Lab: Logistic Regression LDA QDA KNN

#4.6.1 The Stock Market data

In [89]:
```
library(ISLR)
names(Smarket)
```

'Year' 'Lag1' 'Lag2' 'Lag3' 'Lag4' 'Lag5' 'Volume' 'Today' 'Direction'

In [90]:
```
dim(Smarket)
```

1250 9

In [91]:

```
summary(Smarket)
```

```
      Year           Lag1                Lag2                Lag3
 Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92
2000
 1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.64
0000
 Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.03
8500
 Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.00
1716
 3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59
6750
 Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73
3000
      Lag4                Lag5              Volume           Today
 Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.9
22000
 1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.6
39500
 Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.0
38500
 Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.0
03138
 3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.5
96750
 Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.7
33000
 Direction
 Down:602
 Up  :648
```

In [92]:

```
#cor(Smarket)
#generating error, because Dir is not numremic

#lag1-lag5- %return of previous trading days
#Volume-No. of shares traded of previous days in billion
#Today- %age return on the date in question
#Direction- whether market was up or dowm
?Smarket
```

In [93]:

```
cor(Smarket[, -9])
#produces a matrix that contains all of the pairwise
#correlations among the predictors in a data set
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	
Year	1.00000000	0.029699649	0.030596422	0.033194581	0.035688718	0.029787995	0.5
Lag1	0.02969965	1.000000000	-0.026294328	-0.010803402	-0.002985911	-0.005674606	0.0
Lag2	0.03059642	-0.026294328	1.000000000	-0.025896670	-0.010853533	-0.003557949	-0.0
Lag3	0.03319458	-0.010803402	-0.025896670	1.000000000	-0.024051036	-0.018808338	-0.0
Lag4	0.03568872	-0.002985911	-0.010853533	-0.024051036	1.000000000	-0.027083641	-0.0
Lag5	0.02978799	-0.005674606	-0.003557949	-0.018808338	-0.027083641	1.000000000	-0.0
Volume	0.53900647	0.040909908	-0.043383215	-0.041823686	-0.048414246	-0.022002315	1.0
Today	0.03009523	-0.026155045	-0.010250033	-0.002447647	-0.006899527	-0.034860083	0.0

In [94]:

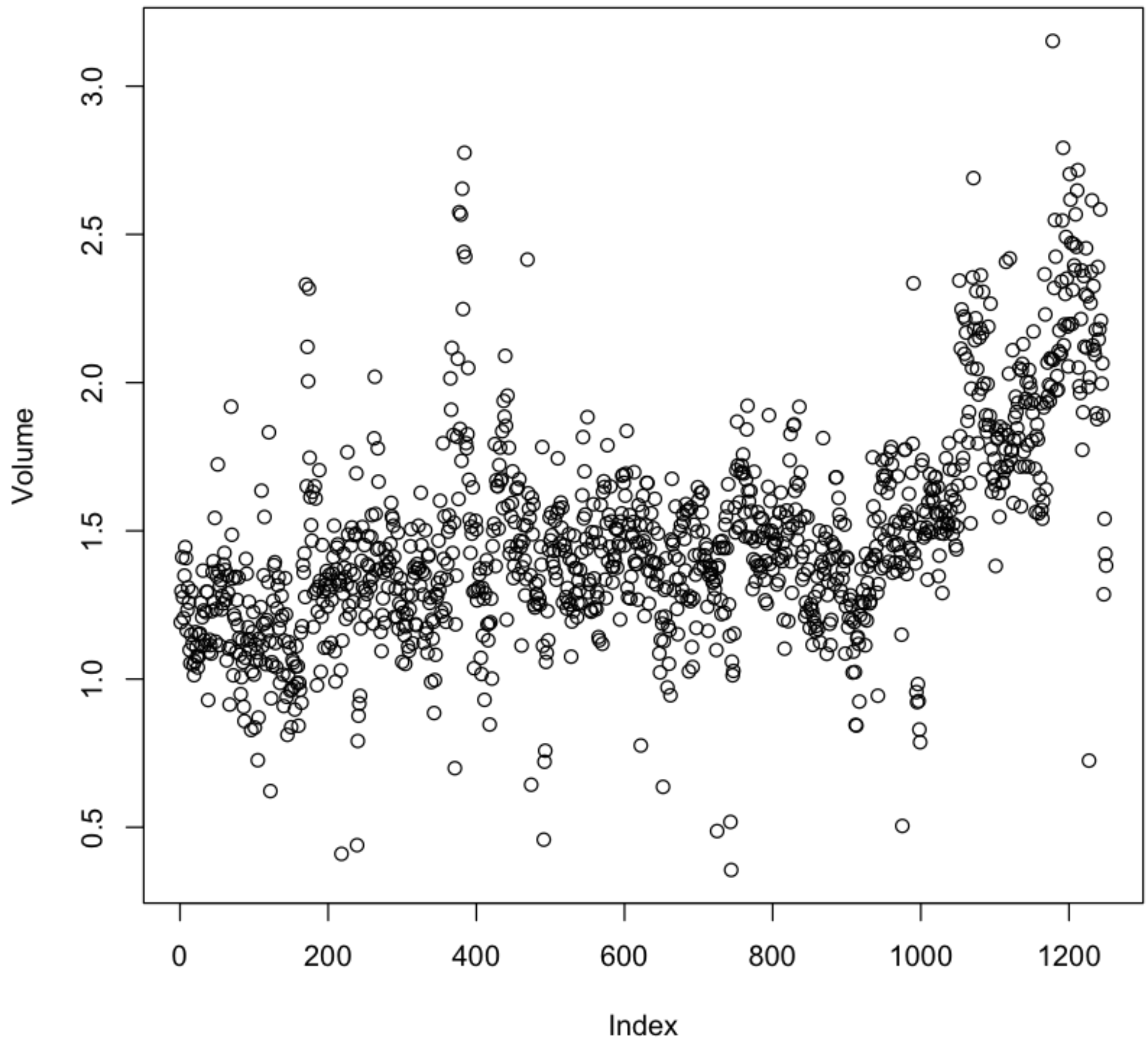
```
#observation:correlations between the lag variables and
#today’s returns are close to zero
#little correlation between today’s returns and previous days’(vol
```

In [95]:
```
attach(Smarket)
plot(Volume)

