**Predicting MBA Student Success**

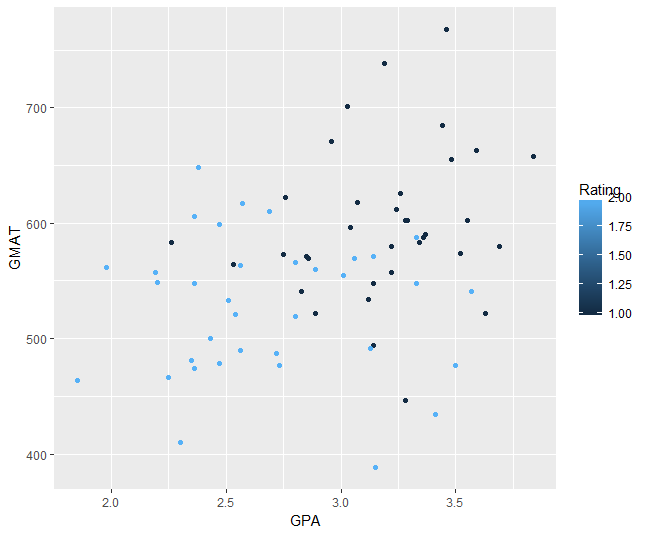
The MBA program staff has observed that students who are admitted to the program with a low GPA or GMAT score will often have difficulty completing the program. Students who do no complete the program or have difficulty degrade the reputation of the program. However, the MBA program would like to admit the maximum number of students possible. In an effort to set reasonable minimum GPA and GMAT scores the program staff have rated the students currently in the program into two groups – group 1 is “satisfactory” and group 2 “challenged.” The file MBAStudents.xlsm has this data.

1. Create a scatterplot of GPA and GMAT scores with different colors for the two groups and 95% confidence ellipses for the two groups.
   1. Draw a line that approximately separates the two groups. What is the classification decision rule for this?
   2. Draw a smooth curve that approximately separates the two groups. What is the classification decision rule for this?
   3. Draw any number of segments that approximately separates the two groups. What is the classification decision rule for this?
   4. What is the difference between these rules? Why would you to choose one over another?

> ggplot (MBAStudents, aes (x = GPA, y = GMAT, colour = Rating, group = Rating)) + geom\_point() + stat\_ellipse()

Without ggplot:

> car::dataEllipse(MBAStudents$GPA, MBAStudents$GMAT, groups=factor(MBAStudents$Rating), levels=.95, plot.points = FALSE)



1. Use Naïve Bayes classification to You can use the e1071 package.
   1. Create a classification decision rule where Prob(Rating | MBA,GPA) is greater than ½. How does the rule compare to your estimate in 1a? What is the accuracy and reliability of this classification?
   2. Use the model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted?
   3. Plot the decision boundary and compare with your estimates in 1a and 1b

> mdl\_bayes <- naiveBayes(factor(Rating) ~ GPA + GMAT, data = MBAStudents)

> library("caret")

> confusionMatrix(predict(mdl\_bayes, as.data.frame(MBAStudents)[,3:4])

, MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 26 8

2 9 27

Accuracy : 0.7571

95% CI : (0.6399, 0.8517)

No Information Rate : 0.5

P-Value [Acc > NIR] : 9.6e-06

Kappa : 0.5143

Mcnemar's Test P-Value : 1

Sensitivity : 0.7429

Specificity : 0.7714

Pos Pred Value : 0.7647

Neg Pred Value : 0.7500

Prevalence : 0.5000

Detection Rate : 0.3714

Detection Prevalence : 0.4857

Balanced Accuracy : 0.7571

'Positive' Class : 1

> bayes\_pred <- predict(mdl\_bayes, as.data.frame(NewApplicants)[,2:3])

> bayes\_pred

[1] 2 1 1 2 1

Levels: 1 2

Confidence is prob of being in the group classified (largest prob)

> apply(predict(model, as.data.frame(NewApplicants)[,2:3], type="raw"),1,max)

[1] 0.7974068 0.6602387 0.8299070 0.9650213 0.5938648

The decision boundary (or separator) is where P[Rating=1|MBA,GMAT] = P[Rating=2|MBA,GMAT] = ½. Not so easy to compute so let’s use a function to generate it from the predictions.

decisionplot <- function(model, data, class = NULL, predict\_type = "class",

resolution = 100, showgrid = TRUE, ...) {

if(!is.null(class)) cl <- data[,class] else cl <- 1

data <- data[,1:2]

k <- length(unique(cl))

plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)

# make grid

r <- sapply(data, range, na.rm = TRUE)

xs <- seq(r[1,1], r[2,1], length.out = resolution)

ys <- seq(r[1,2], r[2,2], length.out = resolution)

g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))

colnames(g) <- colnames(r)

g <- as.data.frame(g)

