Instructions

This R Markdown file includes the questions, the empty code chunk sections for your code, and the text blocks for your responses. Answer the questions below by completing this R Markdown file. You must answer the questions using this file. You can change the format from pdf to Word or html and make other slight adjustments to get the file to knit but otherwise keep the formatting the same. Once you've finished answering the questions, submit your responses in a single knitted file (just like the homework peer assessments).

There are 16 questions divided among 5 sections, each worth 2-8 points. Partial credit may be given if your code is correct but your conclusion is incorrect or vice versa.

Next Steps:

1. Save this .Rmd file in your R working directory - the same directory where you will download the heart_failure.csv data file into. Having both files in the same directory will help in reading the .csv file.
2. Read the question and create the R code necessary within the code chunk section immediately below each question. Knitting this file will generate the output and insert it into the section below the code chunk.
3. Type your answer to the questions in the text block provided immediately after the question prompt: **This is how the question prompt will be formatted**.
4. Once you've finished answering all questions, knit this file and submit the knitted file on Canvas.

Example Question Format:

(8a) This will be the exam question - each question is already copied from Canvas and inserted into individual text blocks below, *you do not need to copy/paste the questions from the online Canvas exam.*

```
# Example code chunk area. Enter your code below the comment and between the ```{r} and ```
```

Response to question (8a) Example prompt for the question being asked: This is the section where you type your written answers to the question. Depending on the question asked, your typed response may be a number, a list of variables, a few sentences, or a combination of these elements.

** Ready? Let's begin. We wish you the best of luck! **

Final Exam Part 2 - Data Set Background

For this exam, you will be building a model to predict whether a heart failure patient will die during their follow-up period based on their characteristics.

The heart_failure.csv data set consists of the following 13 variables:

1. age: age of patient in years (integer)
2. anemia: was there a decrease in RBC/hemoglobin (1 = yes or 0 = no) (binary)
3. high blood pressure: does the patient have hypertension (1 = yes or 0 = no) (binary)
4. creatinine phosphokinase: CPK level in mgc/L (integer)
5. diabetes: does the patient have diabetes (1 = yes or 0 = no) (binary)

6. ejection fraction: percent of blood leaving (integer)
7. platelets: number of platelets in kiloplatelets/mL (integer)
8. sex: is the patient female (0) or male (1) (binary)
9. serum creatinine: ceratinine level in mg/dL (numeric)
10. serum sodium: sodium level in mEq/L (integer)
11. smoking: does the patient smoke (1 = yes or 0 = no) (binary)
12. time: follow-up period in days (integer)
13. death event: did the patient die during the follow-up period (1 = yes or 0 = no) (binary)

Read the data and answer the questions below. Assume a significance level of 0.05 for hypothesis tests.

Read Data

```
# Load relevant libraries (add here if needed)
library(car)
## Loading required package: carData
library(CombMSC)
## Warning: package 'CombMSC' was built under R version 4.0.3
##
## Attaching package: 'CombMSC'
## The following object is masked from 'package:car':
##
##     subsets
## The following object is masked from 'package:stats':
##
##     BIC
library(aod)
## Warning: package 'aod' was built under R version 4.0.3
library(bestglm)
## Warning: package 'bestglm' was built under R version 4.0.3
## Loading required package: leaps
## Warning: package 'leaps' was built under R version 4.0.3
library(boot)
##
## Attaching package: 'boot'
## The following object is masked from 'package:car':
##
##     logit
library(glmnet)
## Warning: package 'glmnet' was built under R version 4.0.3
## Loading required package: Matrix
## Loaded glmnet 4.0-2

# Ensure that the sampling type is correct
RNGkind(sample.kind="Rejection")

# Set seed
```

```r
set.seed(0)

# Read the data
dataFull = read.csv("heart_failure.csv", header=TRUE)

# Split data for traIning and testing
testRows = sample(nrow(dataFull), 0.2*nrow(dataFull))
dataTest = dataFull[testRows, ]
dataTrain = dataFull[-testRows, ]
```

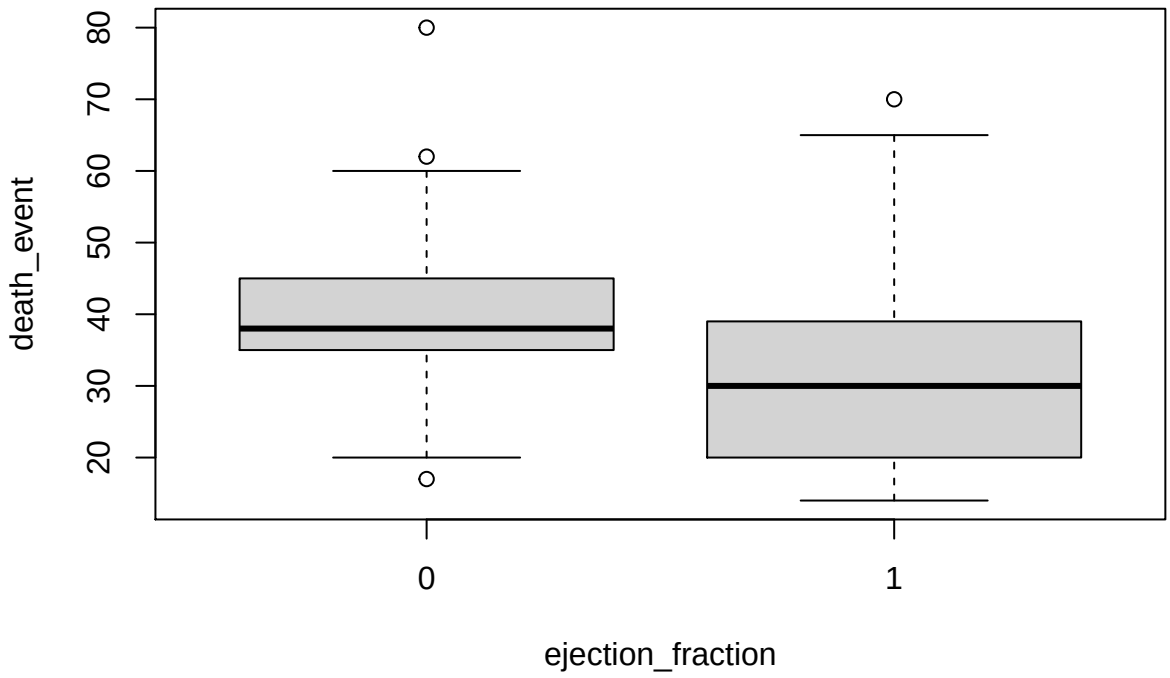
Note: Use *dataTrain* as your dataset for the following questions unless otherwise stated.

