**BANK TERM DEPOSIT CODE**

**PERCEPTRONS**

banking <- bank\_marketing[,c("age","job1","marital1","education1","balance","housing1","default1","loan1",

"contact1","day","month1","duration","campaign","pdays","previous","poutcome1","y")]

ggplot(banking, aes(x =day, y = campaign)) + geom\_point(aes(colour = y, shape= y), size = 3) + xlab("day") + ylab("campaign") + ggtitle("Term Deposit V/S Day and Campaign") + theme(plot.title = element\_text(hjust = 0.5))

bankingSubset2D$class<- lapply(bankingSubset2D$y, function(x) {

if(x == 'no')

bankingSubset2D$class <- -1

else if(x == 'yes')

bankingSubset2D$class <- 1

else

bankingSubset2D$class <- NULL

})

X<-bankingSubset2D[,c("age","job1","marital1","education1","balance","housing1","default1","loan1","contact1","day","month1","duration","campaign","pdays","previous","poutcome1")] # Input Matrix

y <- bankingSubset2D$class # Output Vector

banking$class<- lapply(banking$y, function(x) {

if(x == 'no')

banking$class <- -1

else if(x == 'yes')

banking$class <- 1

else

banking$class <- NULL

})

X<-banking[,c("age","job1","marital1","education1","balance","housing1","default1","loan1","contact1","day","month1","duration","campaign","pdays","previous","poutcome1")] # Input Matrix

y <- banking$class # Output Vector

#Creating Perceptron Learning Algorithm

perceptron <- function(X, y, numEpochs) {

results <- list()

w <- runif(ncol(X), -10, 10) #Initalize weights

# For loop - number of generations(epochs) - number of times dataset is ran through

for(j in 1:numEpochs) {

predictedResult <- numeric(length=100) # Initalize predictedResult vector

numIncorrect = 0 # Keeps track of # of missclassified points

# For loop - loop throught dataset

for(i in 1:length(y)) {

xi = as.numeric(unlist(X[i,])) # Convert dataframe to vector

predictedResult[i] = sign(w %\*% xi) # Predict the point

# If predicted point is incorrect - change weight

if(predictedResult[i] != y[i]) {

numIncorrect = numIncorrect + 1 # Add one to # of missclassified points

w <- w + as.numeric(y[i]) \* xi # Update the weight w <- w + WiXi

}

}

# Print results of this generation(epoch)

cat("\nEpoch #: ", j)

cat("\nNumber Incorrect: ", numIncorrect)

cat("\nFinal Weight: ", w)

}

}

results = perceptron(X,y, 8)

**NAÏVE BAYES CODE**

table(bankmarketing$y)

summary(bankmarketing)

bankmarketing$y <- factor(bankmarketing$y, levels=c("yes","no"), labels=c("Term Deposit","No Term Deposit"))

bankmarketing\_index <- sample(nrow(bankmarketing), 1/2 \* nrow(bankmarketing))

bank\_train <- bankmarketing[bankmarketing\_index, ]

bank\_test <- bankmarketing [-bankmarketing\_index, ]

bank\_model <- naiveBayes(y ~ ., data=bank\_train, laplace=1)

bank\_model

bank\_pred <- predict(bank\_model, bank\_test, type="class")

bank\_pred\_table <- table(bank\_test$y, bank\_pred)

bank\_pred\_table

sum(diag(bank\_pred\_table))/nrow(bank\_test)

**KNN CODE**

bankmarketing$y1 <- factor(bankmarketing$y1, levels=c(1,0), labels=c("Term Deposit","No Term Deposit"))

normalize <- function(x) { return ((x-min(x))/(max(x)-min(x))) }

bank\_norm <- as.data.frame(lapply(bankmarketing[1:16], normalize))

bank\_index <- sample(nrow(bank\_norm), 1/2 \* nrow(bank\_norm))

bank\_train <- bank\_norm[bank\_index, ]

bank\_test <- bank\_norm[-bank\_index, ]

bank\_train\_labels <- bankmarketing[bank\_index,17]

bank\_test\_labels <- bankmarketing[-bank\_index,17]

install.packages("class")

library(class)

bank\_test\_pred <- knn(train = bank\_train, test = bank\_test, cl=bank\_train\_labels, k=30)

install.packages("gmodels")

library(gmodels)

CrossTable(x=bank\_test\_labels, y=bank\_test\_pred, prop.chisq=FALSE)

bank\_test\_pred <- knn(train = bank\_train, test = bank\_test, cl=bank\_train\_labels, k=200)

CrossTable(x=bank\_test\_labels, y=bank\_test\_pred, prop.chisq=FALSE)

**NEURAL NETWORK CODE**

install.packages("neuralnet", dependencies = TRUE)

library(neuralnet)

banking\_index <- sample(nrow(banking), 1/2 \* nrow(banking))

banking\_train <- banking[banking\_index, ]

banking\_test <- banking[-banking\_index, ]

head(banking\_train)

head(banking\_test)

bankingnet <- neuralnet(y1~ age +job1+marital1+education1+default1+balance+housing1+loan1+contact1+day+month1+duration+campaign+pdays+previous+poutcome1,banking\_train, hidden=2, lifesign="minimal", linear.output=FALSE, threshold=0.01)

bankingnet$result.matrix

plot(bankingnet)

bankingnet.results <- compute(bankingnet, banking\_test)

results <- data.frame(actual=banking\_test$y1, prediction=bankingnet.results$net.result)

results[1:20,]

results$prediction <- round(results$prediction)

results [1:20,]

cor(bankingnet.results$net.result, banking\_test$y1)

**NEURAL NETWORK CODE WITH DATA NORMALIZATION**

install.packages("neuralnet", dependencies = TRUE)

library(neuralnet)

normalize <- function(x) {return((x-min(x))/(max(x)-min(x)))}

banking\_index <- sample(nrow(banking), 1/2 \* nrow(banking))

banking\_train <- banking[banking\_index, ]

banking\_test <- banking[-banking\_index, ]

head(banking\_train)

head(banking\_test)

trainingnorm <- as.data.frame(lapply(banking\_train,normalize))

testingnorm <- as.data.frame(lapply(banking\_test,normalize))

bankingnet <- neuralnet(y1~ age +job1+marital1+education1+default1+balance+housing1+loan1+contact1+day+month1+duration+campaign+pdays+previous+poutcome1,trainingnorm, hidden=c(3,2), lifesign="minimal", linear.output=FALSE, threshold=0.01)

bankingnet$result.matrix

plot(bankingnet)

bankingnet.results <- compute(bankingnet, testingnorm)

results <- data.frame(actual=testingnorm$y1, prediction=bankingnet.results$net.result)

results[1:20,]

results$prediction <- round(results$prediction)

results [1:20,]

cor(bankingnet.results$net.result, testingnorm$y1)

**SVM CODE**

install.packages("e1071",dependencies=TRUE)

library(e1071)

banking$y <- as.factor(banking$y)

model1 <- svm(y~ age+duration, data=banking)

print(model1)

summary(model1)

plot(model1,banking)

**SVM example with more than two variables – small dataset**

bankingsubset <- banking[ , c("y" , "age", "job1", "marital1","education1")]

bankingsvm <- svm(y~. , data=bankingsubset)

print(bankingsvm)

summary(bankingsvm)

plot(bankingsvm, bankingsubset,age ~ marital1,slice = list(job1=2,education1=2))