Classification

Code

3. Linear models for classification find a decision boundary between classes. In logistic regression, our target variable is qualitative: we want to know what class an observation is in. In the most common classification scenario, the target variable is a binary output so that we classify into one class or the other. It is a linear (parametric) algorithm which has low variance but high bias. Logistic regression performs well on larger data. Naive Bayes is a popular classification algorithm. The mathematical foundations of Naive Bayes go back to the 18th Century and the mathematician and minister, Thomas Bayes, who formalized this probabilistic equation that bears his name [aka posterior = (likelihood × prior) / marginal]. The algorithm makes the simplifying assumption that all the predictors are independent, which is usually not true but works well (and so handles high dimensions well). It performs well on smaller data and has a lower variance than logistic regression. However, it has a higher bias and the naive assumption may limit the performance of the algorithm if the predictors are not independent.
4. This notebook explores credit card data from Kaggle (https://www.kaggle.com/datasets/mariosfish/default-of-credit-card-clients)

Load the heart_disease.csv file.

Hide

```
df <- read.csv("credit_card.csv")
str(df)
```

```
'data.frame':   30000 obs. of  25 variables:
 $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL: int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX      : int  2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION: int  2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE : int  1 2 2 1 1 2 2 2 1 2 ...
 $ AGE      : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_1    : int  2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2    : int  2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3    : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4    : int  -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5    : int  -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6    : int  -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1: int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
 $ BILL_AMT2: int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
 $ BILL_AMT3: int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
 $ BILL_AMT4: int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
 $ BILL_AMT5: int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
 $ BILL_AMT6: int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
 $ PAY_AMT1 : int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
 $ PAY_AMT2 : int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
 $ PAY_AMT3 : int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4 : int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
 $ PAY_AMT5 : int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
 $ PAY_AMT6 : int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
 $ dpnm     : int  1 1 0 0 0 0 0 0 0 0 ...
```

a. Divide df into 80/20 train/test

Hide

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

b. Data Exploration

c. List the column names

dpnm is the target variable and the rest are predictors.

Hide

```
names(df)
```

```
 [1] "ID"        "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "AGE"       "PAY_1"
 [8] "PAY_2"     "PAY_3"     "PAY_4"     "PAY_5"     "PAY_6"     "BILL_AMT1" "BILL_AMT2"
[15] "BILL_AMT3" "BILL_AMT4" "BILL_AMT5" "BILL_AMT6" "PAY_AMT1"  "PAY_AMT2"  "PAY_AMT3"
[22] "PAY_AMT4"  "PAY_AMT5"  "PAY_AMT6"  "dpnm"
```

ii. See the first 5 rows

Hide

```
head(train, n=5)
```

	ID <int>	LIMIT_BAL <int>	S… <int>	EDUCATION <int>	MARRIAGE <int>	A… <int>	PAY_1 <int>	PAY_2 <int>	PAY_3 <int>
7452	7452	360000	2	2	2	26	1	-2	-2
8016	8016	80000	2	1	2	26	0	0	2
7162	7162	100000	2	2	2	37	0	0	0
8086	8086	500000	1	2	2	35	0	0	0
23653	23653	20000	2	1	1	37	1	-2	-1

5 rows | 1-10 of 25 columns

iii. Check the number of NAs in each column.

Hide

```
sapply(train, function(x) sum(is.na(x)))
```

```
       ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_1     PAY_2
        0         0         0         0         0         0         0         0
    PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
        0         0         0         0         0         0         0         0
BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
        0         0         0         0         0         0         0         0
     dpnm
        0
```

There are no NAs in this dataset. This means we do not have to delete/modify any rows.

iv. Converting columns into factors.

