Similarity and Ensamble : Regression

Fredrick Horn

Data from: https://www.kaggle.com/datasets/mitishaagarwal/patient
(Kaggle:%20Patient%20Survival%20Prediction)

Inital Data Read

First I read in the data from the csv. I then select the most relavent columns, keeping in mind that there are 85 total and choosing too many column will drastically slow down these algorithms. Next I remove all N/A's frim the data set. There are enough cases that are complete that it shoudl not matter to remove all the incomplete ones.

```
set.seed(0)
patients <- read.csv("dataset.csv", header = TRUE, stringsAsFactors = TRUE) # read in csv
patients <- patients[, c(4, 6, 27, 29, 34, 37, 41, 42, 43, 49, 54, 61, 69, 70, 72, 73, 75, 76, 7
8, 81, 85)] # select relavent columns
patients <- patients[complete.cases(patients), ] # remove all rows with NA in any column
patients <- patients[patients$apache_4a_hospital_death_prob >= 0,] #remove invalid values
patients <- patients[patients$apache_4a_icu_death_prob >= 0,] #remove invalid values
patients$hospital_death <- as.factor(patients$hospital_death) # labels to compare agains.
levels(patients$hospital_death) <- c("N", "Y")
str(patients)
```

```
## 'data.frame':    65458 obs. of  21 variables:
##  $ age                        : int  68 77 81 67 59 70 50 72 65 81 ...
##  $ elective_surgery           : int  0 0 1 0 0 0 0 1 1 1 ...
##  $ resprate_apache            : num  36 33 4 35 53 28 46 15 42 31 ...
##  $ ventilated_apache          : int  0 1 1 0 1 1 0 0 0 0 ...
##  $ d1_heartrate_max           : int  119 118 116 113 112 118 96 101 116 119 ...
##  $ d1_mbp_min                 : int  46 38 84 80 97 60 59 70 80 77 ...
##  $ d1_resprate_min            : int  10 12 7 10 16 12 14 14 11 16 ...
##  $ d1_spo2_max                : int  100 100 100 97 100 100 100 99 100 100 ...
##  $ d1_spo2_min                : int  74 70 95 91 87 92 96 92 84 89 ...
##  $ d1_temp_min                : num  37.2 35.1 34.8 36.6 35 36.6 36.4 36.7 36.6 36.3 ...
##  $ h1_heartrate_max           : int  119 114 100 83 79 118 96 90 108 116 ...
##  $ h1_resprate_min            : int  18 28 11 12 18 26 17 14 19 16 ...
##  $ d1_glucose_min             : int  109 128 88 125 129 129 134 133 119 120 ...
##  $ d1_potassium_max           : num  4 4.2 5 3.9 5 5.8 4.1 4.2 4.4 4.9 ...
##  $ apache_4a_hospital_death_prob: num  0.1 0.47 0.04 0.05 0.1 0.11 0.02 0.01 0.01 0.03 ...
##  $ apache_4a_icu_death_prob   : num  0.05 0.29 0.03 0.02 0.05 0.06 0.01 0 0 0.01 ...
##  $ cirrhosis                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diabetes_mellitus          : int  1 1 0 1 1 0 0 0 0 0 ...
##  $ immunosuppression          : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ solid_tumor_with_metastasis : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hospital_death             : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
```

