Linear Models: Classification

Fredrick Horn

Data from: [https://www.kaggle.com/datasets/gauravtopre/credit-card-defaulter-prediction (https://www.kaggle.com/datasets/gauravtopre/credit-card-defaulter-prediction)]

Linear classification uses a linear model to find a decision boundry between classes. The model will take a set of inputs and try to find a line that would split up the data into classes as best it can. Linear classification is great because it is simple and easy to undersand and fast with larger data sets. But linear models simplicity is also one of its down falls. It will alawys assume a linear relationship in the input data, even if its not there.

About Logistic Regression and Naive Bayes

I will be using the a logistic regression model and a naive bayes model to try to predict using my choosen data set. The logistic regression is great for this application as it was made to classify a binary response using its set of predicotrs. It is simple and fast to execute as well as a dsicrimitive algorithm, meaning that it directly predict a classification for the new data using its predictors. Its also robust to noise in the data. But it is still only a linear model. It will try to fit the data to the linear relationship its looking for, even if its not there. A problem that I ran into with a data set I tried to use previously, is that the more predictors it has, the longer it takes to do the computational work. I decided to switch data sets previously as the computation time was taking too long.

The naive bayes is a classification model based on Bayes theorem. Naive bayes tends to perform well on smaller data sets than the logistic regression does. While unneeded for this application, it can also be applied to multiclass classification. But naive bayes has a few problems. The first and largest being its assumption that all the predictors are independent of each other, which is not always the case. Naive bayes as tends to have a higher bias than the logistic regression.

Data Exploration

The data set I choose is from Kaggle and it contains data about credit card useage and defaulters in Taiwan in 2005. The goal is to use the data in this set to predict if a given person defaulted on their payments or not. After reading in the csv, I remove an unneeded ID row, as well as four sets of three related columns of payment information in defferent months. I chose to only use the first 2 of 6 months of data for simplicity and to improve the models. The last bit of cleaning is to remove some rows that had invalid values and insignificantly small subsets and then convert some columns to factors.

```
set.seed(1)
credit <- read.csv("creditcard.csv", header = TRUE) # read in csv

# data cleaning
credit <- credit[, c(-1, -9, -10, -11, -12, -15, -16, -17, -18, -21, -22, -23, -24)] # remove un
needed columns
credit <- subset(credit, credit$education != "0") # remove rows with invalid values
credit <- subset(credit, credit$education != "Others") # remove rows with this education type as
its insignificant
credit <- subset(credit, credit$education != "Unknown") # remove rows with this education type a
s its insignificant
credit <- subset(credit, credit$marriage != "0") # remove rows with invalid values
credit <- subset(credit, credit$marriage != "Other") # remove rows with this marrage type as its
insignificant

credit$defaulted <- factor(credit$defaulted)
credit$sex <- factor(credit$sex)
credit$education <- factor(credit$education)
credit$marriage <- factor(credit$marriage)
```

