Classification

Aleezah Athar

In classification, our target variable is qualitative. We want to know what class an observation falls into. Usually, the target variable in classification is binary. Linear models for classification create decision boundaries to separate observations into regions in which a majority of the observations belong to the same class. Classification linear models are useful when the target variable is qualitative which is an advantage. A disadvantage would be that we cannot use these when our target variable is quantitative, in which case linear regression would have more advantages. Classification linear models also have the general advantages and disadvantages of all linear models. Linear models' advantages are that they are easy to interpret, computationally efficient, can handle missing data points, and can be extended to non-linear relationships if we use complex transformation functions. However, the disadvantages are that they assume a linear relationship where there might not be one and tend to overfit.

This dataset is from https://archive.ics.uci.edu/ml/datasets/Bank+Marketing (https://archive.ics.uci.edu/ml/datasets/Bank+Marketing)

Data Exploration

First we set the seed to 3 to get the same results each time Then we read in the csv file which contains our data

```
set.seed(3)
df <- read.csv("bank-full.csv", sep = ";", quote = "")
df
```

X.age. <int>	X.job. <chr>	X.marital. <chr>	X.education. <chr>	X.default. <chr>	X.balance. <int>	X.housing. <chr>	X.loan <chr>
58	"management"	"married"	"tertiary"	"no"	2143	"yes"	"no"
44	"technician"	"single"	"secondary"	"no"	29	"yes"	"no"
33	"entrepreneur"	"married"	"secondary"	"no"	2	"yes"	"yes"
47	"blue-collar"	"married"	"unknown"	"no"	1506	"yes"	"no"
33	"unknown"	"single"	"unknown"	"no"	1	"no"	"no"
35	"management"	"married"	"tertiary"	"no"	231	"yes"	"no"
28	"management"	"single"	"tertiary"	"no"	447	"yes"	"yes"
42	"entrepreneur"	"divorced"	"tertiary"	"yes"	2	"yes"	"no"
58	"retired"	"married"	"primary"	"no"	121	"yes"	"no"
43	"technician"	"single"	"secondary"	"no"	593	"yes"	"no"

1-10 of 10,000 rows | 1-8 of 17 columns Previous **1** 2 3 4 5 6 … 1000 Next

We change all the character variables to factor variables so we can do classification. Next, we divide into 80/20 train/test Then we randomly select 80% of the rows to be the training data and 20% to be the testing data.

```
set.seed(3)
for(i in 1:ncol(df)){
  if(is.character(df[,i])){
    df[,i] <- as.factor(df[,i])
  }
}
i<-sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train<-df[i,]
test<-df[-i,]
```