Note: All the categorical variables in the dataset only take on two levels, and the categories are already coded as 0's and 1's. *Please don't change the data types of the variables.*

Question 1: Exploratory Analysis

(1a) 3pts - Create a side-by-side boxplot for the variable *ejection_fraction* versus *death_event*. Does *ejection_fraction* appear useful in predicting whether a heart failure patient will die during their follow-up period? Include your reasoning.

```r
# Code to create plot
boxplot(dataTrain$ejection_fraction~dataTrain$death_event,xlab="ejection_fraction", ylab="death_event")
```



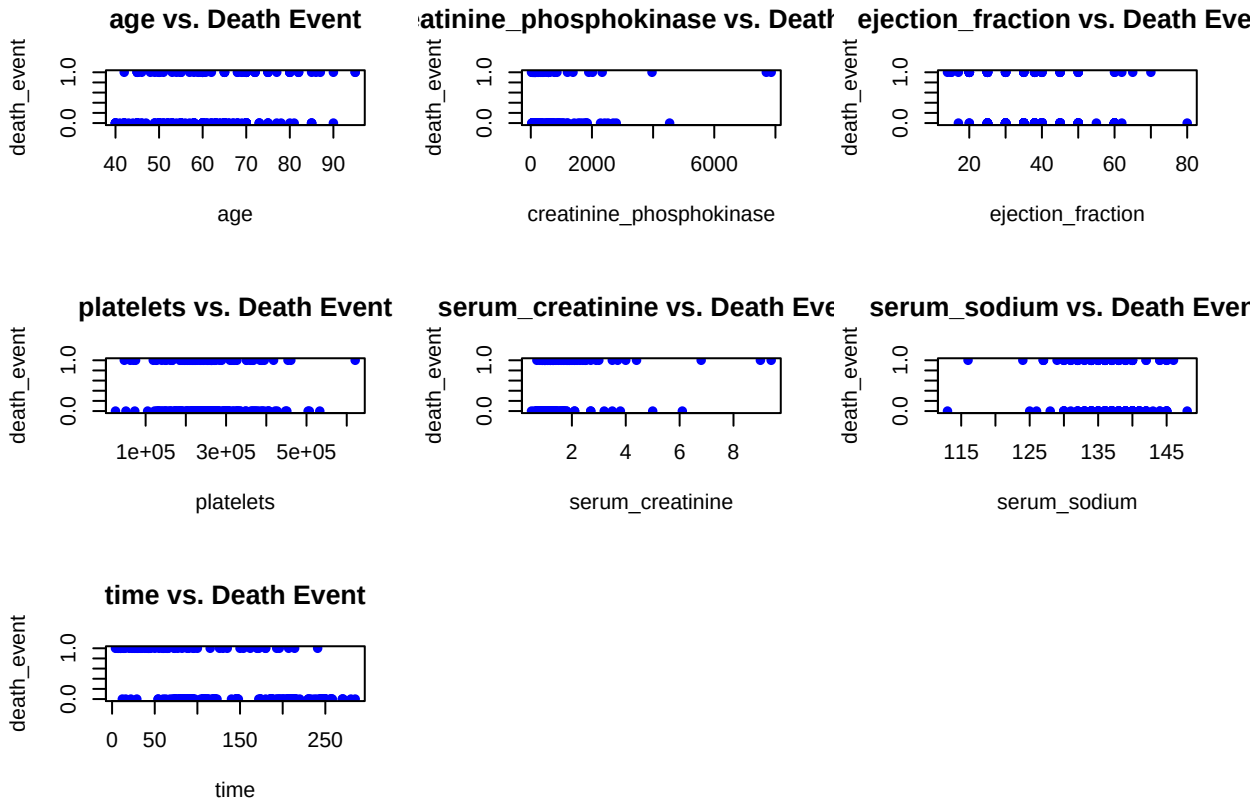
Response to question (1a):

Interpretation: Based on our results we can see that there is a few outlier observations outside the 1st and 3rd quantiles, but not enough to cause a significant effect, we have method like Cook's distance to be able to mitigate the potential effects and determine if they're actually influential points.

Most importantly, we can see that there is a difference between the median results for the ejection_fraction variable, which means it could be a valuable predictor for the death_event response variable.

