R Notebook

Hiep Nguyen

02/14/2023

How do linear models for classification work?

For Linear models for classification, we can imagine having a linear line called the decision boundary that separates categories to classify them. Generally for a binary classification, we would calculate the probability of an target to be true/false based on our predictors. If the probability is >50%, we designate it as true. If it's <50%, it's false. This can be done in many ways depending on the algorithm used. In logistic regression, we draw an S-shaped line on our graph and then use the line to calculate our probability. In Naive Bayes, we manipulate probabilities using the formula posterior = (likelihood*prior)/marginal to train our model. Although great for binary classification, these linear models are usually high bias and low variance, which means that often times it will underfit the data.

Data Cleaning and 80/20 Train/Test

Before I split the data, I did a little bit of simple data cleaning by factoring some of the attributes. I then did a 80/20 train/test split on the dataset.

```
# Data Cleaning
heart_data <- read.csv("heart_data.csv")
to_factor <- c("gender", "gluc", "cholesterol", "smoke", "alco", "active", "cardio")
heart_data[to_factor] <- lapply(heart_data[to_factor], factor)

# Added BMI to the dataset
heart_data['bmi'] = heart_data$weight / (heart_data$height/100)^2

# Train/Test
set.seed(123)
i <- sample(1:nrow(heart_data), 0.80*nrow(heart_data), replace=FALSE)
train <- heart_data[i,]
test <- heart_data[-i,]
```

Data Exploration

ap_hi = Systolic blood pressure ap_lo = Diastolic blood pressure gluc = Glucose alco = Alcohol intake cardio = Presence or absence of cardiovascular disease (Our target)

```
head(train)
```

```
##       index    id   age gender height weight ap_hi ap_lo cholesterol gluc smoke
## 51663 51662 73677 18910      1    162     80   120    79           1    1     0
```

```
## 57870 57869 82586 21971      2    164     63   140    90           1    1     0
## 2986   2985  4208 20398      1    168     78   140  8044           3    3     0
## 29925 29924 42794 20585      1    167     69   120    80           1    1     0
## 68293 68292 97533 19722      2    175     88   160   100           1    1     0
## 62555 62554 89316 16678      2    180    123   110    70           1    1     0
##       alco active cardio      bmi
## 51663    0      1      0 30.48316
## 57870    0      1      1 23.42356
## 2986     0      0      1 27.63605
## 29925    0      1      0 24.74094
## 68293    0      1      1 28.73469
## 62555    0      1      1 37.96296
```

```
tail(train)
```

```
##       index    id   age gender height weight ap_hi ap_lo cholesterol gluc smoke
## 69997 69996 99995 22601      1    158    126   140    90           2    2     0
## 13394 13393 19119 19890      2    170     84   120    70           1    1     0
## 16847 16846 24072 20392      1    163    102   130    80           1    1     0
## 28708 28707 41037 20376      1    164     70   110    70           2    1     0
## 37447 37446 53475 17468      1    160     73   120    80           1    1     0
## 17214 17213 24599 19115      1    167     68   110    70           1    1     0
##       alco active cardio      bmi
## 69997    0      1      1 50.47268
## 13394    0      1      0 29.06574
## 16847    0      0      0 38.39061
## 28708    0      1      0 26.02617
## 37447    0      1      0 28.51562
## 17214    0      1      0 24.38237
```

```
str(train)
```

```
## 'data.frame':    56000 obs. of  15 variables:
##  $ index      : int  51662 57869 2985 29924 68292 62554 45403 65160 46434 9641 ...
##  $ id         : int  73677 82586 4208 42794 97533 89316 64865 93013 66318 13745 ...
##  $ age        : int  18910 21971 20398 20585 19722 16678 20498 20315 19749 15247 ...
##  $ gender     : Factor w/ 2 levels "1","2": 1 2 1 1 2 2 1 1 1 2 ...
##  $ height     : int  162 164 168 167 175 180 169 158 155 180 ...
##  $ weight     : num  80 63 78 69 88 123 67 101 69 72 ...
##  $ ap_hi      : int  120 140 140 120 160 110 120 120 120 120 ...
##  $ ap_lo      : int  79 90 8044 80 100 70 70 80 80 80 ...
##  $ cholesterol: Factor w/ 3 levels "1","2","3": 1 1 3 1 1 1 1 2 1 1 ...
##  $ gluc       : Factor w/ 3 levels "1","2","3": 1 1 3 1 1 1 1 1 1 1 ...
##  $ smoke      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ alco       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ active     : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 1 2 2 2 ...
##  $ cardio     : Factor w/ 2 levels "0","1": 1 2 2 1 2 2 1 2 1 2 ...
##  $ bmi        : num  30.5 23.4 27.6 24.7 28.7 ...
```