We use at least 5 R functions for data exploration, using the training data. We can use the names, dimension, summary, structure and head functions to get information about the data. We can also use the colSums and is.na functions together to check for any NA values.

```
names(train)
```

```
##  [1] "X.age."       "X.job."       "X.marital."   "X.education." "X.default."
##  [6] "X.balance."   "X.housing."   "X.loan."      "X.contact."   "X.day."
## [11] "X.month."     "X.duration."  "X.campaign."  "X.pdays."     "X.previous."
## [16] "X.poutcome."  "X.y."
```

```
dim(train)
```

```
## [1] 36168    17
```

```
summary(train)
```

```
##      X.age.                X.job.          X.marital.         X.education.
##  Min.   :18.00   "blue-collar":7808   "divorced": 4171   "primary"  : 5494
##  1st Qu.:33.00   "management" :7533   "married" :21724   "secondary":18549
##  Median :39.00   "technician" :6067   "single"  :10273   "tertiary" :10648
##  Mean   :40.89   "admin."     :4124                      "unknown"  : 1477
##  3rd Qu.:48.00   "services"   :3384
##  Max.   :95.00   "retired"    :1813
##                  (Other)      :5439
##  X.default.      X.balance.     X.housing.     X.loan.            X.contact.
##  "no" :35528   Min.   : -8019   "no" :15972   "no" :30426   "cellular" :23370
##  "yes":  640   1st Qu.:    75   "yes":20196   "yes": 5742   "telephone": 2296
##                Median :   450                               "unknown"  :10502
##                Mean   :  1359
##                3rd Qu.:  1416
##                Max.   :102127
##
##      X.day.          X.month.      X.duration.       X.campaign.
##  Min.   : 1.00   "may"  :11076   Min.   :   0.0   Min.   : 1.000
##  1st Qu.: 8.00   "jul"  : 5480   1st Qu.: 103.0   1st Qu.: 1.000
##  Median :16.00   "aug"  : 4919   Median : 179.0   Median : 2.000
##  Mean   :15.79   "jun"  : 4306   Mean   : 257.6   Mean   : 2.773
##  3rd Qu.:21.00   "nov"  : 3188   3rd Qu.: 318.0   3rd Qu.: 3.000
##  Max.   :31.00   "apr"  : 2365   Max.   :4918.0   Max.   :63.000
##                  (Other): 4834
##     X.pdays.        X.previous.          X.poutcome.       X.y.
##  Min.   : -1.00   Min.   :  0.0000   "failure": 3888   "no" :31948
##  1st Qu.: -1.00   1st Qu.:  0.0000   "other"  : 1484   "yes": 4220
##  Median : -1.00   Median :  0.0000   "success": 1199
##  Mean   : 40.22   Mean   :  0.5758   "unknown":29597
##  3rd Qu.: -1.00   3rd Qu.:  0.0000
##  Max.   :871.00   Max.   :275.0000
##
```

```
str(train)
```

```
## 'data.frame':    36168 obs. of  17 variables:
##  $ X.age.      : int  49 26 43 36 32 60 46 44 43 32 ...
##  $ X.job.      : Factor w/ 12 levels "\"admin.\"","\"blue-collar\"",..: 5 9 5 11 2 6
2 2 2 5 ...
##  $ X.marital.  : Factor w/ 3 levels "\"divorced\"",..: 2 3 2 3 2 2 2 1 2 3 ...
##  $ X.education.: Factor w/ 4 levels "\"primary\"",..: 3 2 3 3 2 2 2 2 1 3 ...
##  $ X.default.  : Factor w/ 2 levels "\"no\"","\"yes\"": 1 1 1 1 1 1 1 1 1 1 ...
##  $ X.balance.  : int  0 100 0 953 112 5789 870 558 -124 0 ...
##  $ X.housing.  : Factor w/ 2 levels "\"no\"","\"yes\"": 1 1 1 2 2 2 1 2 2 2 ...
##  $ X.loan.     : Factor w/ 2 levels "\"no\"","\"yes\"": 1 1 1 1 1 1 2 1 1 2 ...
##  $ X.contact.  : Factor w/ 3 levels "\"cellular\"",..: 1 1 1 1 1 1 1 3 3 1 ...
##  $ X.day.      : int  19 26 6 31 20 12 31 7 27 17 ...
##  $ X.month.    : Factor w/ 12 levels "\"apr\"","\"aug\"",..: 2 9 2 2 1 9 6 9 9 10 ...
##  $ X.duration. : int  78 445 124 102 311 50 87 485 47 107 ...
##  $ X.campaign. : int  6 1 2 2 1 5 2 1 5 2 ...
##  $ X.pdays.    : int  -1 -1 -1 101 321 -1 -1 -1 -1 -1 ...
##  $ X.previous. : int  0 0 0 3 1 0 0 0 0 0 ...
##  $ X.poutcome. : Factor w/ 4 levels "\"failure\"",..: 4 4 4 1 2 4 4 4 4 4 ...
##  $ X.y.        : Factor w/ 2 levels "\"no\"","\"yes\"": 1 2 1 1 1 1 1 1 1 1 ...
```

```
head(train)
```

	X.age. <int>	X.job. <fct>	X.marital. <fct>	X.education. <fct>	X.default. <fct>	X.balance. <int>	X.housing. <fct>	X. <f
21479	49	"management"	"married"	"tertiary"	"no"	0	"no"	"n
39610	26	"student"	"single"	"secondary"	"no"	100	"no"	"n
19307	43	"management"	"married"	"tertiary"	"no"	0	"no"	"n
41352	36	"unemployed"	"single"	"tertiary"	"no"	953	"yes"	"n
33098	32	"blue-collar"	"married"	"secondary"	"no"	112	"yes"	"n
36520	60	"retired"	"married"	"secondary"	"no"	5789	"yes"	"n

