MVA_Assignment_8

Aman

11/05/2020

Assignment 8 - Logistic regression

This document performs Logistic Regression on the Heart Failure Prediction dataset. We iterate over multiple models to come up with the most robust model.

Let us load libraries and data

```r
# clear environment
rm(list = ls())

# defining libraries

library(ggplot2)
library(dplyr)
library(PerformanceAnalytics)
library(data.table)
library(sqldf)
library(nortest)
library(MASS)
library(rpart)
library(class)
library(ISLR)
library(scales)
library(ClustOfVar)
library(GGally)
library(reticulate)
library(ggthemes)
library(RColorBrewer)
library(gridExtra)
library(kableExtra)
library(Hmisc)
library(corrplot)
library(energy)
library(nnet)
library(Hotelling)
library(car)
library(devtools)
library(ggbiplot)
library(factoextra)
library(rgl)
library(FactoMineR)
library(psych)
library(nFactors)
library(scatterplot3d)
library(lmtest)
library(mctest)
library(aod)
library(InformationValue)
library(pROC)
library(tidyverse)
library(caret)
library(Information)
```

```r
# reading data
data <- read.csv('/Users/mac/Downloads/heart_failure_clinical_records_dataset.csv')
str(data)
```

```
## 'data.frame':    299 obs. of  13 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
##  $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
##  $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
##  $ platelets               : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
##  $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
##  $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
##  $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

Fitting a logistic regression model

We recall three key points from third assignment (EDA) -

1. Our data has no missing values
2. We saw 4 observations as outliers
3. We saw no multicollinearity as our VIF values were all below 2

Hence we have a very small pre-processing step of removing outliers.

Data Cleaning - Let's remove these outliers

```r
data <- data[data$ejection_fraction <70,]
data <- data[data$creatinine_phosphokinase <7000,]
str(data)
```

```
## 'data.frame':    295 obs. of  13 variables:
##  $ age                     : num  75 65 50 65 90 75 60 65 80 75 ...
##  $ anaemia                 : int  0 0 1 1 1 1 1 0 1 1 ...
##  $ creatinine_phosphokinase: int  582 146 111 160 47 246 315 157 123 81 ...
##  $ diabetes                : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ ejection_fraction       : int  20 20 20 20 40 15 60 65 35 38 ...
##  $ high_blood_pressure     : int  1 0 0 0 1 0 0 0 1 1 ...
##  $ platelets               : num  265000 162000 210000 327000 204000 ...
##  $ serum_creatinine        : num  1.9 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 4 ...
##  $ serum_sodium            : int  130 129 137 116 132 137 131 138 133 131 ...
##  $ sex                     : int  1 1 1 0 1 1 1 0 1 1 ...
##  $ smoking                 : int  0 1 0 0 1 0 1 0 1 1 ...
##  $ time                    : int  4 7 7 8 8 10 10 10 10 10 ...
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

We remove the 4 outliers before proceeding to modeling exercise.

Converting categorical features and dependent variable to factor

```
data$DEATH_EVENT <- factor(data$DEATH_EVENT)
data$anaemia <- factor(data$anaemia)
data$diabetes <- factor(data$diabetes)
data$high_blood_pressure <- factor(data$high_blood_pressure)
data$sex <- factor(data$sex)
data$smoking <- factor(data$smoking)
str(data)
```

```
## 'data.frame':    295 obs. of  13 variables:
##  $ age                     : num  75 65 50 65 90 75 60 65 80 75 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 2 2 ...
##  $ creatinine_phosphokinase: int  582 146 111 160 47 246 315 157 123 81 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 1 1 1 ...
##  $ ejection_fraction       : int  20 20 20 20 40 15 60 65 35 38 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 2 2 ...
##  $ platelets               : num  265000 162000 210000 327000 204000 ...
##  $ serum_creatinine        : num  1.9 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 4 ...
##  $ serum_sodium            : int  130 129 137 116 132 137 131 138 133 131 ...
##  $ sex                     : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 1 2 2 ...
##  $ smoking                 : Factor w/ 2 levels "0","1": 1 2 1 1 2 1 2 1 2 2 ...
##  $ time                    : int  4 7 7 8 8 10 10 10 10 10 ...
##  $ DEATH_EVENT             : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Two-way contingency table of categorical outcome and predictors

Since we want to make sure there are not 0 cells

```
xtabs(~DEATH_EVENT + anaemia, data = data)
```

```
##            anaemia
## DEATH_EVENT   0   1
##           0 119  83
##           1  48  45
```

```
xtabs(~DEATH_EVENT + diabetes, data = data)
```

```
##            diabetes
## DEATH_EVENT   0   1
##           0 117  85
##           1  54  39
```

```
xtabs(~DEATH_EVENT + high_blood_pressure, data = data)
```

```
##            high_blood_pressure
## DEATH_EVENT   0   1
##           0 136  66
##           1  56  37
```

```
xtabs(~DEATH_EVENT + sex, data = data)
```

```
##            sex
## DEATH_EVENT   0   1
##           0  70 132
##           1  33  60
```

```
xtabs(~DEATH_EVENT + smoking, data = data)
```

```
##            smoking
## DEATH_EVENT   0   1
##           0 136  66
##           1  63  30
```

We note no 0 or low cells in any of the categorical variables.

Checking event rate in data - this will help determine prob. value for thresholds

```
table(data$DEATH_EVENT)
```

```
##
##   0   1
## 202  93
```

We note a 31.5% event rate in the data.

Fitting a model

Iteration - 1 All variables

```
# Model 1
mylogit <- glm(DEATH_EVENT ~ age+anaemia+creatinine_phosphokinase+
    diabetes+ejection_fraction+high_blood_pressure+platelets+
    serum_creatinine+serum_sodium+sex+smoking+time , data = data, family = "binomial")
summary(mylogit)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + anaemia + creatinine_phosphokinase +
##     diabetes + ejection_fraction + high_blood_pressure + platelets +
##     serum_creatinine + serum_sodium + sex + smoking + time, family = "binomial",
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1947  -0.5926  -0.2341   0.4384   2.6516
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               1.127e+01  5.669e+00   1.988  0.04682 *
## age                       5.197e-02  1.623e-02   3.202  0.00137 **
## anaemia1                 -8.307e-02  3.635e-01  -0.229  0.81923
## creatinine_phosphokinase  1.101e-04  2.207e-04   0.499  0.61779
## diabetes1                 1.623e-01  3.528e-01   0.460  0.64551
## ejection_fraction        -8.244e-02  1.723e-02  -4.784 1.72e-06 ***
## high_blood_pressure1     -2.472e-01  3.740e-01  -0.661  0.50859
## platelets                -1.069e-06  1.908e-06  -0.560  0.57538
## serum_creatinine          4.669e-01  2.049e-01   2.279  0.02265 *
## serum_sodium             -7.236e-02  3.970e-02  -1.823  0.06831 .
## sex1                     -5.384e-01  4.136e-01  -1.302  0.19294
## smoking1                 -4.051e-02  4.196e-01  -0.097  0.92308
## time                     -2.160e-02  3.114e-03  -6.936 4.04e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 367.71  on 294  degrees of freedom
## Residual deviance: 215.95  on 282  degrees of freedom
## AIC: 241.95
##
## Number of Fisher Scoring iterations: 6
```

Key observations

1. We note age, ejection_fraction, serum_creatinine and time as significant variables in this iteration
2. The above iteration has an AIC of 241.95

3. Interpreting the coefficient - For every one unit change in age, the log odds of death (versus survival) increases by 5.197e-02
4. None of the categorical variables are significant in predicting the death event

Predicting the outcome

```
glm.probs <- predict(mylogit,type = "response")
glm.probs[1:5]
```

```
##         1         3         4         5         6
## 0.9802885 0.9671714 0.9008819 0.9956762 0.9417182
```

The first five probabilities in this case are all close to 1 as evidenced in the data as well.

Let's use a base calculation to figure out accuracy

```
# Here we try the case of using default 0.5 as threshold
glm.pred <- ifelse(glm.probs > 0.5, "1", "0")
table(data$DEATH_EVENT,glm.pred)
```

```
##    glm.pred
##       0   1
##   0 185  17
##   1  27  66
```

Looking at the diagonal, we're not that bad. We have an overall accuracy of ~85% (Diagonals summed over overall sum) But note this is called a base model because we didn't do splitting into train and test so the model trained on the entire data and predicted on the entire data. Without out of sample testing one cannot claim robustness as there may be overfitting here.

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(data$DEATH_EVENT, glm.probs)[1]
optCutOff
```

```
## [1] 0.4669198
```

We used 0.5 above to classify however we want the cut-off where model is balanced in accuracy measures. We note this as 0.466. This tells us that we can use this prob cutoff to classify an observation as 0 or 1. If the prob-value is below this threshold we can classify it as survival else death event.

Mis-classification error

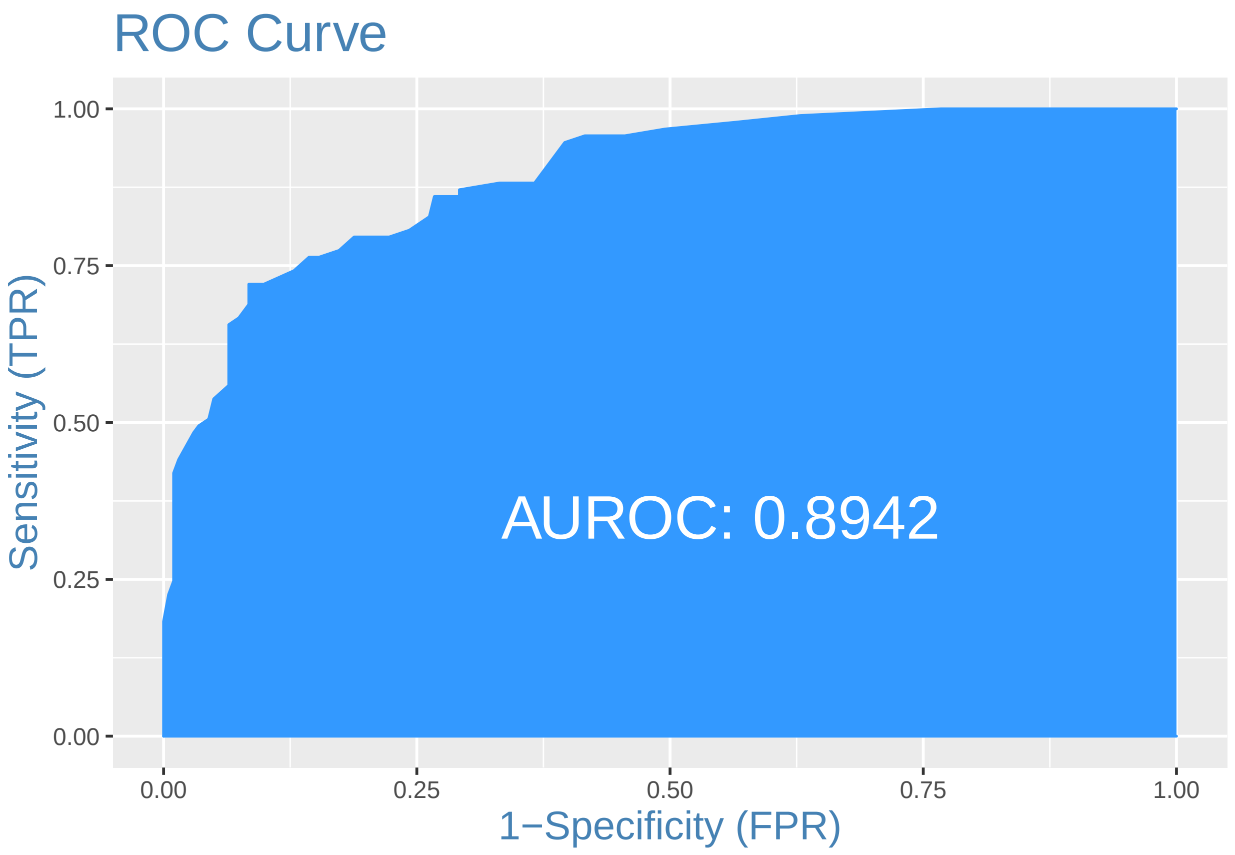
```
misClassError(data$DEATH_EVENT, glm.probs, threshold = optCutOff)
```

```
## [1] 0.1458
```