(1b) 3pts - Create a scatterplot matrix and a correlation table that includes the following seven quantitative variables: *age*, *creatinine_phosphokinase*, *ejection_fraction*, *platelets*, *serum_creatinine*, *serum_sodium*, and *time*. Does there appear to be correlation among these seven variables? Will these potentially suggest multicollinearity? Include your reasoning.

```
# Code to create plot
par(mfrow=c(3,3))
plot(death_event~age, data=dataTrain, main="age vs. Death Event", col="blue", pch = 16)
plot(death_event~creatinine_phosphokinase, data=dataTrain, main="creatinine_phosphokinase vs. Death Ever
plot(death_event~ejection_fraction, data=dataTrain, main="ejection_fraction vs. Death Event", col="blue
plot(death_event~platelets, data=dataTrain, main="platelets vs. Death Event", col="blue", pch = 16)
plot(death_event~serum_creatinine, data=dataTrain, main="serum_creatinine vs. Death Event", col="blue",
plot(death_event~serum_sodium, data=dataTrain, main="serum_sodium vs. Death Event", col="blue", pch = 1
plot(death_event~time, data=dataTrain, main="time vs. Death Event", col="blue", pch = 16)
```




```
# Code to create correlation table
as.matrix(cor(dataTrain[c("age","creatinine_phosphokinase","ejection_fraction","platelets","serum_creati
##                                age creatinine_phosphokinase ejection_fraction
## age                    1.00000000             -0.084021726        0.08826702
## creatinine_phosphokinase -0.08402173              1.000000000       -0.02868456
```

```
## ejection_fraction         0.08826702             -0.028684564        1.00000000
## platelets                -0.02235132              0.009931885        0.05073244
## serum_creatinine          0.17651971              0.017239155        0.01547671
## serum_sodium             -0.02942660              0.055995651        0.24416143
## time                     -0.24144848              0.001908416        0.03936372
##                             platelets serum_creatinine serum_sodium
## age                      -0.022351318       0.17651971  -0.02942660
## creatinine_phosphokinase  0.009931885       0.01723916   0.05599565
## ejection_fraction         0.050732437       0.01547671   0.24416143
## platelets                 1.000000000      -0.01269547   0.01746740
## serum_creatinine         -0.012695465       1.00000000  -0.23332212
## serum_sodium              0.017467398      -0.23332212   1.00000000
## time                     -0.053201642      -0.13014791   0.08719919
##                                   time
## age                      -0.241448476
## creatinine_phosphokinase  0.001908416
## ejection_fraction         0.039363716
## platelets                -0.053201642
## serum_creatinine         -0.130147906
## serum_sodium              0.087199187
## time                      1.000000000
```

Response to question (1b):

Interpretation:

Based on the results we can see that there does not seem to be a problem of correlation between our variables as a majority of our values have a | correlation score | < 0.3 for each of the predictor combinations. We can further using the VIF (Variance Inflation Factor) to get a more precise result, however based on this matrix we're not seeing any signs of heavy correlation.

From you the result of our scatterplot, we can see that the each dependent variable seems to be well spread out and do not seem to form any groups or clustering. We do see potential outliers though in the two creatinine variables.

Question 2: Full Model

(2a) 2pts - Fit a logistic regression model with *death_event* as the response variable and all other variables as predicting variables. Include an intercept. Call it *model1*. Display the summary table for the model.

```
# Code to fit model and display summary
model1 <- glm(death_event~.,data=dataTrain)
summary(model1)
##
## Call:
## glm(formula = death_event ~ ., data = dataTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.76391  -0.25771  -0.03183   0.22334   0.82067
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               6.220e-01  7.653e-01   0.813  0.41723
```

```
## age                        5.845e-03  2.074e-03   2.818  0.00525 **
## anaemia                   -2.398e-02  4.796e-02  -0.500  0.61759
## creatinine_phosphokinase   4.231e-05  2.568e-05   1.648  0.10084
## diabetes                   7.686e-03  4.756e-02   0.162  0.87176
## ejection_fraction         -1.119e-02  2.007e-03  -5.575 6.99e-08 ***
## high_blood_pressure        1.122e-02  4.883e-02   0.230  0.81849
## platelets                 -1.777e-07  2.640e-07  -0.673  0.50149
## serum_creatinine           9.936e-02  2.213e-02   4.490 1.13e-05 ***
## serum_sodium               1.103e-04  5.572e-03   0.020  0.98422
## sex                       -3.976e-02  5.581e-02  -0.712  0.47696
## smoking                    8.384e-03  5.527e-02   0.152  0.87958
## time                      -2.684e-03  3.099e-04  -8.660 8.90e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1232956)
##
##     Null deviance: 50.400  on 239  degrees of freedom
## Residual deviance: 27.988  on 227  degrees of freedom
## AIC: 193.36
##
## Number of Fisher Scoring iterations: 2
```

(2b) 3pts - Evaluate the overall regression for *model1*. Use an alpha level of 0.05. What do you conclude? Include your reasoning.

```
# Code for overall regression test. P-value must be displayed
1-pchisq((model1$null.dev - model1$deviance),
(model1$df.null - model1$df.resid))
## [1] 0.03315456
```

Response to question (2b):

Interpretation Based on our results we can see that the p-value obtained is less than our alpha level of 0.05, indicating that the model is significant overall.

