**BLOOD TRANSFUSION SYSTEM**

**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)

**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%2BTransfusion%2BService%2BCenter

Lvq code:

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)

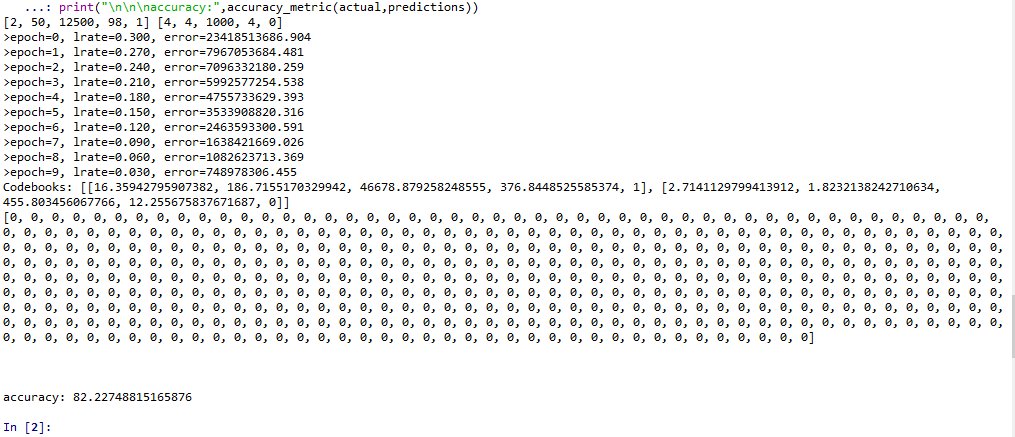
predictions.append(output)

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))



Accuracy of 82.27% if found using lvq.