We note a misclassification error of 14.58%

ROC curve

```r
plotROC(data$DEATH_EVENT, glm.probs)
```



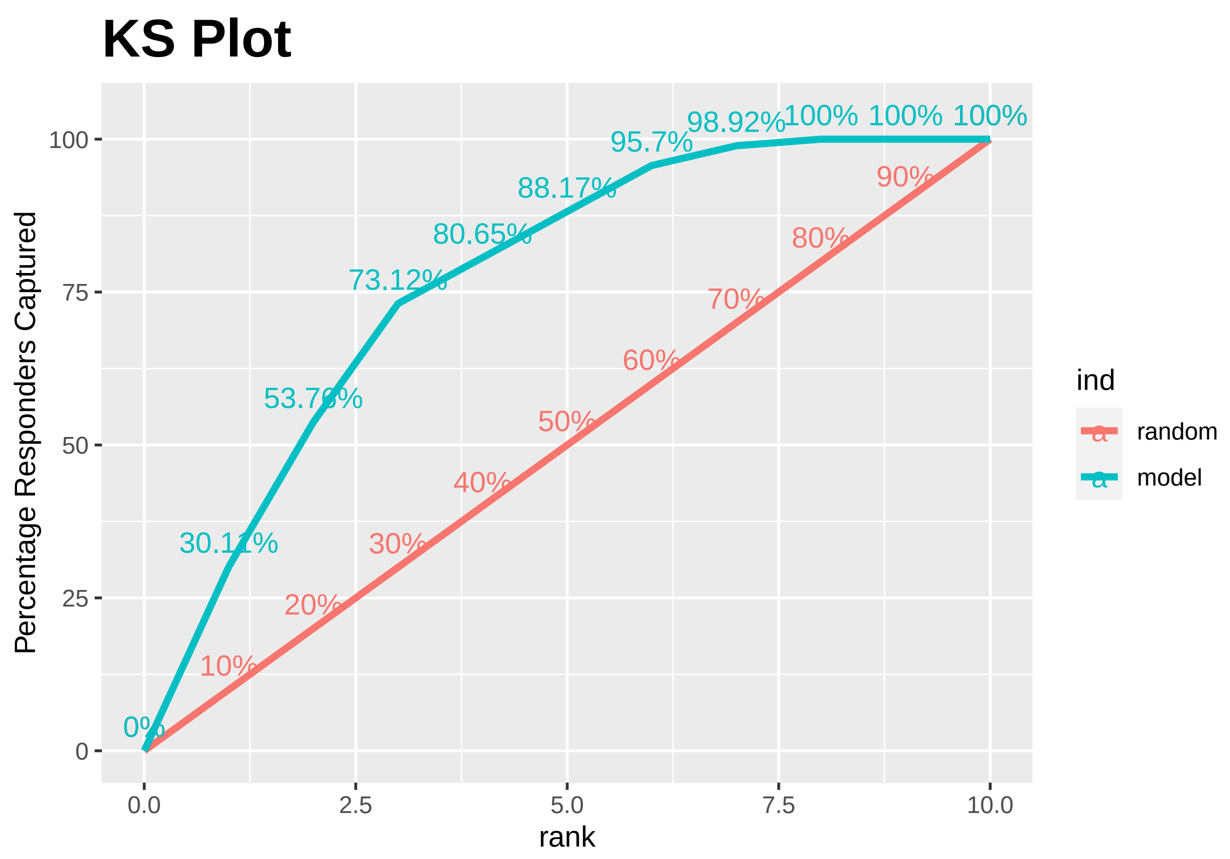
We see an AUC of 0.894 which is decent.

KS plot

```r
ks_stat(data$DEATH_EVENT, glm.probs)
```

```
## [1] 0.6223
```

```r
ks_plot(data$DEATH_EVENT, glm.probs)
```



A KS plot answers the question how many responders/ deaths can we capture if we target x% of the population. Here, we see we can capture 73% responders if we target first 30% of the population.

Confusion Matrix and all accuracy measures

```
confusionMatrix(data = as.factor(glm.pred),
        reference = as.factor(data$DEATH_EVENT), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 185  27
##          1  17  66
##
##                Accuracy : 0.8508
##                  95% CI : (0.805, 0.8895)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 4.394e-11
##
##                   Kappa : 0.6442
##
##  Mcnemar's Test P-Value : 0.1748
##
##               Precision : 0.8726
##                  Recall : 0.9158
##                       F1 : 0.8937
##               Prevalence : 0.6847
##           Detection Rate : 0.6271
##     Detection Prevalence : 0.7186
##        Balanced Accuracy : 0.8128
##
##         'Positive' Class : 0
##
```

We have to be careful here. From above, we can see accuracy of the model is 85.0% as before. Precision is 0.87 and Recall is 0.91 while F1- score is 0.89 but this is for positive class taken as '0'. However, we want to understand precision, recall for positive class as predicting death events is more important than survival events.

Confusion Matrix and all accuracy measures for positive class chosen as death=1

```
confusionMatrix(data = as.factor(glm.pred),
    reference = as.factor(data$DEATH_EVENT),positive='1', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 185  27
##          1  17  66
##
##                Accuracy : 0.8508
##                  95% CI : (0.805, 0.8895)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 4.394e-11
##
##                   Kappa : 0.6442
##
##  Mcnemar's Test P-Value : 0.1748
##
##               Precision : 0.7952
##                  Recall : 0.7097
##                      F1 : 0.7500
##              Prevalence : 0.3153
##          Detection Rate : 0.2237
##    Detection Prevalence : 0.2814
##       Balanced Accuracy : 0.8128
##
##        'Positive' Class : 1
##
```

We give positive class as '1' as we want to understand precision and recall of death events more than survival events so we know what to maximize for. Our accuracy of the model is 85.0% as before however precision is 0.79 and recall is 0.70 while F1- score is 0.75. This is a more true picture of our model than before and we know that the overall accuracy is slightly dominant towards predicting survival events better than death events.

Using optimal cutoff to determine accuracy measures

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit,type="response")>threshold,1,0)
actual_values<-data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 185  26
##          1  17  67
##
##                Accuracy : 0.8542
##                  95% CI : (0.8087, 0.8925)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 1.641e-11
##
##                   Kappa : 0.6533
##
##  Mcnemar's Test P-Value : 0.2225
##
##               Precision : 0.8768
##                  Recall : 0.9158
##                      F1 : 0.8959
##              Prevalence : 0.6847
##          Detection Rate : 0.6271
##    Detection Prevalence : 0.7153
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : 0
##
```

Our new accuracy has gone up from 85% to 85.4% all by optimizing the prob. cutoff threshold.

Using optimal cutoff to determine accuracy measures with positive class as 1

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit,type="response")>threshold,1,0)
actual_values<-data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 185  26
##          1  17  67
##
```

```
##                Accuracy : 0.8542
##                  95% CI : (0.8087, 0.8925)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 1.641e-11
##
##                   Kappa : 0.6533
##
##  Mcnemar's Test P-Value : 0.2225
##
##               Precision : 0.7976
##                  Recall : 0.7204
##                      F1 : 0.7571
##              Prevalence : 0.3153
##          Detection Rate : 0.2271
##    Detection Prevalence : 0.2847
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : 1
##
```

Here, we see that we have improved our recall while keeping precision same and hence consequently our F1 score.

However we may be over-fitting here as we haven't kept a hold out set. This is something we will explore in future iterations.

Iteration - 2 Using only significant variables from Iteration 1

```
# Model 2
mylogit_2 <- glm(DEATH_EVENT ~ age
    +ejection_fraction+
    serum_creatinine+time , data = data, family = "binomial")
summary(mylogit_2)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine +
##     time, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1419  -0.6114  -0.2489   0.4607   2.8716
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.61105    1.06948   0.571  0.56776
## age                0.04813    0.01533   3.140  0.00169 **
## ejection_fraction -0.07854    0.01628  -4.824 1.41e-06 ***
## serum_creatinine   0.56967    0.19991   2.850  0.00438 **
## time              -0.02058    0.00292  -7.047 1.83e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 367.71  on 294  degrees of freedom
## Residual deviance: 222.40  on 290  degrees of freedom
## AIC: 232.4
##
## Number of Fisher Scoring iterations: 5
```

We note the lower AIC value in this iteration of 232.4

Predicting the outcome

```
glm.probs_2 <- predict(mylogit_2,type = "response")
glm.probs_2[1:5]
```

```
##         1         3         4         5         6
## 0.9746764 0.9407823 0.9156974 0.9718717 0.9444430
```

The first five probabilities in this case are all close to 1 as evidenced in the data as well.

Let's use a base calculation to figure out accuracy

```
# Here we try the case of using default 0.5 as threshold
glm.pred_2 <- ifelse(glm.probs_2 < 0.5, "0", "1")
table(data$DEATH_EVENT,glm.pred_2)
```

```
##    glm.pred_2
##       0   1
##   0 182  20
##   1  30  63
```

Looking at the diagonal, we have an overall accuracy of ~83% (Diagonals summed over overall sum). This is lower than before which makes sense since we removed the unnecessary independent variables But how do we know this is more robust than the model before ?