Hide

```
#str(df)
train$MARRIAGE = factor(train$MARRIAGE)
train$EDUCATION = factor(train$EDUCATION)
train$SEX = factor(train$SEX)
str(train)
```

```
'data.frame':   24000 obs. of  25 variables:
 $ ID       : int  7452 8016 7162 8086 23653 9196 623 15241 10885 934 ...
 $ LIMIT_BAL: int  360000 80000 100000 500000 20000 30000 90000 180000 450000 30000 ...
 $ SEX      : Factor w/ 2 levels "1","2": 2 2 2 1 2 1 2 2 1 1 ...
 $ EDUCATION: Factor w/ 7 levels "0","1","2","3",..: 3 2 3 3 2 4 2 3 2 3 ...
 $ MARRIAGE : Factor w/ 4 levels "0","1","2","3": 3 3 3 3 2 3 3 2 2 3 ...
 $ AGE      : int  26 26 37 35 37 31 27 42 53 32 ...
 $ PAY_1    : int  1 0 0 0 1 0 0 1 -1 2 ...
 $ PAY_2    : int  -2 0 0 0 -2 0 0 -2 -1 0 ...
 $ PAY_3    : int  -2 2 0 0 -1 0 -2 -2 -1 0 ...
 $ PAY_4    : int  -2 2 0 0 2 0 -2 -2 -1 2 ...
 $ PAY_5    : int  -2 2 0 0 2 0 -2 -2 -1 2 ...
 $ PAY_6    : int  -2 2 0 0 -2 -2 -2 -2 0 2 ...
 $ BILL_AMT1: int  0 38174 177961 207237 -113 27838 7624 0 3873 7851 ...
 $ BILL_AMT2: int  0 40550 108173 224007 -113 28791 0 0 3119 9065 ...
 $ BILL_AMT3: int  0 41577 15697 275615 10887 27788 0 0 8970 11595 ...
 $ BILL_AMT4: int  0 41595 11353 220088 10413 29784 0 0 1323 11112 ...
 $ BILL_AMT5: int  0 43264 9306 216482 -245 0 0 0 11990 12923 ...
 $ BILL_AMT6: int  0 43402 9693 136086 -245 0 0 0 8838 12566 ...
 $ PAY_AMT1 : int  0 3000 3082 20001 1575 1703 0 0 3128 1500 ...
 $ PAY_AMT2 : int  0 2000 2022 30168 11000 1200 0 0 9008 3000 ...
 $ PAY_AMT3 : int  0 1000 1000 6022 0 2196 0 0 1323 0 ...
 $ PAY_AMT4 : int  0 2500 1000 6375 0 2500 0 0 12008 2000 ...
 $ PAY_AMT5 : int  0 1000 500 5005 0 0 0 0 214 0 ...
 $ PAY_AMT6 : int  0 2000 300 5000 5100 0 0 0 1327 1000 ...
 $ dpnm     : int  0 1 1 0 1 1 0 0 0 1 ...
```

Hide

```
test$MARRIAGE = factor(test$MARRIAGE)
test$EDUCATION = factor(test$EDUCATION)
test$SEX = factor(test$SEX)
```

The MARRIAGE (4 possible values), EDUCATION (7 possible values) and SEX (2 possible values) columns can be turned into factors from integers.

v. Checking distribution of levels in MARRIAGE, EDUCATION, and SEX columns.

Hide

```
summary(train$SEX)
```

```
    1     2
 9463 14537
```

Hide

```
summary(train$MARRIAGE)
```

```
    0     1     2     3
   41 10915 12781   263
```

Hide

```
summary(train$EDUCATION)
```

```
    0     1     2     3     4     5     6
   14  8462 11243  3916   103   218    44
```

There are about a third more people of gender “2” than “1”. Most people’s marriage status is either “1” or “2”, with only a couple hundred being “0” and “3” combined. The education level is mostly skewed towards the “1” and “2” levels, with some in “3” and the others in “4”, “5”, “6”, and “0”.

c. Creating informative graphs using training data.