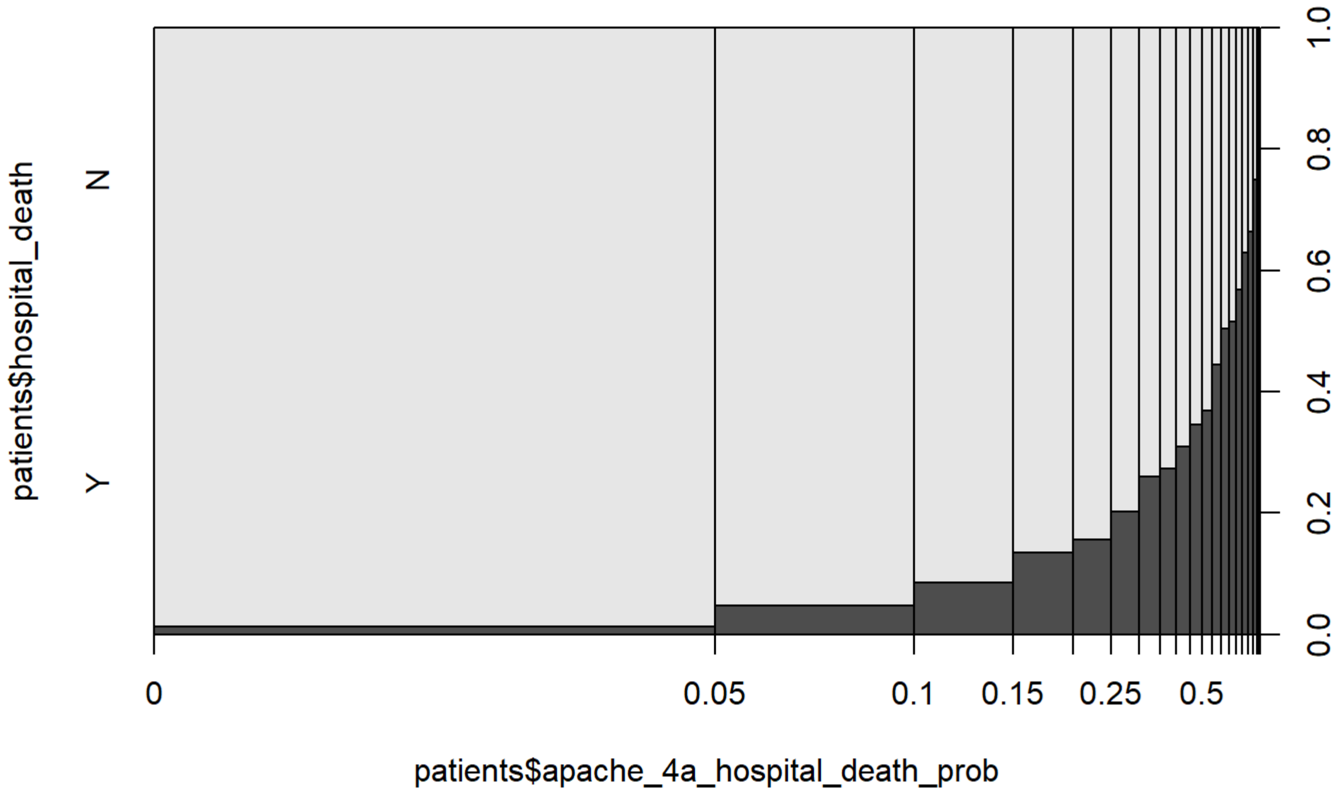
```r
summary(patients)
```

```
##       age        elective_surgery resprate_apache ventilated_apache
##  Min.   :16.00   Min.   :0.0000   Min.   : 4.00   Min.   :0.0000   
##  1st Qu.:53.00   1st Qu.:0.0000   1st Qu.:11.00   1st Qu.:0.0000   
##  Median :65.00   Median :0.0000   Median :28.00   Median :0.0000   
##  Mean   :62.63   Mean   :0.1994   Mean   :25.85   Mean   :0.3499   
##  3rd Qu.:75.00   3rd Qu.:0.0000   3rd Qu.:36.00   3rd Qu.:1.0000   
##  Max.   :89.00   Max.   :1.0000   Max.   :60.00   Max.   :1.0000   
##  d1_heartrate_max   d1_mbp_min     d1_resprate_min  d1_spo2_max    
##  Min.   : 58.0    Min.   : 22.00   Min.   : 0.00   Min.   : 13.00  
##  1st Qu.: 88.0    1st Qu.: 54.00   1st Qu.:10.00   1st Qu.: 99.00  
##  Median :102.0    Median : 63.00   Median :13.00   Median :100.00  
##  Mean   :103.9    Mean   : 64.34   Mean   :12.68   Mean   : 99.32  
##  3rd Qu.:117.0    3rd Qu.: 74.00   3rd Qu.:16.00   3rd Qu.:100.00  
##  Max.   :177.0    Max.   :112.00   Max.   :96.00   Max.   :100.00  
##   d1_spo2_min      d1_temp_min    h1_heartrate_max h1_resprate_min 
##  Min.   :  0.00   Min.   :31.89   Min.   : 46.00   Min.   :  0.00  
##  1st Qu.: 89.00   1st Qu.:36.10   1st Qu.: 77.00   1st Qu.: 13.00  
##  Median : 92.00   Median :36.40   Median : 90.00   Median : 16.00  
##  Mean   : 90.49   Mean   :36.25   Mean   : 92.78   Mean   : 17.11  
##  3rd Qu.: 95.00   3rd Qu.:36.60   3rd Qu.:106.00   3rd Qu.: 20.00  
##  Max.   :100.00   Max.   :37.80   Max.   :164.00   Max.   :129.00  
##  d1_glucose_min  d1_potassium_max apache_4a_hospital_death_prob
##  Min.   : 33.0   Min.   :2.800    Min.   :0.0000               
##  1st Qu.: 90.0   1st Qu.:3.800    1st Qu.:0.0200               
##  Median :107.0   Median :4.200    Median :0.0500               
##  Mean   :113.5   Mean   :4.254    Mean   :0.1205               
##  3rd Qu.:130.0   3rd Qu.:4.600    3rd Qu.:0.1400               
##  Max.   :288.0   Max.   :7.000    Max.   :0.9800               
##  apache_4a_icu_death_prob   cirrhosis       diabetes_mellitus immunosuppression
##  Min.   :0.00000          Min.   :0.00000   Min.   :0.0000    Min.   :0.00000  
##  1st Qu.:0.01000          1st Qu.:0.00000   1st Qu.:0.0000    1st Qu.:0.00000  
##  Median :0.02000          Median :0.00000   Median :0.0000    Median :0.00000  
##  Mean   :0.07543          Mean   :0.01691   Mean   :0.2354    Mean   :0.02797  
##  3rd Qu.:0.07000          3rd Qu.:0.00000   3rd Qu.:0.0000    3rd Qu.:0.00000  
##  Max.   :0.97000          Max.   :1.00000   Max.   :1.0000    Max.   :1.00000  
##  solid_tumor_with_metastasis hospital_death
##  Min.   :0.0000              N:59875       
##  1st Qu.:0.0000              Y: 5583       
##  Median :0.0000                            
##  Mean   :0.0216                            
##  3rd Qu.:0.0000                            
##  Max.   :1.0000
```

