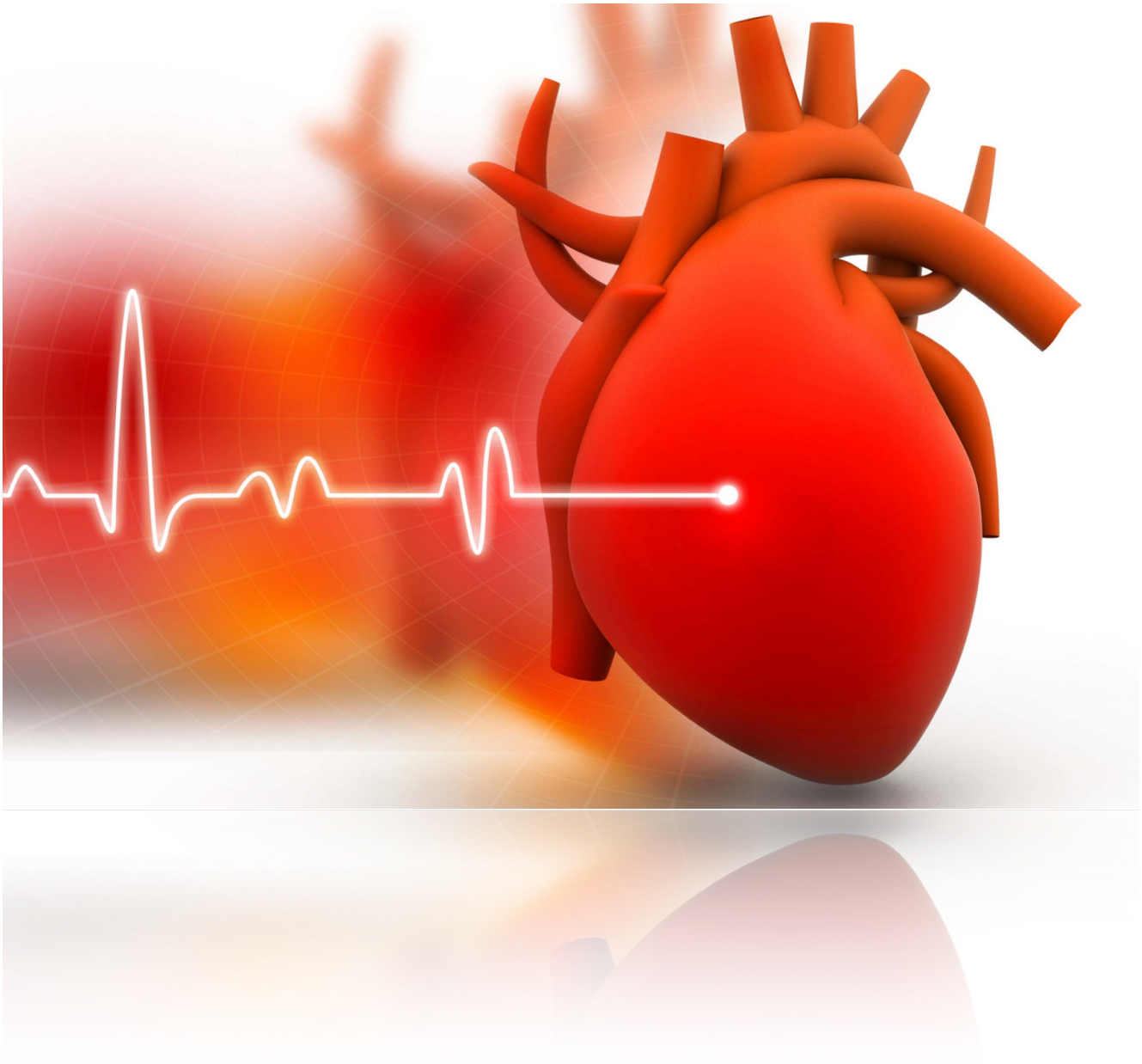
Heart Failure Probability Regression & Classification

Estimating risk of death from heart failure with machine learning

Daniel Peslherbe (dpeslherbe@outlook.com & https://dpeslherbe.wixsite.com/website) -
July 30, 2020



Introduction

Cars need fuel pumped into engines for the engine to work properly, water-cooled computers need water to flow through tubes to cool them to prevent overheating, broadband internet goes from a central control system through cabling and optical fibre to deliver its signals to homes and businesses. Just like these systems, the heart is needed to pump blood through the veins of the body to insure survival. However, unlike previous examples, we only have one heart working continuously through our entire life. Sometimes, the heart (though combinations of genetics, lifestyle habits, and other factors which are not the topic of discussion here) is not capable of shouldering this immense load. When the heart tends to give out on a chronic basis, this is condition is referred to as Heart Failure. The goal of this project is to try an establish models that would predict the event of death of heart failure based on several indicators.