6 rows | 1-9 of 18 columns

```
colSums(is.na(train))
```

```
##       X.age.       X.job.   X.marital. X.education.   X.default.   X.balance.
##            0            0            0            0            0            0
##   X.housing.      X.loan.   X.contact.       X.day.     X.month.  X.duration.
##            0            0            0            0            0            0
##  X.campaign.     X.pdays.  X.previous.  X.poutcome.         X.y.
##            0            0            0            0            0
```

Data Visualization

Creating 2 informative graphs using the training data.

```
plot(train$X.y.)
```

```
boxplot(train$X.age.)
```

Logistic Regression

Building a logistic regression model and outputting the summary using the glm and summary functions.

```
glm1<-glm(X.y.~., data=train, family=binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = X.y. ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.7706  -0.3727  -0.2520  -0.1486   3.4054
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.595e+00  2.052e-01 -12.643  < 2e-16 ***
## X.age.                  -7.079e-04  2.484e-03  -0.285 0.775699
## X.job."blue-collar"     -3.167e-01  8.216e-02  -3.855 0.000116 ***
## X.job."entrepreneur"    -1.987e-01  1.372e-01  -1.448 0.147586
## X.job."housemaid"       -5.748e-01  1.570e-01  -3.662 0.000250 ***
## X.job."management"      -1.164e-01  8.259e-02  -1.409 0.158846
## X.job."retired"          2.476e-01  1.097e-01   2.257 0.024014 *
## X.job."self-employed"   -3.231e-01  1.276e-01  -2.531 0.011379 *
## X.job."services"        -1.538e-01  9.276e-02  -1.658 0.097340 .
## X.job."student"          4.959e-01  1.208e-01   4.107 4.01e-05 ***
## X.job."technician"      -1.830e-01  7.782e-02  -2.351 0.018722 *
## X.job."unemployed"      -1.363e-01  1.248e-01  -1.092 0.274615
## X.job."unknown"         -3.816e-01  2.745e-01  -1.390 0.164488
## X.marital."married"     -1.869e-01  6.628e-02  -2.821 0.004793 **
## X.marital."single"       6.236e-02  7.562e-02   0.825 0.409613
## X.education."secondary"  1.938e-01  7.270e-02   2.666 0.007687 **
## X.education."tertiary"   3.695e-01  8.447e-02   4.374 1.22e-05 ***
## X.education."unknown"    2.182e-01  1.172e-01   1.862 0.062620 .
## X.default."yes"         -1.122e-02  1.828e-01  -0.061 0.951040
## X.balance.               1.249e-05  5.806e-06   2.152 0.031434 *
## X.housing."yes"         -6.955e-01  4.907e-02 -14.172  < 2e-16 ***
## X.loan."yes"            -4.310e-01  6.758e-02  -6.377 1.80e-10 ***
## X.contact."telephone"   -1.645e-01  8.459e-02  -1.944 0.051839 .
## X.contact."unknown"     -1.643e+00  8.195e-02 -20.049  < 2e-16 ***
## X.day.                   9.211e-03  2.801e-03   3.288 0.001009 **
## X.month."aug"           -6.912e-01  8.788e-02  -7.865 3.70e-15 ***
## X.month."dec"            6.900e-01  2.052e-01   3.363 0.000770 ***
## X.month."feb"           -1.726e-01  1.005e-01  -1.717 0.085998 .
## X.month."jan"           -1.157e+00  1.341e-01  -8.629  < 2e-16 ***
## X.month."jul"           -8.307e-01  8.700e-02  -9.548  < 2e-16 ***
## X.month."jun"            5.338e-01  1.044e-01   5.113 3.17e-07 ***
## X.month."mar"            1.725e+00  1.346e-01  12.815  < 2e-16 ***
## X.month."may"           -3.906e-01  8.075e-02  -4.837 1.32e-06 ***
## X.month."nov"           -8.419e-01  9.445e-02  -8.914  < 2e-16 ***
## X.month."oct"            9.153e-01  1.195e-01   7.662 1.84e-14 ***
## X.month."sep"            8.455e-01  1.350e-01   6.264 3.74e-10 ***
## X.duration.              4.257e-03  7.278e-05  58.491  < 2e-16 ***
## X.campaign.             -8.155e-02  1.105e-02  -7.377 1.62e-13 ***
## X.pdays.                 1.361e-04  3.391e-04   0.401 0.688199
## X.previous.              8.063e-03  6.373e-03   1.265 0.205787
## X.poutcome."other"       2.039e-01  1.009e-01   2.020 0.043402 *
## X.poutcome."success"     2.278e+00  9.265e-02  24.586  < 2e-16 ***
```

```
## X.poutcome."unknown"    -4.025e-02  1.039e-01  -0.387 0.698486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26059  on 36167  degrees of freedom
## Residual deviance: 17166  on 36125  degrees of freedom
## AIC: 17252
##
## Number of Fisher Scoring iterations: 6
```