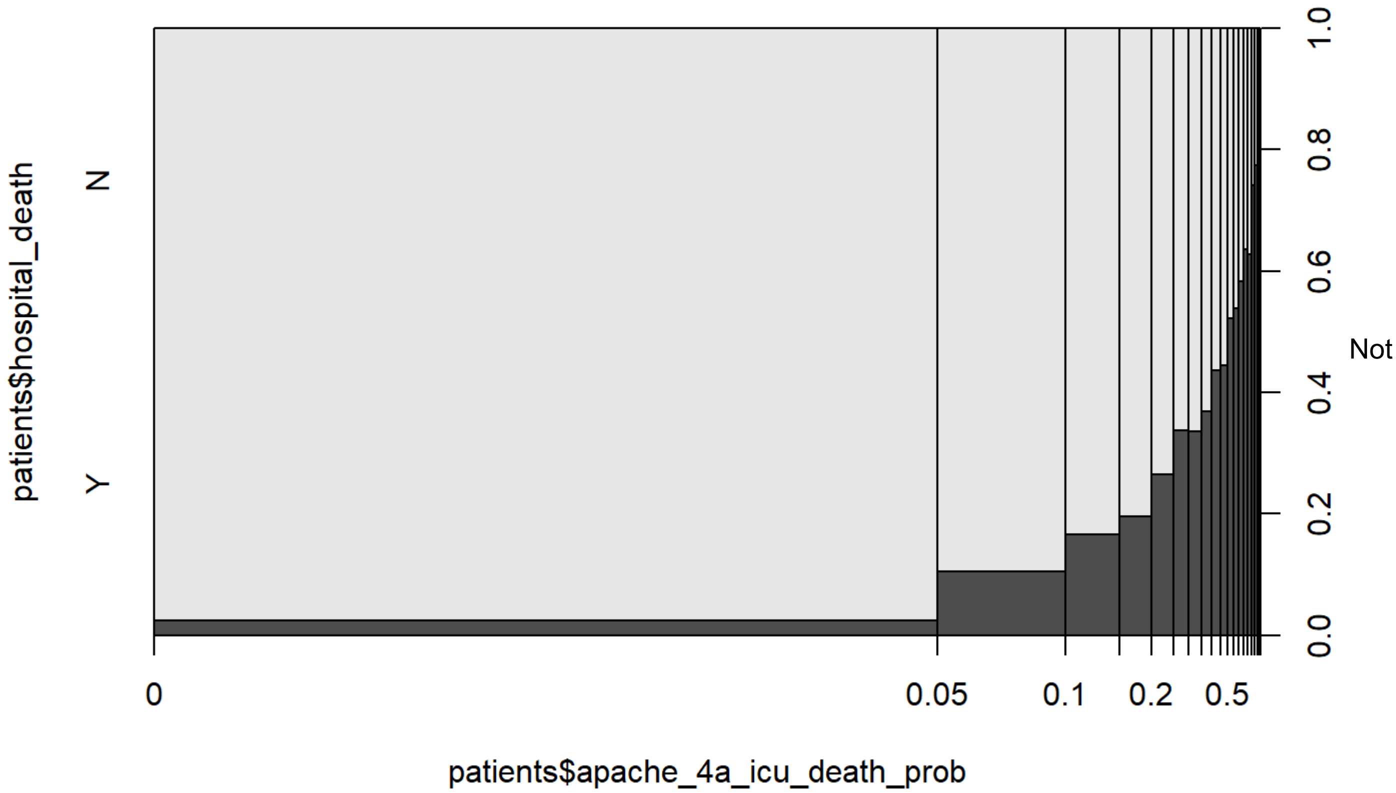
Data Exploration

This data set includes a calculated probabilty of death based on many of the other columns and it performs very well as a predictor.

```