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

```
##       index          id         age      gender      height      weight
```

```
##           0           0           0           0           0           0
##       ap_hi       ap_lo cholesterol        gluc       smoke        alco
##           0           0           0           0           0           0
##      active      cardio         bmi
##           0           0           0
```

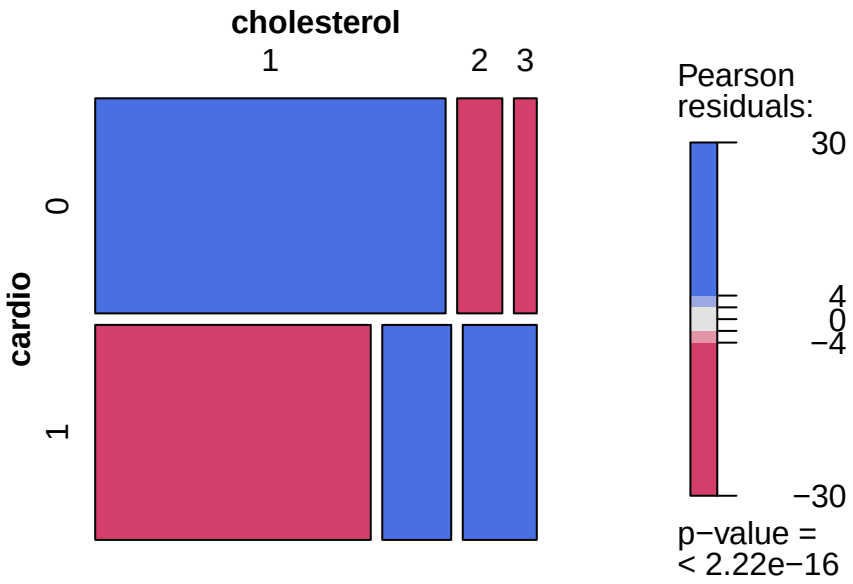
Graphing

As we can see from the graphs below, both higher levels of cholesterol and glucose seem to have an effect on cardiovascular health. For age, the older you are, the more likely you are to develop the disease, too. It can also be seen that if you weigh too little, you're most likely to have the disease. The goldilock's spot appears around 50 kgs and gradually gets larger as we go up the weight. Looking at the ap_hi and ap_lo graphs, we can see the graph being heavily skewed due to some rather extremely high outliers.

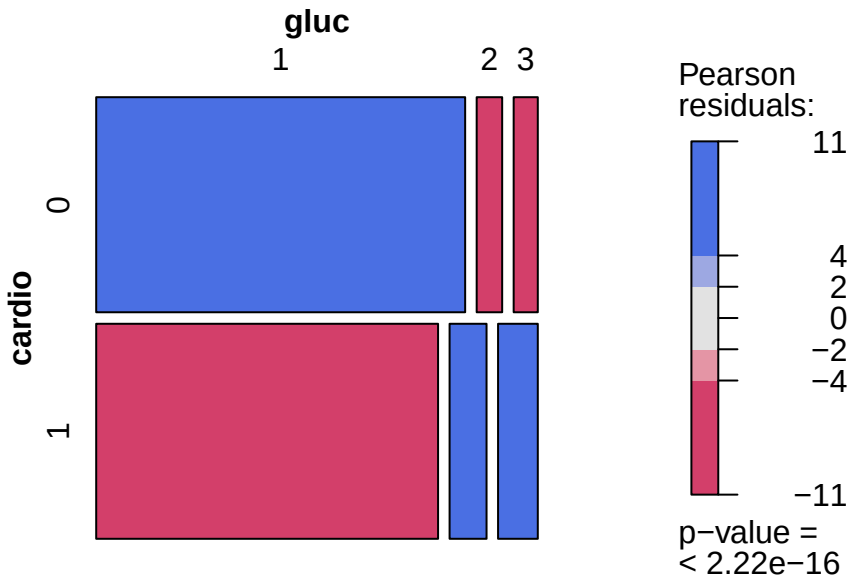
```
library(vcd)
```

```
## Loading required package: grid
```