(2c) 3pts - Conduct a multicollinearity test on *model1*. Using a VIF threshold of 10, what can you conclude? Is your conclusion consistent with your observations from question (1b)?

```
# Code to conduct multicollinearity test
cat("VIF Threshold:", max(10, 1/(1-summary(model1)$r.squared)), "\n")
## VIF Threshold: 10
```

```
vif(model1)
##                      age                  anaemia creatinine_phosphokinase
##                 1.128061                 1.101743                 1.062038
##                 diabetes        ejection_fraction      high_blood_pressure
##                 1.056797                 1.116894                 1.055877
##                platelets         serum_creatinine             serum_sodium
##                 1.037411                 1.110782                 1.181563
##                      sex                  smoking                     time
##                 1.393999                 1.321580                 1.127134
```

Response to question (2c):

Interpretation Based on our results, we can see that the VIF factor for each variable is less than the threshold of 10. Meaning we are not experiencing multicollinearity in our model. This matches up with the results we saw in our correlation matrix as part of question 1b.

Question 3: Variable Selection

(3a) 3pts - Conduct a complete search to find the submodel with the smallest BIC. Fit this model. Include an intercept. Call it *model2*. Display the summary table for the model. *Note: Remember to set family to binomial.* Which variables are in your *model2*?

```
# Code to conduct exhaustive search
regsub = regsubsets(x=as.matrix(dataTrain[-13]),y=dataTrain$death_event,nbest=1,nvmaz=4096,method="exhau
summary_regsub = summary(regsub)
as.matrix(cbind(summary_regsub$which,summary_regsub$bic))
##   (Intercept) age anaemia creatinine_phosphokinase diabetes ejection_fraction
## 1           1   0       0                        0        0                 0
## 2           1   0       0                        0        0                 1
## 3           1   0       0                        0        0                 1
## 4           1   1       0                        0        0                 1
## 5           1   1       0                        1        0                 1
## 6           1   1       0                        1        0                 1
## 7           1   1       0                        1        0                 1
## 8           1   1       1                        1        0                 1
##   high_blood_pressure platelets serum_creatinine serum_sodium sex smoking time
## 1                   0         0                0            0   0       0    1
## 2                   0         0                0            0   0       0    1
## 3                   0         0                1            0   0       0    1
## 4                   0         0                1            0   0       0    1
## 5                   0         0                1            0   0       0    1
## 6                   0         0                1            0   1       0    1
## 7                   0         1                1            0   1       0    1
## 8                   0         1                1            0   1       0    1
##
## 1  -67.76394
## 2  -87.94793
## 3 -107.79953
## 4 -109.42701
## 5 -107.00806
## 6 -101.96216
## 7  -96.93177
## 8  -91.74316
```

To find the best submodel we're going to use the regsubets command so that we can perform an exhaustive search to determine the best model. Since we have 12 predicting variables, our nvmax variables will be set to 2^12 = 4096, as the option used to determine how many submodels we're going to search through, the nbest option will provide back number of subsets of each size to record, in our case 1.

The results show us that the model with the lowest BIC score is the 4th row, which provides back a BIC score of -109.42701.

```
# Code to conduct exhaustive search
summary_regsub$which[4,]
##             (Intercept)                     age                 anaemia
```

```
##                      TRUE                      TRUE                     FALSE
## creatinine_phosphokinase                  diabetes        ejection_fraction
##                     FALSE                     FALSE                      TRUE
##      high_blood_pressure                 platelets         serum_creatinine
##                     FALSE                     FALSE                      TRUE
##             serum_sodium                       sex                   smoking
##                     FALSE                     FALSE                     FALSE
##                     time
##                      TRUE
```

```
# Code to conduct exhaustive search
model2 = glm(death_event~age+ejection_fraction+serum_creatinine+smoking+time,data=dataTrain)
summary(model2)
##
## Call:
## glm(formula = death_event ~ age + ejection_fraction + serum_creatinine +
##     smoking + time, data = dataTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.77537  -0.25471  -0.02714   0.22155   0.82881
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.6151627  0.1519677   4.048 7.02e-05 ***
## age                0.0053976  0.0020326   2.656  0.00846 **
## ejection_fraction -0.0111449  0.0019084  -5.840 1.74e-08 ***
## serum_creatinine   0.1006309  0.0212868   4.727 3.93e-06 ***
## smoking           -0.0143284  0.0480409  -0.298  0.76577
## time              -0.0026772  0.0003008  -8.901  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1217439)
##
##     Null deviance: 50.400  on 239  degrees of freedom
## Residual deviance: 28.488  on 234  degrees of freedom
## AIC: 183.61
##
## Number of Fisher Scoring iterations: 2
```

Responses to question (3a)

Variables selected: Based on our results explained above, the variables used for model2 are the following: Age, Ejection_fraction, Serum_creatinine, Smoking and Time

(3b) 3pts - Conduct forward-backward stepwise regression using AIC. Allow the minimum model to be the model with only an intercept, and the full model to be *model1*. Call it *model3*. Display the summary table for the model. Which variables are in your *model3*?

```
#  Code to conduct stepwise regression
set.seed(100)
# Create the minimum model
minmod = lm(death_event~1, data = dataTrain)
```

```r
# Step Forward from the minimum model using AIC as the complexity penalty
model3 = step(model1, scope = list(lower = minmod, upper = model1),
direction = "both", k=2, trace=F)
# Show the optimal model
summary(model3)
##
## Call:
## glm(formula = death_event ~ age + creatinine_phosphokinase +
##     ejection_fraction + serum_creatinine + time, data = dataTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.77865  -0.25726  -0.03205   0.22466   0.84236
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.652e-01  1.515e-01   3.730  0.00024 ***
## age                       5.699e-03  2.028e-03   2.810  0.00537 **
## creatinine_phosphokinase  4.284e-05  2.472e-05   1.733  0.08436 .
## ejection_fraction        -1.102e-02  1.887e-03  -5.840 1.73e-08 ***
## serum_creatinine          9.946e-02  2.117e-02   4.699 4.46e-06 ***
## time                     -2.670e-03  2.989e-04  -8.930  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1202463)
##
##     Null deviance: 50.400  on 239  degrees of freedom
## Residual deviance: 28.138  on 234  degrees of freedom
## AIC: 180.64
##
## Number of Fisher Scoring iterations: 2
```

Responses to question (3b):

Variables selected: Based on the specification of forward-backward in the question, we will set the direction to both. Meaning it will conduct forward and backward stepwise regression at the same time.

