**Report - Analysis of different Classifiers in ML**

**I. Choose dataset:**

**Dataset**: Ionosphere Data Set

**Number of instances**: 351

**Number of attributes**: 34

**Number of classes in the predicted variable**:

It is a binary classification. The value can be good or bad.

**If there are a number of NA or null values, how you plan to handle that situation**:

The dataset does not have any null values.

**Any other pre-processing that may be needed**: The file was read as a csv file and the data was filtered and the column that had N/A values were removed. The filter statement was used for the purpose of removing the V2 column as the standard deviation of this column of this was 0. Then the columns were shifted and the existing columns were worked on. It did not require much of preprocessing as it did not contain any null values.

**II. Understand the dataset:**

**How was the data obtained (source of the data) :**

The radar data used were collected by the Space Physics Group of The Johns Hopkins University Applied Physics Laboratory. The radar system, located in Goose Bay, Labrador, consists of a phased array of 16 high-frequency antennas, with a total transmitted power on the order of 6.4 kW and an antenna gain of about 30 dBm at frequency ranges of 8 to 20 MHz. The radar returns are used to study the physics of the ionosphere at the E- and F-layers (100- to 500-km altitude). Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

**What did the authors use the dataset for (use cases for the dataset):**

The dataset was used to analyze the essential features needed to understand the input to the neural network. Networks were trained to discriminate "good" from "bad" radar returns from the ionosphere. Also the dataset was used to demonstrate that neural networks would operate at a level of performance high enough to be a real aid in the automation of the classification task.

**What type of experiments were done on the dataset (experiments) :**

The ACF of a radar return is described by the 17 discrete returns. Since each discrete return is composed of a real and an imaginary part, 34 values per ACF result. These 34 values serve as input to the network. Each input was normalized to the range [ - 1, 1]. The number of hidden nodes was varied from o (no hidden layer) to 15. Perceptrons are used for quantifying the additional power obtained by having hidden nodes. The network is trained with 200 returns (101 good, 99 bad). Networks having 0, 3, 5, 8, 10, and 15 hidden nodes were used. The testing set, was composed of 150 returns, of which 123 were good and 27 were bad. Networks were analyzed for differences in sensitivity and specificity.

**Summary of the authors’ results :**

Curves were plotted on the training set for two perceptrons and for a typical MLFN with five hidden nodes. one of the perceptron used linear transformation for its output function (identity function) and the other used the sigmoid trans-formation (1/1 + e- X ) for its output function. All MLFN'S used this sigmoid transformation. The learning curves began at values of 38070 to 58% correct, and all moved to above 80% correct after 25 presentations of the training set. All nearly reached their final values by 100 presentations.

The black curve represents the linear perceptron, which eventually converged to 87.5% correct. The blue line represents the nonlinear perceptron, which eventually converged to 94.5% correct. The red curve represents an MLFN with five hidden nodes; it, and the other MLFN'S converged to 99.5% to 100% correct. It is clear that the MLFN'S are superior to the perceptrons in learning the classification task. The linear perceptron was able to correctly classify 90.67% from the testing set; the nonlinear perceptron, 92%. The MLFN'S averaged greater than 96% correct, with a range from 94% to 98%. The sensitivity of the linear perceptron was 95.9%; that for the nonlinear perceptron was 98.4% (121 of 123); and that for the best MLFN'S was 100%.

Specificity is a measure of how well the networks correctly classify bad returns. The specificity of the linear perceptron was only 66.7% (it correctly classified 18 of 27 bad returns); that for the nonlinear perceptron was 63 % (17 of 27); and that for the best MLFN'S was 88.9% (24 of 27). The worst MLFN'S had a sensitivity of 1000/0 and a specificity of 66.7%. Thus, the worst MLFN did as well as the best perceptron. For a threshold of 0.5, the MLFN accounted for 83.8% of the output variance; the perceptron accounted for only 49.1%.

**Pre Processing**

**Summary of preprocessing**

The file was read as a csv file and the data was filtered and the column that had N/A values were removed. The filter statement was used for the purpose of removing the V2 column as the standard deviation of this column of this was 0. Then the columns were shifted and the existing columns were worked on. It did not require much of preprocessing as it did not contain any null values.