plot(patients$hospital_death ~ patients$apache_4a_hospital_death_prob)
```

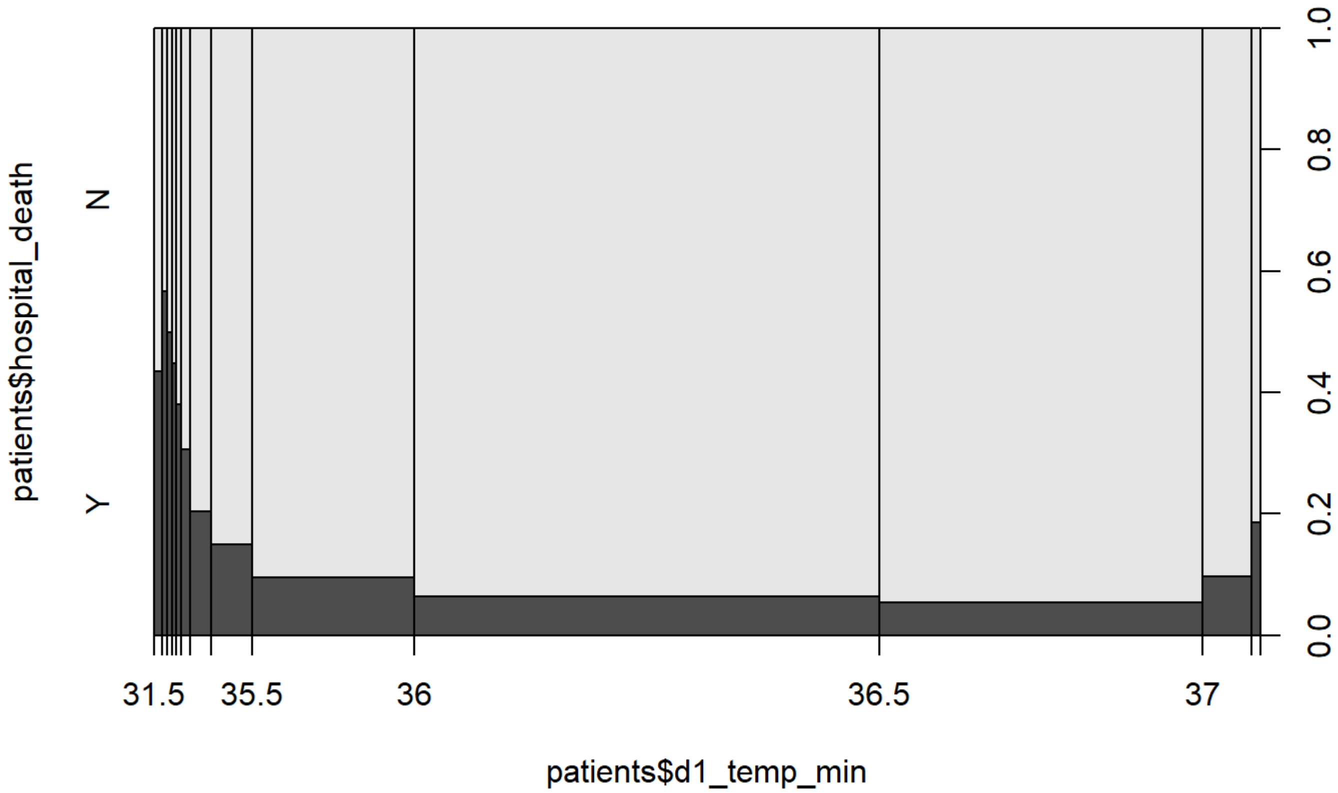



```
plot(patients$hospital_death ~ patients$apache_4a_icu_death_prob)
```