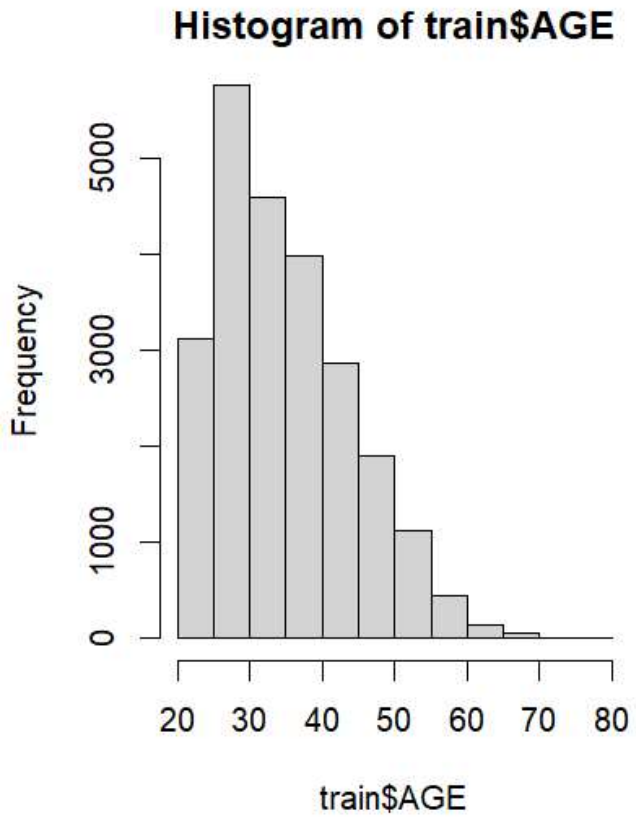
d. Explore distribution of AGE and dpnm (our target variable).

Hide

```
opar <- par()
par(mfrow=c(1,2))  # plots show up side by side
hist(train$AGE)
hist(train$dpnm)   # the model may be biased as there are triple the 0s as 1s.
```

Hide

```
par(opar)
```



Age is (understandably) skewed to the left, with most people being around the ages between 25 and 40. Any model made from this data may be biased as there are triple the 0s as 1s.

d. Simple logistic regression model

Hide

```
glm1 <- glm(dpnm~LIMIT_BAL+SEX+MARRIAGE+PAY_1+PAY_2+PAY_3+AGE+BILL_AMT1+BILL_AMT2+PAY_AMT1+PAY_A
MT2+PAY_AMT4, data=train, family=binomial)
summary(glm1)
```

```
Call:
glm(formula = dpnm ~ LIMIT_BAL + SEX + MARRIAGE + PAY_1 + PAY_2 +
    PAY_3 + AGE + BILL_AMT1 + BILL_AMT2 + PAY_AMT1 + PAY_AMT2 +
    PAY_AMT4, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1358  -0.6993  -0.5483  -0.2977   3.2335

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.547e+00  5.873e-01  -4.336 1.45e-05 ***
LIMIT_BAL   -7.014e-07  1.654e-07  -4.241 2.23e-05 ***
SEX2        -1.026e-01  3.425e-02  -2.997  0.00273 **
MARRIAGE1    1.496e+00  5.818e-01   2.571  0.01015 *
MARRIAGE2    1.297e+00  5.819e-01   2.230  0.02578 *
MARRIAGE3    1.423e+00  6.001e-01   2.371  0.01773 *
PAY_1        5.818e-01  1.960e-02  29.689  < 2e-16 ***
PAY_2        8.785e-02  2.240e-02   3.922 8.80e-05 ***
PAY_3        1.099e-01  2.077e-02   5.290 1.22e-07 ***
AGE          4.643e-03  2.022e-03   2.296  0.02166 *
BILL_AMT1   -7.342e-06  1.280e-06  -5.736 9.68e-09 ***
BILL_AMT2    5.941e-06  1.329e-06   4.472 7.76e-06 ***
PAY_AMT1    -1.813e-05  2.725e-06  -6.654 2.85e-11 ***
PAY_AMT2    -5.975e-06  1.858e-06  -3.216  0.00130 **
PAY_AMT4    -4.677e-06  1.777e-06  -2.631  0.00850 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25433  on 23999  degrees of freedom
Residual deviance: 22379  on 23985  degrees of freedom
AIC: 22409

Number of Fisher Scoring iterations: 6
```

