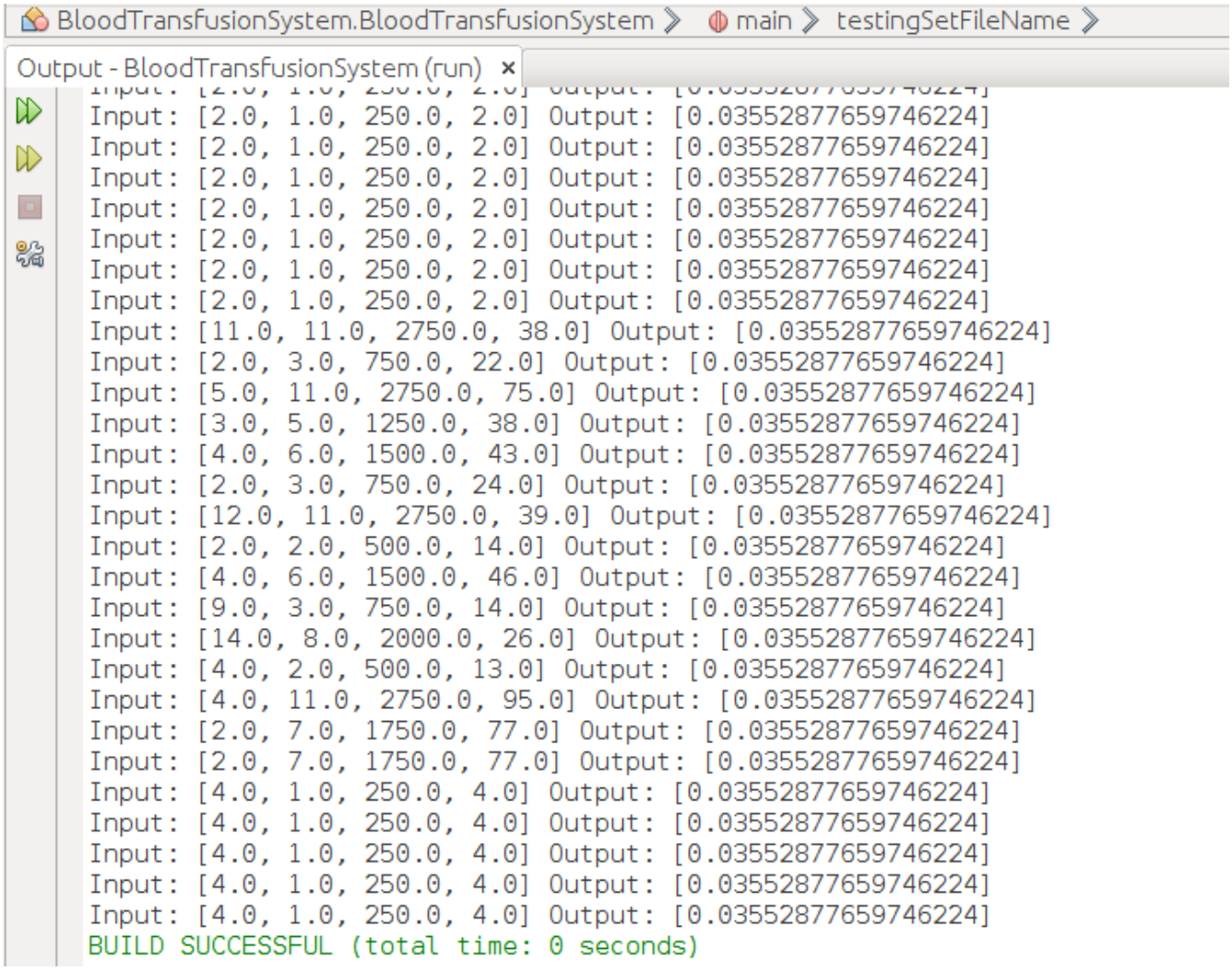
Output 75%Testing 25%Training

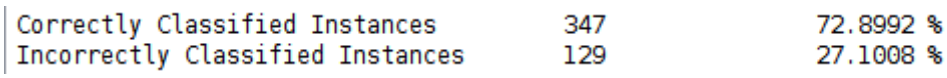


```
Correctly Classified Instances          91               91      %
Incorrectly Classified Instances         9                9      %
```