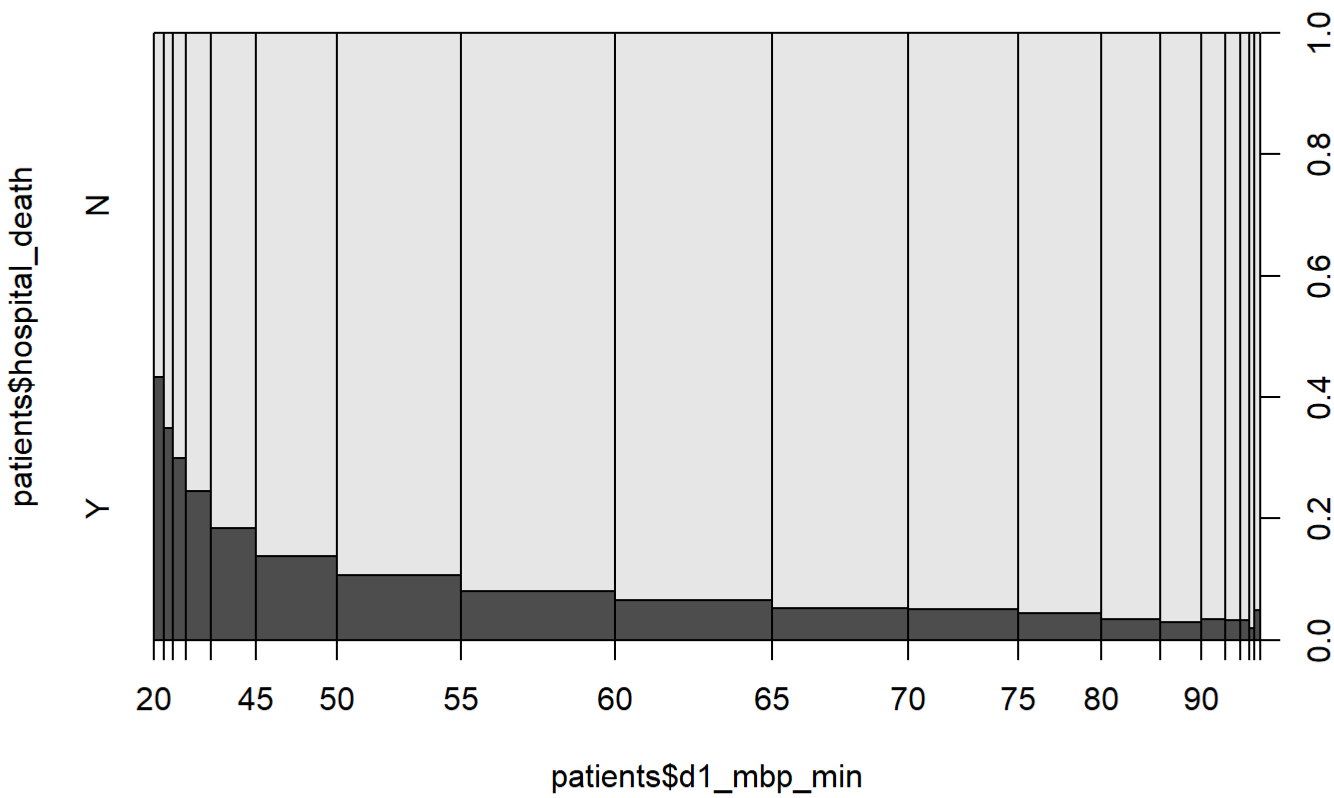


every signficant column is shown in these graphs but generally the min recorded value columns show a signficant increase in deaths in their minimums and the max recorded values show an increase in deaths in their max recorded values.

```
plot(patients$hospital_death ~ patients$d1_temp_min)
```




```
plot(patients$hospital_death ~ patients$d1_mbp_min)
```



Logistic Regression

For organization, I will move the data into a new var for the logistic regression to use and will load the libraries needed for the logistic regresion.

```
logData <- patients

library(ROCR)
library(caret)
```

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

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

Next I split the data on into 80% train and 20% test

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

Create the logistic regression model and output its summary.

```r
logModel <- glm(hospital_death ~ ., data = train, family = "binomial")
summary(logModel)
```

```
##
## Call:
## glm(formula = hospital_death ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6384  -0.3312  -0.2254  -0.1627   3.3746
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 10.0361667  1.4393336   6.973 3.11e-12 ***
## age                          0.0136049  0.0014553   9.349  < 2e-16 ***
## elective_surgery            -0.5442998  0.0673080  -8.087 6.13e-16 ***
## resprate_apache              0.0054584  0.0013866   3.937 8.27e-05 ***
## ventilated_apache            0.5104063  0.0453462  11.256  < 2e-16 ***
## d1_heartrate_max             0.0132074  0.0011747  11.243  < 2e-16 ***
## d1_mbp_min                  -0.0215673  0.0013413 -16.079  < 2e-16 ***
## d1_resprate_min              0.0248534  0.0040226   6.179 6.47e-10 ***
## d1_spo2_max                 -0.0588046  0.0123119  -4.776 1.79e-06 ***
## d1_spo2_min                 -0.0210296  0.0014220 -14.788  < 2e-16 ***
## d1_temp_min                 -0.2216483  0.0187681 -11.810  < 2e-16 ***
## h1_heartrate_max            -0.0051413  0.0011753  -4.375 1.22e-05 ***
## h1_resprate_min              0.0115345  0.0033233   3.471 0.000519 ***
## d1_glucose_min               0.0022885  0.0004545   5.035 4.77e-07 ***
## d1_potassium_max             0.1708806  0.0246455   6.934 4.10e-12 ***
## apache_4a_hospital_death_prob  5.7881623  0.3675660  15.747  < 2e-16 ***
## apache_4a_icu_death_prob    -2.0695766  0.4168978  -4.964 6.90e-07 ***
## cirrhosis                    0.4786115  0.1141170   4.194 2.74e-05 ***
## diabetes_mellitus           -0.1913360  0.0453811  -4.216 2.48e-05 ***
## immunosuppression            0.2667489  0.0929993   2.868 0.004127 **
## solid_tumor_with_metastasis  0.4074337  0.1046730   3.892 9.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 30357  on 52365  degrees of freedom
## Residual deviance: 21201  on 52345  degrees of freedom
## AIC: 21243
##
## Number of Fisher Scoring iterations: 6
```

To evaluate the model, I create a factor of the prediction on the test data and output the confusion matrix from the caret library. I also plot the ROC curve and area under that curve.