The deviance residuals are not small, which is expected for a simple logistic regression model like this. Considering the fact that dpnm can only be 0 or 1, the maximum residual of 3.2335 may seem alarming but is normal. The model has 2 dummy variables related to SEX, due to it being a predictor with factors. Their estimates are in relation to the estimate made by the first factor level of SEX (where EDUCATION = 1). For example, SEX1 is -1.026e-01 units away from SEX0. I have only included the significant predictors which all have one or more “*” next to them. The null deviance reduced from 25433 to 22379, which is a small decrease.

e. Naive Bayes model

Hide

```
library(e1071)
nb1 <- naiveBayes(dpnm~LIMIT_BAL+SEX+MARRIAGE+PAY_1+PAY_2+PAY_3+AGE+BILL_AMT1+BILL_AMT2+PAY_AMT1
+PAY_AMT2+PAY_AMT4, data=train)
nb1
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.7776667 0.2223333

Conditional probabilities:
   LIMIT_BAL
Y       [,1]     [,2]
  0 177889.6 131810.5
  1 129815.9 115126.0

   SEX
Y           1         2
  0 0.3845371 0.6154629
  1 0.4284108 0.5715892

   MARRIAGE
Y              0            1            2            3
  0 0.0019824261 0.4461530219 0.5415773682 0.0102871839
  1 0.0007496252 0.4850074963 0.5009370315 0.0133058471

   PAY_1
Y         [,1]      [,2]
  0 -0.2086905 0.9583333
  1  0.6752249 1.3853000

   PAY_2
Y         [,1]     [,2]
  0 -0.3039006 1.038463
  1  0.4636432 1.500200

   PAY_3
Y         [,1]     [,2]
  0 -0.3181526 1.050991
  1  0.3697526 1.502605

   AGE
Y       [,1]     [,2]
  0 35.38561 9.074010
  1 35.68722 9.686917

   BILL_AMT1
Y       [,1]     [,2]
  0 51902.78 73238.79
  1 48612.40 73472.70
```

```
   BILL_AMT2
Y       [,1]     [,2]
  0 49602.30 70717.87
  1 47516.98 71725.95

   PAY_AMT1
Y       [,1]      [,2]
  0 6274.820 18103.268
  1 3351.269  8708.317

   PAY_AMT2
Y       [,1]     [,2]
  0 6629.747 25324.62
  1 3468.872 12433.34

   PAY_AMT4
Y       [,1]     [,2]
  0 5268.031 17121.32
  1 3110.524 10488.95
```

First, the overall probability of dpnm being 0 or 1 is shown, and then conditional probabilities of each predictor used in the model are shown. [The mathematical foundations of Naive Bayes go back to the 18th Century and the mathematician and minister, Thomas Bayes, who formalized this probabilistic equation that bears his name [aka posterior = (likelihood × prior) / marginal]. The algorithm makes the simplifying assumption that all the predictors are independent, which is usually not true but works well (and so handles high dimensions well)]. So for the predictors that are factors, the probabilities for having a value of 0 or 1 given each level is printed. For example, if the SEX is 1, the probability of 0 is 0.3845371 and the probability of 1 is 0.4284108. In addition, if SEX is 2 the probability of 0 is 0.6154629 and the probability of 1 is 0.6154629. For the predictors that are not factors, the mean and then standard deviation considering the dpnm values of 0 and 1 are displayed.

f. Predict and test models

Logistic regression:

Hide

```
#library("psych")
#library(irr)
#install.packages("irr")
probs1 <- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1>0.5, 1, 0)

mean(pred1==as.integer(test$dpnm)) #accuracy
```

```
[1] 0.8118333
```

Hide

```
cm <- table(pred1, as.integer(test$dpnm)) #confusion matrix
cm
```

```
pred1    0    1
    0 4582 1011
    1  118  289
```

Hide

```
(4582)/(4582+118) #sensitivity/recall
```

```
[1] 0.9748936
```

Hide

```
(289)/(289+1011) #specificity
```

```
[1] 0.2223077
```

Hide

```
(4582)/(4582+1011) #precision
```

```
[1] 0.8192383
```

Hide

```
p0 <- (4582+289)/6000
p1 <- (4582+1011)/6000
p2 <- (4582+118)/6000
p3 <- p1 * p2
#p3
p4 <- (118+289)/6000
p5 <- (1011+289)/6000
p6 <- p4 * p5
cohen_kappa = (p0 - (p3+p6)) / (1-(p3+p6))
cohen_kappa  # fair agreement
```

```
[1] 0.2623968
```