### guess how to get class labels from predict

### (unfortunately not very consistent between models)

p <- predict(model, g, type = predict\_type)

if(is.list(p)) p <- p$class

p <- as.factor(p)

if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")

z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)

contour(xs, ys, z, add = TRUE, drawlabels = FALSE,

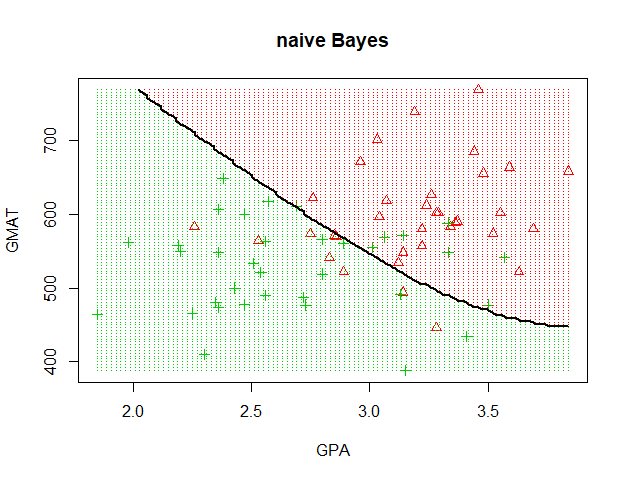
lwd = 2, levels = (1:(k-1))+.5)

invisible(z)

}

> mdl\_bayes <- naiveBayes(Rating ~ GPA + GMAT, data=MBAStudents)

> decisionplot(model, as.data.frame(cbind(MBAStudents[,3:4], MBAStudents[,2])), class = "Rating", main = "naive Bayes")



1. Use linear regression with a dummy variable for group 2 as the dependent variable and GPA, GMAT as the dependent variables.
   1. Look at a boxplot and density plot of the predicted value (discriminant scores) for each group. Does there seem to be clear differences between the groups? Can you see a cutoff point? Can you reasonably interpret the discriminant score as a probability of being in group 2?
   2. Interpret the slopes, intercept. Does the standard error and R-squared have interpretations here? What about the p-values for the slopes?
   3. Create a classification decision rule where the discriminant score value is greater than ½. How does the rule compare to your estimate in 1a? What is the accuracy and reliability of this classification?

> library("ggformula”)

plt <- ggplot(data=MBAStudents)+geom\_point(mapping=aes(x=GPA,y=GMAT,color=Rating))

> mdl\_lin <- lm(formula = (Rating - 1) ~ GPA + GMAT, data = MBAStudents)

> summary(mdl\_lin)

Call:

lm(formula = (Rating - 1) ~ GPA + GMAT, data = MBAStudents)

Residuals:

Min 1Q Median 3Q Max

-0.76491 -0.25814 0.03232 0.30898 0.74844

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.2807027 0.4169944 7.867 4.17e-11 \*\*\*

GPA -0.4680791 0.1076201 -4.349 4.76e-05 \*\*\*

GMAT -0.0025007 0.0006898 -3.625 0.000558 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

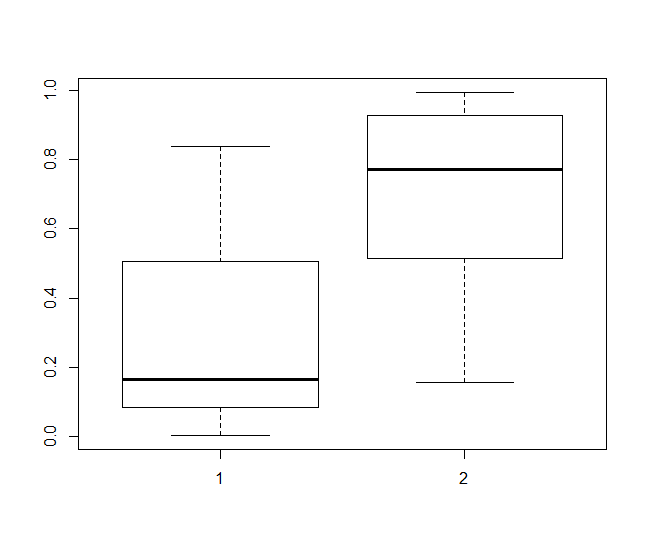
Residual standard error: 0.3918 on 67 degrees of freedom

Multiple R-squared: 0.4124, Adjusted R-squared: 0.3949

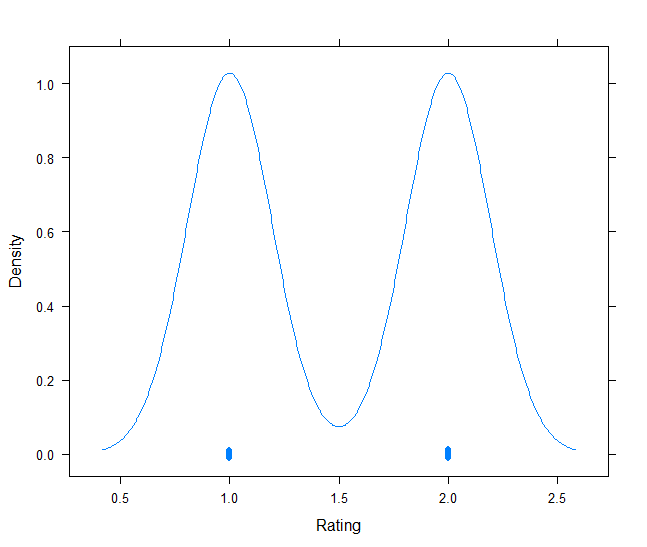
F-statistic: 23.51 on 2 and 67 DF, p-value: 1.835e-08

> MBAStudents$predict <- predict(mdl\_lin, type="response")

> boxplot(predict ~ Rating, data = MBAStudents)