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(data$DEATH_EVENT, glm.probs_2)[1]
optCutOff
```

```
## [1] 0.5989542
```

Mis-classification error

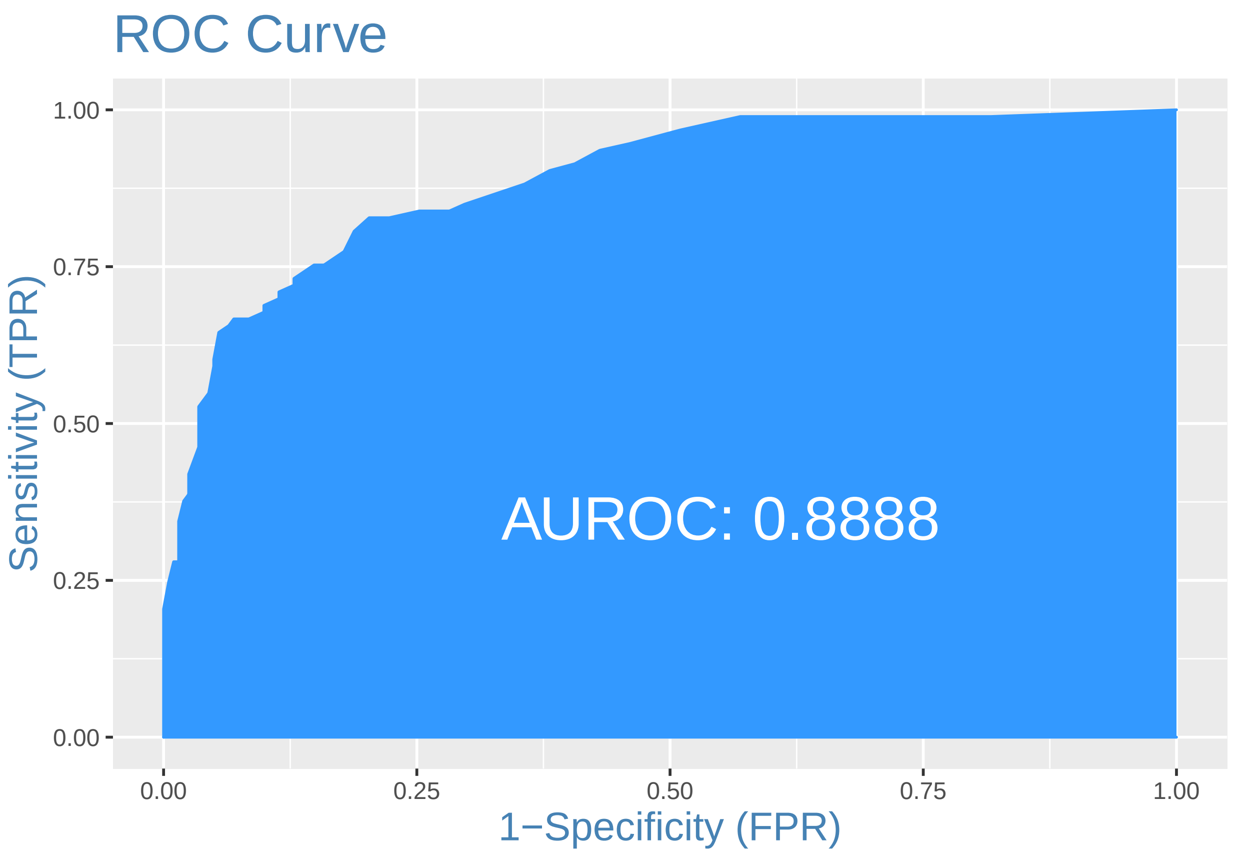
```
misClassError(data$DEATH_EVENT, glm.probs_2, threshold = optCutOff)
```

```
## [1] 0.1492
```

We note a misclassification error of 14.92%

ROC curve

```
plotROC(data$DEATH_EVENT, glm.probs_2)
```



We see an AUC of 0.888 which is decent.

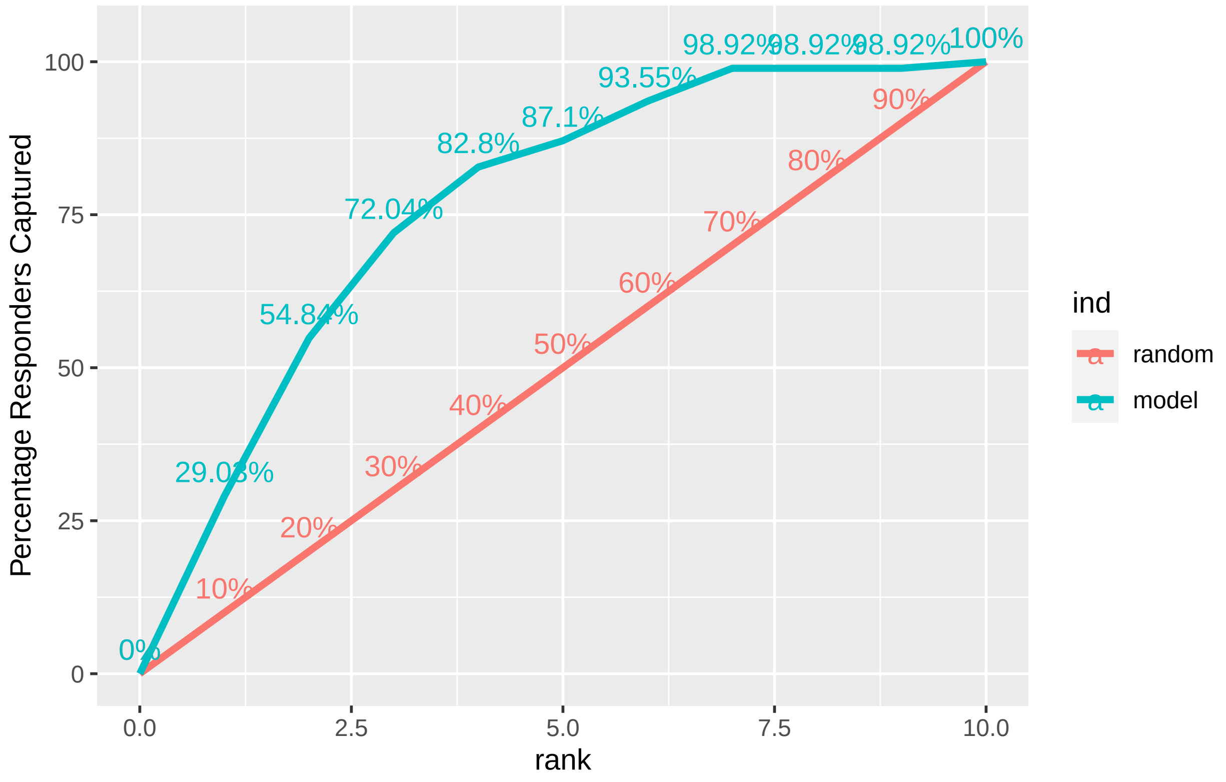
KS plot

```
ks_stat(data$DEATH_EVENT, glm.probs_2)
```

```
## [1] 0.6151
```

```
ks_plot(data$DEATH_EVENT, glm.probs_2)
```

KS Plot



Confusion Matrix and all accuracy measures

```
confusionMatrix(data = as.factor(glm.pred_2),
                reference = as.factor(data$DEATH_EVENT), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 182  30
##          1  20  63
##
##                Accuracy : 0.8305
##                  95% CI : (0.7827, 0.8715)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 9.527e-09
##
##                   Kappa : 0.5957
##
##  Mcnemar's Test P-Value : 0.2031
##
##               Precision : 0.8585
##                  Recall : 0.9010
##                      F1 : 0.8792
##              Prevalence : 0.6847
##          Detection Rate : 0.6169
##    Detection Prevalence : 0.7186
##       Balanced Accuracy : 0.7892
##
##        'Positive' Class : 0
##
```

Our new accuracy is 83.0%. This is slightly lower than before which makes sense since we have eliminated some independent variables in this iteration.

Confusion Matrix and all accuracy measures for positive class chosen as death=1

```
confusionMatrix(data = as.factor(glm.pred_2),
    reference = as.factor(data$DEATH_EVENT),positive='1', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 182  30
##          1  20  63
##
##                Accuracy : 0.8305
##                  95% CI : (0.7827, 0.8715)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 9.527e-09
##
##                   Kappa : 0.5957
##
##  Mcnemar's Test P-Value : 0.2031
##
##               Precision : 0.7590
##                  Recall : 0.6774
##                      F1 : 0.7159
##              Prevalence : 0.3153
##          Detection Rate : 0.2136
##    Detection Prevalence : 0.2814
##       Balanced Accuracy : 0.7892
##
##        'Positive' Class : 1
##
```

Our accuracy of the model is 83.0% as before however precision is 0.75 and recall is 0.67 while F1- score is 0.71. This is a more true picture of our model than before and we know that the overall accuracy is slightly dominant towards predicting survival events better than death events. We can see that all precision, recall, F1 score, accuracy and AUC are slightly lower in this iteration.

Using optimal cutoff to determine accuracy measures

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_2,type="response")>threshold,1,0)
actual_values<-data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
            reference = as.factor(actual_values), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 191  33
##          1  11  60
##
##                Accuracy : 0.8508
##                  95% CI : (0.805, 0.8895)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 4.394e-11
##
##                   Kappa : 0.631
##
##  Mcnemar's Test P-Value : 0.001546
##
##               Precision : 0.8527
##                  Recall : 0.9455
##                      F1 : 0.8967
##              Prevalence : 0.6847
##          Detection Rate : 0.6475
##    Detection Prevalence : 0.7593
##       Balanced Accuracy : 0.7954
##
##        'Positive' Class : 0
##
```

On using the optimal cutoff however, Our new accuracy has gone up from 83% to 85.0% all by optimizing the prob. cutoff threshold.