Then, I split the data into 80% training and 20% testing data and output the structure and summary of the training data.

```
i <- sample(1:nrow(credit), 0.8 * nrow(credit), replace = FALSE) # split data
train <- credit[i, ] # 80% train
test <- credit[-i, ] # 20% test
str(train) # structure of training data
```

```
## 'data.frame':    23330 obs. of  12 variables:
##  $ creditLimit    : int  250000 260000 100000 160000 20000 470000 180000 50000 380000 80000
...
##  $ sex            : Factor w/ 2 levels "F","M": 2 2 1 2 2 2 2 2 2 1 ...
##  $ education      : Factor w/ 3 levels "Graduate school",..: 3 3 3 3 3 3 1 2 1 3 ...
##  $ marriage       : Factor w/ 2 levels "Married","Single": 2 1 1 2 2 2 1 2 1 2 ...
##  $ age            : int  26 50 38 59 36 35 32 36 34 26 ...
##  $ paymentDelaySep: int  0 0 1 0 0 0 -2 2 0 0 ...
##  $ paymentDelayAug: int  0 0 -1 0 0 0 -2 2 0 0 ...
##  $ billTotalSep   : int  115497 133208 0 86249 15198 262980 0 2400 106065 47788 ...
##  $ billTotalAug   : int  114716 122199 199 85764 19590 204078 0 2400 97979 45221 ...
##  $ billPaidSep    : int  3835 6400 199 5000 5000 10929 0 0 3385 4000 ...
##  $ billPaidAug    : int  4068 5744 0 4214 2000 4177 0 0 4074 3000 ...
##  $ defaulted      : Factor w/ 2 levels "N","Y": 1 1 1 1 2 1 1 2 1 1 ...
```

```
head(train) # first few lines
```

```
##       creditLimit sex  education marriage age paymentDelaySep paymentDelayAug
## 17848      250000   M University   Single  26               0               0
## 25079      260000   M University  Married  50               0               0
## 4895       100000   F University  Married  38               1              -1
## 27506      160000   M University   Single  59               0               0
## 13549       20000   M University   Single  36               0               0
## 26839      470000   M University   Single  35               0               0
##       billTotalSep billTotalAug billPaidSep billPaidAug defaulted
## 17848       115497       114716        3835        4068         N
## 25079       133208       122199        6400        5744         N
## 4895             0          199         199           0         N
## 27506        86249        85764        5000        4214         N
## 13549        15198        19590        5000        2000         Y
## 26839       262980       204078       10929        4177         N
```

```
tail(train) # last few lines
```

```
##       creditLimit sex       education marriage age paymentDelaySep
## 25502      200000   F Graduate school   Single  31               1
## 13192      120000   F      University  Married  48               0
## 26621       10000   M      University   Single  23              -2
## 26330      350000   F      University   Single  34               1
## 23633      210000   F     High School   Single  43              -1
## 23458       80000   F     High School  Married  41              -1
##       paymentDelayAug billTotalSep billTotalAug billPaidSep billPaidAug
## 25502              -1            0          896         896           0
## 13192               0        78384        77085        4000        3505
## 26621              -1          998          780         780         390
## 26330              -1          -20          630         650           0
## 23633              -1         7605         1170        1170           0
## 23458              -1         3526        10129       10129        6100
##       defaulted
## 25502         N
## 13192         N
## 26621         Y
## 26330         N
## 23633         N
## 23458         N
```