Output Training 75% Testing 25%







Artificial Neural Network-16BIT0169(ROHIT BURA)

Installing Theano

pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git# Installing Tensorflow

Install Tensorflow from the website:

https://www.tensorflow.org/versions/r0.12/get_started/os_setup.html

Installing Keras

pip install --upgrade keras

Part 1 - Data Preprocessing

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('dataset_soft_c.csv')
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Part 2 - Now let's make the ANN!
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
# Initialising the ANN
classifier = Sequential()# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 3, init = 'uniform', activation = 'relu', input_dim =
4))
# Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

```
# Part 3 - Making the predictions and evaluating the model
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.47)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
# Find all results
TP=cm[1][1]
TN=cm[0][0]
FP=cm[0][1]
FN=cm[1][0]
print(TP,TN,FP,FN)
Accuracy=(TP+TN)/(TP+TN+FP+FN)
Sensitivity=TP/(TP+FN)
Specificity=TN/(TN+FP)
Precision=TP/(TP+FP)
print("for Neural Network :","\n")
print("Accuracy\tSensitivity\tSpecificity\tPrecision","\n")
print(f"{Accuracy:.4f}","\t","\t",f"{Sensitivity:.4f}","\t",f"{Specificity:.4f}","\
t",f"{Precision:.4f}")
```

Lvq code-16BIT0117(APOORV GUPTA):

```
from math import sqrt
from random import randrange
from random import seed

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
	distance = 0.0
	for i in range(len(row1)-1):
		distance += (row1[i] - row2[i])**2
	return sqrt(distance)

# Locate the best matching unit
def get_best_matching_unit(codebooks, test_row):
	distances = list()
	for codebook in codebooks:
		dist = euclidean_distance(codebook, test_row)
		distances.append((codebook, dist))
	distances.sort(key=lambda tup: tup[1])
	return distances[0][0]

# Create a random codebook vector
def random_codebook(train):
	n_records = len(train)
	n_features = len(train[0])
	codebook = [train[randrange(n_records)][i] for i in range(n_features)]
	return codebook

# Train a set of codebook vectors
def train_codebooks(train, n_codebooks, lrate, epochs):

	codebooks = [train[0],train[5]]
	for epoch in range(epochs):
		rate = lrate * (1.0-(epoch/float(epochs)))
		sum_error = 0.0
		for row in train:
			bmu = get_best_matching_unit(codebooks, row)
			for i in range(len(row)-1):
				error = row[i] - bmu[i]
				sum_error += error**2
				if bmu[-1] == row[-1]:
					bmu[i] += rate * error
				else:
					bmu[i] -= rate * error
		print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, rate, sum_error))
	return codebooks
```

```
# Make a prediction with codebook vectors
def predict(codebooks, test_row):
	bmu = get_best_matching_unit(codebooks, test_row)
	return bmu[-1]


def accuracy_metric(actual, predicted):
	correct = 0
	for i in range(len(actual)):
		if actual[i] == predicted[i]:
			correct += 1
	return correct / float(len(actual)) * 100.0



# Test the training function.

import pandas as pd
import numpy as np
df = pd.read_csv('g:/data/blood-donation.csv')

x1=np.array(df["'Months since Last Donation"])
x2=np.array(df['Number of Donations'])
x3=np.array(df['Total Volume Donated (c.c.)'])
x4=np.array(df['Months since First Donation'])
y=np.array(df['Made Donation in March 2007'])
x_train=[]
x_test=[]



#20% train data and 80% test data
for i in range(int(len(x1)/4)):
   x_train.append([x1[i],x2[i],x3[i],x4[i],y[i]])

for j in range(int(len(x1)/4),int(len(x1)-10)):
   x_test.append([x1[j],x2[j],x3[j],x4[j],y[j]])

# for list conversion print df.iloc[:, 0].tolist()
u=x_train[0]
v=x_train[5]
print(u,v)

#train dataset
learn_rate = 0.3
n_epochs = 10
n_codebooks = 2
```

codebooks = train_codebooks(x_train, n_codebooks, learn_rate, n_epochs)
print('Codebooks: %s' % codebooks)

#predicting the values
predictions = list()
actual=[]
for row in x_test:
    output = predict(codebooks, row)
    predictions.append(output)
print(predictions)