Using optimal cutoff to determine accuracy measures with positive class as 1

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_2,type="response")>threshold,1,0)
actual_values<-data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 191  33
##          1  11  60
##
##                Accuracy : 0.8508
##                  95% CI : (0.805, 0.8895)
##     No Information Rate : 0.6847
##     P-Value [Acc > NIR] : 4.394e-11
##
##                   Kappa : 0.631
##
##  Mcnemar's Test P-Value : 0.001546
```

```
##
##               Precision : 0.8451
##                  Recall : 0.6452
##                      F1 : 0.7317
##              Prevalence : 0.3153
##          Detection Rate : 0.2034
##    Detection Prevalence : 0.2407
##       Balanced Accuracy : 0.7954
##
##        'Positive' Class : 1
##
```

Here, we see that we have improved our pecision however recall is much worse.

The problem in the first two iterations however is that we haven't kept a test set and may have over-fitted the model unknowingly.

Iteration - 3 All variables but splitting into train and test

Splitting into train and test - 70%, 30% split

```
set.seed(123)
trainIndex <- createDataPartition(data$DEATH_EVENT, p = .7,
                                  list = FALSE,
                                  times = 1)
train_data<-data[trainIndex,]
test_data<-data[-trainIndex,]
table(train_data$DEATH_EVENT)
```

```
##
##   0   1
## 142  66
```

```
table(test_data$DEATH_EVENT)
```

```
##
##  0  1
## 60 27
```

We see our train data has event rate of 31.7% and our test data has event rate of 31.0%. This can happen in reality as well and hence a good accuracy on test will ensure we have built a robust model.

We will now train the model on training set and test on test set.

```
# Model 3
mylogit_3 <- glm(DEATH_EVENT ~ age+anaemia+creatinine_phosphokinase+
    diabetes+ejection_fraction+high_blood_pressure+platelets+
    serum_creatinine+serum_sodium+sex+smoking+time ,
    data = train_data, family = "binomial")
summary(mylogit_3)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + anaemia + creatinine_phosphokinase +
##     diabetes + ejection_fraction + high_blood_pressure + platelets +
##     serum_creatinine + serum_sodium + sex + smoking + time, family = "binomial",
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0286  -0.5529  -0.2011   0.4010   2.6545
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              1.240e+01  6.840e+00   1.813  0.06989 .
## age                      5.037e-02  1.910e-02   2.637  0.00835 **
## anaemia1                 3.813e-01  4.407e-01   0.865  0.38686
## creatinine_phosphokinase  6.435e-05  2.707e-04   0.238  0.81212
## diabetes1                2.028e-01  4.338e-01   0.467  0.64017
## ejection_fraction       -1.024e-01  2.208e-02  -4.636 3.55e-06 ***
## high_blood_pressure1    -4.740e-01  4.734e-01  -1.001  0.31670
## platelets               -2.502e-07  2.293e-06  -0.109  0.91313
## serum_creatinine         3.950e-01  2.313e-01   1.708  0.08767 .
## serum_sodium             -7.384e-02  4.764e-02  -1.550  0.12117
## sex1                     -8.848e-01  5.098e-01  -1.736  0.08262 .
## smoking1                 -3.864e-01  5.298e-01  -0.729  0.46584
## time                     -2.112e-02  3.921e-03  -5.387 7.15e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 259.93  on 207  degrees of freedom
## Residual deviance: 147.47  on 195  degrees of freedom
## AIC: 173.47
##
## Number of Fisher Scoring iterations: 6
```

```
predicted <- predict(mylogit_3, test_data, type="response")
```

We note a key difference here - we do not see the serum_creatinine as a significant variable in this iteration. We see only age, time and ejection_fraction as significant variables.

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(test_data$DEATH_EVENT, predicted)[1]
optCutOff
```

```
## [1] 0.6984528
```

This tells us that we can use this prob cutoff to classify an observation as 0 or 1. If the prob-value is below this threshold we can classify it as survival else death event.

Mis-classification error

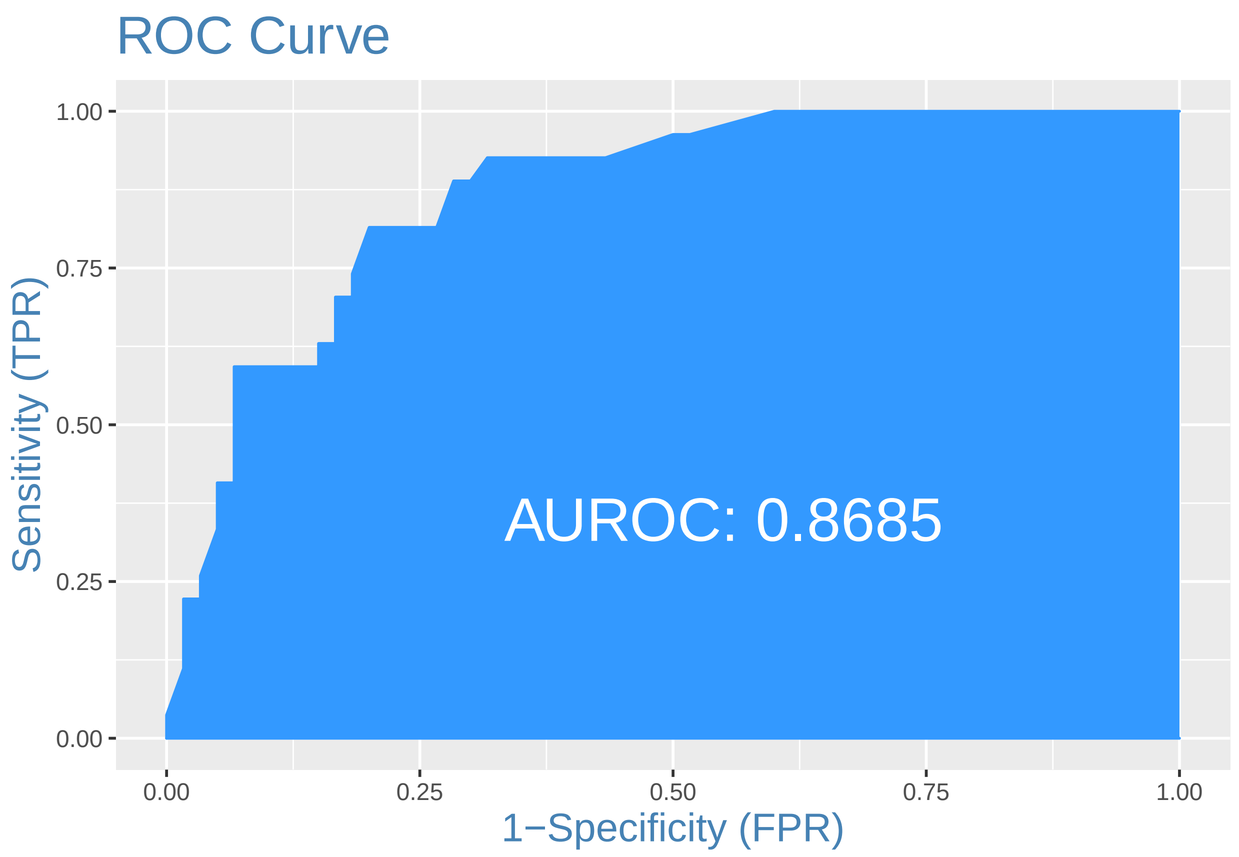
```
misClassError(test_data$DEATH_EVENT, predicted, threshold = optCutOff)
```

```
## [1] 0.1724
```

We note a misclassification error on test set of 17.24%

ROC curve

```
plotROC(test_data$DEATH_EVENT, predicted)
```



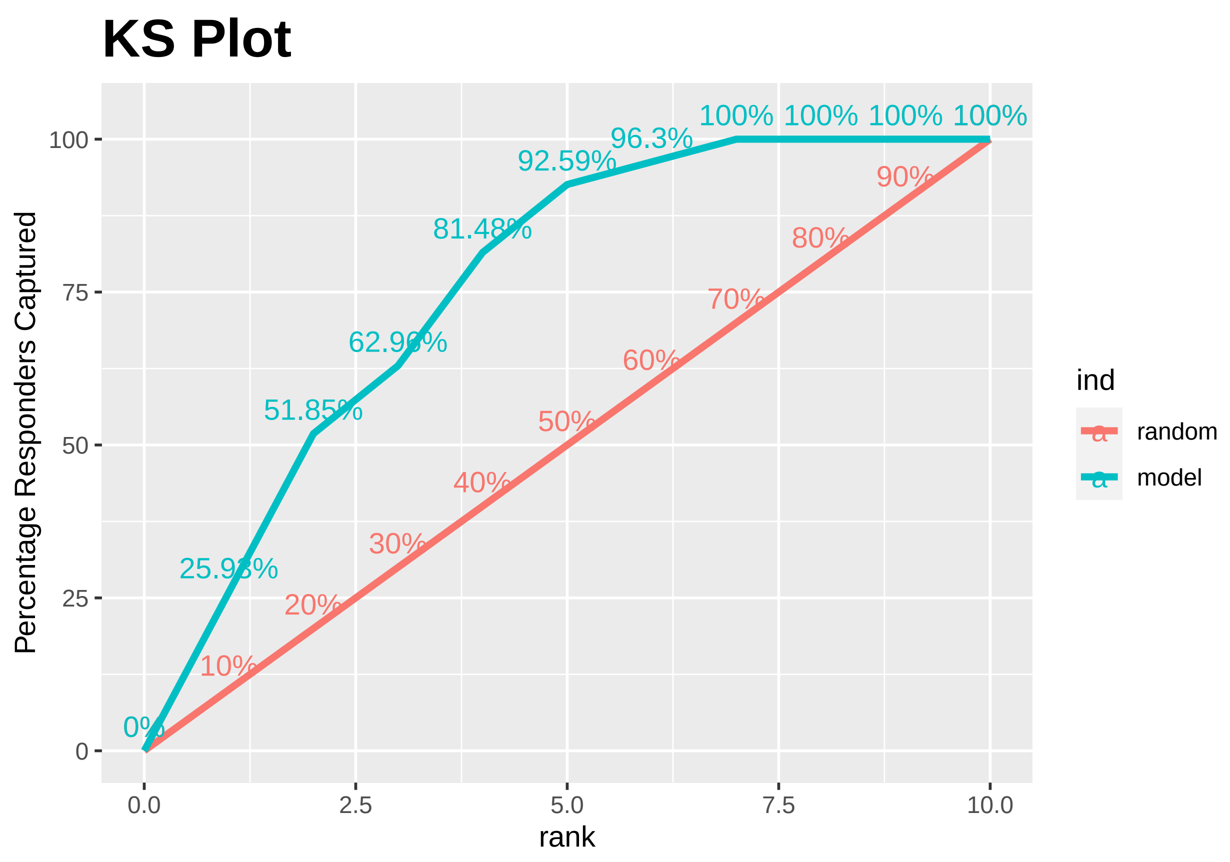
We see an AUC of 0.868 which is decent.