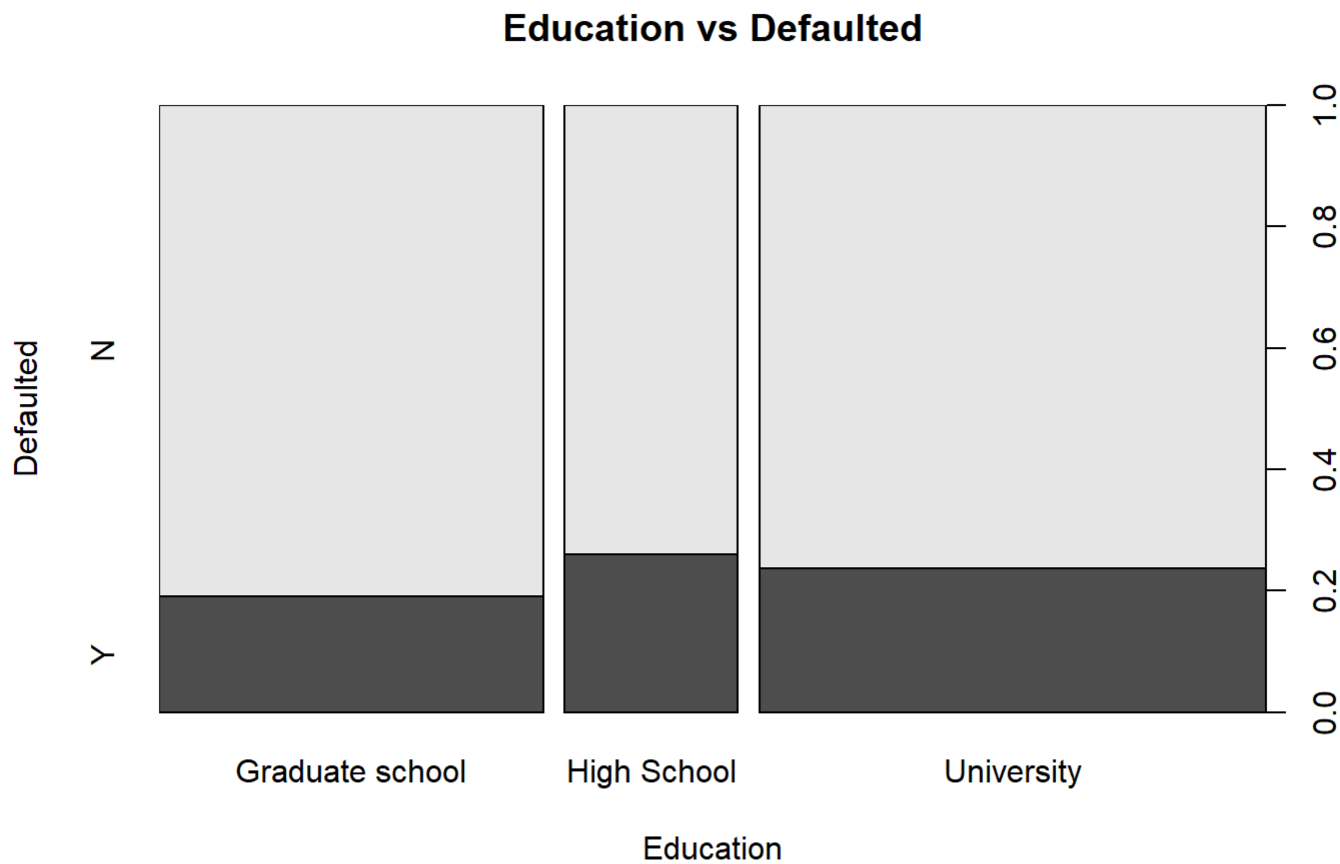
```
summary(train) # summary of data columns
```

```
##   creditLimit      sex                education       marriage
##  Min.   :  10000   F:14183   Graduate school: 8428   Married:10759
##  1st Qu.:  50000   M: 9147   High School    : 3795   Single :12571
##  Median : 140000             University     :11107
##  Mean   : 168031
##  3rd Qu.: 240000
##  Max.   :1000000
##       age        paymentDelaySep    paymentDelayAug    billTotalSep
##  Min.   :21.00   Min.   :-2.00000   Min.   :-2.0000   Min.   :-165580
##  1st Qu.:28.00   1st Qu.:-1.00000   1st Qu.:-1.0000   1st Qu.:   3510
##  Median :34.00   Median : 0.00000   Median : 0.0000   Median :  22316
##  Mean   :35.36   Mean   :-0.01547   Mean   :-0.1293   Mean   :  50870
##  3rd Qu.:41.00   3rd Qu.: 0.00000   3rd Qu.: 0.0000   3rd Qu.:  66503
##  Max.   :79.00   Max.   : 8.00000   Max.   : 8.0000   Max.   : 964511
##   billTotalAug     billPaidSep        billPaidAug        defaulted
##  Min.   :-69777   Min.   :     0.0   Min.   :      0.0   N:18105
##  1st Qu.:  2975   1st Qu.:   961.2   1st Qu.:    788.2   Y: 5225
##  Median : 21066   Median :  2100.0   Median :   2009.0
##  Mean   : 48917   Mean   :  5580.3   Mean   :   5742.1
##  3rd Qu.: 63327   3rd Qu.:  5005.0   3rd Qu.:   5000.0
##  Max.   :983931   Max.   :505000.0   Max.   :1227082.0
```

