**Homework 1 (Problem 1)**

**MATH 6333 Statistical Learning**

**Md Salman Rahman**

**Instructor: Dr. Tamer Oraby**

**Problem 1:** Write a detailed report to explain every step in the file Lab 1 - R.R that was covered in the third week.

**Solution:**

**Code line 1:** #### Lab 1 Intro to Supervised Learning in R ###########

**Explanation:** # means comments in R. This indicates the comments of the code.

**Code line 2:** #### Tamer Oraby ########################################

**Explanation:** This is also a comment and express the instructor's name inside the comment.

**Code line 3:** setwd("C:/Users/day109/OneDrive - The University of Texas-Rio Grande Valley/MATH 6333 90L - SL - F21/R Codes")

**Explanation:** This set the working directory in C:/Users/day109/OneDrive - The University of Texas-Rio Grande Valley/MATH 6333 90L - SL - F21/R Codes this location.

**Code line 4:** #set.seed(5)

**Explanation:** Though it is a comment, set.seed() generally used to generate random number which is particularly useful for creating random object that can be possible to reproduce. Here, Professor oraby start it so that all of us get the same result. Later drop the idea because we want randomness.

**Code line 5:** library("MASS")

**Explanation:** it is a library and it is called so that we can use mvrnorm(multivariate random norm) from this library.

**Code line 6:** library("ggplot2")

**Explanation:** ggplot2 is a great R library for making nice plot and we will use it during plot.

**Code line 8:** mean_blue<-mvrnorm(n=10,c(1,0),diag(2))

**Explanation:** mvrnorm: multivariate random norm; n=10 sample required, c(1,0): generate 10 centers at the corner c(1,0); diag(2): we need variance covariance matrix to be the identity matrix, because it is bivariates so it should be 2 by 2 identity matrix.

**Code line 9:** is.matrix(mean_blue)

**Explanation:** asking R, whether the above mean_blue is matrix or not.

**Code line 10:** nrow(mean_blue)

**Explanation:** this finds how many rows the mean_blue matrix has

**Code line 11:** DF_mean_blue<-as.data.frame(mean_blue)

**Explanation:** making the mean blue matrix as a data frame. Many of the R function prefer dataframe rather than a matrix.

**Code line 12:** colnames(DF_mean_blue)<-c('cx','cy')

**Explanation:** Changing the column name to cx and cy. (cx means center of x and cy means center of y).

**Code line 14 to 17:** p1=ggplot(DF_mean_blue,aes(x= cx, y=cy ))+

```
  geom_point(colour="blue",shape=23)+

  xlab('x')+

  ylab('y')
```

**Explanation:** making plot using ggplot library, here DF_mean: we want to plot this DF_mean_blue data frame; aes: this is for axis, and here on the x axis it will be cx, and on the y axis it will be cy; geom_point(colour="blue",shape=23): this is for plotting geometric plot where colour will be blue and the shape 23 give the diamond shape, xlab("x"): the name the label of x axis as x; ylab("y"): name the label of y axis as y; this will overall give 10 randomly generated mean with blue color using bivariate gaussian.

**Code line 18:** p1

**Explanation:** calling the p1 variable.

**Code line 21:** mean_orange<-mvrnorm(n=10,c(0,1),diag(2))

**Explanation:** this is for orange, mvrnorm: multivariate random norm; n=10 sample required, c(1,0): generate 10 centers at the corner c(1,0); diag(2): we need variance covariance matrix to be the identity matrix, because it is bivariates so it should be 2 by 2 identity matrix.

**Code line 22:** DF_mean_orange<-as.data.frame(mean_orange)

**Explanation:** making the mean_orange matrix as a data frame. The purpose many of the R function prefer dataframe rather than a matrix.

**Code line 23:** colnames(DF_mean_orange)<-c('cx','cy')

**Explanation:** Changing the column name to cx and cy. (cx means center of x and cy means center y).

**Code line 25 to 28:** p2=ggplot(DF_mean_orange,aes(x= cx, y=cy ))+

```
  geom_point(colour="orange",shape=23)+
  xlab('x')+
  ylab('y')
```

**Explanation:** making plot using ggplot library, here DF_mean_orange: we want to plot this DF_mean_orange data frame; aes: this is for axis, and here on the x axis it will be cx, and on the y axis it will be cy; geom_point(colour="orange",shape=23): this is for plotting geometric plot where color will be orange and the shape 23 give the diamond shape, xlab(“x”): the x label will change to x; ylab(“y”): the y label change to y; this will overall give 10 randomly generated mean with orange color using bivariate gaussian.