KS plot

```
ks_stat(test_data$DEATH_EVENT, predicted)
```

```
## [1] 0.5926
```

```
ks_plot(test_data$DEATH_EVENT, predicted)
```



Concordance check

```r
Concordance(test_data$DEATH_EVENT, predicted)
```

```
## $Concordance
## [1] 0.8685185
##
## $Discordance
## [1] 0.1314815
##
## $Tied
## [1] 5.551115e-17
##
## $Pairs
## [1] 1620
```

Usually concordance is in-line with AUC and we see that 86.8% pairs are concordant (the model calculated prob scores of 1s being greater than model calculated prob scores of 0s)

Using optimal cutoff to determine accuracy measures

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_3,test_data,type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
```

```r
confusionMatrix(data = as.factor(predicted_values),
            reference = as.factor(actual_values), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 11
##          1  4 16
##
##                Accuracy : 0.8276
##                  95% CI : (0.7316, 0.9002)
##     No Information Rate : 0.6897
##     P-Value [Acc > NIR] : 0.002661
##
##                   Kappa : 0.5663
##
##  Mcnemar's Test P-Value : 0.121335
##
##               Precision : 0.8358
##                  Recall : 0.9333
##                      F1 : 0.8819
##              Prevalence : 0.6897
##          Detection Rate : 0.6437
##    Detection Prevalence : 0.7701
##       Balanced Accuracy : 0.7630
##
##        'Positive' Class : 0
##
```

We see that on test set our accuracy is 82.7%

Using optimal cutoff to determine accuracy measures with positive class as 1

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_3, test_data,
                                 type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 11
##          1  4 16
##
##                Accuracy : 0.8276
##                  95% CI : (0.7316, 0.9002)
##     No Information Rate : 0.6897
##     P-Value [Acc > NIR] : 0.002661
```

```
##
##                  Kappa : 0.5663
##
##  Mcnemar's Test P-Value : 0.121335
##
##              Precision : 0.8000
##                 Recall : 0.5926
##                     F1 : 0.6809
##             Prevalence : 0.3103
##         Detection Rate : 0.1839
##   Detection Prevalence : 0.2299
##      Balanced Accuracy : 0.7630
##
##       'Positive' Class : 1
##
```

We however note a key difference in this accuracy. Our recall has fallen to 0.59 while the precision is 0.80 with F1-score of 0.68

Iteration - 4 Stepwise regression

We now perform a stepwise regression which computes a null model and a full model first and adds variables as long as the added variable's AIC is below the previous computation of AIC.

```
null_model<-glm(DEATH_EVENT~1,data=train_data,family='binomial')
full_model<-glm(DEATH_EVENT~.,data=train_data,family='binomial')
step_model <- step(null_model,
                   scope = list(lower = null_model,
                                upper = full_model),
                   direction = "forward")
```

```
## Start:  AIC=261.93
## DEATH_EVENT ~ 1
##
##                            Df Deviance    AIC
## + time                      1   195.18 199.18
## + ejection_fraction         1   235.92 239.92
## + serum_creatinine          1   237.52 241.52
## + age                       1   244.83 248.83
## + serum_sodium              1   252.47 256.47
## + anaemia                   1   257.23 261.23
## <none>                          259.93 261.93
## + high_blood_pressure       1   258.97 262.97
## + creatinine_phosphokinase  1   259.29 263.29
## + smoking                   1   259.37 263.37
## + sex                       1   259.49 263.49
## + platelets                 1   259.74 263.74
## + diabetes                  1   259.76 263.76
##
## Step:  AIC=199.18
## DEATH_EVENT ~ time
##
##                            Df Deviance    AIC
## + ejection_fraction         1   172.94 178.94
## + serum_creatinine          1   184.74 190.74
## + serum_sodium              1   188.05 194.05
## + age                       1   191.07 197.07
## + smoking                   1   192.53 198.53
## <none>                          195.18 199.18
## + sex                       1   193.86 199.86
## + high_blood_pressure       1   194.86 200.86
## + diabetes                  1   194.91 200.91
## + anaemia                   1   195.00 201.00
## + platelets                 1   195.00 201.00
## + creatinine_phosphokinase  1   195.10 201.10
##
## Step:  AIC=178.94
## DEATH_EVENT ~ time + ejection_fraction
##
##                            Df Deviance    AIC
## + serum_creatinine          1   163.51 171.51
## + age                       1   164.56 172.56
## + serum_sodium              1   168.52 176.52
## + sex                      1   168.68 176.68
## + smoking                  1   168.76 176.76
## <none>                         172.94 178.94
## + anaemia                  1   171.98 179.98
## + diabetes                 1   172.38 180.38
## + high_blood_pressure      1   172.50 180.50
## + creatinine_phosphokinase  1   172.85 180.85
## + platelets                1   172.93 180.93
##
## Step:  AIC=171.51
## DEATH_EVENT ~ time + ejection_fraction + serum_creatinine
##
##                           Df Deviance    AIC
## + age                      1   157.00 167.00
## + sex                      1   160.28 170.28
## + smoking                  1   161.07 171.07
## + serum_sodium             1   161.22 171.22
## <none>                         163.51 171.51
## + anaemia                  1   162.06 172.06
## + creatinine_phosphokinase  1   163.03 173.03
## + diabetes                 1   163.05 173.05
## + high_blood_pressure      1   163.41 173.41
## + platelets                1   163.50 173.50
##
## Step:  AIC=166.99
## DEATH_EVENT ~ time + ejection_fraction + serum_creatinine + age
##
##                           Df Deviance    AIC
## + sex                      1   152.78 164.78
## + smoking                  1   154.68 166.68
## + serum_sodium             1   154.78 166.78
## <none>                         157.00 167.00
## + anaemia                  1   155.71 167.71
## + diabetes                 1   156.09 168.09
## + creatinine_phosphokinase  1   156.79 168.79
## + high_blood_pressure      1   156.81 168.81
## + platelets                1   156.93 168.93
##
## Step:  AIC=164.78
## DEATH_EVENT ~ time + ejection_fraction + serum_creatinine + age +
##     sex
##
##                           Df Deviance    AIC
## + serum_sodium             1   150.14 164.14
## <none>                         152.78 164.78
## + high_blood_pressure      1   151.81 165.81
## + anaemia                  1   152.07 166.07
## + smoking                  1   152.08 166.08
## + diabetes                 1   152.20 166.20
## + platelets                1   152.61 166.61
## + creatinine_phosphokinase  1   152.75 166.75
##
## Step:  AIC=164.14
## DEATH_EVENT ~ time + ejection_fraction + serum_creatinine + age +
##     sex + serum_sodium
##
##                           Df Deviance    AIC
## <none>                         150.14 164.14
## + high_blood_pressure       1   149.20 165.20
## + anaemia                   1   149.34 165.34
## + smoking                   1   149.53 165.53
## + diabetes                  1   149.91 165.91
## + platelets                 1   150.05 166.05
## + creatinine_phosphokinase  1   150.13 166.13
```

We see the results of the stepwise regression lowering our AIC to 164.14 with variables like time, ejection_fraction, serum_creatinine, age, sex and serum_sodium

```r
summary(step_model)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ time + ejection_fraction + serum_creatinine +
##     age + sex + serum_sodium, family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9074  -0.5536  -0.2155   0.4005   2.6098
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       12.185514   6.446110   1.890  0.05871 .
## time              -0.020332   0.003657  -5.560 2.70e-08 ***
## ejection_fraction -0.096216   0.021094  -4.561 5.08e-06 ***
## serum_creatinine   0.451644   0.215623   2.095  0.03621 *
## age                0.048070   0.018369   2.617  0.00888 **
## sex1              -0.958956   0.453200  -2.116  0.03435 *
## serum_sodium      -0.074058   0.045182  -1.639  0.10120
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 259.93  on 207  degrees of freedom
## Residual deviance: 150.14  on 201  degrees of freedom
## AIC: 164.14
##
## Number of Fisher Scoring iterations: 6
```

We see a lower AIC but unfortunately serum_sodium isn't significant. We can remove this variable and re-compute the ideal model.

```r
# Model 4
mylogit_4 <- glm(DEATH_EVENT ~ time + ejection_fraction + serum_creatinine +
    age + sex , data = train_data, family = "binomial")
summary(mylogit_4)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ time + ejection_fraction + serum_creatinine +
##     age + sex, family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9095  -0.5704  -0.2370   0.4689   2.6793
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.939576   1.357212   1.429  0.15298
## time              -0.019475   0.003506  -5.554 2.79e-08 ***
## ejection_fraction -0.096797   0.020669  -4.683 2.82e-06 ***
## serum_creatinine   0.509987   0.221317   2.304  0.02120 *
## age                0.047654   0.018083   2.635  0.00841 **
## sex1              -0.902833   0.445665  -2.026  0.04278 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 259.93  on 207  degrees of freedom
## Residual deviance: 152.78  on 202  degrees of freedom
## AIC: 164.78
##
## Number of Fisher Scoring iterations: 6
```

We note our lowest AIC yet of 164.78

Let's predict the outcome for this model

```
predicted <- predict(mylogit_4, test_data, type="response")
```

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(test_data$DEATH_EVENT, predicted)[1]
optCutOff
```

```
## [1] 0.5069049
```

This tells us that we can use this prob cutoff to classify an observation as 0 or 1. If the prob-value is below this threshold we can classify it as survival else death event.

Mis-classification error

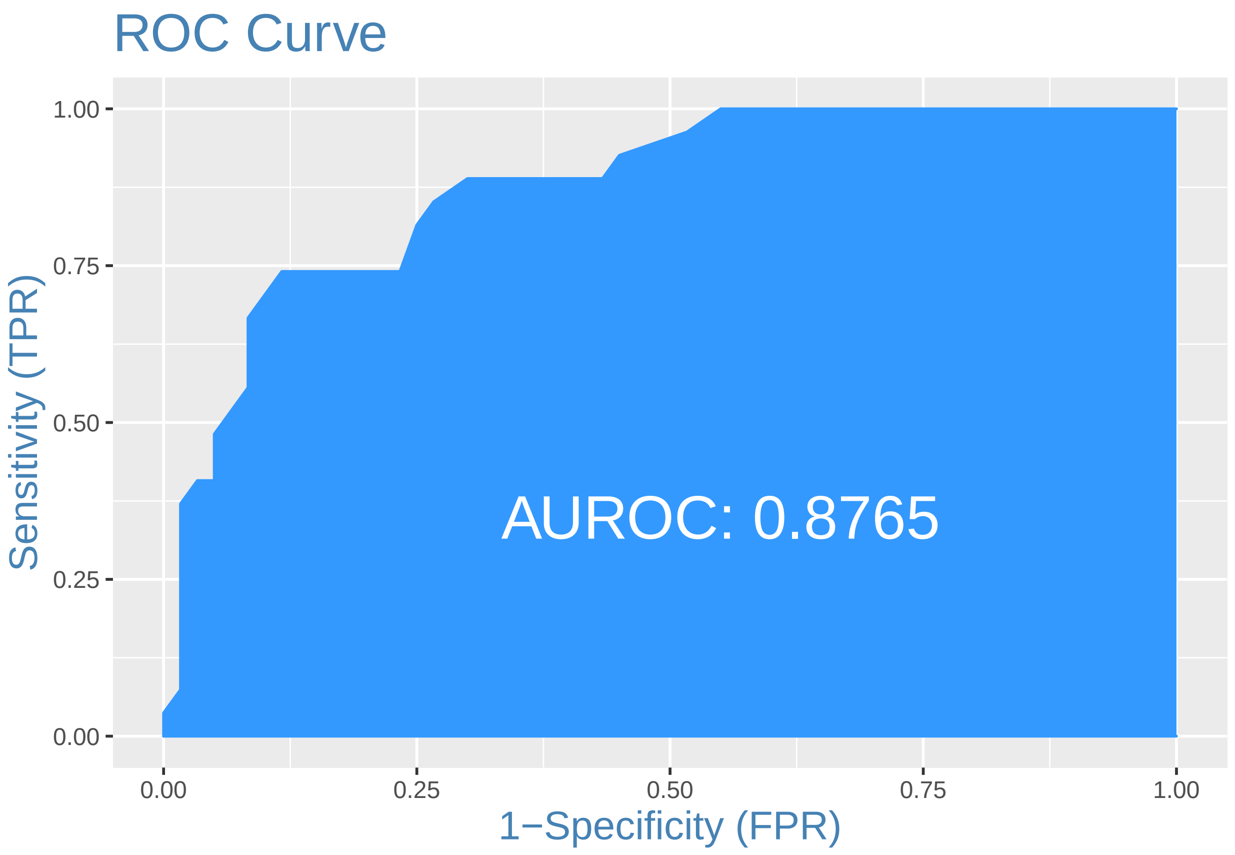
```
misClassError(test_data$DEATH_EVENT, predicted, threshold = optCutOff)
```