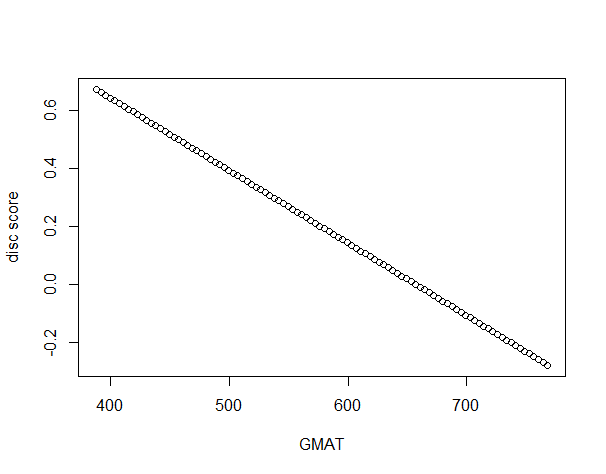
> densityplot(predict ~ Rating, data = MBAStudents)



The discriminate scores are only approximately probabilities. They do not behave appropriately as probabilities even within the relevant range. Let’s look at the discriminant values versus GMAT score with GPA = 3.4

> p\_GMAT <- predict(mdl\_lin, data.frame(GMAT= seq(min(MBAStudents$GMAT),max(MBAStudents$GMAT),length=100), GPA=rep(3.5,times=100)))

> plot(seq(min(MBAStudents$GMAT),max(MBAStudents$GMAT),length=100), p\_GMAT, xlab="GMAT", ylab="disc score")



At around a GMAT score of 660 the score goes negative which of course makes no sense as a probability.

> MBAStudents$predict\_half <- (predict(mdl\_lin, type="response") > .5)\*1 + 1

> confusionMatrix(MBAStudents$predict\_half, MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 26 8

2 9 27

Accuracy : 0.7571

95% CI : (0.6399, 0.8517)

No Information Rate : 0.5

P-Value [Acc > NIR] : 9.6e-06

Kappa : 0.5143

Mcnemar's Test P-Value : 1

Sensitivity : 0.7429

Specificity : 0.7714

Pos Pred Value : 0.7647

Neg Pred Value : 0.7500

Prevalence : 0.5000

Detection Rate : 0.3714

Detection Prevalence : 0.4857

Balanced Accuracy : 0.7571

'Positive' Class : 1

> coef(mdl\_lin)

(Intercept) GPA GMAT

3.280702702 -0.468079136 -0.002500741

So linear separator is:

3.280702702 -0.468079136\*GPA -0.002500741\*GMAT = 0.5

Can solve this to plot discriminant function:

> plot(MBAStudents[,3:4],col=MBAStudents$Rating)

> usr <- par('usr')

> x <- c(usr[1],usr[2])

> y <- c(usr[3],usr[4])

> sep\_half\_GMAT <- function(GPA) {(3.280702702 -0.468079136\*GPA - 0.5)/0.002500741}

> plot(MBAStudents[,3:4],col=MBAStudents$Rating)

> lines(x,sep\_half\_GMAT(x), lty=2)



Or use our decisionplot function:

|  |
| --- |
| > mdl\_lda <- lda(factor(Rating) ~ GPA + GMAT, data = MBAStudents)  > decisionplot(mdl\_lda, as.data.frame(cbind(MBAStudents[,3:4], MBAStudents[,2])), class = "Rating", main = "LDA") |
|  |

* 1. Determine a discriminant cutoff by averaging the mean discriminant scores for each group to create a classification rule. How does the rule compare to your estimate in 1a? What is the accuracy and reliability of this classification? Explain if this rule better than the 50% rule in (c).

> mean2 <- mean(predict(mdl\_lin, type="response")[MBAStudents$Rating==2])

> mean1 <- mean(predict(mdl\_lin, type="response")[MBAStudents$Rating==1])

> cutoff <- (mean1 + mean2)/2

> cutoff

[1] 0.5

Cutoff same because equal number of members in each group. Moving cutoff will not improve accuracy.

* 1. If you believe outliers are affecting the discriminant, determine a discriminant cutoff by averaging the median discriminant scores for each group to create a classification rule. does the rule compare to your estimate in 1a? What is the accuracy and reliability of this classification?

> median1 <- median(predict(mdl\_lin, type="response")[MBAStudents$Rating==1])

> median2 <- median(predict(mdl\_lin, type="response")[MBAStudents$Rating==2])> cutoff\_median <- (median1 + median2)/2

> MBAStudents$predict\_median <- (predict(mdl\_lin, type="response") > cutoff\_median)\*1 + 1

> confusionMatrix(MBAStudents$predict\_median, MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 25 6

2 10 29

Accuracy : 0.7714

95% CI : (0.6555, 0.8633)

No Information Rate : 0.5

P-Value [Acc > NIR] : 2.927e-06

Kappa : 0.5429

Mcnemar's Test P-Value : 0.4533

Sensitivity : 0.7143

Specificity : 0.8286

Pos Pred Value : 0.8065

Neg Pred Value : 0.7436

Prevalence : 0.5000

Detection Rate : 0.3571

Detection Prevalence : 0.4429

Balanced Accuracy : 0.7714

'Positive' Class : 1

Interesting. Median separator seems better. Let’s add it to our plot to see why:

> sep\_median\_GMAT <- function(GPA) {(3.280702702 -0.468079136\*GPA - cutoff\_median)/0.002500741}

> lines(x,sep\_median\_GMAT(x), lty=2, col="green")



Looks like some oulier 1’s drop the mean a little too much.