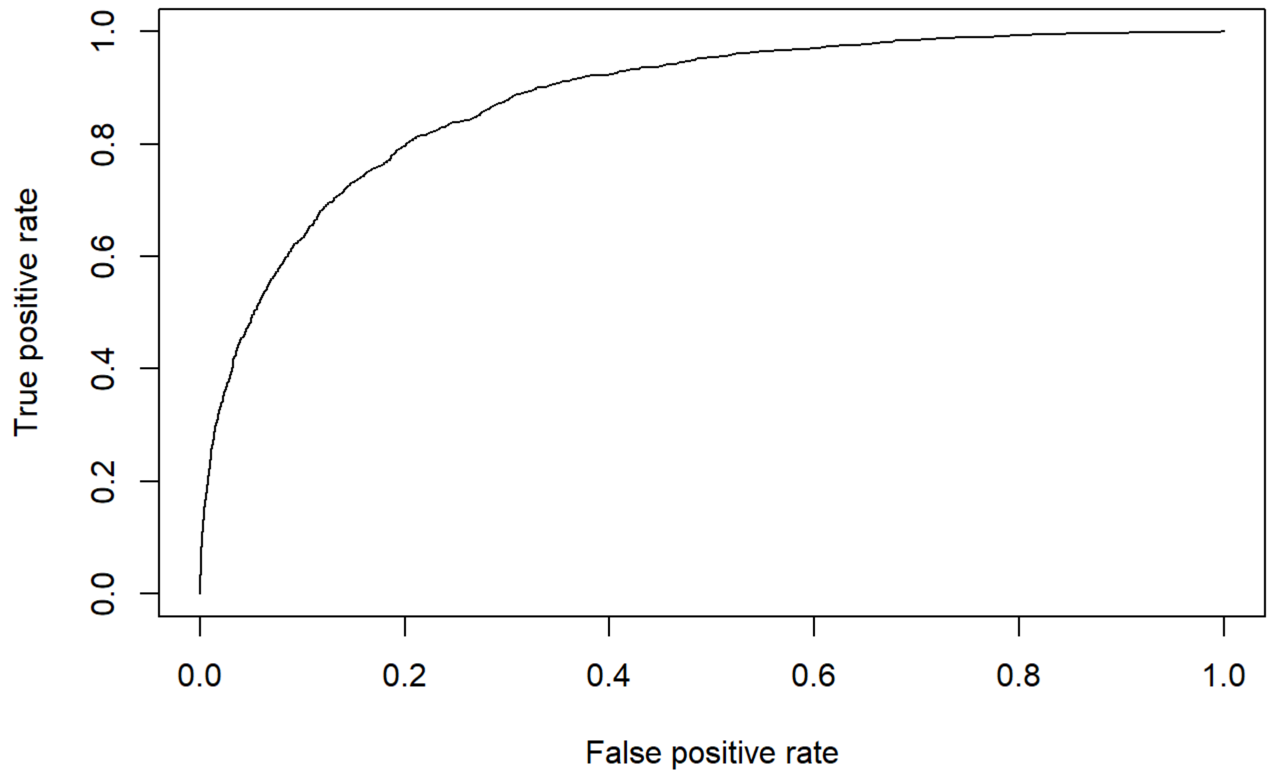
```
predLog <- ifelse(predict(logModel, newdata = test, type = "response") > 0.5, "Y", "N")
cMat <- confusionMatrix(as.factor(predLog), reference = test$hospital_death)
cMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 11768   818
##          Y   171   335
##
##                Accuracy : 0.9245
##                  95% CI : (0.9198, 0.9289)
##     No Information Rate : 0.9119
##     P-Value [Acc > NIR] : 1.284e-07
##
##                   Kappa : 0.37
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9857
##             Specificity : 0.2905
##          Pos Pred Value : 0.9350
##          Neg Pred Value : 0.6621
##              Prevalence : 0.9119
##          Detection Rate : 0.8989
##    Detection Prevalence : 0.9614
##       Balanced Accuracy : 0.6381
##
##        'Positive' Class : N
##
```

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

```
## [1] "AUC: 0.879347001492809"
```

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



K Nearest Neighbors

For kNN, I move the data into a new var and then convert all the int types to numeric types as kNN does not play nicely with the int type.

```r
knnData <- patients

#make everything numeric
knnData$age <- as.numeric(knnData$age)
knnData$elective_surgery <- as.numeric(knnData$elective_surgery)
knnData$ventilated_apache <- as.numeric(knnData$ventilated_apache)
knnData$d1_heartrate_max <- as.numeric(knnData$d1_heartrate_max)
knnData$d1_mbp_min <- as.numeric(knnData$d1_mbp_min)
knnData$d1_resprate_min <- as.numeric(knnData$d1_resprate_min)
knnData$d1_spo2_max <- as.numeric(knnData$d1_spo2_max)
knnData$d1_spo2_min <- as.numeric(knnData$d1_spo2_min)
knnData$h1_heartrate_max <- as.numeric(knnData$h1_heartrate_max)
knnData$h1_resprate_min <- as.numeric(knnData$h1_resprate_min)
knnData$d1_glucose_min <- as.numeric(knnData$d1_glucose_min)
knnData$cirrhosis <- as.numeric(knnData$cirrhosis)
knnData$diabetes_mellitus <- as.numeric(knnData$diabetes_mellitus)
knnData$immunosuppression <- as.numeric(knnData$immunosuppression)
knnData$solid_tumor_with_metastasis <- as.numeric(knnData$solid_tumor_with_metastasis)

library(class)
```