Models & Data

For the purpose of this project, ten simple models will be used to try and evaluate the risk of a deadly event from Heart Failure given predictors. The data used here is from the Heart failure clinical records Data Set which can be found at http://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records with variable descriptions which are also available at https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5. In the dataset, the following information is given; whether the patient had anaemia (where 0 is not anaemic and 1 is anaemic, as boolean operator logic is applied for many other variables in the dataset), high blood pressure (boolean operator logic), diabetes (boolean operator logic). We are also given sex of the patient (where 0 signifies the patient is female, and 1 is male), if the patient was a smoker (boolean operator logic). The following biological indicators were also included Creatinine PhosphoKinase (CPK) in mcg/L, Ejection fraction in percentage, Platelets in kiloplateletes/mL, Serum Creatine in mg/dL, and Serum Sodium in mEq/L. Finally, we are also given the number of days in the follow-up period (time between being admitted for left ventricular systolic dysfunction and either being discharged or dying), and whether the patient had died or not (following Boolean operator logic once again).

Methodology (Initial Regression)

Using packages MASS, mgcv, mda, tree, matrixStats, caret, pROC, & ggplot2 in R, we start by splitting our dataset into 10 randomized groups using split() function; the first eight groups become both our training and validation sets (we will use leave-one-group-out as a cross-validation method), while the last two groups become our test set to draw fine conclusions from. To be able to judge model accuracy, we must have some sort of metric to base our models on. For this we create a *model type*.RMSE empty vector for each different model; we will use it to store the Root Mean Square Error of the model validated on one of the eight training/validation groups when the model is trained on the seven others. Thus, we train our models eight different times, recording RMSE values for each group, then after taking a small break to look over those results, we train our models over the entire training/validation set, and use our test set to give final conclusions on model efficiency.

Our first model is the most simplistic one: it is a simple linear regression model using lm() function and taking into account all the predictor variables in the dataset.

The second model is a revised version of the simple linear regression model: instead of using all predictors in the dataset, we first run a simple linear regression on the entire dataset, and through the use of information given by summary() and anova() functions on the model, we keep this time only predictors with a minimum of significance; here these predictors are age, ejection fraction, serum creatinine & time.

The third model is a generalized linear model using the glm() function with the binomial family and the logic link function (in essence the method for logistic regression using glms). To insure that we use the best forumla for this type of model, we use the stepAIC() function with direction set to 'both' to get the set of predictors leading to the lowest Akaike Information Criterion.

The fourth model is a generalized additive model using the function gam() with splines for non-boolean operator logic variables and just regression for boolean operator logic variables (if the splines were not necessary, the function would still calculate as a straight line anyways), then by using the same variable selection method as with our revised linear model using the summary() function.

The fifth model is a basic linear discriminant model. It uses the lda() function and takes into account all the variables which do not need to be standardized for discriminant analysis (as it was also not necessary for the above variables).

The sixth, seventh, and eight models are respectively quadratic discriminant, mixed discriminant, and flexible discriminant models (using qda(), mda(), fda() functions respectively) following the same predictor variables logic as the previous linear discriminant model.

The ninth and final are models are closely related; the ninth is a an extensive decision tree model built using tree() function, while the tenth is a pruned version of the decision tree model using set.seed(5) and the tree.cv() function set to minimize deviations based on the tree size.

Results (Initial Regression)

To keep this report clean and uncluttered, we will not include every numerical result, and model built, instead giving small examples below of code inputs (the simple linear model computations with fifth group as validation, the linear discriminant model with third group as validation and the pruned decision tree with fifth group as validation) :

Simple Linear Code:

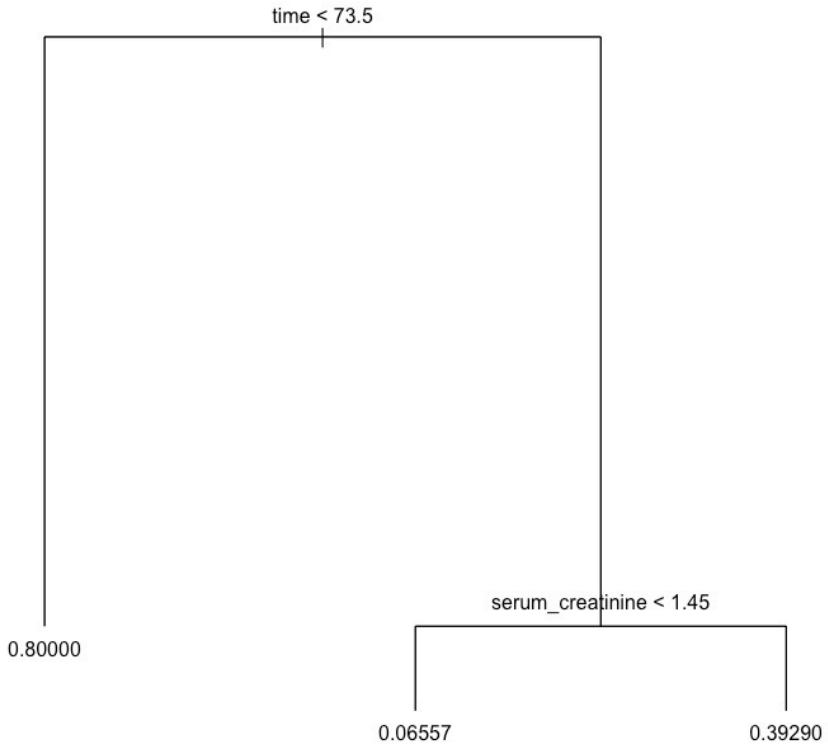
```
> simplelinear.model <- lm(DEATH_EVENT ~ ., data = trainset)
> simplelinear.model
Call:
lm(formula = DEATH_EVENT ~ ., data = trainset)
Coefficients:
          (Intercept)                   age               anaemia
            1.233e+00             8.370e-03            -2.398e-02
creatinine_phosphokinase              diabetes      ejection_fraction
            2.694e-05             5.462e-02            -9.024e-03
  high_blood_pressure             platelets       serum_creatinine
            2.701e-02             3.944e-08             5.541e-02
         serum_sodium                   sex                smoking
           -6.063e-03            -5.786e-02             1.294e-02
                 time
           -2.687e-03
> simplelinear.valid <- predict(simplelinear.model, groups[[5]])
> simplelinear.valid
        5         12         14         18         39         63         82         90        102        107        119        124
1.06911883 0.69870589 0.46981702 0.68428635 0.76091578 0.40238311 0.35413414 0.41536614 0.45073169 0.26473326 0.25262733 0.40352637
      125        161        164        181        189        204        230        235        244        249        254        255
0.47508833 0.30987964 0.28234294 0.14612165 0.21796691 0.36835123 0.32419074 -0.10212863 0.14914807 -0.14503608 0.28464973 -0.26252017
      296
-0.09453922
> simplelinear.RMSE <- c(simplelinear.RMSE, sqrt(sum((groups[[5]]$DEATH_EVENT - simplelinear.valid)^2))/nrow(groups[[5]]))
```

Linear Discriminant Code:

```
> lindiscr.model <- lda(DEATH_EVENT ~., data = trainset)
> lindiscr.model
Call:
lda(DEATH_EVENT ~ ., data = trainset)
Prior probabilities of groups:
        0         1
0.6868687 0.3131313
Group means:
      age          anaemia      creatinine_phosphokinase  diabetes    ejection_fraction   high_blood_pressure   platelets   serum_creatinine     serum_sodium
0   58.81373    0.4117647    607.5515                    0.4264706   40.25735            0.3455882             261789.7    .217279              137.0588
1   67.02689    0.4516129    701.6129                    0.4032258   33.17742            0.4516129             256050.8    1.796452             135.5645
      sex              smoking        time
0    0.6470588      0.3161765     159.98529
1    0.6612903.     0.3064516     69.91935
Coefficients of linear discriminants:
                          LD1
age                3.003853e-02
```

```
anaemia                   5.982976e-02
creatinine_phosphokinase  1.066581e-04
diabetes                  1.091380e-01
ejection_fraction        -4.208915e-02
high_blood_pressure       2.310405e-02
platelets                -7.540239e-08
serum_creatinine          2.350919e-01
serum_sodium             -2.311929e-02
sex                      -2.667262e-01
smoking                   6.143551e-02
time                     -1.243420e-02
> lindiscr.valid <- predict(lindiscr.model, groups[[3]], type = 'response')
> lindiscr.valid <- lindiscr.valid$posterior[,2]
> lindiscr.valid
        8         9        21        40        42        57        58        85        95       105
0.420577508 0.378931221 0.928158850 0.825311414 0.769936798 0.793067784 0.400613625 0.678533537 0.321928155 0.228009322
      106       108       109       110       113       146       147       155       157       174
0.761001488 0.238859271 0.380126638 0.174893337 0.456730891 0.142444687 0.186184320 0.328684525 0.218071201 0.157624384
      217       243       247       252       257       258       272       274       277       287
0.030172763 0.009377192 0.050835559 0.025365119 0.041484947 0.012859043 0.011066553 0.002135598 0.028234546 0.011908151
      290       293       295
0.045583524 0.004999411 0.006365670
> lindiscr.RMSE <- c(lindiscr.RMSE, sqrt(sum((groups[[3]]$DEATH_EVENT - lindiscr.valid)^2))/nrow(groups[[3]]))
```

Pruned Decision Tree Output:



Below, we include in a table the average RMSE scores from cross-validation in the first row, and the RMSE scores from our test section in the second row.