The deviance residual is a mathematical transformation of the loss function and quantifies a given point's contribution to the overall likehood. They can be used to form RSS-like statistics We notice that the residual deviance, which is the lack of fit of the entire model, is much lower than the null deviance, which is the lack of fit of the model considering only the intercept. This is what we want to see and indicates that our model is good. The AIC stands for Akaike Information Criterion, is based on deviance and should be lower. Finally, the coefficients quantifies the difference in log odds of a target variable for each predictor.

Naive Bayes Model

Building a Naive Bayes model and output what the model learned using the naiveBayes function. We first need to load the library e1071 to use this function

```
library(e1071)
nb1 <- naiveBayes(X.y.~.,data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      "no"     "yes"
## 0.8833223 0.1166777
##
## Conditional probabilities:
##        X.age.
## Y           [,1]    [,2]
##   "no"  40.80227 10.1631
##   "yes" 41.53815 13.3866
##
##        X.job.
## Y          "admin." "blue-collar" "entrepreneur" "housemaid" "management"
##   "no"  0.113622136   0.227119068    0.033335420 0.028202078  0.203205208
##   "yes" 0.117061611   0.130805687    0.025355450 0.019431280  0.246682464
##        X.job.
## Y         "retired" "self-employed"  "services"   "student" "technician"
##   "no"  0.043977714     0.034587455 0.096062351 0.016839865  0.169087267
##   "yes" 0.096682464     0.033886256 0.074644550 0.052606635  0.157582938
##        X.job.
## Y       "unemployed"   "unknown"
##   "no"   0.027732565 0.006228872
##   "yes"  0.039336493 0.005924171
##
##        X.marital.
## Y       "divorced" "married"  "single"
##   "no"   0.1152811 0.6111807 0.2735382
##   "yes"  0.1156398 0.5208531 0.3635071
##
##        X.education.
## Y        "primary" "secondary" "tertiary"  "unknown"
##   "no"  0.15709904  0.51921873 0.28349192 0.04019031
##   "yes" 0.11255924  0.46469194 0.37701422 0.04573460
##
##        X.default.
## Y             "no"      "yes"
##   "no"  0.98125078 0.01874922
##   "yes" 0.99028436 0.00971564
##
##        X.balance.
## Y           [,1]     [,2]
##   "no"  1302.084 2982.862
##   "yes" 1788.949 3243.416
##
##        X.housing.
## Y            "no"     "yes"
```

```
##   "no"  0.4162076 0.5837924
##   "yes" 0.6338863 0.3661137
##
##        X.loan.
## Y             "no"      "yes"
##   "no"  0.83222737 0.16777263
##   "yes" 0.90947867 0.09052133
##
##        X.contact.
## Y       "cellular" "telephone"  "unknown"
##   "no"  0.62263678  0.06210091 0.31526230
##   "yes" 0.82417062  0.07393365 0.10189573
##
##        X.day.
## Y           [,1]     [,2]
##   "no"  15.87555 8.301302
##   "yes" 15.10213 8.501258
##
##        X.month.
## Y             "apr"       "aug"       "dec"       "feb"       "jan"       "jul"
##   "no"  0.059659447 0.136941280 0.002629273 0.055402529 0.031551271 0.156191311
##   "yes" 0.108767773 0.128909953 0.017772512 0.081990521 0.027962085 0.116113744
##        X.month.
## Y             "jun"       "mar"       "may"       "nov"       "oct"       "sep"
##   "no"  0.120664830 0.005321147 0.323463128 0.089583072 0.010892701 0.007700013
##   "yes" 0.106872038 0.047867299 0.175829384 0.077251185 0.061611374 0.049052133
##
##        X.duration.
## Y           [,1]     [,2]
##   "no"  220.5720 206.3221
##   "yes" 537.9088 386.6043
##
##        X.campaign.
## Y           [,1]     [,2]
##   "no"  2.852917 3.270860
##   "yes" 2.164455 1.982136
##
##        X.pdays.
## Y           [,1]      [,2]
##   "no"  36.44375  97.10827
##   "yes" 68.79076 119.88205
##
##        X.previous.
## Y            [,1]     [,2]
##   "no"  0.4985915 2.355944
##   "yes" 1.1606635 2.479261
##
##        X.poutcome.
## Y        "failure"    "other"  "success"  "unknown"
##   "no"  0.10607863 0.03884437 0.01352197 0.84155503
##   "yes" 0.11824645 0.05758294 0.18175355 0.64241706
```