Based on our summary output of our model, we can see that the variables selected from the optimal model are: age, creatinine_phosphokinase, ejection_fraction, serum_creatinine and time.

(3c) 8pts - Conduct Lasso regression on the train data set (*dataTrain*). Use *death_event* as the response, and all other variables as the predicting variables. Use 10-fold cross validation on the *classification error* to select the optimal lambda value.

(3c.1) What is the optimal lambda value?

(3c.2) Display the estimated coefficients at the optimal lambda value.

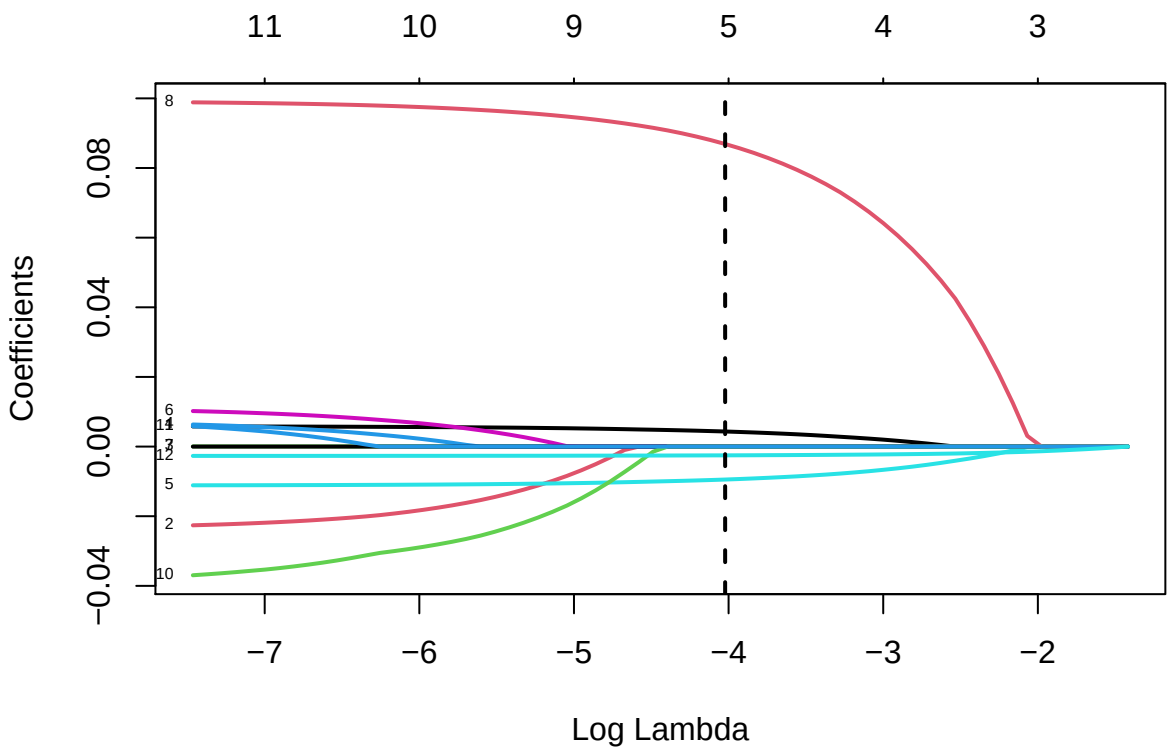
(3c.3) Which variables were selected?

(3c.4) Plot the paths of the lasso coefficients. Which variable entered the model first?

(3c.5) Fit a logistic regression model with *death_event* as the response variable and the variables selected from lasso regression as predicting variables. Include an intercept. Call it *model4*. Display the summary table for the model.

```r
# Setting the seed (please do not change)
set.seed(0)

# Code to conduct 10-fold CV and find optimal lambda
cv.lasso = cv.glmnet(as.matrix(dataTrain[,-13]), dataTrain[,13],
family='gaussian', alpha=1, nfolds=10)
cat("CV Optimized lambda:\n")
## CV Optimized lambda:
cat(c(cv.lasso$lambda.min,"\n"))
## 0.0179104287105811

# Display coefficients at optimal lambda (do not output anything else)
set.seed(100)
lasso.mod = glmnet(as.matrix(dataTrain[,-13]), dataTrain[,13],
family='gaussian', alpha=1)
# Extract coefficents at optimal lambda
cat("Coefficients at Optimal Lambda:\n")
## Coefficients at Optimal Lambda:
coef(lasso.mod, s = cv.lasso$lambda.min)
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                      1
## (Intercept)               5.980521e-01
## age                       4.328428e-03
## anaemia                   .
## creatinine_phosphokinase  2.249997e-05
## diabetes                  .
## ejection_fraction        -9.465683e-03
## high_blood_pressure       .
## platelets                 .
## serum_creatinine          8.689876e-02
## serum_sodium              .
## sex                       .
## smoking                   .
## time                     -2.520091e-03
```

```r
# Plot of coefficent path
plot(lasso.mod,xvar="lambda",label=TRUE,lwd=2)
abline(v=log(cv.lasso$lambda.min),col='black',lty = 2,lwd=2)
```




```
# Code to fit model4 and display summary
model4 <- glm(death_event ~ age + creatinine_phosphokinase + ejection_fraction + serum_creatinine + tim
summary(model4)
##
## Call:
## glm(formula = death_event ~ age + creatinine_phosphokinase +
##     ejection_fraction + serum_creatinine + time, data = dataTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.77865  -0.25726  -0.03205   0.22466   0.84236
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              5.652e-01  1.515e-01   3.730  0.00024 ***
## age                      5.699e-03  2.028e-03   2.810  0.00537 **
## creatinine_phosphokinase 4.284e-05  2.472e-05   1.733  0.08436 .
## ejection_fraction       -1.102e-02  1.887e-03  -5.840 1.73e-08 ***
## serum_creatinine         9.946e-02  2.117e-02   4.699 4.46e-06 ***
## time                    -2.670e-03  2.989e-04  -8.930  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1202463)
##
##     Null deviance: 50.400  on 239  degrees of freedom
```

```
## Residual deviance: 28.138  on 234  degrees of freedom
## AIC: 180.64
##
## Number of Fisher Scoring iterations: 2
```