#vol is increasing
```

```
The following objects are masked from Smarket (pos = 6):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year
```



4.6.2 Logistic regression

In [96]:

```
glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarke

# glm() function fits generalized linear models,
#a class of models that includes logistic regression
```

In [97]:

```
summary(glm.fits)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + 
    Volume, family = binomial, data = Smarket)

Deviance Residuals: 
   Min      1Q  Median      3Q     Max  
-1.446  -1.203   1.065   1.145   1.326  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

In [98]:

```
#observations
#lag1 has min p-value,
#negative coefficient for this predictor suggests that-
#if the market had a positive return yesterday,
#then it is less likely to go up today

#however, 0.15 is still large value of p
# so there is no clear evidence of a real association
#between Lag1 and Direction.
```

In [99]:
```
#coef() function in order to access the
#coefficients for this fitted model.
coef(glm.fits)
```

(Intercept)	-0.126000256559266
Lag1	-0.0730737458900261
Lag2	-0.0423013440073083
Lag3	0.0110851083796762
Lag4	0.0093589383702787
Lag5	0.0103130684758179
Volume	0.13544065885916

In [100]:
```
summary(glm.fits)$coef
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.126000257	0.24073574	-0.5233966	0.6006983
Lag1	-0.073073746	0.05016739	-1.4565986	0.1452272
Lag2	-0.042301344	0.05008605	-0.8445733	0.3983491
Lag3	0.011085108	0.04993854	0.2219750	0.8243333
Lag4	0.009358938	0.04997413	0.1872757	0.8514445
Lag5	0.010313068	0.04951146	0.2082966	0.8349974
Volume	0.135440659	0.15835970	0.8552723	0.3924004

In [101]:
```
summary(glm.fits)$coef[,4]
```

(Intercept)	0.600698319413355
Lag1	0.145227211568647
Lag2	0.398349095427021
Lag3	0.824333346101536
Lag4	0.851444506926455
Lag5	0.834997390499829
Volume	0.392400433202429

In [102]:

```
#The predict() function can be used to predict the probability tha
#the market will go up, given values of the predictors
#The predict() function is used similary to generate predictions
#for the response variable.
glm.probs=predict(glm.fits,type="response")
glm.probs[1:10] #printed only the first 10 probabilities
```

1	0.507084133395402
2	0.481467878454591
3	0.481138835214201
4	0.515222355813022
5	0.510781162691538
6	0.506956460534911
7	0.492650874187038
8	0.509229158207377
9	0.517613526170958
10	0.488837779771376

In [103]:

```
contrasts (Direction)
#Use the contrasts() function to see the dummy variables generatec
#for values in the categorical variable Direction.

#In order to make a prediction as to whether the market will go
#up or down on a particular day, we must convert these predicted
#probabilities into class labels, Up or Down.
```

	Up
Down	0
Up	1

In [104]:

```
glm.pred=rep("Down",1250)  #creates a vector of 1,250 Down element
glm.pred[glm.probs >.5]="Up" #transforms to Up all of the elements

#creating a vector of class predictions based on whether
#the predicted probability of a market increase is
#greater than or less than 0.5

```

In [105]:

```
#We can generate a confusion matrix between the predicted directic
#and the actual direction from the variable Direction
#using the table()function.

table(glm.pred,Direction) #confusion matrix
```

```
        Direction
glm.pred Down  Up
    Down  145 141
    Up    457 507
```

In [106]:

```
#diagonal elements of the confusion matrix indicate correct predic
#while the off-diagonals represent incorrect predictions.
#market would go up on 507 days and that it would go down on 145 c

(507+145) /1250  #correct prediction

#logistic regression correctly predicted the movement of the marke
#training error--> 100 - 52.2 = 47.8 %
```

0.5216

In [107]:

```
#We then divide our dataset into training set and test set.
#The training set will include observations from 2001-2004
#and the test set from the year 2005.

train=(Year <2005) #train is a vector of 1,250 elements as observa
Smarket.2005= Smarket[! train ,] #pick submatrix of market data, b
dim(Smarket.2005)
```

252 9

In [108]:

```
Direction.2005=Direction[!train]
```

In [109]:

```
glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume , data=Smar
```

In [110]:

```
glm.probs=predict(glm.fits,Smarket.2005,type="response")
```

In [111]:

```
#we have trained and tested our model on two completely separate-
# training was performed using only the dates before 2005,
#and testing was performed using only the dates in 2005.
```

In [112]:
```
glm.pred=rep("Down",252)
glm.pred[glm.probs >.5]="Up"
table(glm.pred,Direction.2005)

#To improve the preditive performance, we can restrict the predict
#to only those with the strongest relationship to the response var
#In this case, we limit the variables to Lag1 and Lag2.

```