RMSE	Cross-Validation Average	Test Set
Simple Linear Model	0.07002033	0.04457358
Revised Linear Model	0.06794522	0.04571624
Generalized Linear Model	0.06561213	0.04403429
Generalized Additive Model	0.06482254	0.04052547
Linear Discriminant Model	0.06725358	0.04457358
Quadratic Discriminant Model	0.08396521	0.05211896
Mixed Discriminant Model	0.07155041	0.04302420
Flexible Discriminant Model	0.06725752	0.04457326
Decision Tree Model	0.06971369	0.04870489
Pruned Decision Tree Model	0.06542507	0.04855105

From these cross-validation metrics, we can hypothesized that the Generalized Additive Model seemed like the best model, followed closely by the Pruned Decision Tree, Generalized Linear, Revised Linear, Linear Discriminant, and Flexible Discriminant Models. When applying our training/validation sets to train models, all models improve against the Test set predictions (which is normal, given the increase in the size of the training set), with the Generalized Additive Model performing best in both cross-validation and testing phases (Cross-Validation Average RMSE of 0.06482254 and Test RMSE of 0.04052547); in contrast, the Mixed Discriminant has the second lowest RMSE for the Test Set (0.04302420), but a Cross-Validation Average RMSE of 0.07155041 (second highest) which is why it cannot be recommended above the Generalized Additive Model who performed well in both phases.

Interestingly, the Pruned Decision Tree Model boasts good metrics for both phases, even though most iterations of the Pruned Decision Tree Models only have Time as the pruning variable. Which also leads to an interesting question; when patients are admitted, can we know how long their follow-up period will be ? While we could also try to build models to try to estimate the length of the follow-up period, we would rather try to stay on course with our initial goal: Predicting probability of death from Heart Failure. Thus, to build

some purely analytical models, we will have to re-apply the same steps from before, but without taking the Time variable into account (since it is not predictable in advance).

Methodology (Second Regression)

Thus, as already done above, we run our models through cross-validation and testing set, but this time, without taking Time as a predictor variables.

The Simple Linear Model takes into account all variables available except the Time variable.

The Revised Linear Model uses only the predictors with high significance from the above Simple Linear Model, which are Age, Ejection Fraction, & Serum Creatinine.

The Generalized Linear Model finds the relevant through predictors through the stepAIC() function, but disregarding the Time predictor variable on every iteration while searching for the set predictor variables with the lowest associated AIC.

The Generalized Additive Model uses only the significant predictors and splines, which are the following predictor variables as splines; Age, Creatinine PhosphoKinase, Ejection Fraction, & Serum Creatinine.

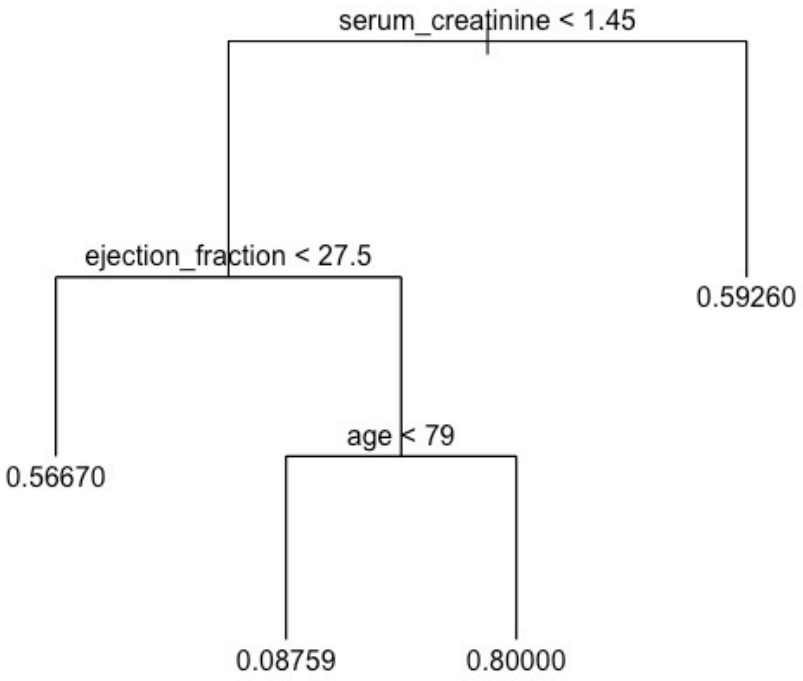
All Discriminant Models follow the same format using all predictor variables except the Time variable.

The Decision Tree Models uses all predictor variables, with the exclusion of the Time variable, and the Pruned Decision Tree is built from the Decision Tree that excludes Time variable.

Results (Second Regression)

As expected, with the absence of the Time variables, the error metrics for all our models is larger than on our Initial Regression. However, now, our models are purely predictive; they should be able to assess the probability of someone dying from Heart Failure, following admittance to a Hospital. The most notable change is for the Pruned Decision Tree Model, where there there a more than one branch on every iteration (in our Initial Regression, almost all Pruned Decision Tree Models had only the Time variable as a branch alone or without another variable). We have included the Pruned Decision Tree from our Testing Set phase, below as an example, followed by a table indicating the new value for our error metrics in the Second Regression.