Responses to question (3c)

(3c.1) Optimal lambda value: 0.0179104287105811 **(3c.3) Variables selected:** Same as model3, the age, creatinine_phosphokinase, ejection_fraction, serum_creatinine and time variables.

(3c.4) Variable that entered the model first? Based on our results of the coefficient paths, we can see that the time variable is the first to enter the model.

Question 4: Model Comparison and Prediction

(4a) 3pts - Compare the AICs, and BICs of *model1*, *model2* and *model3*. Which model is preferred based on these criteria and why?

```
# Code to calculate AICs and BICs
set.seed(100)
n = length(dataTrain$death_event)
comp = rbind(Model1=
               c(AIC(model1,k=2), AIC(model1,k=log(n))),

             Model2=c(AIC(model2,k=2), AIC(model2,k=log(n))),

             Model3=c(AIC(model3,k=2), AIC(model3,k=log(n))))
colnames(comp) = c("AIC", "BIC")
comp
##              AIC      BIC
## Model1 193.3643 242.0932
## Model2 183.6136 207.9781
## Model3 180.6431 205.0076
```

Response to question (4a)

Interpretation: Based on the above output, we can see that Model3 is the preferred model because it contains the lowest AIC and BIC when compared to model1 and model2.

(4b) 4pts - Using *model2*, and *model3*, give a binary classification to each of the test rows in *dataTest*, with 1 indicating a heart failure patient dying during their follow-up period. Use 0.5 as your classification threshold. What is the classification error rate over these data points for each model?

```
# Code to estimate binary classification and classification error rate for model2
set.seed(100)

cost0.5 = function(y, pi){
 ypred=rep(0,length(y))
 ypred[pi>0.5] = 1
 err = mean(abs(y-ypred))
 return(err)
}

## classification error for 10-fold cross-validation
```

```
cv.err_model2 = cv.glm(dataTest,model2,cost=cost0.5, K=10)$delta[1]
cr.err_model3 = cv.glm(dataTest,model3,cost=cost0.5, K=10)$delta[1]
cv.err_model2
## [1] 0.6101695
cr.err_model3
## [1] 0.5932203
```

Response to question (4b)

Model2 classification error rate: 0.6101695

Model3 classification error rate: 0.5932203

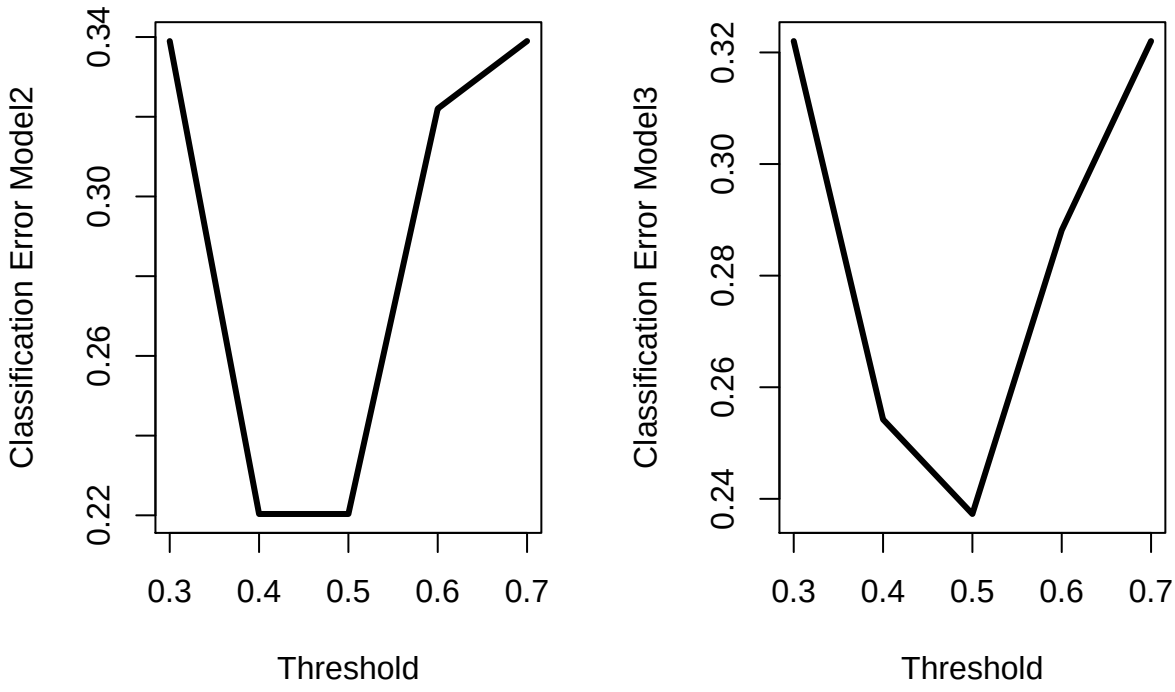
(4c) 4pts - Using *model2* only, provide the prediction accuracy rates for the test data *dataTest* using the following classification thresholds: 0.3, 0.4, 0.5, 0.6, 0.7.