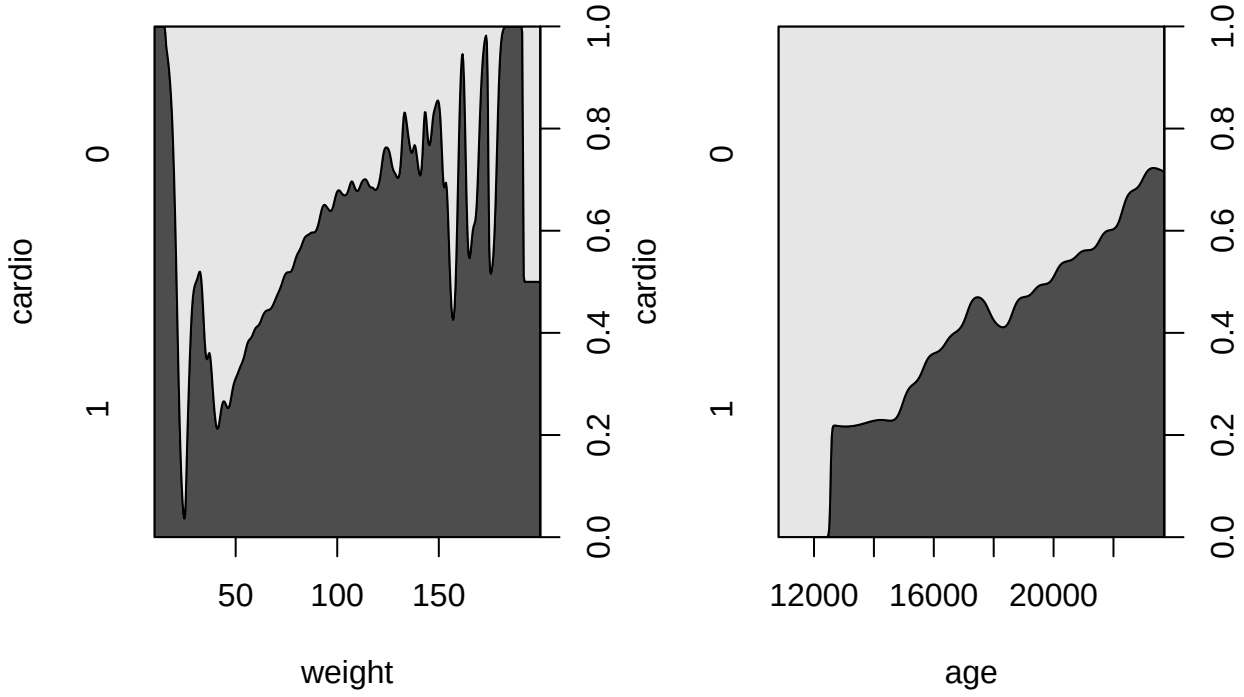
```
mosaic(table(train[,c(14, 9)]), shade=TRUE, legend=TRUE)
```



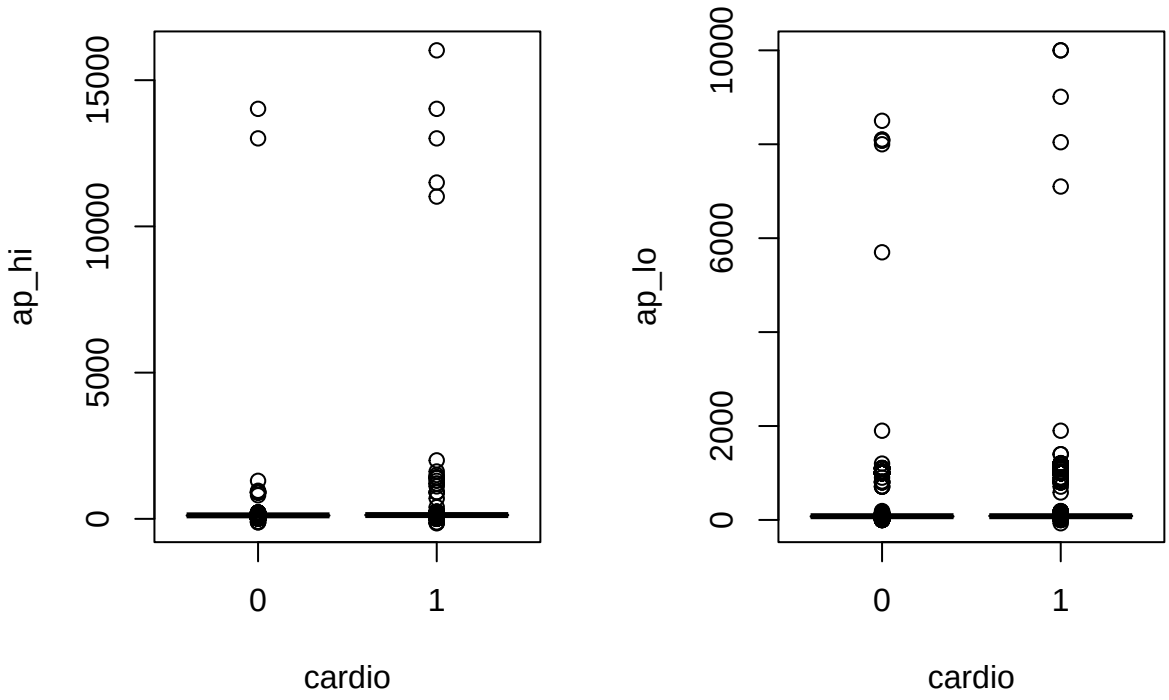

```
mosaic(table(train[,c(14, 10)]), shade=TRUE, legend=TRUE)
```




```
par(mfrow=c(1,2))
cdplot(cardio~weight, data=train)
cdplot(cardio~age, data=train)
```