Pruned Decision Tree Output:



RMSE	Cross-Validation Average	Test Set
Simple Linear Model	0.08049216	0.05095655
Revised Linear Model	0.07796662	0.05252098
Generalized Linear Model	0.08018486	0.04982938
Generalized Additive Model	0.07540968	0.04947659
Linear Discriminant Model	0.08101815	0.05095655
Quadratic Discriminant Model	0.0932875	0.05742224
Mixed Discriminant Model	0.0845517	0.04980596
Flexible Discriminant Model	0.08106155	0.05092428
Decision Tree Model	0.0848603	0.05987955
Pruned Decision Tree Model	0.07600848	0.05287502

We can note an improvement for all models here with the worst being the Original Decision Tree with a 5.99% RMSE. While many of our models are close to its RMSE, the best model we have is our Generalized Additive model, since it had a low error score in our cross-validation as well as in our test phase (others either have a great test score, with unremarkable cross-validation results, or great cross-validation results with a test score in the same range as other models).

However what we get our approximate percentages of dying from Heart Failure according to biological indicators used as predictors (they are approximate because some cases may give us negative probabilities or probabilities above 100% [which could be categorized as impossible to die from Heart Failure or death from Heart Failure inevitable respectively]).

Then, let us test our models this way one more time, but this time, with the goal being classification (death vs no death) instead of computed predicted probability of the death event.

Methodology (Classification)

Classification implies that the model classifies the outcome (here either death or survival) depending on the predictor variables the model is trained. However, using RMSE would be insufficient in this case to determine the best model to be used, since we can have false negatives and false positives which are both wrong, but indicate different weaknesses in the model. Thus, in addition to our empty RMSE vector, we also set up empty Accuracy, Precision, Recall, F1 Score, AUC value vectors, as well as empty list for ROC curves, and our Confusion Matrices.

Accuracy is the percentage of correct predictions by the model. Precision is the percentage of correctly predicted positive values over all predicted positive values. Recall is the percentage of correctly predicted positive values over all actual positive values. The F1 Score is an harmonic mean of both Precision and Recall (it is at its highest when both Precision and Recall are high). The AUC score corresponds to the Area Under Curve of the Receiver Operating Characteristic Curve which we also store in a list to plot all together. A perfect model would have an AUC of 1 (but can be as low as 0 for a model that cannot predict anything correctly, where an average model would have an AUC of 0.5). The Confusion Matrices are simple 2x2 matrices showing true positives, false positives, false negatives, and true positives from our model results.

For our Simple Linear, Revised Linear, Generalized Linear, Generalized Additive Models, Classification is done by computing the probabilities as in the Second Regression, and then classifying the outcome by round up or down to the closest to 0 or 1 (ie. 0.4 becomes 0, 0.7 becomes 1, 0.5 becomes 1, 0.25 becomes 0, etc…)

For our Discriminant based Models, by changing the section type in the function predict() for Linear and Quadratic Discriminant from 'posterior' to 'response' & for Mixed and Flexible Discriminant from 'posterior' to 'class', then the output of the prediction is then outputted as either 0 or 1 (which we convert into numerical values from factors for some calculations).

For our Tree based Models, the change is simply to input DEATH_EVENT as a factor using the as.factor() function in our tree() function to build a classification model instead.

Results (Classification)

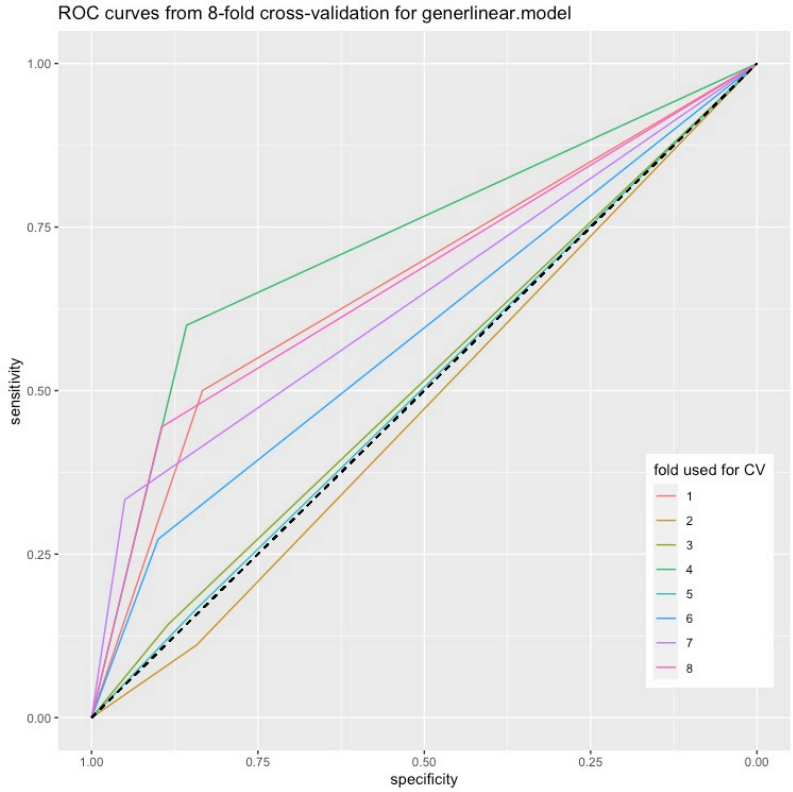
See below, the code output for Generalized Additive Model on the Test Set, the Set of 8 ROC curves from the 8 cross-validation results from Generalized Linear Model, as well the Decision Tree Model, the Cross-Validation Tree Size against Deviation Graph & Pruned Decision Tree Model on the Test Set, as examples for this section:

Generalized Additive Code:

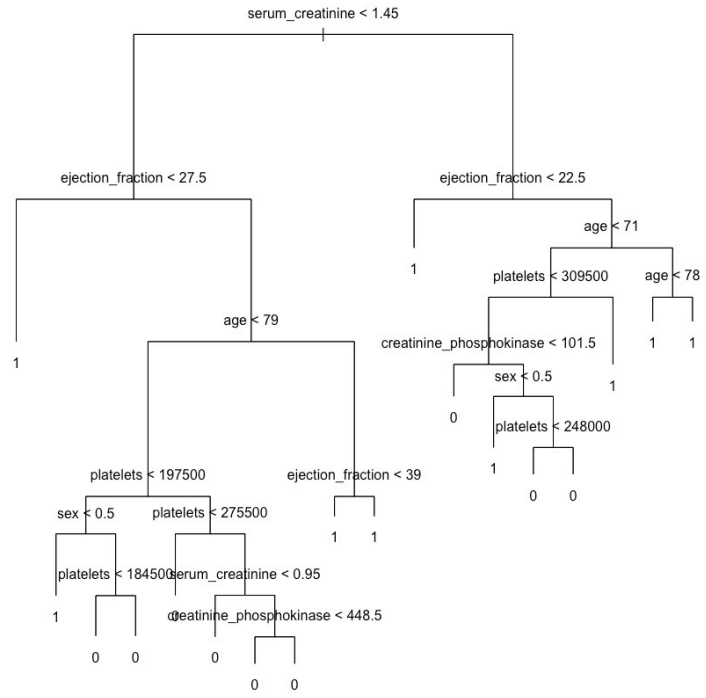
```
> generaddit.model <- gam(DEATH_EVENT ~ s(age)+ s(creatinine_phosphokinase) + s(ejection_fraction) + s(serum_creatinine), data = trainset, method = 'REML')
> generaddit.model
Family: gaussian
Link function: identity
Formula:
DEATH_EVENT ~ s(age) + s(creatinine_phosphokinase) + s(ejection_fraction) +
    s(serum_creatinine)
Estimated degrees of freedom:
2.83 1.00 3.50 3.07  total = 11.4
REML score: 124.8923
> summary(generaddit.model)
Family: gaussian
Link function: identity
Formula:
DEATH_EVENT ~ s(age) + s(creatinine_phosphokinase) + s(ejection_fraction) +
    s(serum_creatinine)
Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.29870    0.02532   11.79   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Approximate significance of smooth terms:
                           edf Ref.df     F  p-value
s(age)                   2.831  3.552 6.561 0.000115 ***
s(creatinine_phosphokinase) 1.000  1.001 1.844 0.175952
s(ejection_fraction)       3.499  4.320 9.410 2.57e-07 ***
s(serum_creatinine)        3.069  3.802 4.310 0.002711 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
R-sq.(adj) =  0.296   Deviance explained = 32.8%
-REML = 124.89  Scale est. = 0.14814   n = 231
> generaddit.test <- predict(generaddit.model, testset, type = 'response')
> for (j in 1:length(generaddit.test)) {
+   if(generaddit.test[j] < 0.5){
+     generaddit.test[j] <- 0
+   }
```

```
+   if(generaddit.test[j] >= 0.5){
+     generaddit.test[j] <- 1
+   }
+ }
> generaddit.test
  2   3   4  15  29  55  60  64  65  79  86  87  91  96 114 116 121 130 131 137
  0   1   1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
156 159 160 165 168 178 200 208 218 220 221 231 250 263 273 284 285 291  11  13
  1   1   0   0   1   0   0   0   1   0   1   0   0   0   0   0   0   0   0   0
 17  19  20  23  30  31  48  83  88  92  94 122 127 139 152 163 180 185 213 225
  0   0   0   0   1   1   0   1   0   0   1   0   1   0   0   0   0   0   0   0
227 242 253 262 265 266 270 299
  0   0   0   0   0   0   0   0
> generaddit.test.RMSE <- sqrt(sum((testset$DEATH_EVENT - generaddit.test)^2))/nrow(testset)
> generaddit.test.ConfMat <- confusionMatrix(factor(generaddit.test), factor(testset$DEATH_EVENT))
> generaddit.test.accur <- generaddit.test.ConfMat$overall[1]
> generaddit.test.precis <- generaddit.test.ConfMat$byClass[5]