#calculating accuracy and printing the accuracy by comapring actual and prediction values
actual = [row[-1] for row in x_test]
print("\n\n\naccuracy:",accuracy_metric(actual,predictions))

Ouput:

```
   ...: print("\n\n\naccuracy:",accuracy_metric(actual,predictions))
[2, 50, 12500, 98, 1] [4, 4, 1000, 4, 0]
>epoch=0, lrate=0.300, error=23418513686.904
>epoch=1, lrate=0.270, error=7967053684.481
>epoch=2, lrate=0.240, error=7096332180.259
>epoch=3, lrate=0.210, error=5992577254.538
>epoch=4, lrate=0.180, error=4755733629.393
>epoch=5, lrate=0.150, error=3533908820.316
>epoch=6, lrate=0.120, error=2463593300.591
>epoch=7, lrate=0.090, error=1638421669.026
>epoch=8, lrate=0.060, error=1082623713.369
>epoch=9, lrate=0.030, error=748978306.455
Codebooks: [[16.35942795907382, 186.7155170329942, 46678.879258248555, 376.8448525585374, 1], [2.7141129799413912, 1.8232138242710634,
455.803456067766, 12.255675837671687, 0]]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]


accuracy: 82.22748815165876

In [2]:
```

Accuracy of 82.27% if found using lvq.

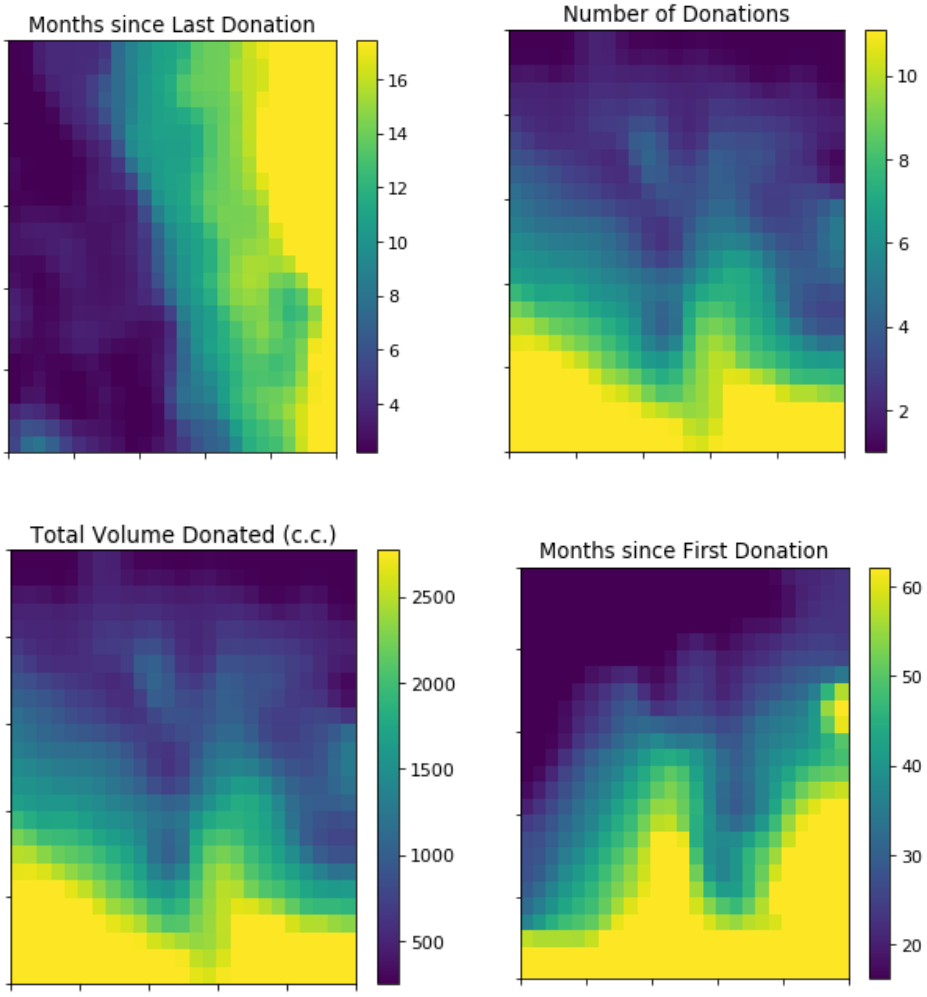
#KSOM-16BIT0160(PRAKHAR RUNGTA):

