Final\_Project\_Shields

Brandon Lloyd Shields

12/6/2021

#load packages needed for analysis  
  
library(dplyr)

## Registered S3 methods overwritten by 'tibble':  
## method from   
## format.tbl pillar  
## print.tbl pillar

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(keras)  
library(ggplot2)  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(Hmisc)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

library(corrplot)

## corrplot 0.84 loaded

#Motivation  
  
#As I am begining to think about possible career transistion after completing my MSBA degree, I thought about opular industries in Northeast Ohio. The first thing that came to mind was the medical industry with world class hospitals in both Cleveland and Akron. Online job boards further indicate a variety of analyst posistions in this industry across the region. It was for this reason that I was motivated to work with medical data.  
  
#Data Source  
  
# I will be working with a data set from Kaggle (https://www.kaggle.com/andrewmvd/heart-failure-clinical-data/code).   
  
#Dataset from Davide Chicco, Giuseppe Jurman: â€œMachine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20, 16 (2020)  
  
#Cardiovascular disease (CVD) is the number one reason for death globally. There are a variety of behavioral and risk factors that can be used to predict mortality of heart of failure. Having an algorithm that can predict mortality could serve as early detection for high risk patients. The dataset consists of 12 explanatory variables and the target variable (death event)  
  
cvd <- read.csv(file = "~/Downloads/heart\_failure\_clinical\_records\_dataset.csv")

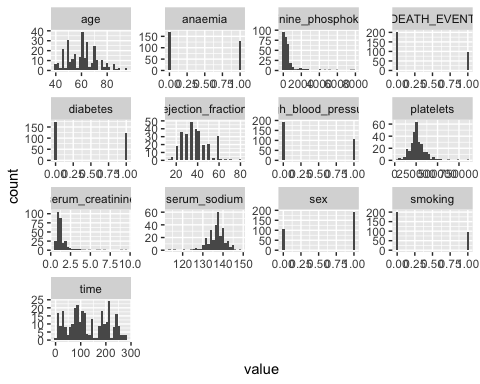
summary(cvd)

## age anaemia creatinine\_phosphokinase  
## Min. :40.00 Min. :0.0000 Min. : 23.0   
## 1st Qu.:51.00 1st Qu.:0.0000 1st Qu.: 116.5   
## Median :60.00 Median :0.0000 Median : 250.0   
## Mean :60.83 Mean :0.4314 Mean : 581.8   
## 3rd Qu.:70.00 3rd Qu.:1.0000 3rd Qu.: 582.0   
## Max. :95.00 Max. :1.0000 Max. :7861.0   
## diabetes ejection\_fraction high\_blood\_pressure platelets   
## Min. :0.0000 Min. :14.00 Min. :0.0000 Min. : 25100   
## 1st Qu.:0.0000 1st Qu.:30.00 1st Qu.:0.0000 1st Qu.:212500   
## Median :0.0000 Median :38.00 Median :0.0000 Median :262000   
## Mean :0.4181 Mean :38.08 Mean :0.3512 Mean :263358   
## 3rd Qu.:1.0000 3rd Qu.:45.00 3rd Qu.:1.0000 3rd Qu.:303500   
## Max. :1.0000 Max. :80.00 Max. :1.0000 Max. :850000   
## serum\_creatinine serum\_sodium sex smoking   
## Min. :0.500 Min. :113.0 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.900 1st Qu.:134.0 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1.100 Median :137.0 Median :1.0000 Median :0.0000   
## Mean :1.394 Mean :136.6 Mean :0.6488 Mean :0.3211   
## 3rd Qu.:1.400 3rd Qu.:140.0 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :9.400 Max. :148.0 Max. :1.0000 Max. :1.0000   
## time DEATH\_EVENT   
## Min. : 4.0 Min. :0.0000   
## 1st Qu.: 73.0 1st Qu.:0.0000   
## Median :115.0 Median :0.0000   
## Mean :130.3 Mean :0.3211   
## 3rd Qu.:203.0 3rd Qu.:1.0000   
## Max. :285.0 Max. :1.0000

#The summary below shows that there are no missing valuables for any of the 13 variables. Some general information that can be gleaned from the summary statitsics include ages ranging from 40 to 95, the average time for follow-up was 130 days.  
  