Graphs

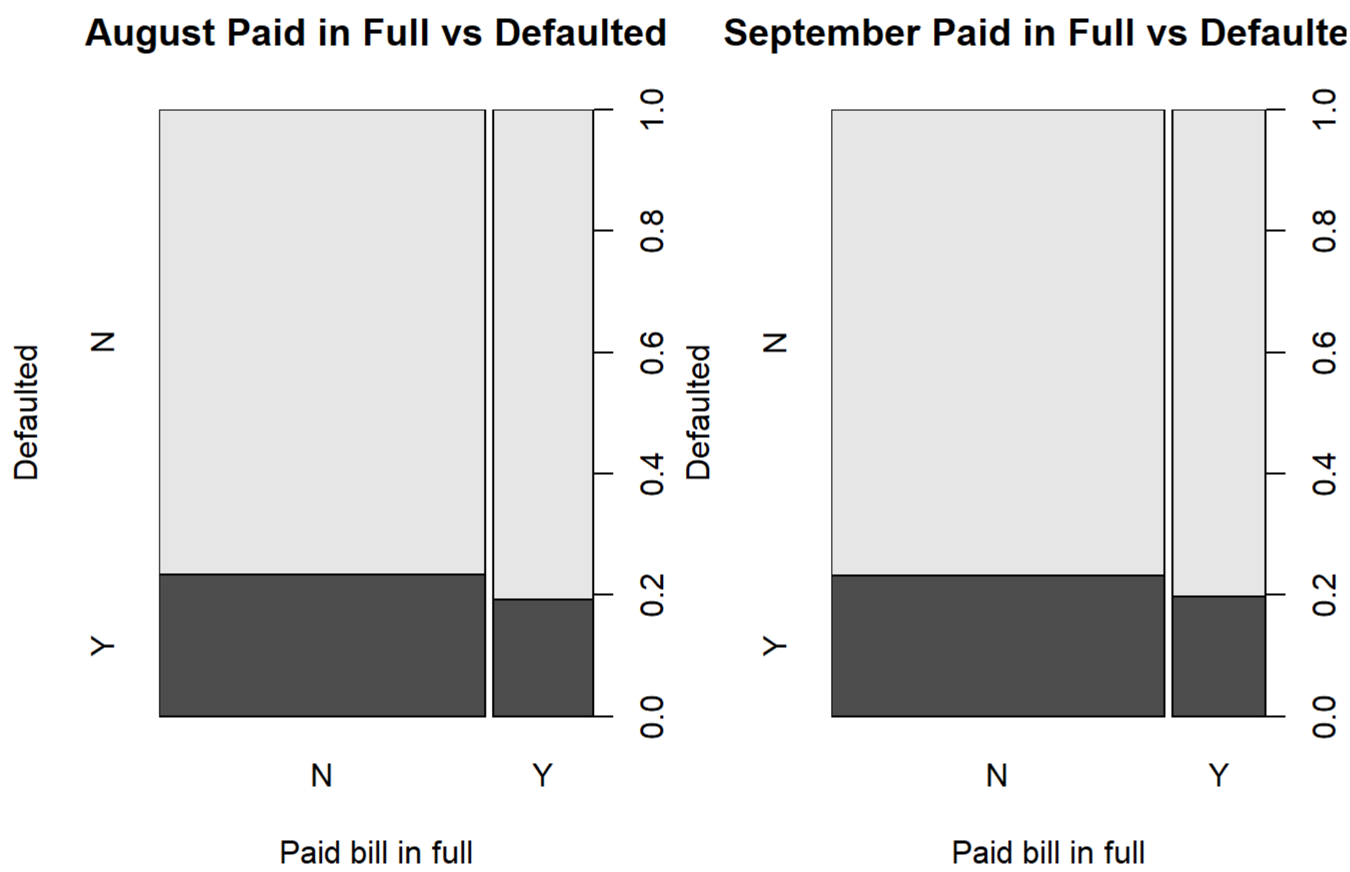
This first graph shows that education level does have an effect on if someone defaults on their payment. The amount of poeple who defaulted that went to grad school is less than the amount of people that went to university is less than the amount of people that only went to high schoool.

```
plot(train$defaulted ~ train$education, xlab = "Education", ylab = "Defaulted", main = "Educatio
n vs Defaulted")
```



These next two graphs are showing that in both months there are less people who paid their bill in full and defaulted than those that didnt pay their bill in full and defaulted.

```
par(mfrow = c(1, 2)) # output graphs in 2x1
plot(factor(ifelse(train$billPaidAug >= train$billTotalAug, "Y", "N")), train$defaulted, xlab =
"Paid bill in full", ylab = "Defaulted", main = "August Paid in Full vs Defaulted")
plot(factor(ifelse(train$billPaidSep >= train$billTotalSep, "Y", "N")), train$defaulted, xlab =
"Paid bill in full", ylab = "Defaulted", main = "September Paid in Full vs Defaulted")
```



Logistic Regression Model

Now, I create a binomial logistic regression model using the training data where I am training it to predict if a given person defaulted on their payment. As shown in the summary, the range of the min and max of the deviance residuals is small which is good for thid model. R seems very confident that the best predictor is the paymentDelaySep and to a lessser degree paymentDelayAug based on the reported z and p values. But overall, most of the predictors are significant. Interestingly, the dummy variables for the high school and university predictors are not reported as significant by R. Finally, the residual deviance is lower than the null deviance which does show that the model is at least improved over just the intercept with no predictors.

```
logModel <- glm(defaulted ~ ., data = train, family = "binomial") # create logistic regression m
odel
summary(logModel)
```

```
##
## Call:
## glm(formula = defaulted ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1167  -0.7006  -0.5505  -0.2979   3.8271
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.172e+00  9.280e-02 -12.624  < 2e-16 ***
## creditLimit           -9.390e-07  1.713e-07  -5.483 4.19e-08 ***
## sexM                   1.114e-01  3.464e-02   3.215 0.001306 **
## educationHigh School -6.635e-02  5.339e-02  -1.243 0.213929
## educationUniversity  -6.729e-02  3.987e-02  -1.688 0.091440 .
## marriageSingle       -1.765e-01  3.880e-02  -4.549 5.39e-06 ***
## age                   7.083e-03  2.108e-03   3.360 0.000779 ***
## paymentDelaySep       5.839e-01  1.965e-02  29.711  < 2e-16 ***
## paymentDelayAug       1.640e-01  1.744e-02   9.400  < 2e-16 ***
## billTotalSep         -7.444e-06  1.292e-06  -5.763 8.28e-09 ***
## billTotalAug          6.082e-06  1.341e-06   4.534 5.79e-06 ***
## billPaidSep          -1.607e-05  2.632e-06  -6.105 1.03e-09 ***
## billPaidAug          -1.169e-05  2.383e-06  -4.905 9.34e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24817  on 23329  degrees of freedom
## Residual deviance: 21890  on 23317  degrees of freedom
## AIC: 21916
##
## Number of Fisher Scoring iterations: 6
```