```
## [1] 0.1494
```

We note a misclassification error on test set of 14.9%

ROC curve

```
plotROC(test_data$DEATH_EVENT, predicted)
```



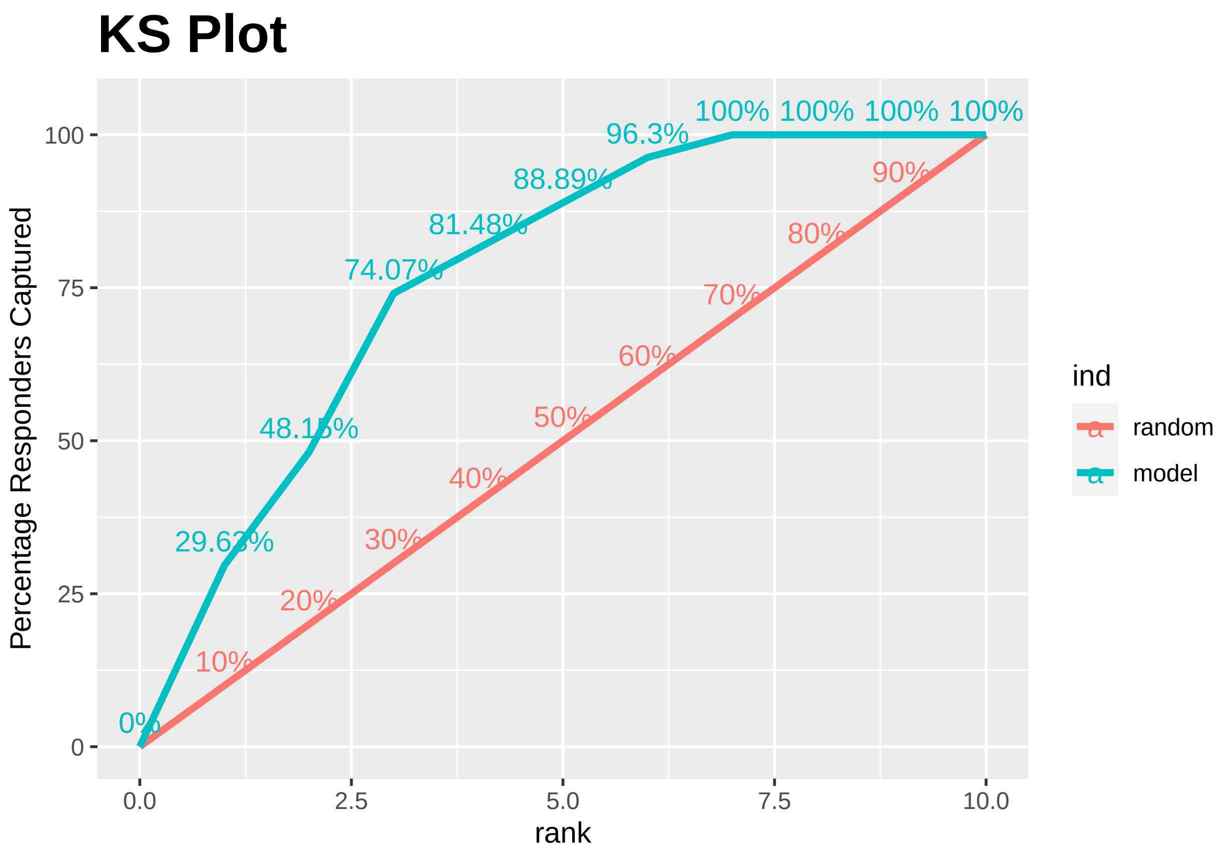
We see an AUC of 0.876 which is our best yet on test data.

KS plot

```
ks_stat(test_data$DEATH_EVENT, predicted)
```

```
## [1] 0.6241
```

```
ks_plot(test_data$DEATH_EVENT, predicted)
```



Concordance check

```
Concordance(test_data$DEATH_EVENT, predicted)
```

```
## $Concordance
## [1] 0.8765432
##
## $Discordance
## [1] 0.1234568
##
## $Tied
## [1] 0
##
## $Pairs
## [1] 1620
```

Usually concordance is in-line with AUC and we see that 87.6% pairs are concordant (the model calculated prob scores of 1s being greater than model calculated prob scores of 0s)

Using optimal cutoff to determine accuracy measures

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_4,test_data,
                                 type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
```

```r
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 55  8
##          1  5 19
##
##                Accuracy : 0.8506
##                  95% CI : (0.758, 0.918)
##     No Information Rate : 0.6897
##     P-Value [Acc > NIR] : 0.0004584
##
##                   Kappa : 0.6399
##
##  Mcnemar's Test P-Value : 0.5790997
##
##               Precision : 0.8730
##                  Recall : 0.9167
##                      F1 : 0.8943
##              Prevalence : 0.6897
##          Detection Rate : 0.6322
##    Detection Prevalence : 0.7241
##       Balanced Accuracy : 0.8102
##
##        'Positive' Class : 0
##
```

We see that on test set our accuracy is 85.0%

Using optimal cutoff to determine accuracy measures with positive class as 1

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_4, test_data,
                            type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 55  8
##          1  5 19
##
##                Accuracy : 0.8506
##                  95% CI : (0.758, 0.918)
##     No Information Rate : 0.6897
##     P-Value [Acc > NIR] : 0.0004584
##
##                   Kappa : 0.6399
##
##  Mcnemar's Test P-Value : 0.5790997
##
##               Precision : 0.7917
##                  Recall : 0.7037
##                      F1 : 0.7451
##              Prevalence : 0.3103
##          Detection Rate : 0.2184
##    Detection Prevalence : 0.2759
##       Balanced Accuracy : 0.8102
##
##        'Positive' Class : 1
##
```

We see improved values of precision to 0.79, recall to 0.70 and F1 score to 0.74

This clearly is our most balanced and best model yet.

Iteration 5 - Computing WOE (weight of evidence) and IV (information value) to improve prediction accuracy

Computing IV

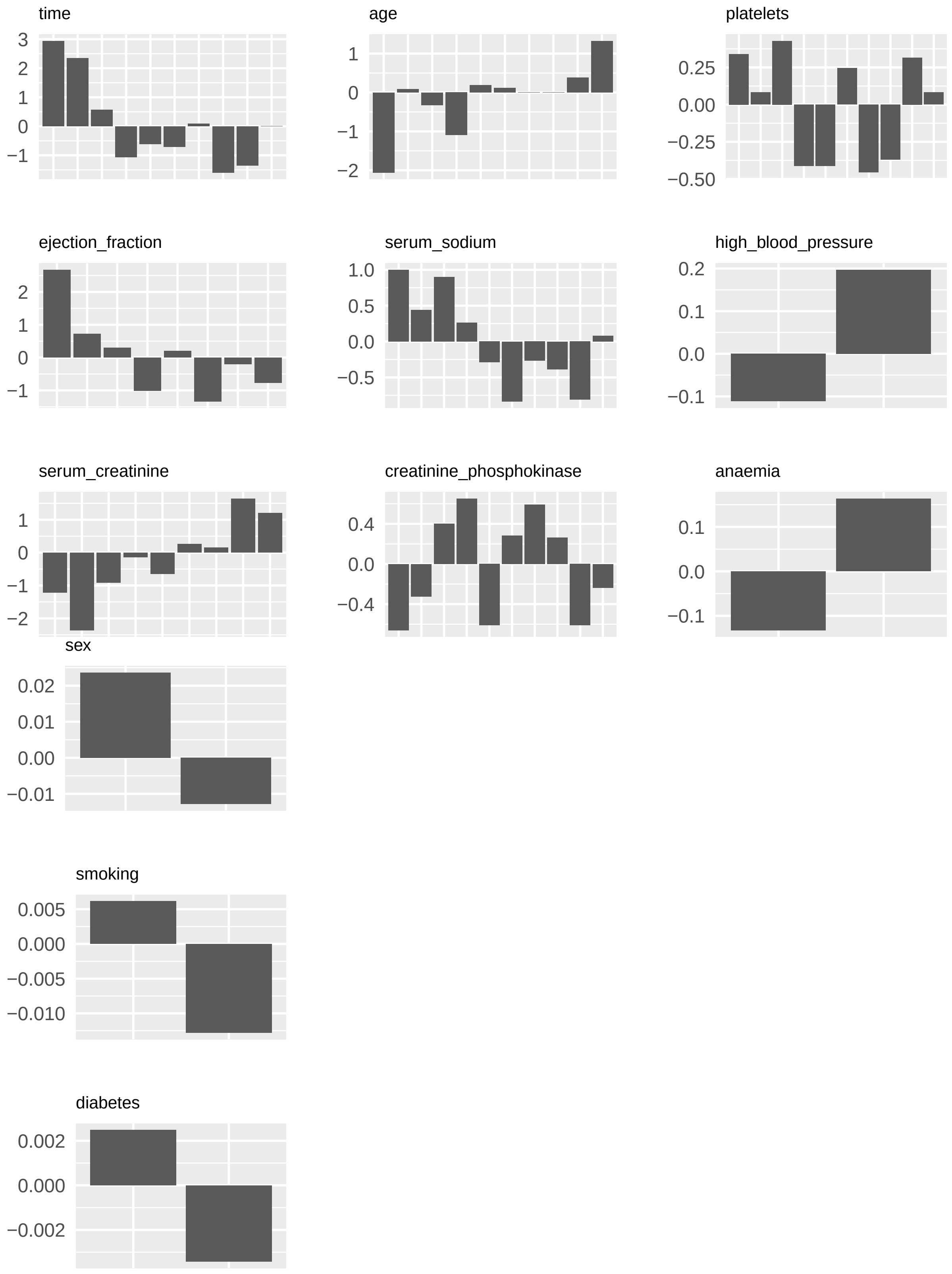
```r
library(Information)
library(gridExtra)
data <- read.csv('/Users/mac/Downloads/heart_failure_clinical_records_dataset.csv')
data <- data[data$ejection_fraction <70,]
data <- data[data$creatinine_phosphokinase <7000,]
data$anaemia <- factor(data$anaemia)
data$diabetes <- factor(data$diabetes)
data$high_blood_pressure <- factor(data$high_blood_pressure)
data$sex <- factor(data$sex)
data$smoking <- factor(data$smoking)
# this package needs the dependent variable in numeric format
# hence we reload data here
IV <- create_infotables(data=data, y="DEATH_EVENT",
                bins=10, parallel=FALSE)
IV_Value = data.frame(IV$Summary)
IV$Summary
```