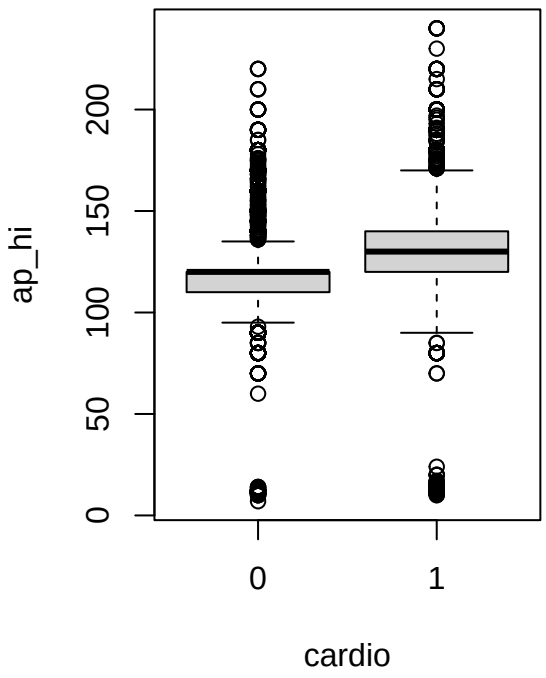


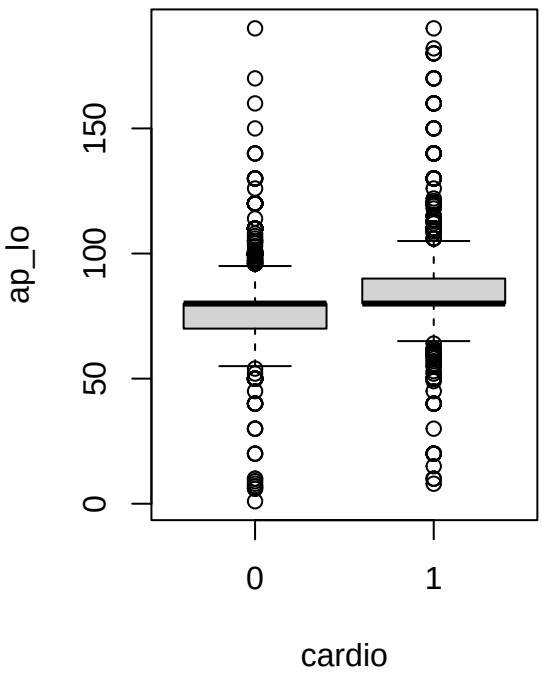

```
par(mfrow=c(1,2))
boxplot(ap_hi~cardio, data=train)
boxplot(ap_lo~cardio, data=train)
```



I decided to remove some of the outliers in the ap_hi and ap_lo graphs by limiting the range to be (0, 250).

```
train <- train[train$ap_hi < 250,]
train <- train[train$ap_lo < 250,]
train <- train[train$ap_hi > 0,]
train <- train[train$ap_lo > 0,]
par(mfrow=c(1,2))
boxplot(ap_hi~cardio, data=train)
boxplot(ap_lo~cardio, data=train)
```