```
        Direction.2005
glm.pred Down Up
    Down   77 97
    Up     34 44
```

In [113]:
```
mean(glm.pred==Direction.2005)
```

0.48015873015873

In [114]:
```
 mean(glm.pred!=Direction.2005)
```

0.51984126984127

In [115]:
```
glm.fits=glm(Direction~Lag1+Lag2,data=Smarket ,family=binomial, su
```

In [116]:
```
predict(glm.fits,newdata=data.frame(Lag1=c(1.2,1.5),
Lag2=c(1.1,-0.8)),type="response")
```

1	0.479146239171912
2	0.496093872956532

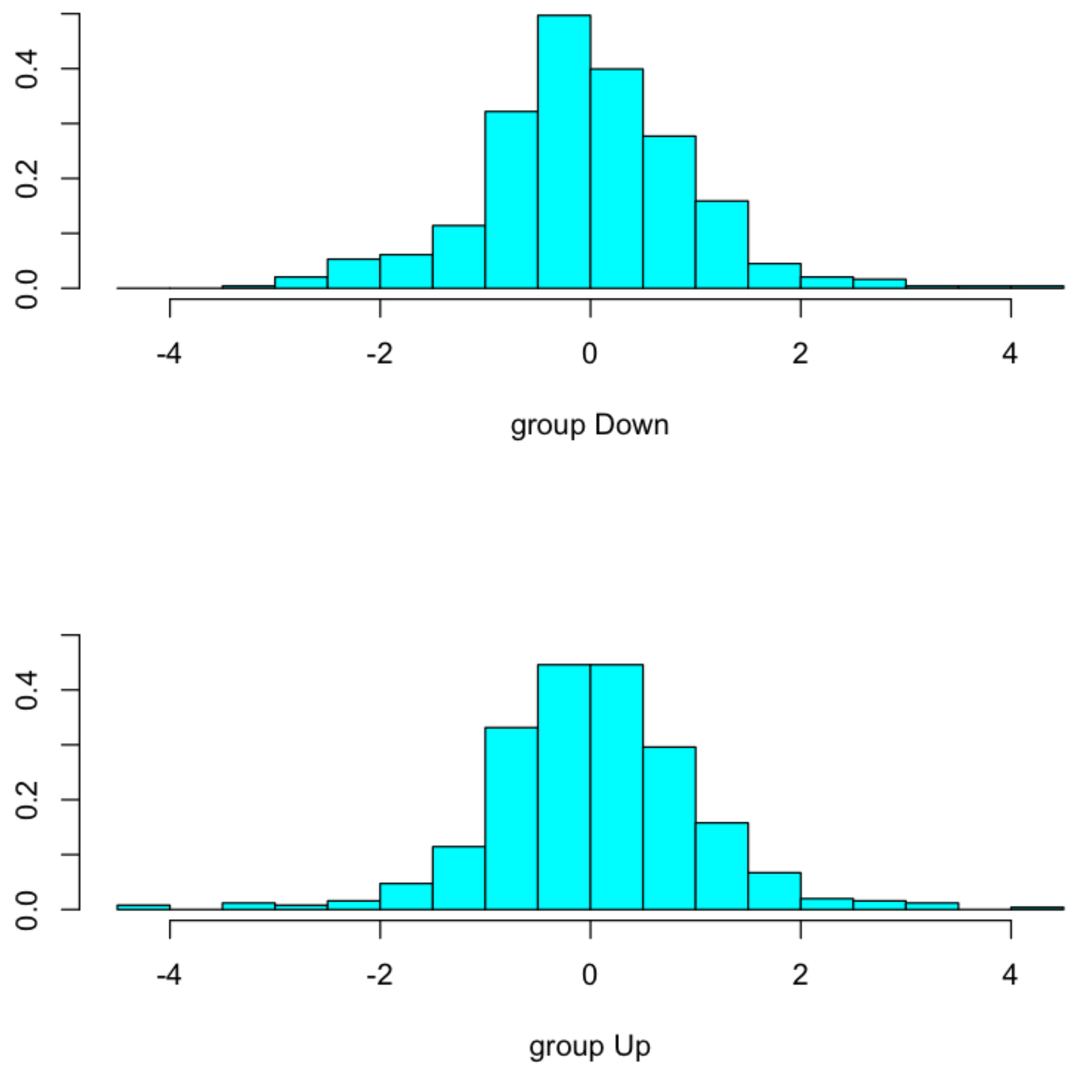
Linear Discriminant analysis

In [117]:
```
library(MASS)
```

In [118]:
```
lda.fit=lda(Direction~Lag1+Lag2,data=Smarket ,subset=train)
```

In [119]:

```
plot(lda.fit)
```



In [120]:

```
lda.pred = predict(lda.fit, Smarket.2005)
names(lda.pred)

#The preditc() function for an LDA model returns
#list of three elements representing the predicted class,
#the posterior probabilities and the linear discriminants
#
```

'class' 'posterior' 'x'

In [121]:
```
lda.class = lda.pred$class
table(lda.class, Direction.2005)

#comparing the predicted class with the predicted directed obtaine
#from logistic regression reviously and stored in the vector Direc
```

```
          Direction.2005
lda.class Down  Up
     Down   35  35
     Up     76 106
```

In [122]:
```
mean(lda.class == Direction.2005)
```

0.55952380952381

In [123]:
```
sum(lda.pred$posterior[, 1] >= 0.5)
```

70

In [124]:
```
sum(lda.pred$posterior[, 1] < 0.5)
```

182

In [125]:
```
lda.pred$posterior[1:20, 1]
#We can inspect the posterior probabilities of the LDA model
#from the posterior vector of the fitted model.
```

999	0.490179249818258
1000	0.479218499099683
1001	0.466818479852065
1002	0.474001069455248
1003	0.492787663967445
1004	0.493856154997504
1005	0.495101564646223
1006	0.487286099421815
1007	0.490701348960405
1008	0.484402624071869
1009	0.490696276120968
1010	0.511998846261919
1011	0.489515226936648
1012	0.470676122211879
1013	0.474459285611829
1014	0.479958339148108
1015	0.493577529465861
1016	0.503089377118306
1017	0.497880612141404
1018	0.488633086516518

In [126]:
```
lda.class[1:20]
```

Up Up Up Up Up Up Up Up Up Up Up Down Up Up Up Up Up Down Up Up

▶ **Levels**:

In [127]:
```
#We can also set the posterior probabilities to different
#thresholds for making predictions.
```

4.6.4 Quadratic Discriminant Analysis

In [128]:
```
qda.fit = qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = tr
qda.fit
```

```
Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
    Down       Up
0.491984 0.508016

Group means:
            Lag1        Lag2
Down  0.04279022  0.03389409
Up   -0.03954635 -0.03132544
```

In [129]:
```
#We can make predictions using predict() just as we did for an LDA
#and compare them to the results from the logistic regression
qda.class = predict(qda.fit, Smarket.2005)$class
table(qda.class, Direction.2005)
```

```
         Direction.2005
qda.class Down  Up
     Down   30  20
     Up     81 121
```

In [130]:
```
mean(qda.class == Direction.2005)
```

0.599206349206349

#4.6.5 K-Nearest Neighbors

In [131]:
```
#The class package offers a number of classification algorithms
#including K-Nearest Neighbors. Before we can run the KNN algorith
#we need to split our dataset into training and test subsets.
#After splitting the dataset, the cbind() is used to bind the Lag1
#into a matrix for each subset.

library(class)
train.X=cbind(Lag1 ,Lag2)[train ,]  # training data,
test.X=cbind(Lag1,Lag2)[!train,]   # predictors associated with th
train.Direction=Direction [train]  #class labels for the training
```

In [132]:
```
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2005)

# set.seed() to ensure that repeated runs produce consistent resul
#make predictions about the market direction in 2005.
```

```
        Direction.2005
knn.pred Down Up
    Down   43 58
    Up     68 83
```

In [133]:
```
(83 + 43)/252
```

0.5

In [134]:
```
knn.pred <- knn(train.X, test.X, train.Direction, k = 3)
table(knn.pred, Direction.2005)

#can repeat the fit with K = 3.
```

```
        Direction.2005
knn.pred Down Up
    Down   48 54
    Up     63 87
```

In [135]:
```
mean(knn.pred == Direction.2005)
```

0.535714285714286

4.6.6 An Application to Caravan Insurance Data

In [136]:

```
attach(Caravan) #function to make the Caravan dataset available
```

```
The following objects are masked from Caravan (pos = 4):