(4c.1) Provide a plot of the thresholds versus the prediction accuracy rates.

(4c.2) Which threshold(s) result(s) in the highest prediction accuracy rate(s)?

```
# Code for calculating prediction accuracy rates for different thresholds
cost0.3 = function(y, pi){
 ypred=rep(0,length(y))
 ypred[pi>0.3] = 1
 err = mean(abs(y-ypred))
 return(err)
}

cost0.4 = function(y, pi){
 ypred=rep(0,length(y))
 ypred[pi>0.4] = 1
 err = mean(abs(y-ypred))
 return(err)
}

cost0.6 = function(y, pi){
 ypred=rep(0,length(y))
 ypred[pi>0.6] = 1
 err = mean(abs(y-ypred))
 return(err)
}
cost0.7 = function(y, pi){
 ypred=rep(0,length(y))
 ypred[pi>0.7] = 1
 err = mean(abs(y-ypred))
 return(err)
}

## Prediction given a set of new observations
pred_model2 = predict.glm(model2,dataTest[-13],type="response")
pred_model3 = predict.glm(model3,dataTest[-13],type="response")

err0.3 = cost0.3(dataTest$death_event,pred_model2)
err0.4 = cost0.4(dataTest$death_event,pred_model2)
err0.5 = cost0.5(dataTest$death_event,pred_model2)
err0.6 = cost0.6(dataTest$death_event,pred_model2)
```

```r
err0.7 = cost0.7(dataTest$death_event,pred_model2)
err_model2 = c(err0.3,err0.4,err0.5,err0.6,err0.7)

err0.3_1 = cost0.3(dataTest$death_event,pred_model3)
err0.4_1 = cost0.4(dataTest$death_event,pred_model3)
err0.5_1 = cost0.5(dataTest$death_event,pred_model3)
err0.6_1 = cost0.6(dataTest$death_event,pred_model3)
err0.7_1 = cost0.7(dataTest$death_event,pred_model3)
err_model3 = c(err0.3_1,err0.4_1,err0.5_1,err0.6_1,err0.7_1)



# Plot of the classification threshold vs. prediction accuracy rates.
par(mfrow=c(1,2))
plot(c(0.3,0.4,0.5,0.6,0.7),err_model2,
     type="l",lwd=3,xlab="Threshold",ylab="Classification Error Model2")
plot(c(0.3,0.4,0.5,0.6,0.7),err_model3,
     type="l",lwd=3,xlab="Threshold",ylab="Classification Error Model3")
```



Responses to question (4c)

Threshold(s) with the highest prediction accuracy rate(s):

From our results we can see that model2 has the best prediction accuracy for either a threshold value of 0.4 or 0.5 and model3 has the highest prediction accuracy at a thresold = 0.5

Question 5: Goodness of fit

For residual analysis, we need replications. We will provide the code for you to aggregate the data across the variables *anaemia*, *diabetes*, *high_blood_pressure*, *sex*, and *smoking*.

```
# Aggregate data and display new model summary
data.agg.n = aggregate(death_event~anaemia+diabetes+high_blood_pressure+sex+smoking,
                       data=dataTrain,FUN=length)
data.agg.y = aggregate(death_event~anaemia+diabetes+high_blood_pressure+sex+smoking,
                       data=dataTrain,FUN=sum)
data.agg = cbind(data.agg.y,total=data.agg.n$death_event)
head(data.agg,1)
##   anaemia diabetes high_blood_pressure sex smoking death_event total
## 1       0        0                   0   0       0           4    11
```

(5a) 2pts - Fit a logistic regression model with this aggregated data called *model5*. Use all five predicting variables. Include an intercept. Remember the replications. Display the summary table for the model.

```
# Code for fitting the model here
model5 <- glm(data.agg,family="binomial")
summary(model5)
##
## Call:
## glm(formula = data.agg, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6559  -1.0545   0.3773   0.9853   1.6380
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)            4.1514     2.5008   1.660   0.0969 .
## diabetes              -0.6434     0.9845  -0.653   0.5134
## high_blood_pressure   -1.9769     1.3128  -1.506   0.1321
## sex                    0.1971     0.9377   0.210   0.8335
## smoking               -0.7504     1.1178  -0.671   0.5020
## death_event            0.2022     0.4124   0.490   0.6238
## total                 -0.3529     0.2244  -1.572   0.1159
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 37.393  on 26  degrees of freedom
## Residual deviance: 32.415  on 20  degrees of freedom
## AIC: 46.415
##
## Number of Fisher Scoring iterations: 5
```

(5b) 3pts - Using model5, conduct a goodness-of-fit hypothesis test using the Pearson residuals. What do you conclude at the alpha level of 0.05? Explain.