Logistics Regression Model

The null deviance determines how well our model fits the data using only the intercept, while the residual deviance determines how well it fits using all the predictors. Generally, we want to see a large difference between the null deviance and the residual deviance. In this case, we do see a difference going from 76501 to 63905, but it's not a very large one, which means our predictors only marginally improves the model. The deviance residuals show how much influence a datapoint may have on the model, producing stats similar to RSS. Our AIC is based on deviance and can be used for comparison between models. In this case, we don't have another logistics model, so it may not be useful for us. Most of the predictors can be seen with three stars, which means that the p-value is low enough to reject the null hypothesis for those predictors.

```
glm1 <- glm(cardio~bmi+cholesterol+gluc+ap_hi+ap_lo, data=train, family=binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = cardio ~ bmi + cholesterol + gluc + ap_hi + ap_lo,
##     family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7932  -0.9511  -0.3812   0.9416   3.5930
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -8.9440512  0.1094131 -81.746  < 2e-16 ***
## bmi           0.0239720  0.0018711  12.812  < 2e-16 ***
## cholesterol2  0.4026022  0.0298733  13.477  < 2e-16 ***
## cholesterol3  1.1789148  0.0390688  30.175  < 2e-16 ***
## gluc2         0.0302181  0.0393633   0.768    0.443
## gluc3        -0.2804216  0.0434390  -6.456 1.08e-10 ***
## ap_hi         0.0502419  0.0009221  54.488  < 2e-16 ***
## ap_lo         0.0220369  0.0014455  15.245  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 76501  on 55187  degrees of freedom
## Residual deviance: 63905  on 55180  degrees of freedom
## AIC: 63921
##
## Number of Fisher Scoring iterations: 4
```

Naive Bayes Model

Our A-priori shows us the probability of having cardiovascular disease prior from doing any data analysis. We could calculate this just by comparing the number of people who have the disease to the total number of people in the data set. We would have a 49.5% chance of having cardiovascular disease and a 50.5% of not having cardiovascular disease. For our quantitative data, such as bmi, we would get the mean and the standard deviation. For example, out of the patients who had cardiovascular disease, their bmi was average 28.5 with a standard deviation of 6.4. For qualitative data, such as cholesterol, we get the probability of each situation. If the patient had cardiovascular disease, there was a 65.9% probability they had a normal cholesterol level and a 17.7% chance they had way above average cholesterol levels.

```
library(e1071)
nb1 <- naiveBayes(cardio~bmi+cholesterol+gluc+ap_hi+ap_lo, data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.5049105 0.4950895
##
## Conditional probabilities:
##    bmi
## Y       [,1]     [,2]
##   0 26.53622 5.678451
##   1 28.51095 6.402257
##
##    cholesterol
## Y            1          2          3
```

```
##   0 0.83789700 0.10759017 0.05451283
##   1 0.65959082 0.16341544 0.17699374
##
##    gluc
## Y           1          2          3
##   0 0.88171541 0.06064956 0.05763503
##   1 0.81671120 0.08809428 0.09519452
##
##    ap_hi
## Y       [,1]     [,2]
##   0 119.2416 13.82109
##   1 133.5382 18.32452
##
##    ap_lo
## Y       [,1]      [,2]
##   0 78.13275  8.482959
##   1 84.65384 10.047909
```

Testing and Predictions

The Naive Bayes Model had both a lower kappa value (.433 vs .444), a lower accuracy value (.716 vs .722), and a lower AUC value (.716 vs .722) than the logistics regression model, which means the logistics regressin model is better just from its stats. This is likely due to the fact that Naive Bayes assumes that the variables are all independent of each other, when in reality, all of the predictors used probably had an effect on each other (e.g., cholesterol levels and blood pressure). Both the models are extremely similar in score though, so I don't think it would be fair to pick one model over the other.

```
library(caret)
```

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

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

```
library(ROCR)
par(mfrow=c(1,1))

# Logistics Regression Model
cat("Logistics Regression Model:\n")
```