    AAANHANG, ABESAUT, ABRAND, ABROM, ABYSTAND, AFIETS, AGEZONG,
    AINBOED, ALEVEN, AMOTSCO, APERSAUT, APERSONG, APLEZIER, ATRACTOR,
    AVRAAUT, AWABEDR, AWALAND, AWAOREG, AWAPART, AWERKT, AZEILPL,
    MAANTHUI, MAUT0, MAUT1, MAUT2, MBERARBG, MBERARBO, MBERBOER,
    MBERHOOG, MBERMIDD, MBERZELF, MFALLEEN, MFGEKIND, MFWEKIND,
    MGEMLEEF, MGEMOMV, MGODGE, MGODOV, MGODPR, MGODRK, MHHUUR, MHKOOP
,
    MINK123M, MINK3045, MINK4575, MINK7512, MINKGEM, MINKM30, MKOOPKL
A,
    MOPLHOOG, MOPLLAAG, MOPLMIDD, MOSHOOFD, MOSTYPE, MRELGE, MRELOV,
    MRELSA, MSKA, MSKB1, MSKB2, MSKC, MSKD, MZFONDS, MZPART, PAANHANG
,
    PBESAUT, PBRAND, PBROM, PBYSTAND, PFIETS, PGEZONG, PINBOED, PLEVE
N,
    PMOTSCO, PPERSAUT, PPERSONG, PPLEZIER, PTRACTOR, PVRAAUT, PWABEDR
,
    PWALAND, PWAOREG, PWAPART, PWERKT, PZEILPL, Purchase
```

In [137]:

```
dim(Caravan)
```

5822 86

In [138]:

```
summary(Purchase)
```

No	5474
Yes	348

In [139]:

```
348/5822
```

0.0597732737890759

In [140]:

```
standardized.X <- scale(Caravan[, -86])
var(Caravan[, 1])

#scale() function to scale the dataset with a mean of zero and sta
```

165.037847395189

In [141]:
```
var(Caravan[, 2])
```

0.164707781931954

In [142]:
```
var(standardized.X[, 1])
```

1

In [143]:
```
var(standardized.X[, 2])
```

1

In [144]:
```
test = 1:1000
train.X = standardized.X[-test, ]
test.X = standardized.X[test, ]
train.Y = Purchase[-test]
test.Y = Purchase[test]
set.seed(1)
knn.pred = knn(train.X, test.X, train.Y, k = 1)
mean(test.Y != knn.pred)
```

0.118

In [145]:
```
mean(test.Y != "No")
```

0.059

In [146]:
```
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  873  50
     Yes  68   9
```

In [147]:
```
9/(68 + 9)
```

0.116883116883117

In [148]:
```
knn.pred = knn(train.X, test.X, train.Y, k = 3)  #takeing k = 3
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  920  54
     Yes  21   5
```

In [149]:
```
5/26
```

0.192307692307692

In [150]:
```
knn.pred = knn(train.X, test.X, train.Y, k = 5) #k = 5
table(knn.pred, test.Y)
```

```
        test.Y
knn.pred  No Yes
     No  930  55
     Yes  11   4
```

In [151]:
```
glm.fit = glm(Purchase ~ ., data = Caravan, family = binomial, sub

#compare the KNN model with a logistic regression using glm() and
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

In [152]:
```
glm.probs = predict(glm.fit, Caravan[test, ], type = "response")
glm.pred = rep("No", 1000)
glm.pred[glm.probs > 0.5] <- "Yes"
table(glm.pred, test.Y)
```

```
        test.Y
glm.pred  No Yes
     No  934  59
     Yes   7   0
```

In [155]:
```
glm.pred = rep("No", 1000)
glm.pred[glm.probs > 0.25] = " Yes"
table(glm.pred, test.Y)
```

```
        test.Y
glm.pred  No Yes
     Yes  22  11
    No   919  48
```

In [156]:
```
11/(22 + 11)
```

0.333333333333333