Naive Bayes Model

Next, I create a naive bayes model using the e1071 library. The first notable data in the model's output is the A-priori probabilties. These are what the model thinks the overall probabilties that a given person will have defaulted or not. Next is the conditional probabilities, where each group is what the model thinks the probability that a person would default given each column of data.

```
library(e1071) # import lib for naive bayes
bayesModel <- naiveBayes(defaulted ~ ., data = train) # create naive bayes model
bayesModel
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         N         Y
## 0.7760394 0.2239606
##
## Conditional probabilities:
##    creditLimit
## Y       [,1]   [,2]
##   N 178948.1 131947
##   Y 130204.3 115571
##
##    sex
## Y           F         M
##   N 0.6179508 0.3820492
##   Y 0.5732057 0.4267943
##
##    education
## Y   Graduate school High School University
##   N       0.3766363   0.1552610  0.4681027
##   Y       0.3079426   0.1883254  0.5037321
##
##    marriage
## Y    Married   Single
##   N 0.452306 0.547694
##   Y 0.491866 0.508134
##
##    age
## Y       [,1]     [,2]
##   N 35.26407 9.002781
##   Y 35.67254 9.649452
##
##    paymentDelaySep
## Y         [,1]      [,2]
##   N -0.2104391 0.9472449
##   Y  0.6600957 1.3919982
##
##    paymentDelayAug
## Y         [,1]     [,2]
##   N -0.2989229 1.035447
##   Y  0.4583732 1.513543
##
##    billTotalSep
## Y       [,1]     [,2]
##   N 51733.21 73193.86
##   Y 47879.87 73460.41
##
```

```
##    billTotalAug
## Y       [,1]     [,2]
##   N 49552.02 70811.78
##   Y 46714.99 71446.26
##
##    billPaidSep
## Y       [,1]      [,2]
##   N 6232.587 16998.128
##   Y 3319.910  9145.588
##
##    billPaidAug
## Y       [,1]     [,2]
##   N 6460.940 21015.84
##   Y 3251.403 11054.06
```

Evaluation of Models on Test Set

In evaluating the logistic regression model, it performs pretty well. R reports an accuracy of 0.8092 which is higher than I expected. But looking closer at the confusion matrix and statistics, there are a lot of false positive. This is further shown by the fact that the specificity is quite low. The Kappa value is also on the low end of 0.2815 which is not great but good enough. Looking at the ROC graph, the model clearly a little lackcing as the slope of the line does go up significantly in the beginning, indicating a good true positive rate, but it does not ever flatten out. Ideally the true positive rate would be as close to 1 as possible and the false positive rate would be close to 0. The reported AUC is also repoted to be at least better than randomly guessing, with a 0.7293 chance to correctly distinguish between if someone did default or not.

```r
library(ROCR)
library(caret)
```