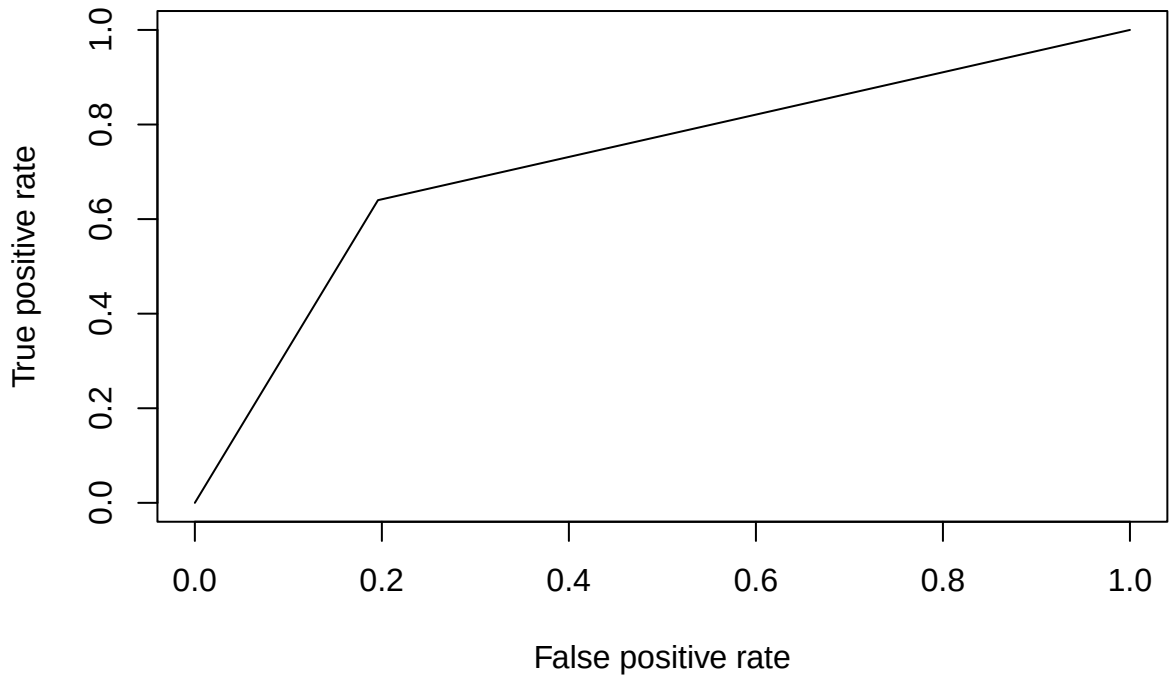
```
## Logistics Regression Model:
```

```
probs1 <- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1>0.5, 1, 0)
confusionMatrix(as.factor(pred1), reference=test$cardio)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5642 2514
##          1 1374 4470
```

```
##
##                Accuracy : 0.7223
##                  95% CI : (0.7148, 0.7297)
##     No Information Rate : 0.5011
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4444
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8042
##             Specificity : 0.6400
##          Pos Pred Value : 0.6918
##          Neg Pred Value : 0.7649
##              Prevalence : 0.5011
##          Detection Rate : 0.4030
##    Detection Prevalence : 0.5826
##       Balanced Accuracy : 0.7221
##
##        'Positive' Class : 0
##
```

```
pr1 <- prediction(pred1, test$cardio)
prf1 <- performance(pr1, measure="tpr", x.measure="fpr")
plot(prf1)
```



```
auc1 <- performance(pr1, measure="auc")
auc1 <- auc1@y.values[[1]]
print(paste("AUC: ", auc1))
```

```
## [1] "AUC:  0.722098139941302"
```