1. Use the linear discriminant model to predict what groups the applicants in the New Applicants tab would be in. Start by plotting the applicants points on the scatter plot.
   1. What is the classification decision rule for this model? How does it compare with any of your answers in question (1)?
   2. Use the model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted? Hint: change the prediction interval confidence until the interval does not contain the discriminant value.
   3. Compare the results to the other approaches used. What are the advantages and disadvantages of this approach? Is this approach reliable?

> pred.applicants <- predict(mdl\_lin,NewApplicants, interval="prediction", se.fit = TRUE)

> NewApplicants$Pred\_lda <- (pred.applicants$fit[,1] > cutoff\_median)\*1+1

> NewApplicants$Pred\_lda

[1] 2 1 1 2 1

Same as Naïve Bayes. Let’s look at these predictions on the plot

points(NewApplicants[,2:3], col=3, cex = 4)

text(NewApplicants[,2:3],labels=1:nrow(NewApplicants))



The farther from the separator the more confident the classification. But it’s not clear on how far gives what confidence. We can try using the regression prediction intervals as a measure.

> l=.95;n=1;a<-predict(mdl\_lin,NewApplicants, interval="prediction", se.fit = TRUE, level = l);a$fit[n,]; a$fit[n,]>cutoff\_median

fit lwr upr

0.74177017 -0.06221268 1.54575302

fit lwr upr

TRUE FALSE TRUE

Want to see prediction interval totally above or below cutoff. So try lower confidence:

> l=.5;n=1;a<-predict(mdl\_lin,NewApplicants, interval="prediction", se.fit = TRUE, level = l);a$fit[n,]; a$fit[n,]>cutoff\_median

fit lwr upr

0.7417702 0.4686068 1.0149335

fit lwr upr

TRUE TRUE TRUE

So can be about 50% confident that applicant 1 is in group 2. Let’s write a script to do it automatically for all:

for(n in 1:nrow(NewApplicants)) {

for(l in seq(.99,0,-.01)) {

a<-predict(mdl\_lin,NewApplicants, interval="prediction", level = l)

b <- (a[n,] > cutoff)

if((b[[1]]==b[[2]]) & (b[[2]] == b[[3]])) {

cat(c("student", n, "in group", (a[n,1]>cutoff)\*1+1, "with confidence", l\*100, "%\n"))

break

}

}

}

student 1 in group 2 with confidence 50 %

student 2 in group 1 with confidence 8 %

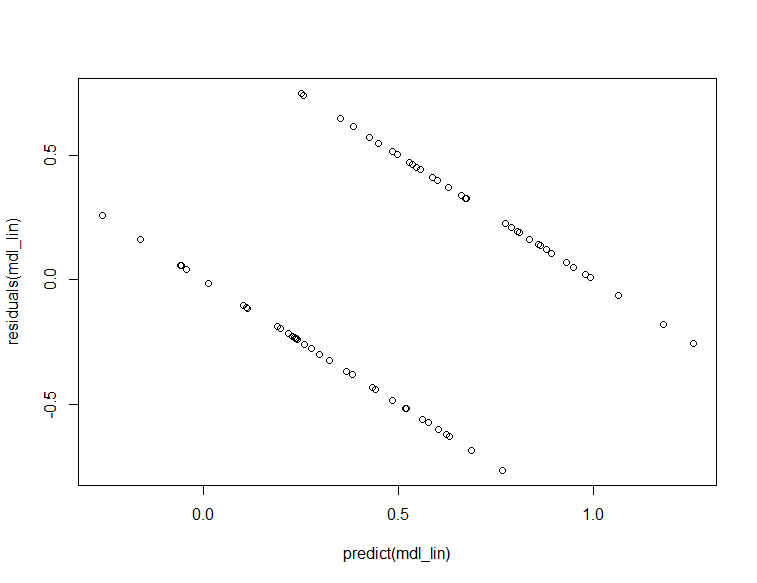
student 3 in group 1 with confidence 68 %

student 4 in group 2 with confidence 74 %

student 5 in group 1 with confidence 3.99999999999999 %

But these confidence levels are reliable only is the linear regression assumptions are valid. No need to worry about independence, normality, or linearity, but equal variance will always be an issue when the dependent variable is categorical

> plot(predict(mdl\_lin), residuals(mdl\_lin))



So equal variance is questionable here and the prediction intervals are probably unreliable.

1. Use a nearest to the centroid of each group approach to classifying the new applicants. This is actually a k-means clustering approach where the clusters are given (the centers) rather than discovered. Again, start with a plot of the centroids and the distances from the points to be classified.
2. What is the classification decision rule for this model? How does it compare with any of your answers in question (1)?
3. Use the model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted? Hint: confidence has to do with how sure you are in the result of the decision rule.
4. Compare the results to the other approaches used. What are the advantages and disadvantages of this approach?

Let’s get the centers of each group.