#age = age  
#anameia = Decrease of red blood cells or hemoglobin (boolean)  
#creatiine\_phosphokinase = Level of the CPK enzyme in the blood (mcg/L)  
#diabetes = f the patient has diabetes (boolean)  
#ejection\_fraction = Percentage of blood leaving the heart at each contraction (percentage)  
#high\_blood\_pressure = If the patient has hypertension (boolean)  
#platelets = Platelets in the blood (kiloplatelets/mL)  
#serum\_creatinine = Level of serum creatinine in the blood (mg/dL)  
#serum\_sodium = Level of serum sodium in the blood (mEq/L)  
#sex = Woman or man (binary)  
#smoking = If the patient smokes or not (boolean)  
#time = follow-up period (days)  
#death\_event = If the patient deceased during the follow-up period (boolean)

#Histogram of Values  
  
h\_grams <- cvd %>% gather() %>%  
 ggplot(aes(value)) +   
 facet\_wrap(~key, scales = "free") +  
 geom\_histogram()  
  
h\_grams

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



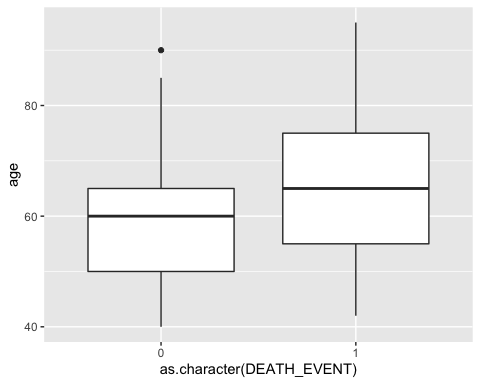
#Histograms below paint a clearer picture of the data. We can see there are some outliers in creatine\_phosphokinase and serum creatinine. While some numeric variables follow a gausician distribution, time is the most clear example of one that does not.

#Look to see if there is any correlation betwen variables, paying significant attention to correlation between the target and predictor variables.   
  
corr\_cvd <- rcorr(as.matrix(cvd))  
  
flattenCorrMatrix <- function(cormat, pmat) {  
 ut <- upper.tri(cormat)  
 data.frame(  
 row = rownames(cormat)[row(cormat)[ut]],  
 column = rownames(cormat)[col(cormat)[ut]],  
 cor =(cormat)[ut],  
 p = pmat[ut]  
 )  
}  
  
flattenCorrMatrix(corr\_cvd$r, corr\_cvd$P)