**Code line 29:** p2

**Explanation:** calling the p2 variable.

**Code line 31 to 32:** p3=p1+

```
  geom_point(data=DF_mean_orange,aes(x= cx, y=cy ),colour="orange",shape=23)
```

**Explanation:** Putting p1 and p2 altogether (adding). We add the geometric point of orange and the data is DF_mean_orange with the p1(blue point). And putting them altogether in p3 variable.

**Code line 33:** p3

**Explanation:** calling the p3 variable.

**Code line 35:** N=200

**Explanation:** There will be N=200 data point.

**Code line 37:** data_blue<-replicate(n=N/2 , c(mvrnorm(n=1,mean_blue[sample(nrow(mean_blue),1), ],diag(2)/5),0))

**Explanation:** mean_blue [sample(nrow(mean_blue),1)]: sampling 1 value from the n of rows of the mean_blue which is 10, it will give the row number of the means of blue. We take this sampling mean of blue of 10 randomly generated rows of both columns.

(mvrnorm(n=1,mean_blue[sample(nrow(mean_blue),1), ],diag(2)/5): generating mean of gaussian distribution for n=1, and randomly generated mean which I discussed above with diagonal matrix of 2 by 2 divided by 5 c(mvrnorm(n=1,mean_blue[sample(nrow(mean_blue),1), ],diag(2)/5),0): marking blue by 0 and concatenating

replicate(n=N/2 , c(mvrnorm(n=1,mean_blue[sample(nrow(mean_blue),1), ],diag(2)/5),0)): here replicate use instead of for loop (for loop consume time), here replicate is use to replicate the line of code inside the parentheses in N/2 (N =200) number of times.

**Code line 38:** data_blue<-t(data_blue)

**Explanation:** transposing the data_blue

**Code line 40:** data_orange<-replicate(n=N/2 , c(mvrnorm(n=1,mean_orange[sample(nrow(mean_orange),1), ],diag(2)/5),1))

**Explanation:** Explanation is given in codeline 37, only difference is that code line 40 is for orange and mark by 1 and code line 37 is for blue and mark by 0.

**Code line 41:** data_orange<-t(data_orange)

**Explanation:** transposing the data_orange

**Code line 43:** mix_data<-rbind(data_blue,data_orange)

**Explanation:** Binding data_blue and data_orange together using rbind, blue one in the top and orange one in the bottom. There will be total of 200 entries.

**Code line 45:** DF_mix_data<-as.data.frame(mix_data)

**Explanation:** making the mix_data matrix as a data frame naming DF_mix_data

**Code line 46:** colnames(DF_mix_data)<-c('cx','cy','Y')

**Explanation:** Changing the column name to cx, cy, and Y.

**Code line 47:** rownames(DF_mix_data)<-c()

**Explanation:** removing any order or changes in the rownames.

**Code line 49:** DF_mix_data<-DF_mix_data[sample(nrow(DF_mix_data)), ]

**Explanation:** The purpose here is to bring the real world feeling in the data by random sampling.

sample(nrow(DF_mix_data): sampling all of them from the n number of rows of the DF_mix_data,

DF_mix_data<-DF_mix_data[sample(nrow(DF_mix_data)), ]: the sampling is done for DF_mix_data data frame

**Code line 50:** rownames(DF_mix_data)<-c()

**Explanation:** removing any order or changes in the rownames.

**Code line 52 to 54:** p4=ggplot(DF_mix_data[ ,1:2],aes(x=cx,y=cy))+

geom_point(colour= ifelse( DF_mix_data[,3]==1, "orange", "blue"))+

theme_bw()+xlim(-3,4)+ylim(-3,3)

**Explanation:** DF_mix_data[ ,1:2],aes(x=cx,y=cy): [ ,1:2] define all rows(all 200 rows) from 1 to 2 column and x axis will be cx, and y axis will be cy.

geom_point(colour= ifelse( DF_mix_data[,3]==1, "orange", "blue")): plotting geometic point of the above define data, and here the colouring is done by if else statement; here if the DF_mix_data[,3]==1 (if the third column is equal to 1) condition is true then it will color as “orange”, and if the statement is false then color will be “blue”.

theme_bw()+xlim(-3,4)+ylim(-3,3): theme_bw(): tweak the display of an existing theme and xlim means limiting x axis from -3 to 4 and ylim means limiting y axis from -3 to 3.

ggplot(DF_mix_data[ ,1:2],aes(x=cx,y=cy))+ geom_point(colour= ifelse( DF_mix_data[,3]==1, "orange", "blue"))+ theme_bw()+xlim(-3,4)+ylim(-3,3): this make the plot using the ggplot of the above explanation. This will generate 200 datapoints. And there are 100 data points of orange came from mixture and 100 data points of blue came from the mixture.