> centers <- with(MBAStudents, data.frame(GPA=c(mean(GPA[Rating==1]),mean(GPA[Rating==2])),GMAT=c(mean(GMAT[Rating==1]),mean(GMAT[Rating==2])), row.names = c("C1","C2") ) )

> centers

GPA GMAT

C1 3.185143 598.2286

C2 2.684286 527.0571

> row.names(NewApplicants) = paste("A", 1:5, sep="")

> par(pty="s")

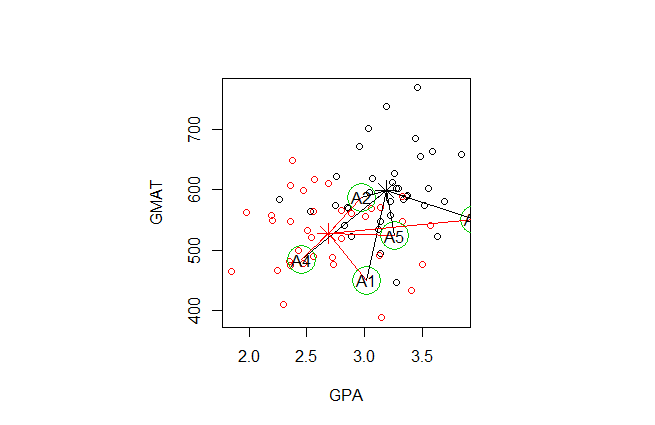
> plot(MBAStudents[,3:4],col=MBAStudents$Rating)

> points(centers, col=1:2, pch=8, cex = 2)

> points(NewApplicants[,2:3], col=3, cex = 4)

> for(j in 1:nrow(centers)) {for(i in 1:nrow(NewApplicants)) {lines(x=c(NewApplicants[i,2], centers[j,1]), y=c(NewApplicants[i,3], centers[j,2]), lty="solid", col=j)}}

> text(NewApplicants[,2:3],labels=row.names(NewApplicants))



We can make our own distance function but let’s use an R package that computes for a matrix for convenience. There are also many choices for distance measures. We’ll use the default “Euclidean” distance as it is is easy to interpret. Also, the scales are not in comparable units so the distance from GPA and GMAT will not be equal. We can re-scale by dividing by the maximum of GPA (4.0) and GMAT (800) or maybe better to “normalize” the data by computing Z-scores for each.

> library("stats", lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")

> library("BBmisc", lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")

par(pty="s")



Because normalizing must normalize the New Applicants with the MBAstudents data and centers.

N<-scale(rbind(MBAStudents[,3:4],NewApplicants[,2:3],centers))

> DT <- as.matrix(stats::dist(N))

RT <- (DT["C1",] > DT["C2",])\*1 + 1

MBAStudents$Pred\_dist <- RT[1:70]

> confusionMatrix(MBAStudents$Pred\_dist, MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 26 8

2 9 27

Accuracy : 0.7571

95% CI : (0.6399, 0.8517)

No Information Rate : 0.5

P-Value [Acc > NIR] : 9.6e-06

Kappa : 0.5143

Mcnemar's Test P-Value : 1

Sensitivity : 0.7429

Specificity : 0.7714

Pos Pred Value : 0.7647

Neg Pred Value : 0.7500

Prevalence : 0.5000

Detection Rate : 0.3714

Detection Prevalence : 0.4857

Balanced Accuracy : 0.7571

'Positive' Class : 1

Seems a little less accurate than discriminant, but not by much.

> RT[paste("A", 1:5, sep="")]

A1 A2 A3 A4 A5

2 1 1 2 1

> NewApplicants$Pred\_dist <- RT[paste("A", 1:5, sep="")]

Even still, it agrees with discriminant classification. Confidence is ratio of difference in distance over the max distance

> abs(DT["C1",paste("A", 1:5, sep="")] - DT["C2",paste("A", 1:5, sep="")])/max(DT["C1",paste("A", 1:5, sep="")], DT["C2",paste("A", 1:5, sep="")])

A1 A2 A3 A4 A5

0.29501701 0.20143124 0.35347003 0.53386949 0.06767402

The classification rule is if Dist(GPA,GMAT,C1) < Dist(GPA,GMAT,C2) rating is 1, otherwise rating is 2. The discriminant is Dist(P, C1) = Dist(P ,C2) for some point P = (GPA,GMAT) and centers C1, C2. If the distance function is Euclidian, this is a simple line (easy to work out but let’s just plot the implicit function):

> usr <- par('usr')

> x <- c(usr[1],usr[2])

> y <- c(usr[3],usr[4])

> out <- outer(x,y,function(x,y) (x-N[76,1])^2-(x-N[77,1])^2 + (y-N[76,2])^2-(y-N[77,2])^2)

> contour(x,y,out,levels=0, add=TRUE, lty=2)



Not surprising that it is similar to the linear discriminant

1. Use a nearest k neighbors (or knn) approach to classifying the new applicants.
2. What is the classification decision rule for this model? How does it compare with any of your answers in question (1)?
3. Use the model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted?
4. Compare the results to the other approaches used. What are the advantages and disadvantages of this approach?

Yet another approach is to look at what classification of a few neighbors and use the majority. This is the k-nearest neighbors or knn classification. The nice thing with this method is that the data does not need to be normalized. We’ll use the class package’s knn function but there are other packages or can do this visually.

> library(caret)

> library(class)

Have to use normalized data because of different distance scales

> NN<- as.data.frame(cbind(N[1:70,], MBAStudents[,2]))

> class::knn(train=NN[,1:2], test=N[71:75,], cl= NN[,3])

[1] 1 1 1 2 1

Levels: 1 2

Hmmm, kinda crappy! Let’s see if we can do better with more neighbors. But how many more?

> indxTrain <- createDataPartition(y = NN[,3],p = 0.75,list = FALSE)

> training <- MBAStudents[indxTrain,]

> testing <- MBAStudents[-indxTrain,]

> set.seed(400)

> ctrl <- trainControl(method="repeatedcv",repeats = 3)

> knnFit <- train(factor(Rating) ~ GPA + GMAT, data = training, method = "knn", trControl = ctrl,tuneLength = 20)

> knnFit

k-Nearest Neighbors

54 samples

2 predictor

2 classes: '1', '2'

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 3 times)

Summary of sample sizes: 48, 49, 50, 48, 49, 49, ...