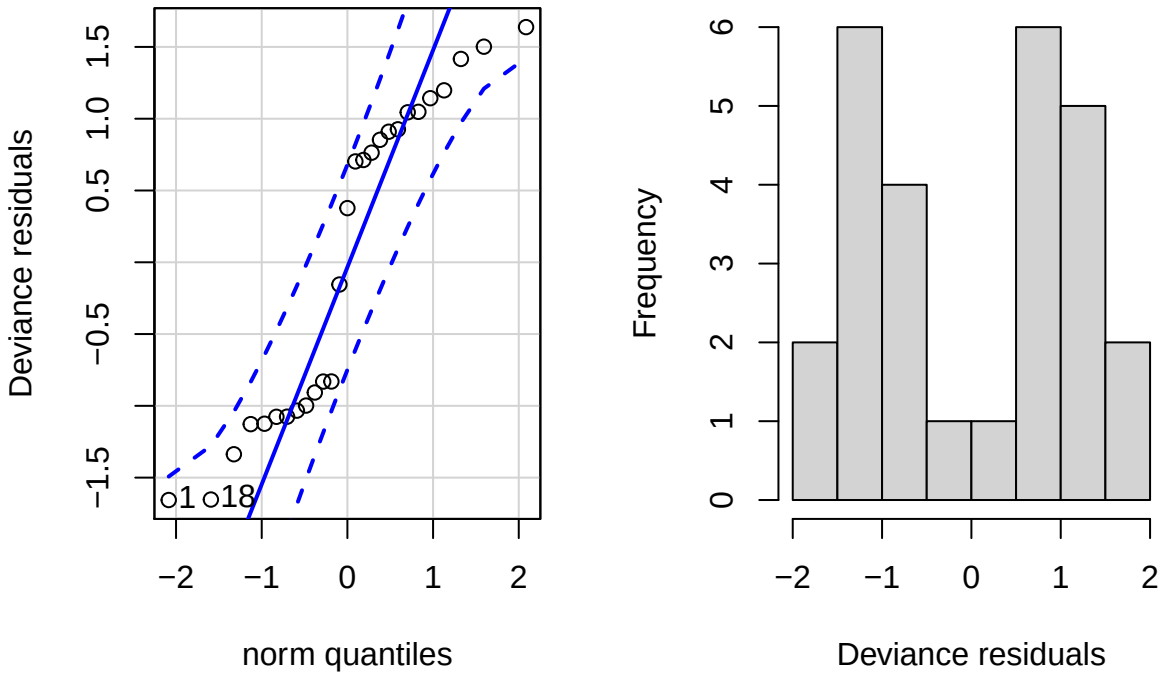
```r
# Code for GOF test. P-value must be displayed
# Pearson residuals test
pResid <- resid(model5, type = "pearson")
cat("Pearson residuals test p-value:",
1-pchisq(sum(pResid^2), model5$df.residual))
## Pearson residuals test p-value: 0.170152
```

Response to question (5b)

Interpretation: The p-values for the goodness of fit test > alpha 0.5, suggesting that we fail to reject the null hypothesis that the model is a good fit. Meaning our model does not fit the data well.

(5c) 2pts - Evaluate whether the deviance residuals are normally distributed by producing a QQ plot of the residuals. What assessment can you make about the goodness of fit of *model5* based on these plots?

```r
#All code needed for 5c
res = resid(model5,type="deviance")

# QQ-plot and Histogram
par(mfrow=c(1,2))
qqPlot(res, ylab="Deviance residuals")
## [1]  1 18
hist(res,10,xlab="Deviance residuals", main="")
```



Response to question (5c)

Interpretation:

Based on our results, we see that the residuals clearly do not follow a normal distribution, as our deviance residuals more closely resemble a bimodal distribution. We can also see from the qq plot that our data does not follow closely to our center line, and is going in and out of the center line several times throughout the graph, further backing up histogram result.

(5d) 2pts - Using *model5*, estimate the overdispersion parameter. What do you conclude?

```
#All code needed for 5d
model5$deviance/model5$df.res
## [1] 1.620745
```

Responses to question (5d)

Overdispersion parameter: 1.620745

Interpretation: We can conclude that there is not overdispersion in our model. I believe we're not exhibiting this because it's been clearly shown throughout the model that we don't experience any multicollinearity, based on the results of our correlation plot and VIF calculation. This can cause overdispersion in addition to heterogenity in the success probablity that hasn't been modeled.

(5e) 2pts - Why might a logistic regression model not be a good fit? Provide two reasons. How can you try to improve the fit in each situation? *Note: This is a general question and is not related to question 5b.*

Responses to question (5e)

Reason 1: One reason may be that the link function used in our case (logit) simply may not fit our data well.

How can you try to improve the fit? In our case the model may be improved by switching the link function that is used to fit the model such as changing the function from logit to cloglog. This may help reduce the deviance and leader to a better fitting model.

Reason 2: A second reason we may be getting bad results is due to outlier / influential observations throughout each variable.

How can you try to improve the fit? We can try and alleviate this by conducting a Cook's test on each of the impacted variables and removes outliers. After removing outliers and refitting the model, we can determine if this provides a better fit to our model.

Providing 3rd reason. **Reason 2:** Another reason we might not be getting a food fit is that there's likely additional variables that we're excluding which could provide explanatory power against our death_event response variable.

How can you try to improve the fit? We can try and use different combinations of factor's by running a boxcox transformation on each variable to determine what transformation may be applied to improve against the assumptions (i.e. Linearity, Independence, Constant V, etc.). In addition we could also provide interaction terms to see if this also helps.

This is the End of Final Exam Part 2

We hope you enjoyed the course - and we wish you the best in your future coursework!