> generaddit.test.recall <- generaddit.test.ConfMat$byClass[6]
> generaddit.test.F1score <- generaddit.test.ConfMat$byClass[7]
> generaddit.test.ConfMat <- generaddit.test.ConfMat$table
> generaddit.test.ROCcurv <- roc(testset$DEATH_EVENT, generaddit.test)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> generaddit.test.AUCsc <- auc(testset$DEATH_EVENT, generaddit.test)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

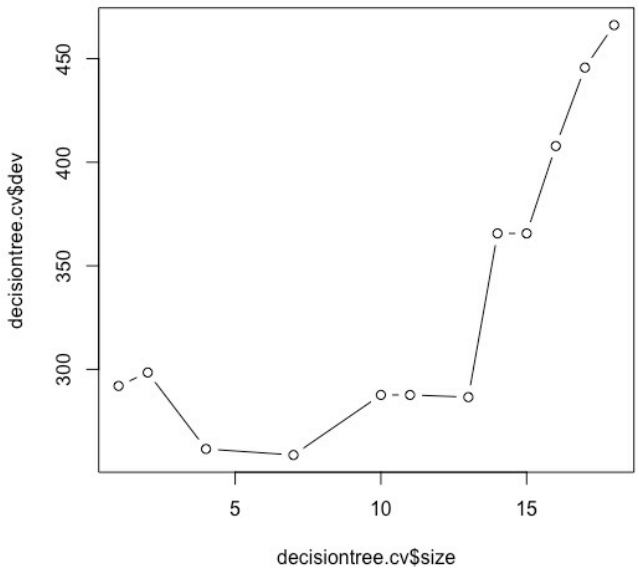
ROC curves combined plot for Generalized Linear Output:



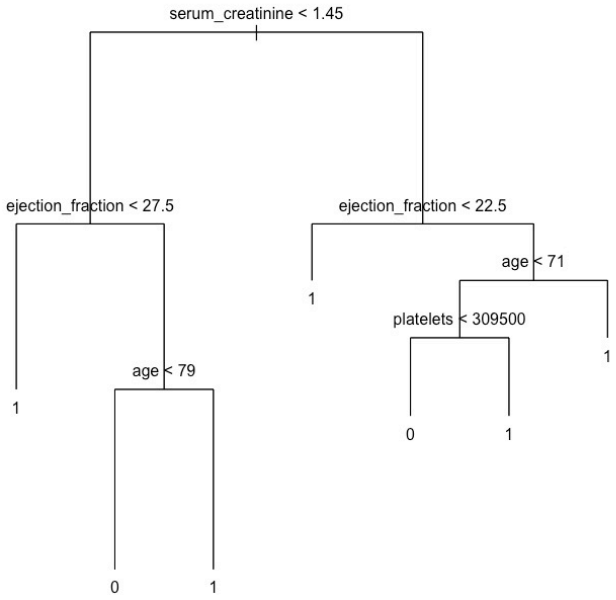
Decision Tree Output:



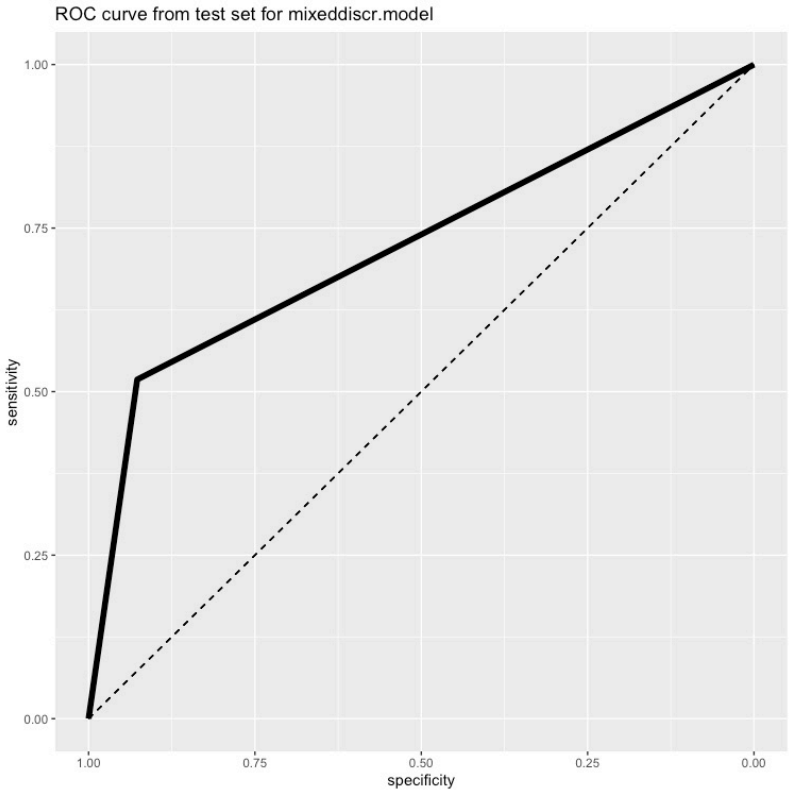
Pruning Tree through Cross-Validation Graph:



Pruned Decision Tree Output:



ROC Curve for Mixed Discriminant on Test set:



Thus, we get the following metrics from our models:

Cross-Validation Results	RMSE	Accuracy	Precision	Recall	F1 Score	AUC Score
Simple Linear Model	0.09959962	0.713211	0.7479634	0.8928126	0.8127616	0.5883525
Revised Linear Model	0.09501127	0.738739	0.7592342	0.9175776	0.8302935	0.6130614
Generalized Linear Model	0.09966705	0.7131773	0.7567957	0.8755049	0.8106114	0.5984487
Generalized Additive Model	0.09467038	0.7405892	0.7770175	0.888062	0.8278937	0.6328439
Linear Discriminant Model	0.1036575	0.6880544	0.7391981	0.8557315	0.7921428	0.569812
Quadratic Discriminant Model	0.1066421	0.6668111	0.7142866	0.867331	0.7821984	0.5332461
Mixed Discriminant Model	0.1055745	0.6767798	0.7269975	0.8608368	0.7870189	0.5501898
Flexible Discriminant Model	0.1036575	0.6880544	0.7391981	0.8557315	0.7921428	0.569812
Decision Tree Model	0.09723775	0.7230848	0.8048025	0.08015724	0.8021641	0.6670245
Pruned Decision Tree Model	0.09531263	0.7355376	0.8165335	0.812353	0.8121774	0.6824053

From these results, we can already identify certain issues with our models; on one part, all models (Tree based models aside) have a decent F1 score (around 80%), but whereas our Tree based models have balanced Precision and Recall averages (~80% as well), the other models have a worse Precision (~high 70%), and a better Recall (~high 80%). This indicates that the models are good at identifying true non-fatal cases amongst the non-fatal cases predicted, but that models also tend to predict non-fatal cases where the actual outcome is

fatal. Thus, based on Cross-Validation, the Tree based Models seem to be superior, even if the other modes are not far behind.