Resampling results across tuning parameters:

k Accuracy Kappa

5 0.6383333 0.2554779

7 0.6550000 0.2917749

9 0.6383333 0.2659396

11 0.6633333 0.3074204

13 0.6777778 0.3473193

15 0.6811111 0.3447941

17 0.7000000 0.3882340

19 0.7138889 0.4195416

21 0.7205556 0.4341936

23 0.7300000 0.4507382

25 0.7216667 0.4340715

27 0.7272222 0.4451826

29 0.7344444 0.4619714

31 0.7450000 0.4802253

33 0.7411111 0.4706294

35 0.7050000 0.4007382

37 0.6983333 0.3879176

39 0.7150000 0.4212510

41 0.7216667 0.4340715

43 0.7316667 0.4545843

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 31.

K=31 neighbors seems most accurate with training data.

> mdl\_knn31 <- knn3(factor(Rating) ~ GPA + GMAT, NN, k=31)

> Pred\_knn31 <- predict(mdl\_knn33, newdata = N[71:75,])

>

> Pred\_knn31

1 2

[1,] 0.3030303 0.6969697

[2,] 0.6363636 0.3636364

[3,] 0.6969697 0.3030303

[4,] 0.2727273 0.7272727

[5,] 0.6363636 0.3636364

This is the proportion of the 31 neighbors each new applicant has for group 1 and 2. It is also the degree of confidence we can have in the classification.

The classification rule is if #neighbors group 1 > #neighbors group 2 then group 1

> NewApplicants$Pred\_knn31 <- predict(mdl\_knn31, newdata = N[71:75,], type="class")

> NewApplicants$Pred\_knn31

[1] 2 1 1 2 1

Levels: 1 2

Same predictions as the other methods.

> decisionplot(mdl\_knn31, NN, class = "Rating", main = "31nn")



Not that much different than the linear discriminant but impossible to interpret.

1. Use a decision tree (or regression tree) approach to classifying the new applicants.
2. What is the classification decision rule for this model? How does it compare with any of your answers in question (1)?
3. Use the model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted?
4. Compare the results to the other approaches used. What are the advantages and disadvantages of this approach?

> library("rpart")

> tree <- rpart(factor(Rating) ~ GPA + GMAT, MBAStudents)

> plot(tree, uniform=TRUE,compress=TRUE,lty=3,branch=0.7)

> text(tree ,all=TRUE,digits=3,use.n=TRUE,cex=0.8,xpd=TRUE)



> TP <- predict(tree, newdata = NewApplicants[,2:3]); TP

1 2

1 0.28571429 0.71428571

2 0.95833333 0.04166667

3 0.53333333 0.46666667

4 0.08333333 0.91666667

5 0.53333333 0.46666667

> (TP[,1] < TP[,2])\*1 + 1

1 2 3 4 5

2 1 1 2 1

Predicts same as other methods. lol

plot(MBAStudents[,3:4], col=MBAStudents$Rating)

lims = par("usr")

segments(2.74,0,2.74,lims[4])

text((2.74+lims[1])/2, (lims[3] + lims[4])/2,labels="2",cex=2,col="red")

segments(2.74,572,lims[2],572)

text((2.74+lims[2])/2, (572 + lims[4])/2,labels="1",cex=2,col="black")

segments(2.74,520.5,lims[2],520.5)

text((2.74+lims[2])/2, (lims[3] + 520.5)/2,labels="2",cex=2,col="red")

text((2.74+lims[2])/2, ( 520.5 + 572)/2,labels="1",cex=2,col="black")



> decisionplot(tree, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "CART")



1. Create a logistic regression model Prob(in group 2) ~ GPA + GMAT
   1. What is the classification decision rule for this model? How does it compare with any of your answers in question (1)?
   2. Use the logistic regression model to predict what groups the applicants in the New Applicants tab would be in. How confident can you be for each student that they will be in the group predicted?
   3. Advantages and disadvantages?

> MBA.logistic <- glm(factor(Rating) ~ GPA + GMAT, data=MBAStudents, family = "binomial")

> summary(MBA.logistic)

Call:

glm(formula = factor(Rating) ~ GPA + GMAT, family = "binomial",

data = MBAStudents)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.91225 -0.58816 0.00844 0.67481 1.92975

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 19.542992 4.735321 4.127 3.67e-05 \*\*\*

GPA -3.039031 0.880379 -3.452 0.000557 \*\*\*

GMAT -0.018905 0.006216 -3.041 0.002355 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 97.041 on 69 degrees of freedom

Residual deviance: 60.101 on 67 degrees of freedom

AIC: 66.101

Number of Fisher Scoring iterations: 5

GPA and GMAT are both significantly predictive. For every increase in GPA by one point, the log odds of unsatisfactory 2) (versus satisfactory) decreases by 3.039.

For a one point increase in GMAT, the log odds of being unsatisfactory decreases by 0.0189.