**Code for Preprocessing**

mydata<-Filter(function(x) sd(x) != 0, mydata)

colnames(mydata)[colnames(mydata)=="V3"] <- "V2"

colnames(mydata)[colnames(mydata)=="V4"] <- "V3"

colnames(mydata)[colnames(mydata)=="V5"] <- "V4"

colnames(mydata)[colnames(mydata)=="V6"] <- "V5"

colnames(mydata)[colnames(mydata)=="V7"] <- "V6"

colnames(mydata)[colnames(mydata)=="V8"] <- "V7"

colnames(mydata)[colnames(mydata)=="V9"] <- "V8"

colnames(mydata)[colnames(mydata)=="V10"] <- "V9"

colnames(mydata)[colnames(mydata)=="V11"] <- "V10"

colnames(mydata)[colnames(mydata)=="V12"] <- "V11"

colnames(mydata)[colnames(mydata)=="V13"] <- "V12"

colnames(mydata)[colnames(mydata)=="V14"] <- "V13"

colnames(mydata)[colnames(mydata)=="V15"] <- "V14"

colnames(mydata)[colnames(mydata)=="V16"] <- "V15"

colnames(mydata)[colnames(mydata)=="V17"] <- "V16"

colnames(mydata)[colnames(mydata)=="V18"] <- "V17"

colnames(mydata)[colnames(mydata)=="V19"] <- "V18"

colnames(mydata)[colnames(mydata)=="V20"] <- "V19"

colnames(mydata)[colnames(mydata)=="V21"] <- "V20"

colnames(mydata)[colnames(mydata)=="V22"] <- "V21"

colnames(mydata)[colnames(mydata)=="V23"] <- "V22"

colnames(mydata)[colnames(mydata)=="V24"] <- "V23"

colnames(mydata)[colnames(mydata)=="V25"] <- "V24"

colnames(mydata)[colnames(mydata)=="V26"] <- "V25"

colnames(mydata)[colnames(mydata)=="V27"] <- "V26"

colnames(mydata)[colnames(mydata)=="V28"] <- "V27"

colnames(mydata)[colnames(mydata)=="V29"] <- "V28"

colnames(mydata)[colnames(mydata)=="V30"] <- "V29"

colnames(mydata)[colnames(mydata)=="V31"] <- "V30"

colnames(mydata)[colnames(mydata)=="V32"] <- "V31"

colnames(mydata)[colnames(mydata)=="V33"] <- "V32"

colnames(mydata)[colnames(mydata)=="V34"] <- "V33"

colnames(mydata)[colnames(mydata)=="V35"] <- "V34"

colnames(mydata)[colnames(mydata)=="V34"] <- "myclass"

mydata$myclass <- mapvalues(mydata$myclass,from = c("b","g"),to = c(0,1))

str(mydata)

**Code for correlation and Histogram display**

#Correlation

for(i in 1:33){

for(j in i:33){

y=cor(mydata[i],mydata[j])

cat(c("Corelation between variables",i," ",j," ",is," ",y,"\n"),file=fileConn,append=TRUE,sep="")

}

}

for(i in 1:33){

y=cor(mydata[i],as.numeric(levels(mydata$myclass)[mydata$myclass]))

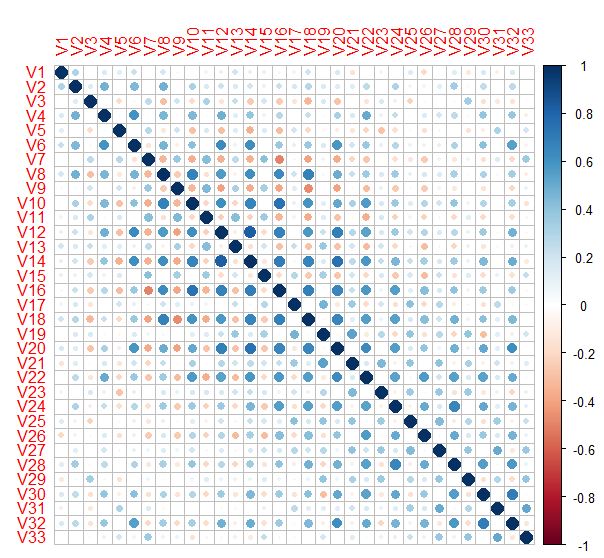
cat(c("Corelation between variables",i," ","ClassLabel"," ","is",y,"\n"),file=fileConn,append=TRUE,sep="")

}

#cat(c("Corelation matrix is",cor(mydata[,unlist(lapply(mydata, is.numeric))])),file=fileConn,append=TRUE)

y=cor(mydata[1:33])

corrplot(y,method="circle")



hist(y)