Code:

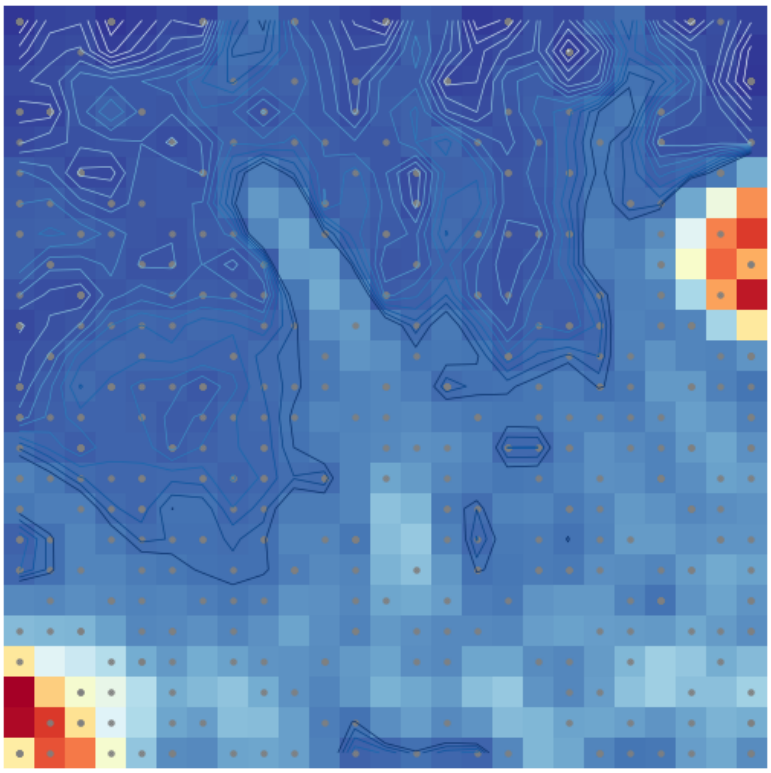
```
import sys
sys.path.append('../sompy')
import numpy as np
from sompy import SOMFactory
import pandas as pd
import glob
import os

ogdf = pd.read_csv('blood-train.csv')
data = ogdf[['Months since Last Donation', 'Number of Donations', 'Total Volume Donated
(c.c.)','Months since First Donation']]
names = ['Months since Last Donation', 'Number of Donations', 'Total Volume Donated (c.c.)','Months
since First Donation']

print(data.head())

sm = SOMFactory().build(data.values, normalization = 'var', mapsize=(25,25), initialization='pca',
component_names=names)
sm.train(n_job=1, verbose=False, train_rough_len=2, train_finetune_len=5)

# The quantization error: average distance between each data vector and its BMU.
# The topographic error: the proportion of all data vectors for which first and second BMUs are not
adjacent units.
topographic_error = sm.calculate_topographic_error()
quantization_error = np.mean(sm._bmu[1])
print ("Topographic error = %s; Quantization error = %s" % (topographic_error, quantization_error))

# component planes view
import sys
sys.path.append('../sompy/visualization')
from mapview import View2D
view2D  = View2D(10,10,"rand data",text_size=11)
view2D.show(sm, col_sz=3, which_dim="all", desnormalize=True)

# U-matrix plot
from umatrix import UMatrixView
umat  = UMatrixView(width=2,height=2,title='U-matrix')
umat.show(sm)

# K-means clustering on the SOM grid
from hitmap import HitMapView
number_of_clusters = 2
[labels, km, norm_data] = sm.cluster(10,number_of_clusters)
hits  = HitMapView(7,7,"Clustering",text_size=12)
a=hits.show(sm)
```

Output:

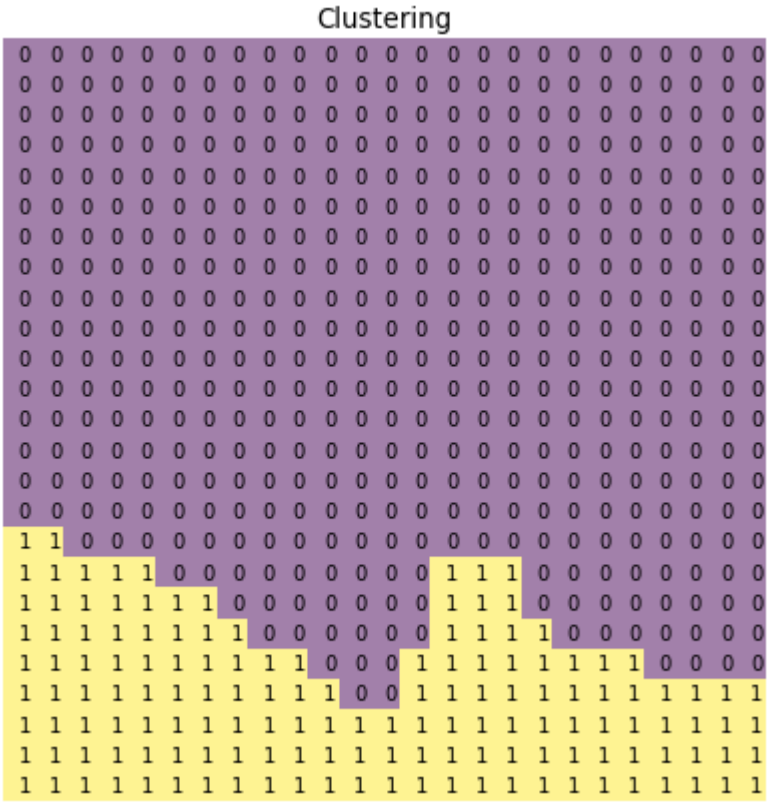
Components plane view



U-matrix plot



K-means clustering on SOM grid



Conclusion:

This project is utilized to complete an examination of different algorithms in classification and to gauge the precision with low time consumption for testing. The algorithm Multi-Layer Perceptron has demonstrated 79% exactness inside brief term when contrasted and different algorithms in classification. This examination among algorithms is utilized to give appropriate usage of best algorithm to give precise answer for issues in brief time. This examination ought to be utilized as a part of further procedures in logical research and in the forecast of blood benefactors. The solution info presented at San Raffaele Hospital is a piece without bounds of human services around the world, which will incorporate a totally straightforward blood store network.

Future Scope:

This method can be utilized on huge scale at blood donation centres and furthermore amid different cases, for example, orthopaedic surgeries, spinal surgeries, open heart surgeries and so forth. Blood loss in during a surgical procedure should be minimized. The following points work in favour of the patients who are looking for blood donors:

1.No need to find matching donors

2.No worries for rare blood groups

3.No infectious or immunologic effects after blood transfusion

4.Safe method

5.Faster recovery

The blood benefactor process, from the contributor through various blood donation centres and doctor's facilities to the patient, is being contemplated by governments and private

human services associations and will get incredible consideration later on .

References:

[1]Creutzfeldt Jakob disease surveillance in the UK: Thirteenth Annual Report 2004

[2]Provan D. Better blood transfusion: we must use donated blood better and consider alternatives. BMJ 1999; 318: 1435–6

[3]James V. A national blood conservation strategy for NBTC and NBS— Report from the Working Party on Autologous Transfusion and the Working Party on Alternatives to Transfusion of the NBS Sub-Group on Appropriate Use of Blood, 2004.

[4]Gombotzand H, Kulier A. Intraoperative autotransfusion of red blood cells and platelet- rich plasma. Transfusion Alternatives in Transfusion Medicine 1999; 1: 5–13.

[5]Bull BS, Bull MH. The salvaged blood syndrome: a sequel to mechanochemical activation of platelets and leukocytes? Blood Cells 1990; 16: 5–20

[6]Heimer P, Knels R (2007) Safe blood transfusions in Pakistan–Report of the joint mission of GIZ and KfW experts for feasibility study. Document of the German Governmental Bank (KfW).

[7]Carless PA, Henry DA, Moxey AJ, O'Connell DL, Fergusson DA. Cell salvage for minimising perioperative allogenic blood transfusion. Cochrane Database Syst Rev 2003; CD001888. DOI: 10.1002/14651858

[8]National Institute for Clinical Excellence. Interventional Procedures Advisory Committee. IPG144 Intraoperative blood cell salvage in obstetrics—guidance.