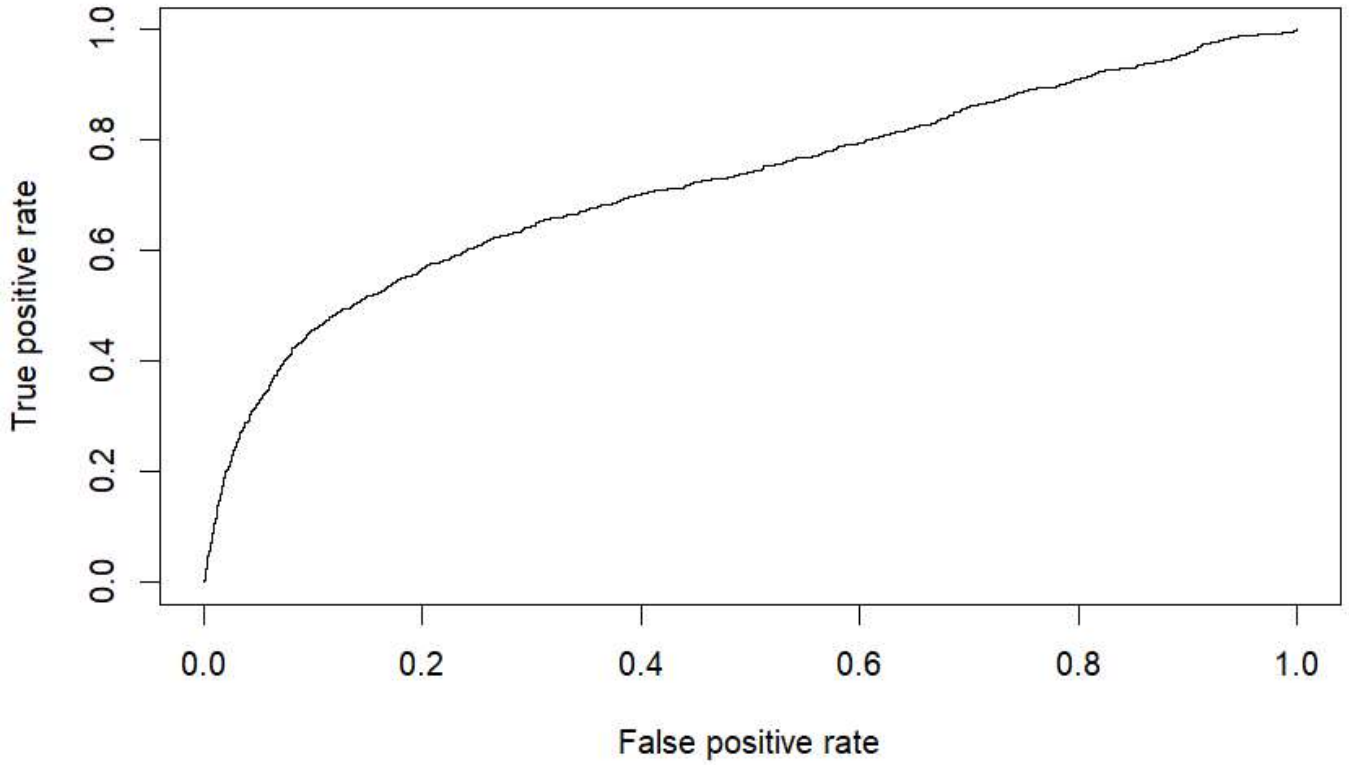
Hide

```
#ROC and AUC
#install.packages("ROCR")
library("ROCR")
```

```
Warning: package ‘ROCR’ was built under R version 4.0.5
```

Hide

```
pr1 <- prediction(probs1, test$dpnm)
prf1 <- performance(pr1, measure="tpr", x.measure="fpr")
plot(prf1)
```



Naive Bayes:

Hide

```
probs2 <- predict(nb1, newdata=test, type="raw")
pred2 <- ifelse(probs2>0.5, 1, 0)
pred2 <- apply(pred2==1, 1, function(x) {
                      if(any(x)) {
                           as.integer(names(which(x)))
                         }
                      else NA
                    })
#summary(probs2)
mean(pred2==as.integer(test$dpnm))  #accuracy
```

```
[1] 0.7526667
```

Hide

```
table(pred2, test$dpnm)  #confusion matrix
```

```
pred2    0    1
    0 3729  513
    1  971  787
```

Hide

```
(3729)/(3729+971) #sensitivity/recall
```

```
[1] 0.7934043
```

Hide

```
(787)/(787+513) #specificity
```

```
[1] 0.6053846
```

Hide

```
(3729)/(3729+513) #precision
```

```
[1] 0.8790665
```

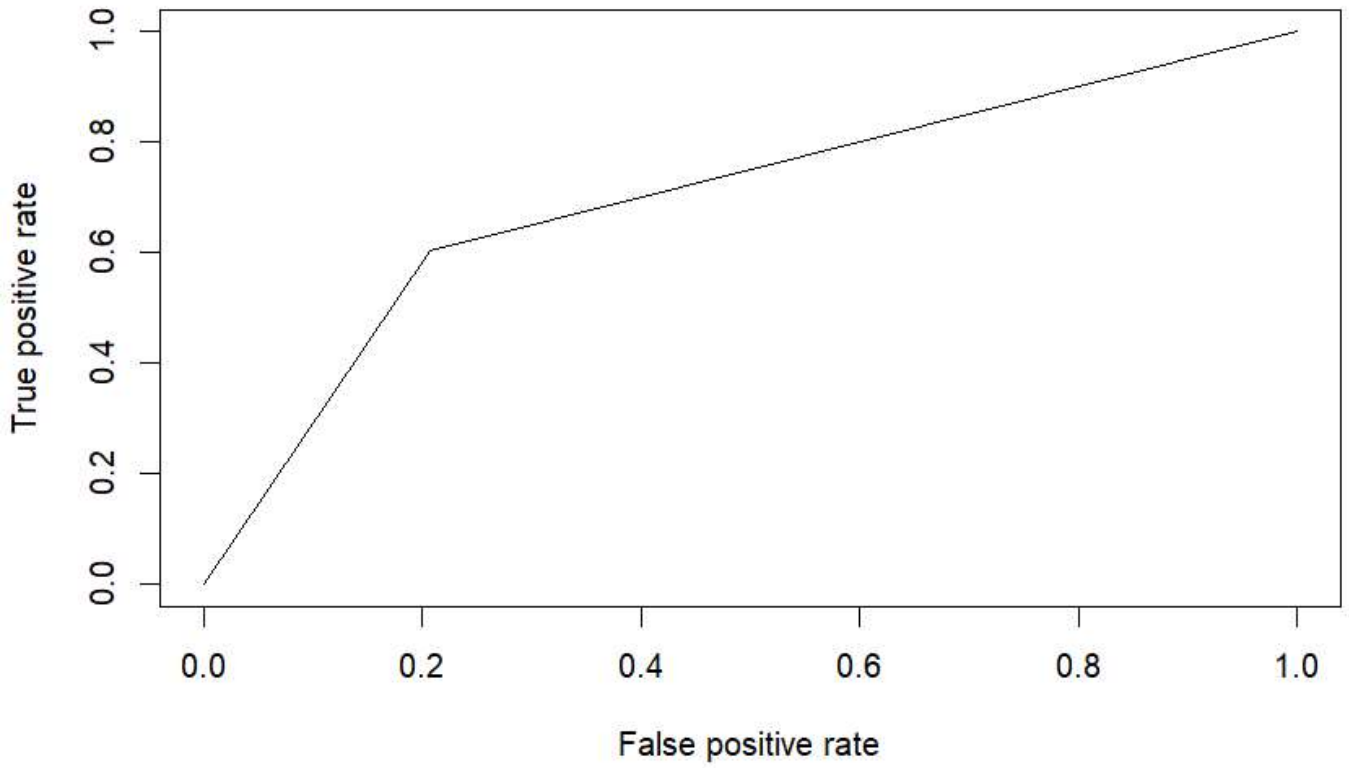
Hide

```
p02 <- (3729+787)/6000
p12 <- (3729+513)/6000
p22 <- (3729+971)/6000
p32 <- p12 * p22
#p32
p42 <- (971+787)/6000
p52 <- (513+787)/6000
p62 <- p42 * p52
cohen_kappa = (p02 - (p32+p62)) / (1-(p32+p62))
cohen_kappa  # fair agreement
```

```
[1] 0.3537148
```