The prior for our target variable y, which is whether or not a client will subscribe to a term deposit, is called A-priori and is equal to 0.88 for “no” and 0.12 for “yes”. Furthermore, the likelihood data is in terms of conditional probabilities. Discrete data tells us how likely someone from each category is to subscribe. For example, out of the people who subscribed, most are likely to be married then single and then divorced. For continuous variables, the mean and standard deviation are given as we can see for age. The mean age for the people who subscribed is 41 vs 40 for those who didn’t. However, our data exploration told us that the median amount of people who participated is around 40 so this makes sense.

Evaluate on Test Data

We use the predict function to test our model on the test data. And we also check the accuracy of using the Logistic regression.

```
probs <- predict(glm1,newdata=test, type="response")
pred <- ifelse(probs>0.5,2,1)
pred <- as.factor(pred)
levels(pred) <- list("no"="1","yes"="2")
acc <- mean(as.integer(pred)==as.integer(test$X.y.))
print(paste("glm1 accuracy: ", acc))
```

```
## [1] "glm1 accuracy:  0.9010284197722"
```

```
table(pred, test$X.y.)
```

```
##
## pred  "no" "yes"
##   no  7764   685
##   yes  210   384
```

On the first run we install the caret package. Then we change the labels of the factor variables in test to be the same as pred. And then check the structure. Finally, we run the confusionMatrix function.

```
#install.packages('caret', dependencies = T)
levels(test$X.y.)<-c("no", "yes")
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
str(test$X.y.)
```

```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
str(pred)
```