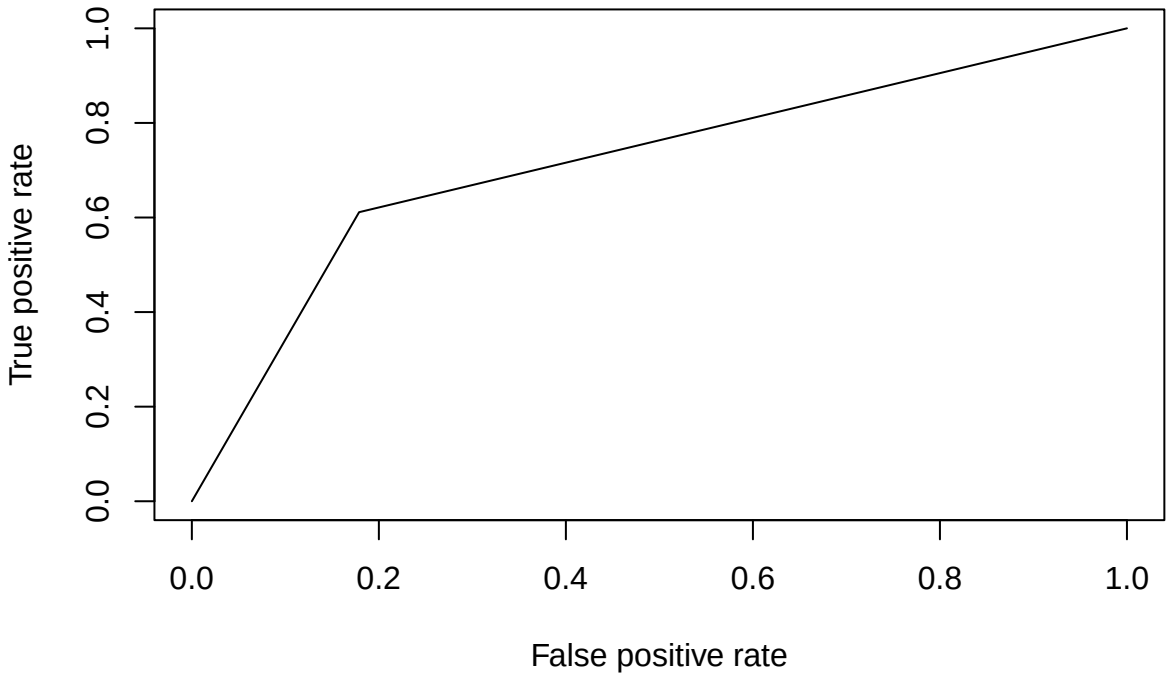
```
library(caret)
library(ROCR)
par(mfrow=c(1,1))

# Naive Bayes Model
cat("\nNaive Bayes Model:\n")
```

```
##
## Naive Bayes Model:
```

```
pred2 <- predict(nb1, newdata=test, type="class")
confusionMatrix(as.factor(pred2), reference=test$cardio)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5761 2715
##          1 1255 4269
##
##                Accuracy : 0.7164
##                  95% CI : (0.7089, 0.7239)
##     No Information Rate : 0.5011
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4326
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8211
##             Specificity : 0.6113
##          Pos Pred Value : 0.6797
##          Neg Pred Value : 0.7728
##              Prevalence : 0.5011
##          Detection Rate : 0.4115
##    Detection Prevalence : 0.6054
##       Balanced Accuracy : 0.7162
##
##        'Positive' Class : 0
##
```

```
predvec <- ifelse(pred2==1, 1, 0)
pr2 <- prediction(predvec, test$cardio)
prf2 <- performance(pr2, measure="tpr", x.measure="fpr")
plot(prf2)
```




```
auc2 <- performance(pr2, measure="auc")
auc2 <- auc2@y.values[[1]]
print(paste("AUC: ", auc2))
```

```
## [1] "AUC:  0.716188721312503"
```

Strengths and Weaknesses of Naive Bayes vs Logistic Regression

Naive Bayes is widely used as a baseline algorithm, as it's very simple to implement and interpret and it's also very strong in it's own right. It excels when used on smaller data sets, but it can also work with higher dimensions, too. As for its weaknesses, it often performs poorly on larger data sets compared to other models and it also takes under the assumption that all of its predictors are independent, meaning that if they aren't then the algorithm could be bottlenecked.

Logistics Regression is great when it comes to differentiating between linearly separable classes. The algorithm itself isn't very hard computationally either while still giving good results. However, it does share the same weaknesses of linear regression, as often times it'll underfit the data and struggle in cases where the decision boundary is non-linear.

Benefits, drawbacks of each of the classification metrics

The accuracy value is solely based on how many correct predictions were made. The benefit of this is that it's straight forward and easy to understand, but it doesn't really account for pure chance. The kappa value is the accuracy value but adjusted for pure luck/chance. In both of the models, we can see a kappa value

slight above 0.4, which means we have a moderate agreement between our data. The disadvantage of kappa, however, is that it's difficult to interpret on its own, and there are many different definitions of what a 'good' kappa score would be. The ROCR curve is the trade-off of showing TP and avoiding FP. Generally, we would like to see the curve reach as far as it can to the top left. The AUC statistic is the area under the ROCR curve. This is beneficial in deciding where at which point someone would want the TP to FP ratio to be. However, the drawback is that it doesn't account for misclassification.