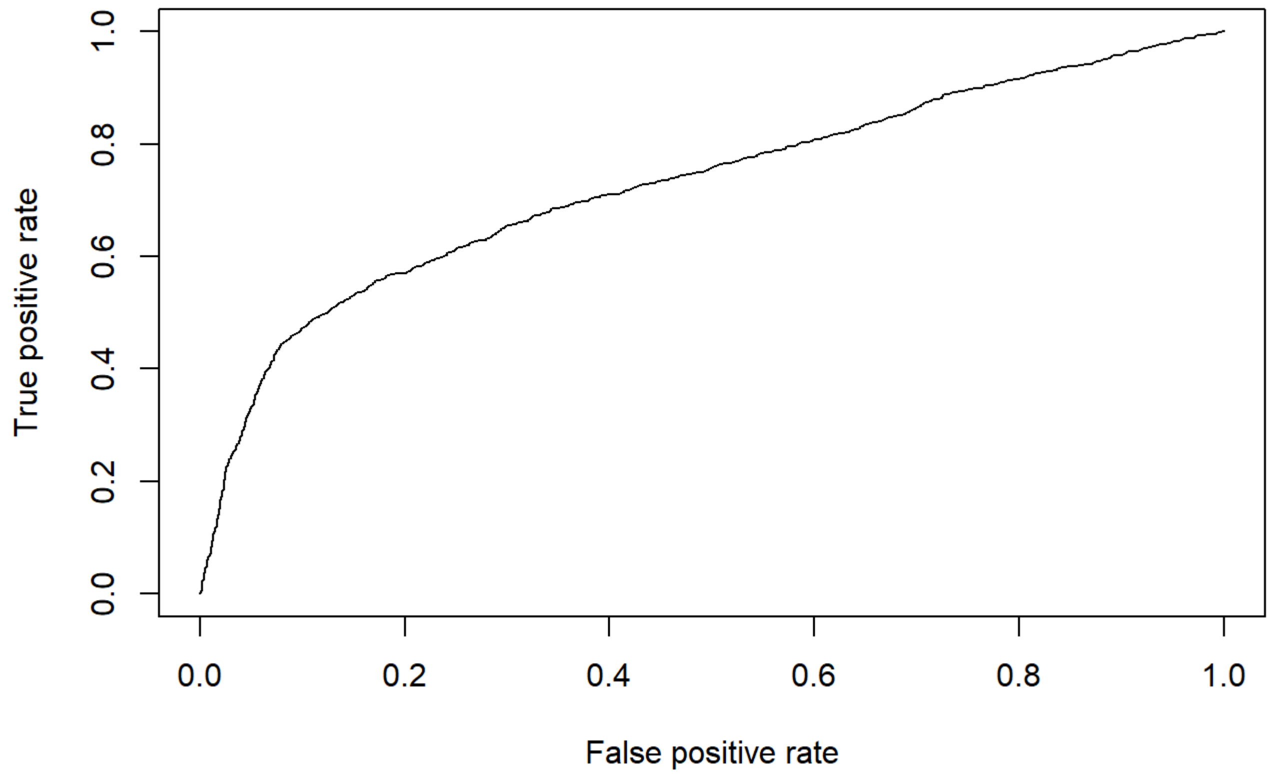
```
## Loading required package: ggplot2
```

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

```r
# logistic regession
predLog <- ifelse(predict(logModel, newdata = test, type = "response") > 0.5, "Y", "N")
confusionMatrix(as.factor(predLog), reference = test$defaulted)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N 4399  968
##          Y  145  321
##
##                Accuracy : 0.8092
##                  95% CI : (0.7989, 0.8192)
##     No Information Rate : 0.779
##     P-Value [Acc > NIR] : 8.915e-09
##
##                   Kappa : 0.2815
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9681
##             Specificity : 0.2490
##          Pos Pred Value : 0.8196
##          Neg Pred Value : 0.6888
##              Prevalence : 0.7790
##          Detection Rate : 0.7542
##    Detection Prevalence : 0.9201
##       Balanced Accuracy : 0.6086
##
##        'Positive' Class : N
##
```

```
plot(performance(prediction(predict(logModel, newdata = test, type = "response"), test$defaulte
d), measure = "tpr", x.measure = "fpr"))
```




```
print(paste("AUC:", performance(prediction(predict(logModel, newdata = test, type = "response"),
test$defaulted), measure = "auc")@y.values[[1]]))
```

```
## [1] "AUC: 0.729258234628874"
```

Now, in evaluating the naive bayes model, its clear that it does not perform as well as the logistic regression model. While the naive bayes model does not have as many false positives as the logistic regression model, it does have more overall incorrect classifications than teh logistic model. This is furth shown by its accuracy being a bit lower at about 0.7876.

```
# naive bayes
predBayes <- predict(bayesModel, newdata = test, type = "class")
predBayes_raw <- predict(bayesModel, newdata = test, type = "raw")
table(predBayes, test$defaulted)
```

```
##
## predBayes    N    Y
##         N 3947  642
##         Y  597  647
```

```
print(paste("Accuracy:", mean(predBayes == test$defaulted)))
```

```
## [1] "Accuracy: 0.787587862163552"
```