```
##                     Variable           IV
## 12                      time 1.840224e+00
## 5          ejection_fraction 9.763676e-01
## 8            serum_creatinine 9.235629e-01
## 1                        age 4.849249e-01
## 9               serum_sodium 4.030774e-01
## 3   creatinine_phosphokinase 2.157046e-01
## 7                  platelets 1.132326e-01
## 6        high_blood_pressure 2.194425e-02
## 2                    anaemia 2.158339e-02
## 10                       sex 3.028463e-04
## 11                   smoking 7.862779e-05
## 4                   diabetes 8.479320e-06
```

Let's analyze IV first - Our IV values are significant for time, ejection_fraction, serum_creatinine, age, serum_sodium, creatinine_phosphokinase and platelets (>0.1). After platelets, IV values are below 0.02 so we do not need to use these variables.

Plotting WOE

```r
library(woe)
# plot woe
plot_infotables(IV, IV$Summary$Variable[1:12], same_scale=FALSE)
```


time
3
2
1
0
−1
age
1
0
−1
−2
platelets
0.25
0.00
−0.25
−0.50
ejection_fraction
2
1
0
−1
serum_sodium
1.0
0.5
0.0
−0.5
high_blood_pressure
0.2
0.1
0.0
−0.1
serum_creatinine
1
0
−1
−2
creatinine_phosphokinase
0.4
0.0
−0.4
anaemia
0.1
0.0
−0.1
sex
0.02
0.01
0.00
−0.01
smoking
0.005
0.000
−0.005
−0.010
diabetes
0.002
0.000
−0.002

Replacing WOE

```r
library(fuzzyjoin)
woe_replace <- function(df_orig, IV) {
  df <- cbind(df_orig)
  df_clmtyp <- data.frame(clmtyp = sapply(df, class))
  df_col_typ <-
    data.frame(clmnm = colnames(df), clmtyp = df_clmtyp$clmtyp)
  for (rownm in 1:nrow(df_col_typ)) {
    colmn_nm <- toString(df_col_typ[rownm, "clmnm"])
    if(colmn_nm %in% names(IV$Tables)){
    column_woe_df <- cbind(data.frame(IV$Tables[[toString(df_col_typ[rownm, "clmnm"])]]))
    if (df_col_typ[rownm, "clmtyp"] == "factor" | df_col_typ[rownm, "clmtyp"] == "character") {
      df <-
        dplyr::inner_join(
          df,
          column_woe_df[,c(colmn_nm,"WOE")],
          by = colmn_nm,
          type = "inner",
          match = "all"
        )
      df[colmn_nm]<-NULL
      colnames(df)[colnames(df)=="WOE"]<-colmn_nm
    } else if (df_col_typ[rownm, "clmtyp"] == "numeric" | df_col_typ[rownm, "clmtyp"] == "integer") {
      column_woe_df$lv<-as.numeric(str_sub(
        column_woe_df[,colmn_nm],
        regexpr("\\[", column_woe_df[,colmn_nm]) + 1,
        regexpr(",", column_woe_df[,colmn_nm]) - 1
      ))
      column_woe_df$uv<-as.numeric(str_sub(
        column_woe_df[,colmn_nm],
        regexpr(",", column_woe_df[,colmn_nm]) + 1,
        regexpr("\\]", column_woe_df[,colmn_nm]) - 1
      ))
      column_woe_df[colmn_nm]<-NULL
      column_woe_df<-column_woe_df[,c("lv","uv","WOE")]
      colnames(df)[colnames(df)==colmn_nm]<-"WOE_temp2381111111111111697"
      df <-
        fuzzy_inner_join(
          df,
          column_woe_df[,c("lv","uv","WOE")],
          by = c("WOE_temp2381111111111111697"="lv","WOE_temp2381111111111111697"="uv"),
          match_fun=list(`>=`,`<=`)
        )
      df["WOE_temp2381111111111111697"]<-NULL
      df["lv"]<-NULL
      df["uv"]<-NULL
      colnames(df)[colnames(df)=="WOE"]<-colmn_nm
    }}
  }
  return(df)
}
df_woe <- woe_replace(data, IV)
```

```r
str(df_woe)
```

```
## 'data.frame':    295 obs. of  13 variables:
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ age                     : num  0.3879 0.00248 -0.32294 0.00248 1.32221 ...
##  $ anaemia                 : num  -0.132 -0.132 0.163 0.163 0.163 ...
##  $ creatinine_phosphokinase: num  0.265 0.651 0.401 0.651 -0.659 ...
##  $ diabetes                : num  0.00248 0.00248 0.00248 -0.00342 0.00248 ...
##  $ ejection_fraction       : num  2.67 2.67 2.67 2.67 -1.33 ...
##  $ high_blood_pressure     : num  0.197 -0.112 -0.112 -0.112 0.197 ...
##  $ platelets               : num  -0.4565 0.0825 0.4274 0.3161 0.4274 ...
##  $ serum_creatinine        : num  1.641 0.157 1.641 1.206 1.206 ...
##  $ serum_sodium            : num  0.999 0.999 -0.834 0.999 0.439 ...
##  $ sex                     : num  -0.0128 -0.0128 -0.0128 0.0237 -0.0128 ...
##  $ smoking                 : num  0.00615 -0.01279 0.00615 0.00615 -0.01279 ...
##  $ time                    : num  2.94 2.94 2.94 2.94 2.94 ...
```

Let’s now use the new dataframe for prediction.

Splitting into train and test - 70%, 30% split

```r
set.seed(123)
trainIndex <- createDataPartition(df_woe$DEATH_EVENT, p = .7,
                                  list = FALSE,
                                  times = 1)
train_data<-df_woe[trainIndex,]
test_data<-df_woe[-trainIndex,]
```

```r
# Model 5
mylogit_5 <- glm(DEATH_EVENT ~ age+anaemia+
                creatinine_phosphokinase+
    diabetes+ejection_fraction+high_blood_pressure+platelets+
    serum_creatinine+serum_sodium+sex+smoking+time ,
    data = train_data, family = "binomial")
summary(mylogit_5)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + anaemia + creatinine_phosphokinase +
##     diabetes + ejection_fraction + high_blood_pressure + platelets +
##     serum_creatinine + serum_sodium + sex + smoking + time, family = "binomial",
##     data = train_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.38680  -0.27533  -0.05267   0.04466   2.86971
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.7806     0.3805  -4.680 2.87e-06 ***
## age                         1.4738     0.4871   3.026 0.002481 **
## anaemia                    -1.0069     2.2465  -0.448 0.654019
## creatinine_phosphokinase    2.6262     0.7736   3.395 0.000687 ***
## diabetes                   41.5959   107.3797   0.387 0.698481
## ejection_fraction          1.8307     0.4350   4.209 2.57e-05 ***
## high_blood_pressure        0.1975     2.1164   0.093 0.925659
## platelets                  0.9531     0.9779   0.975 0.329764
## serum_creatinine           0.6848     0.3016   2.271 0.023159 *
## serum_sodium               1.7529     0.5805   3.020 0.002529 **
## sex                        3.5083    18.9358   0.185 0.853012
## smoking                  -45.4997    38.7693  -1.174 0.240555
## time                       1.6880     0.3265   5.171 2.33e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 251.006  on 206  degrees of freedom
## Residual deviance:  80.756  on 194  degrees of freedom
## AIC: 106.76
##
## Number of Fisher Scoring iterations: 7
```

```
predicted <- predict(mylogit_5, test_data, type="response")
```

We see age, time, ejection_fraction and serum_sodium, creatinine_phosphpkinase, serum_creatinine as significant variables.

```
# Model 5
mylogit_5 <- glm(DEATH_EVENT ~ age+
    ejection_fraction+
    serum_sodium+time+creatinine_phosphokinase+
        serum_creatinine, data = train_data, family = "binomial")
summary(mylogit_5)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_sodium +
##     time + creatinine_phosphokinase + serum_creatinine, family = "binomial",
##     data = train_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.60054  -0.28769  -0.06755   0.07144   2.75776
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.6741     0.3457  -4.843 1.28e-06 ***
## age                        1.4124     0.4665   3.028  0.00246 **
## ejection_fraction          1.6660     0.3847   4.331 1.49e-05 ***
## serum_sodium               1.6252     0.5245   3.099  0.00194 **
## time                       1.5588     0.2864   5.442 5.27e-08 ***
## creatinine_phosphokinase   2.4491     0.7225   3.390  0.00070 ***
## serum_creatinine           0.6359     0.2825   2.251  0.02437 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 251.006  on 206  degrees of freedom
## Residual deviance:  83.779  on 200  degrees of freedom
## AIC: 97.779
##
## Number of Fisher Scoring iterations: 7
```

```
predicted <- predict(mylogit_5, test_data, type="response")
```

We note our lowest AIC yet of 97.77

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(test_data$DEATH_EVENT, predicted)[1]
optCutOff
```

```
## [1] 0.09998813
```

This tells us that we can use this prob cutoff to classify an observation as 0 or 1. If the prob-value is below this threshold we can classify it as survival else death event.