**Code line 55:** p4

**Explanation:** calling the p4 variable.

**Code line 57:**
rm(data_blue,data_orange,DF_mean_blue,DF_mean_orange,mean_blue,mean_orange,mix_data)

**Explanation:** here rm used for removing data_blue,data_orange,DF_mean_blue,DF_mean_orange,mean_blue,mean_orange,mix_data from the memory

**Code line 59:** ## Linear regression for classification

**Explanation:** This is a comment and from here we will start doing linear regression for classification

**Code line 61:** LR<-lm(Y~cx+cy,data=DF_mix_data)

**Explanation:** lm stands for linear model,

LR<-lm(Y~cx+cy,data=DF_mix_data): linear model of Y defined by cx and cy, and the data will be DF_mix_data which we establish earlier.

**Code line 62:** summary(LR)

**Explanation:** This gives the summary of above linear model.

**Code line 63:** sum((predict(LR)>.5)+0 == DF_mix_data$Y)/N

**Explanation:** making a predication of LR, and this will make prediction of value greater than 0.5 (means orange) and +0 means it will label as 0 (orange) and 1 instead of FALSE and TRUE.

((predict(LR)>.5)+0 == DF_mix_data$Y)/N: == DF_mix_data$Y)/N means it will check if the prediction is equal to actual data DF_mix_data of Y or not, and if we divide this by N then it will give us average of success.

**Code line 65:** (predict(LR,data.frame(cx=c(1.3,1.4),cy=c(2,4)))>.5)+0

**Explanation:** Here we will make a separate prediction for a new data point cx=c(1.3,1.4) and cy=c(2,4). And check whether they are greater than 5 or not (greater than 5 means orange).

**Code line 67:** beta=coef(LR)

**Explanation:** this will give us coefficient of cx, cy, and the intercept.

**Code line 69:** meshpixel=100

**Explanation:** we will color the mesh by 100 by 100 mesh pixel

**Code line 70:** x_pixel=seq(-3,4,length=meshpixel)

**Explanation:** seq(-3,4,length=meshpixel): this will give the sequence from -3 to 4 (earlier we cut x axis from -3 to 4) and the length is equal to meshpixel which is 100

**Code line 71:** y_pixel=seq(-3,3,length=meshpixel)

**Explanation:** seq(-3,3,length=meshpixel): this will give the sequence from -3 to 3 (earlier we cut y axis from -3 to 3) and the length is equal to meshpixel which is 100

**Code line 72:** M<-expand.grid(x_pixel,y_pixel)

**Explanation:** expand.grid(x_pixel,y_pixel): it will make the grid

**Code line 73:** DF_M<-as.data.frame(M)

**Explanation:** making the data frame from M

**Code line 74:** colnames(DF_M)<-c('cx','cy')

**Explanation:** changing the column name of the data frame as cx and cy

**Code line 75:** color_M<-(predict(LR,DF_M)>.5)+0

**Explanation:** Here we will color the mesh point according to the type of prediction. By defining which one is greater than 0.5 and label them as 0 (orange) and 1 instead of TRUE and FALSE

**Code line 76:** new_M_LR<-cbind.data.frame(DF_M,color_M)

**Explanation:** now we will bind the DF_M and color_M data frame

**Code line 78 to 80:** p5=p4+geom_point(data=new_M_LR[ , 1:2],aes(x=cx,y=cy),

colour=ifelse( new_M_LR[ , 3]==1,"orange","blue"),alpha=.01)+

geom_abline(aes(intercept= (.5-beta[1])/beta[3], slope=-beta[2]/beta[3] ))

**Explanation:** adding geometic point to p4

geom_point(data=new_M_LR[ , 1:2],aes(x=cx,y=cy), colour=ifelse( new_M_LR[ , 3]==1,"orange","blue"),alpha=.01): the data for the mesh will coming from new_M_LR and it will only use all rows and 1:2 means it will use column 1 to 2 to make this data point and

aes(x=cx,y=cy): it will define the x axis as cx and y axis as cy,

colour=ifelse( new_M_LR[ , 3]==1,"orange","blue"),alpha=.01): color will be if else statement and the condition is if the third column of new_M_LR equal to 1 than it will color them as orange otherwise the color will be blue. Also, opacity of a geom. Here, orange and blue may overlap so that we use alpha =0.01 which will make the color lighter.

geom_abline(aes(intercept= (.5-beta[1])/beta[3], slope=-beta[2]/beta[3] ): adding a straight line where the intercept equal to (.5-beta[1])/beta[3] and the slope is equal to slope=-beta[2]/beta[3] (intercept and slope are coming from the equation of the straight line(beta_0+beta_1*c_x+beta_2*c_y=0.5]) we develop earlier)

**Code line 81:** p5

**Explanation:** calling the p5 variable.

**Code line 82:** ggsave("LRlearn.png",p5)

**Explanation:** saving the p5 plot as png using ggsave