Then, on our Test set, the following error metrics are produced by our analysis:

Test Set Results	RMSE	Accuracy	Precision	Recall	F1 Score	AUC Score
Simple Linear Model	0.06063391	0.75	0.7222222	0.9512195	0.8210526	0.6978
Revised Linear Model	0.06063391	0.75	0.7142857	0.9756098	0.8247423	0.6915
Generalized Linear Model	0.05882353	0.7647059	0.7358491	0.9512195	0.8297872	0.7164
Generalized Additive Model	0.06239177	0.7352941	0.7090909	0.9512195	0.8125	0.6793
Linear Discriminant Model	0.05882353	0.7647059	0.7358491	0.9512195	0.8297872	0.7164
Quadratic Discriminant Model	0.06576671	0.7058824	0.6981132	0.902439	0.787234	0.6549
Mixed Discriminant Model	0.05882353	0.7647059	0.745098	0.9268293	0.826087	0.7227
Flexible Discriminant Model	0.5882353	0.7647059	0.7358491	0.9512195	0.8297872	0.7164
Decision Tree Model	0.06410146	0.7205882	0.7619048	0.7804878	0.7710843	0.7051
Pruned Decision Tree Model	0.06410146	0.7205882	0.72	0.8780488	0.7912088	0.6798

On our Test set, however, our Tree based models seem to falter, where some Linear and Discriminant Models improve (most notably Generalized Linear, Linear Discriminant, Mixed Discriminant, & Flexible Discriminant Models). However, given the last of consistency between our results from cross-validation and testing set phases, there are no models that we can recommend in the absolute.

Conclusions

Ideally, this dataset would be enlarged in the future with more data being sporadically added (as this might help improve the models). It may also be interesting to attempt any kind of clustering analysis to see if there significant differences between clusters (whether it be predictor values or actual death event outcomes) before using our machine learning models (this may lead to better improved, but would also depend on cluster sizes; where a cluster being too small outweighs model improvement). There is also the possibility of using other types of models used in Machine Learning, as the ones used here share a good amount of similarities with each other; but given the sheer amount of models that would need to be tested, this project only made use of those ten, for simplicity's sake.

Any comments or questions about this report, the R code may be redirected to dpeslherbe@outlook.com or the contact me section of my personal website. All relevant files are included in this shareable Google Drive Folder.