Log odds are kinda odd to interpret, so let’s exponentiate:

> exp(coef(MBA.logistic))

(Intercept) GPA GMAT

3.071946e+08 4.788126e-02 9.812726e-01

This is a ratio of decrease, so to make this even easier, look at the reciprocal to get the odds decrease:

> 1/exp(MBA.logistic$coefficients)

(Intercept) GPA GMAT

3.255266e-09 2.088500e+01 1.019085e+00

So we see that each point increase of GPA decreases chance of unsatisfactory 20x and a 10 point increase in GMAT decreases this chance by about 10x.

a<-predict(MBA.logistic, newdata = NewApplicants, type="response")

> a

1 2 3 4 5

0.86504977 0.35889843 0.05326095 0.95013855 0.43279556

> round(a)+1

1 2 3 4 5

2 1 1 2 1

Predicts same as other methods but with more reliable confidence and interpretable coefficients. Just as accurate:

> confusionMatrix(MBAStudents$Rating, round(b)+1)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 26 9

2 8 27

Accuracy : 0.7571

95% CI : (0.6399, 0.8517)

No Information Rate : 0.5143

P-Value [Acc > NIR] : 2.676e-05

Kappa : 0.5143

Mcnemar's Test P-Value : 1

Sensitivity : 0.7647

Specificity : 0.7500

Pos Pred Value : 0.7429

Neg Pred Value : 0.7714

Prevalence : 0.4857

Detection Rate : 0.3714

Detection Prevalence : 0.5000

Balanced Accuracy : 0.7574

'Positive' Class : 1

> class(MBA.logistic) <- c("lr", class(MBA.logistic))

> predict.lr <- function(object, newdata, ...)

+ predict.glm(object, newdata, type = "response") > .5

> decisionplot(MBA.logistic, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "logistic")



Let’s look at the relationship between GPA, GMAT, and likelihood of being in Group 2:

> newdata <- expand.grid(

+ GPA = pretty(MBAStudents$GPA, 20),

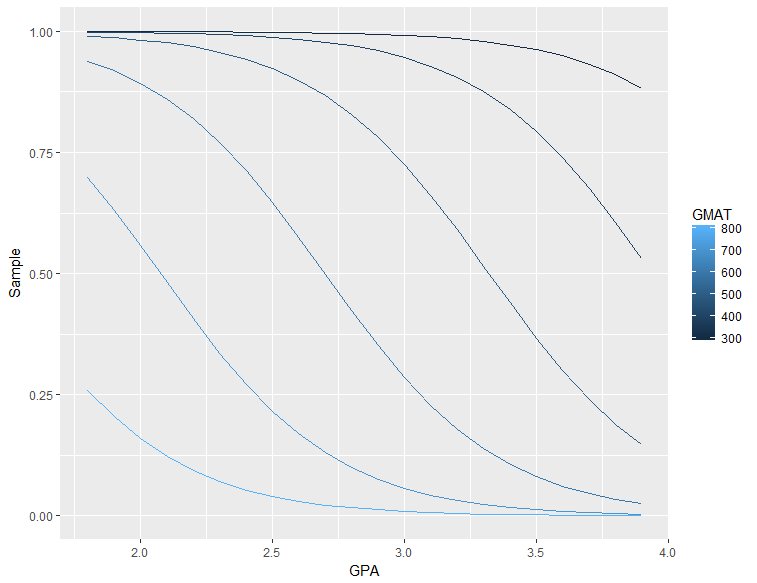
+ GMAT = pretty(MBAStudents$GMAT, 5))

> newdata$Sample <- predict(MBA.logistic, newdata = newdata, type = "response")

> library(ggplot2)

> ggplot(newdata, aes(x = GPA, y = Sample, colour = GMAT, group = GMAT)) +

+ geom\_line()



1. [**Try this exercise on your own]**

Support Vector Machines (SVM) is a general approach to supervised classification and non-linear regression whose discriminant maximizes the separation of groups. Read about SVM (Ch. 9 in the ISLR book or Wikipedia or the R libsvm package overview at <https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf>)

1. Perform an analysis of SVM using different kernels. Compare on accuracy, reliability, and ability to interpret the classification decision rule.
2. Compare the SVM decision rules with the other methods used previously. How are they similar? Where do they differ?
3. Use your SVM models to make predictions of the classification for new applicants. How do these compare with predictions with the methods used previously?
4. Would you recommend using SVM for this classification problem? If so, which kernel and why.

> library(e1071)

>

> mdl\_svm <- svm(factor(Rating) ~ GPA + GMAT, data = MBAStudents, kernel = "linear")

> confusionMatrix(predict(mdl\_svm), MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 25 6

2 10 29

Accuracy : 0.7714

95% CI : (0.6555, 0.8633)

No Information Rate : 0.5

P-Value [Acc > NIR] : 2.927e-06

Kappa : 0.5429

Mcnemar's Test P-Value : 0.4533

Sensitivity : 0.7143

Specificity : 0.8286

Pos Pred Value : 0.8065

Neg Pred Value : 0.7436

Prevalence : 0.5000

Detection Rate : 0.3571

Detection Prevalence : 0.4429

Balanced Accuracy : 0.7714

'Positive' Class : 1

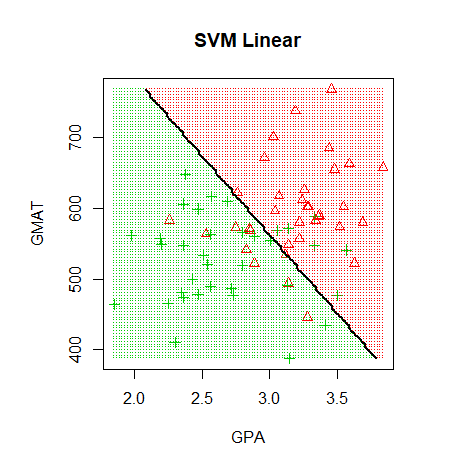
> predict(mdl\_svm, newdata = NewApplicants)