Mis-classification error

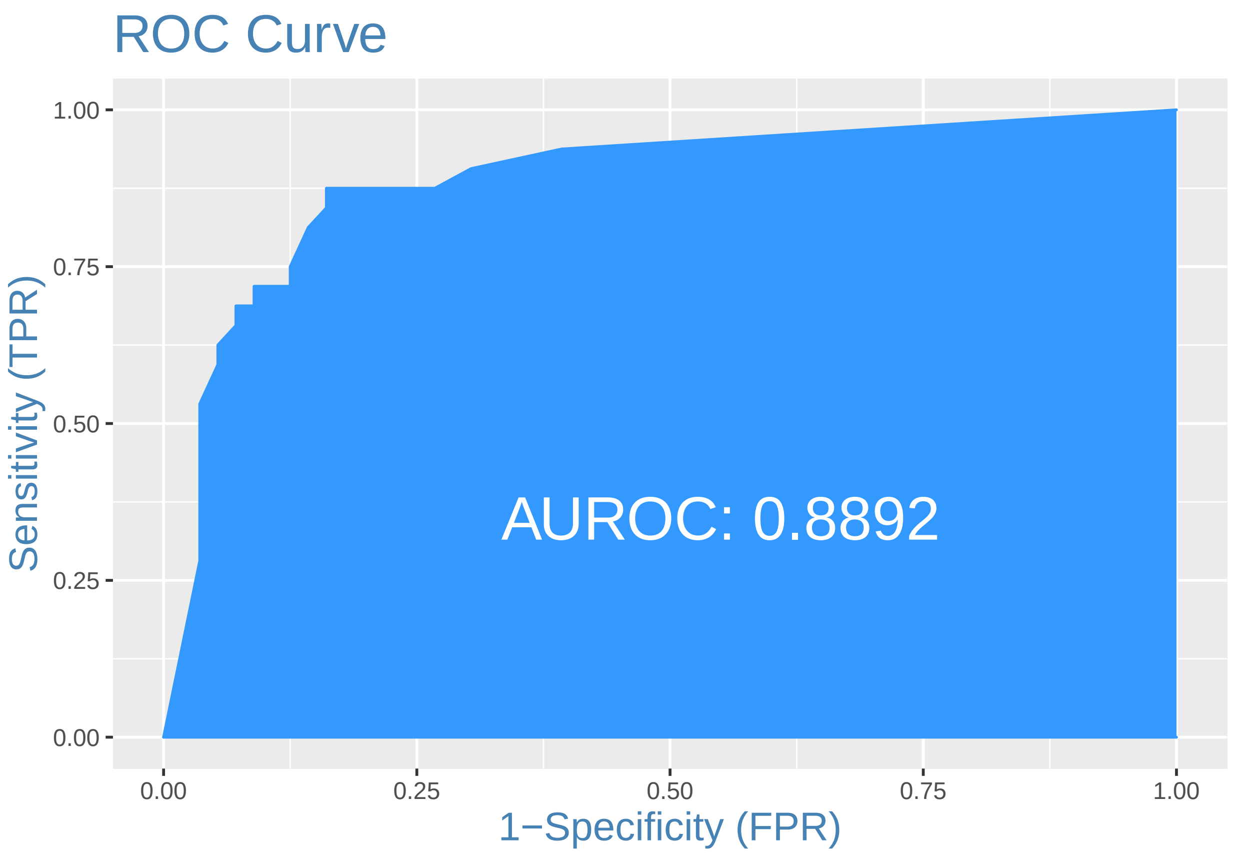
```
misClassError(test_data$DEATH_EVENT, predicted, threshold = optCutOff)
```

```
## [1] 0.1477
```

We note a misclassification error on test set of 14.7%

ROC curve

```
plotROC(test_data$DEATH_EVENT, predicted)
```



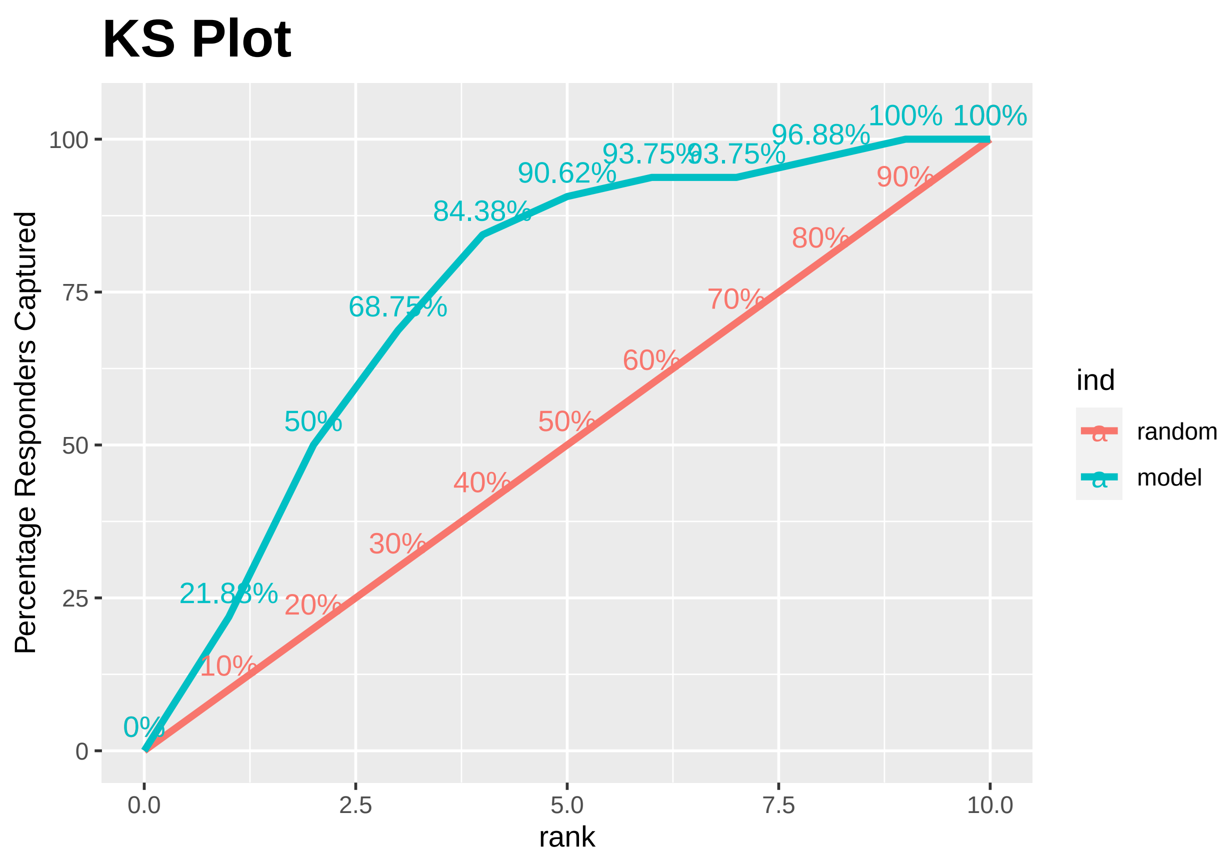
We see an AUC of 0.889 which is decent.

KS plot

```
ks_stat(test_data$DEATH_EVENT, predicted)
```

```
## [1] 0.683
```

```
ks_plot(test_data$DEATH_EVENT, predicted)
```



Concordance check

```
Concordance(test_data$DEATH_EVENT, predicted)
```

```
## $Concordance
## [1] 0.8889509
##
## $Discordance
## [1] 0.1110491
##
## $Tied
## [1] -4.163336e-17
##
## $Pairs
## [1] 1792
```

Usually concordance is in-line with AUC and we see that 88.8% pairs are concordant (the model calculated prob scores of 1s being greater than model calculated prob scores of 0s)

Using optimal cutoff to determine accuracy measures

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_5,test_data,
                    type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
```

```
confusionMatrix(data = as.factor(predicted_values),
    reference = as.factor(actual_values), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  4
##          1  9 28
##
##                Accuracy : 0.8523
##                  95% CI : (0.7606, 0.9189)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 6.225e-06
##
##                   Kappa : 0.6911
##
##  Mcnemar's Test P-Value : 0.2673
##
##               Precision : 0.9216
##                  Recall : 0.8393
##                      F1 : 0.8785
##              Prevalence : 0.6364
##          Detection Rate : 0.5341
##    Detection Prevalence : 0.5795
##       Balanced Accuracy : 0.8571
##
##        'Positive' Class : 0
##
```

We see that on test set our accuracy is 85.2%

Using optimal cutoff to determine accuracy measures with positive class as 1

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit_5, test_data,
                            type="response")>threshold,1,0)
actual_values<-test_data$DEATH_EVENT
confusionMatrix(data = as.factor(predicted_values),
            reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  4
##          1  9 28
##
##                Accuracy : 0.8523
##                  95% CI : (0.7606, 0.9189)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 6.225e-06
```

```
##
##                   Kappa : 0.6911
##
##  Mcnemar's Test P-Value : 0.2673
##
##               Precision : 0.7568
##                  Recall : 0.8750
##                      F1 : 0.8116
##              Prevalence : 0.3636
##          Detection Rate : 0.3182
##    Detection Prevalence : 0.4205
##       Balanced Accuracy : 0.8571
##
##        'Positive' Class : 1
##
```

We however note a key difference in this accuracy. Our recall is 0.87 while the precision is 0.75 with F1-score of 0.81

We see how computing and recoding variables to WOE improved our model accuracy even further. We also see our highest recall yet of 0.87 which is great for the purpose of predicting death events more rigorously.

Summarizing all model results in a table

Model	Variables	AIC	AUC	Accuracy	Precision	Recall	F1-Score	Comments
Model 1	All	241.95	0.894	0.854	0.79	0.72	0.75	No test set (overfitting)
Model 2	Age, ejection_fraction, serum_creatinine, time	232.4	0.888	0.85	0.84	0.64	0.73	No test set (overfitting)
Model 3	All	173.4	0.868	0.827	0.80	0.59	0.68	Test set results- First real model
Model 4	Age, ejection_fraction, serum_creatinine, time, sex	164.7	0.87	0.85	0.79	0.70	0.74	Stepwise Forward selection
Model 5	Age, ejection_fraction, serum_creatinine, time, serum_sodium, creatinine_phosphokinase	97.7	0.89	0.852	0.75	0.88	0.81	WoE dataset

We note that in model 4 stepwise method performed well and gave us better model results than model 3

However, we finally have a model (Model 5) which we can use as a best model outcome from our iterations which came from computing woe and iv values for each of our variables in the dataset.

This concludes our approach to Logistic regression in our dataset