The data is split into 80% test and 20% train again. This time though I remove the data lables and move it into its own var for uses in the evaluation. I also make sure to scale the data as it does improve the performace.

```r
i <- sample(1:nrow(knnData), 0.8 * nrow(knnData), replace = FALSE) # split data
train <- knnData[i, -21] # 80% train
test <- knnData[-i, -21] # 20% test
trainLabels <- knnData[i, 21]
testLabels <- knnData[-i, 21]
#scale data
means <- sapply(train, mean)
stdvs <- sapply(train, sd)
train <- scale(train, center = means, scale = stdvs)
test <- scale(test, center = means, scale = stdvs)
```

Create the kNN model.

```r
knnModel <- knn(train, test, cl = trainLabels, k = 2)
```

First I calclate accuracy

```r
knnAcc <- length(which(knnModel == testLabels))/length(knnModel)
predKnn <- as.factor(knnModel != testLabels)
levels(predKnn) <- c("N","Y")
table(predKnn, knnModel)
```

```
##        knnModel
## predKnn     N     Y
##       N 11401   310
##       Y   770   611
```

```
print(paste("kNN Accuracy:",knnAcc))
```

```
## [1] "kNN Accuracy: 0.894515734799878"
```

Decision Tree

Make seperate var for decision tree data and load the tree library.

```
dtData <- patients

library(tree)
```

```
## Warning: package 'tree' was built under R version 4.2.3
```

Split into 80% train and 20% test.

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

Create the decision tree model and output the tree.

```
dtModel <- tree(hospital_death ~ ., data=train)
dtModel
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 52366 30550 N ( 0.914620 0.085380 )
##   2) apache_4a_hospital_death_prob < 0.165 41359 11310 N ( 0.969414 0.030586 )
##     4) apache_4a_hospital_death_prob < 0.045 23419  2501 N ( 0.990563 0.009437 ) *
##     5) apache_4a_hospital_death_prob > 0.045 17940  7964 N ( 0.941806 0.058194 ) *
##   3) apache_4a_hospital_death_prob > 0.165 11007 13280 N ( 0.708731 0.291269 )
##     6) apache_4a_hospital_death_prob < 0.465 7829  7822 N ( 0.800613 0.199387 ) *
##     7) apache_4a_hospital_death_prob > 0.465 3178  4402 Y ( 0.482379 0.517621 ) *
```

Then run the prediction using the model and the test data and output the accuracy and confusion matrix.

```
dtPred <- predict(dtModel, newdata = test, type="class")
table(dtPred, test$hospital_death)
```

```
##
## dtPred     N     Y
##      N 11583   692
##      Y   397   420
```

```
dtAcc <- mean(dtPred == test$hospital_death)
print(paste("Decision Tree Accuracy:",dtAcc))
```

```
## [1] "Decision Tree Accuracy: 0.916819431714024"
```

Analysis

These are the accuracy's of the 3 models

```
print(paste("Log Reg Accuracy:", cMat$overall["Accuracy"]))
```

```
## [1] "Log Reg Accuracy: 0.924457684081882"
```

```
print(paste("kNN Accuracy:", knnAcc))
```

```
## [1] "kNN Accuracy: 0.894515734799878"
```

```
print(paste("DT Accuracy:", dtAcc))
```

```
## [1] "DT Accuracy: 0.916819431714024"
```

The logistic regresion and decision tree models perform very similarity to each other. This is interesting as the decision tree seems to only care about a single predictor while the logistic regresion is using all the predictors. Logistic regression tends to perform slightly better, though just by changing the seed set at the beginning the gap in accuracy between them can get very small. kNN is the only odd one out here. Before scaling the data it performed the worst with around 87% accuracy. But even after scaling it only gained about 2% accuracy.