Hide

```
#ROC and AUC
#install.packages("ROCR")
library("ROCR")
pr2 <- prediction(pred2, test$dpnm)
prf2 <- performance(pr2, measure="tpr", x.measure="fpr")
plot(prf2)
```



Hide

```
library(ModelMetrics)
```

```
Warning: package 'ModelMetrics' was built under R version 4.0.5Registered S3 method overwritten
by 'data.table':
  method           from
  print.data.table

Attaching package: 'ModelMetrics'

The following object is masked from 'package:base':

    kappa
```

Hide

```
library(Rcpp)
#mcc(TP=3729, FP=513, TN=787, FN=971)
#mcc(test$dpnm, pred2, cutoff=0.5)
```

The accuracy of the logistic regression model is greater than that of Naive Bayes in this case (81% compared to 75%). This might be because logistic regression is better with larger datasets. Looking at the confusion matrices of both the models, the sensitivity is much greater for the first model (~97%) than the second (~79%). Specificity of both the models are 0.22 and 0.60, which is a great difference. Precision is 82% and 88% respectively, and cohen kappa is 0.26 and 0.35. (The greater the kappa value, the better. In this case, it is fair agreement.) The ROCs for

both the models are similarly good, with the logistic regression model having more area under it (AUC). The MCC values for the models are 0.323 and 0.361 respectively, which is less than accuracy but accounts for class distribution (so the accuracy was adjusted based on the number of 0s and 1s in test$dpnm). It is slightly greater for the second model. The Naive Bayes model was less accurate (as it does better on smaller datasets) which might have affected the other metrics. As mentioned before, logistic regression is better with larger datasets.

g. Naive Bayes algorithm makes the simplifying assumption that all the predictors are independent, which is usually not true but works well (and so handles high dimensions well). It performs well on smaller data and has a lower variance than logistic regression. However, it has a higher bias and the naive assumption may limit the performance of the algorithm if the predictors are not independent. Logistic regression is a linear (parametric) algorithm which has low variance but high bias. Logistic regression performs well on larger data compared to smaller data. Assuming linearity between predictors and target variable might be the wrong assumption sometimes.

h. Accuracy is the number of correct predictions over the total number of examples/data rows. This is the simplest and most common metric for classification, but there might be cases when the calculated accuracy is not correct (due to external factors like an imbalanced dataset affecting the model). A confusion matrix is a table or predictions and true values. It is a more complete report than accuracy and can be used to derive other useful classification metrics. The sensitivity measures the true positive rate while specificity measures the true negative rate. They help to quantify the extent to which a given class was misclassified. However, high sensitivity does not help to exclude false positives, and something similar can be said about high specificity. (Cohen's) Kappa adjusts accuracy to account for correct prediction by chance (in order to reduce the probability that we guessed right by chance) and is often used to quantify agreement between two annotators of data. However, the Kappa paradox prevents it from being a perfect metric. ROC (Receiver Operating Characteristics) curve shows the tradeoff between predicting true positives while avoiding false positives. (We want to see the classifier shoot up and leave little space at the top left.) A related metric is AUC, the area under the curve. AUC values range from 0.5 for a classifier with no predictive value to 1.0 for a perfect classifier. This makes us able to understand how well our model works. However, there is a way to trick the AUC ROC metric: dividing all the predictions by 100 will keep the same AUC ROC. MCC (Matthew's correlation coefficient) accounts for differences in class distribution (which accuracy does not) and so is an improvement over it. However, it can only be used for binary classification.