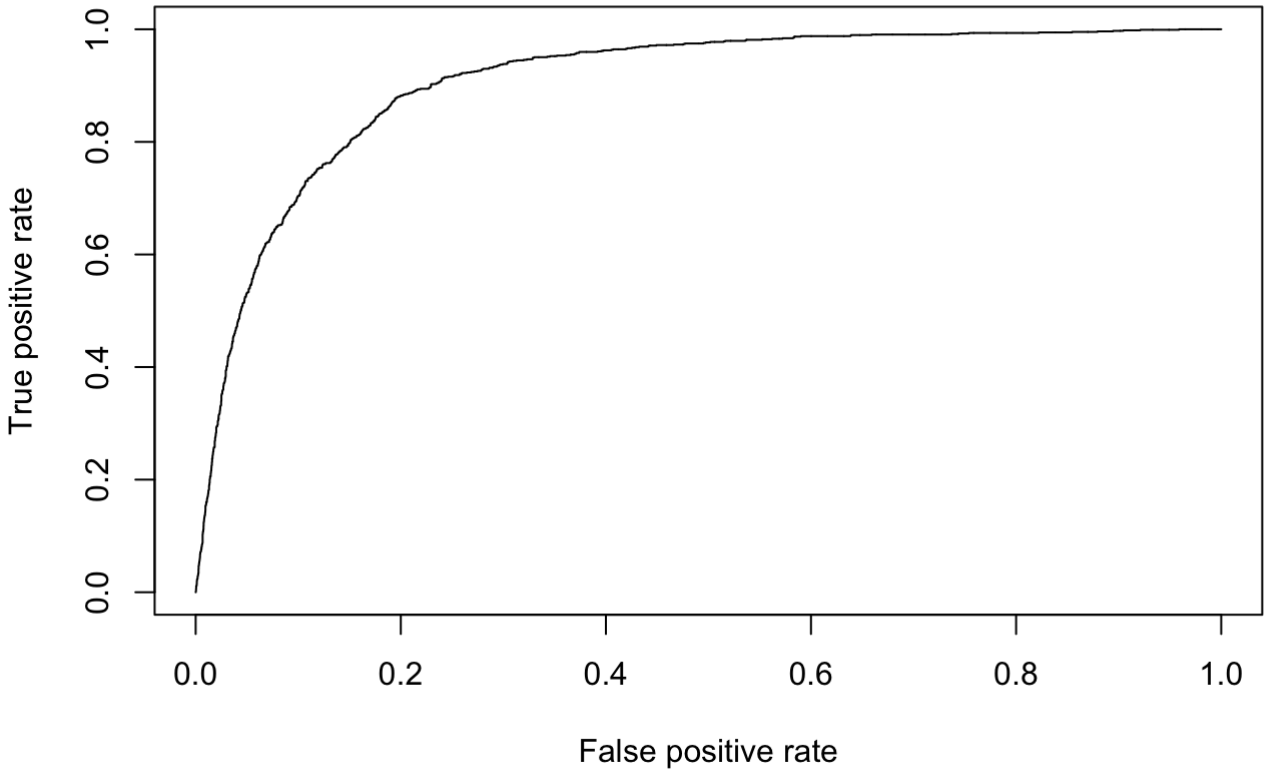
```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "names")= chr [1:9043] "16" "18" "22" "29" ...
```

```
confusionMatrix(pred,reference=test$X.y.)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7764  685
##        yes  210  384
##
##                Accuracy : 0.901
##                  95% CI : (0.8947, 0.9071)
##     No Information Rate : 0.8818
##     P-Value [Acc > NIR] : 3.531e-09
##
##                   Kappa : 0.4122
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9737
##             Specificity : 0.3592
##          Pos Pred Value : 0.9189
##          Neg Pred Value : 0.6465
##              Prevalence : 0.8818
##          Detection Rate : 0.8586
##    Detection Prevalence : 0.9343
##       Balanced Accuracy : 0.6664
##
##        'Positive' Class : no
##
```

We load the library ROCR. The ROC is a curve that plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The AUC is the area under the ROC curve. A good AUC is close to 1 than 0.5. Also we like to see the ROC shoot up rather quickly.

```
library(ROCR)
p <- predict(glm1, newdata=test, type="response")
pr <- prediction(p, test$X.y.)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.9062654
```

Our AUC is close to 1 which is considered good.

Evaluate on test data

Using the predict function.

```
p1<-predict(nb1, newdata=test, type="class")
table(p1, test$X.y.)
```

```
##
## p1        no  yes
##   "no"  7360  473
##   "yes"  614  596
```

```
levels(p1)<-c("no", "yes")
str(p1)
```

```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
mean(p1==test$X.y.)
```

```
## [1] 0.8797965
```

We got a slightly lower value with Naive Bayes (0.88) than we did with logistic regression(0.90). This could be due to the fact that Naive Bayes works better with smaller data sets however, our data set is rather large.

Strengths and Weaknesses of Logistic Regression and Naive Bayes

The strengths of these types of Logistic regression linear models are that they separate classes well if they are linearly separable, are computationally cheap, and have a nice probabilistic output. However, they're prone to underfitting. The strengths of Naive Bayes Linear models are that they works well with small data sets, are easy to implement, easy to interpret, and handle high dimensions well. However, it may be outperformed for larger data sets, it makes guesses, and it assumes predictors are independent.

Benefits and drawbacks of the classification metrics used.

Many metrics can be used to access classification. In this project, we applied sensitivity, specificity, and accuracy. Specificity gets information on the true negative rate. The sensitivity gives information on the true positive rate. Furthermore, accuracy is the number of accurate predictions divided by the total predictions. Although it is a useful tool, it does not give us information on the true positive rate and the true negative rate which is when we use sensitivity and specificity.