A1 A2 A3 A4 A5

2 1 1 2 1

Levels: 1 2

> decisionplot(mdl\_svm, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "SVM")



> mdl\_svm <- svm(factor(Rating) ~ GPA + GMAT, data = MBAStudents, kernel = "polynomial")

> confusionMatrix(predict(mdl\_svm), MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 31 13

2 4 22

Accuracy : 0.7571

95% CI : (0.6399, 0.8517)

No Information Rate : 0.5

P-Value [Acc > NIR] : 9.6e-06

Kappa : 0.5143

Mcnemar's Test P-Value : 0.05235

Sensitivity : 0.8857

Specificity : 0.6286

Pos Pred Value : 0.7045

Neg Pred Value : 0.8462

Prevalence : 0.5000

Detection Rate : 0.4429

Detection Prevalence : 0.6286

Balanced Accuracy : 0.7571

'Positive' Class : 1

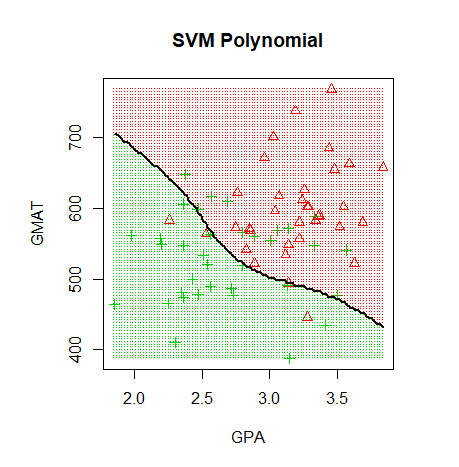
> predict(mdl\_svm, newdata = NewApplicants)

A1 A2 A3 A4 A5

2 1 1 2 1

Levels: 1 2

> decisionplot(mdl\_svm, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "SVM Polynomial")



> mdl\_svm <- svm(factor(Rating) ~ GPA + GMAT, data = MBAStudents, kernel = "radial")

> confusionMatrix(predict(mdl\_svm), MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 28 7

2 7 28

Accuracy : 0.8

95% CI : (0.6873, 0.8861)

No Information Rate : 0.5

P-Value [Acc > NIR] : 2.151e-07

Kappa : 0.6

Mcnemar's Test P-Value : 1

Sensitivity : 0.8

Specificity : 0.8

Pos Pred Value : 0.8

Neg Pred Value : 0.8

Prevalence : 0.5

Detection Rate : 0.4

Detection Prevalence : 0.5

Balanced Accuracy : 0.8

'Positive' Class : 1

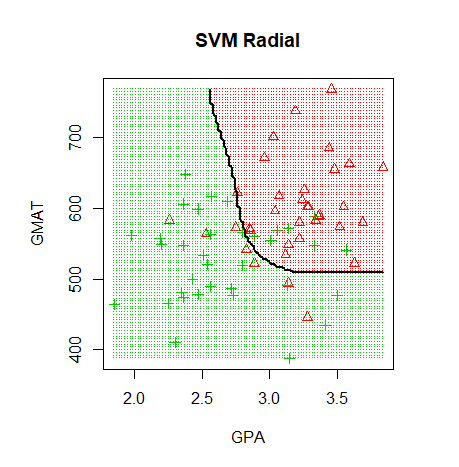
> predict(mdl\_svm, newdata = NewApplicants)

A1 A2 A3 A4 A5

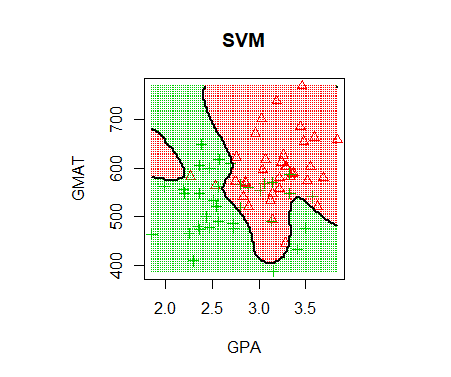
2 1 1 2 1

Levels: 1 2

> decisionplot(mdl\_svm, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "SVM Radial")



> mdl\_svm <- svm(factor(Rating) ~ GPA + GMAT, data = MBAStudents, kernel = "radial", cost=100)



> mdl\_svm <- svm(factor(Rating) ~ GPA + GMAT, data = MBAStudents, kernel = "sigmoid")

> confusionMatrix(predict(mdl\_svm), MBAStudents$Rating)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 24 11

2 11 24

Accuracy : 0.6857

95% CI : (0.5637, 0.7915)

No Information Rate : 0.5

P-Value [Acc > NIR] : 0.001274

Kappa : 0.3714

Mcnemar's Test P-Value : 1.000000

Sensitivity : 0.6857

Specificity : 0.6857

Pos Pred Value : 0.6857

Neg Pred Value : 0.6857

Prevalence : 0.5000

Detection Rate : 0.3429

Detection Prevalence : 0.5000

Balanced Accuracy : 0.6857

'Positive' Class : 1

> predict(mdl\_svm, newdata = NewApplicants)

A1 A2 A3 A4 A5

2 1 1 2 2

> decisionplot(mdl\_svm, as.data.frame(cbind(MBAStudents[1:70,3:4], MBAStudents[,2])), class = "Rating", main = "SVM Sigmoid")