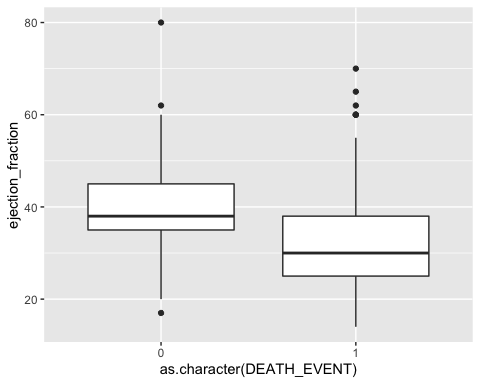
## row column cor  
## 1 age anaemia 0.088006441  
## 2 age creatinine\_phosphokinase -0.081583900  
## 3 anaemia creatinine\_phosphokinase -0.190741030  
## 4 age diabetes -0.101012385  
## 5 anaemia diabetes -0.012729046  
## 6 creatinine\_phosphokinase diabetes -0.009638514  
## 7 age ejection\_fraction 0.060098363  
## 8 anaemia ejection\_fraction 0.031556971  
## 9 creatinine\_phosphokinase ejection\_fraction -0.044079554  
## 10 diabetes ejection\_fraction -0.004850310  
## 11 age high\_blood\_pressure 0.093288685  
## 12 anaemia high\_blood\_pressure 0.038182003  
## 13 creatinine\_phosphokinase high\_blood\_pressure -0.070589980  
## 14 diabetes high\_blood\_pressure -0.012732382  
## 15 ejection\_fraction high\_blood\_pressure 0.024444731  
## 16 age platelets -0.052354367  
## 17 anaemia platelets -0.043785550  
## 18 creatinine\_phosphokinase platelets 0.024463389  
## 19 diabetes platelets 0.092192828  
## 20 ejection\_fraction platelets 0.072177466  
## 21 high\_blood\_pressure platelets 0.049963481  
## 22 age serum\_creatinine 0.159187133  
## 23 anaemia serum\_creatinine 0.052173604  
## 24 creatinine\_phosphokinase serum\_creatinine -0.016408480  
## 25 diabetes serum\_creatinine -0.046975315  
## 26 ejection\_fraction serum\_creatinine -0.011302475  
## 27 high\_blood\_pressure serum\_creatinine -0.004934525  
## 28 platelets serum\_creatinine -0.041198077  
## 29 age serum\_sodium -0.045965841  
## 30 anaemia serum\_sodium 0.041881610  
## 31 creatinine\_phosphokinase serum\_sodium 0.059550156  
## 32 diabetes serum\_sodium -0.089550619  
## 33 ejection\_fraction serum\_sodium 0.175902282  
## 34 high\_blood\_pressure serum\_sodium 0.037109470  
## 35 platelets serum\_sodium 0.062124619  
## 36 serum\_creatinine serum\_sodium -0.189095210  
## 37 age sex 0.065429524  
## 38 anaemia sex -0.094768961  
## 39 creatinine\_phosphokinase sex 0.079790629  
## 40 diabetes sex -0.157729504  
## 41 ejection\_fraction sex -0.148385965  
## 42 high\_blood\_pressure sex -0.104614629  
## 43 platelets sex -0.125120483  
## 44 serum\_creatinine sex 0.006969778  
## 45 serum\_sodium sex -0.027566123  
## 46 age smoking 0.018667868  
## 47 anaemia smoking -0.107289838  
## 48 creatinine\_phosphokinase smoking 0.002421235  
## 49 diabetes smoking -0.147173413  
## 50 ejection\_fraction smoking -0.067314567  
## 51 high\_blood\_pressure smoking -0.055711369  
## 52 platelets smoking 0.028234448  
## 53 serum\_creatinine smoking -0.027414135  
## 54 serum\_sodium smoking 0.004813195  
## 55 sex smoking 0.445891712  
## 56 age time -0.224068420  
## 57 anaemia time -0.141413982  
## 58 creatinine\_phosphokinase time -0.009345653  
## 59 diabetes time 0.033725509  
## 60 ejection\_fraction time 0.041729235  
## 61 high\_blood\_pressure time -0.196439479  
## 62 platelets time 0.010513909  
## 63 serum\_creatinine time -0.149315418  
## 64 serum\_sodium time 0.087640000  
## 65 sex time -0.015608220  
## 66 smoking time -0.022838942  
## 67 age DEATH\_EVENT 0.253728543  
## 68 anaemia DEATH\_EVENT 0.066270098  
## 69 creatinine\_phosphokinase DEATH\_EVENT 0.062728160  
## 70 diabetes DEATH\_EVENT -0.001942883  
## 71 ejection\_fraction DEATH\_EVENT -0.268603312  
## 72 high\_blood\_pressure DEATH\_EVENT 0.079351058  
## 73 platelets DEATH\_EVENT -0.049138868  
## 74 serum\_creatinine DEATH\_EVENT 0.294277561  
## 75 serum\_sodium DEATH\_EVENT -0.195203596  
## 76 sex DEATH\_EVENT -0.004316376  
## 77 smoking DEATH\_EVENT -0.012623153  
## 78 time DEATH\_EVENT -0.526963779  
## p  
## 1 1.289270e-01  
## 2 1.593814e-01  
## 3 9.167881e-04  
## 4 8.119152e-02  
## 5 8.264999e-01  
## 6 8.681796e-01  
## 7 3.003041e-01  
## 8 5.867713e-01  
## 9 4.476233e-01  
## 10 9.334390e-01  
## 11 1.074267e-01  
## 12 5.107290e-01  
## 13 2.235963e-01  
## 14 8.264552e-01  
## 15 6.737687e-01  
## 16 3.669939e-01  
## 17 4.506610e-01  
## 18 6.735340e-01  
## 19 1.116376e-01  
## 20 2.133299e-01  
## 21 3.893106e-01  
## 22 5.803433e-03  
## 23 3.686529e-01  
## 24 7.775144e-01  
## 25 4.183290e-01  
## 26 8.456860e-01  
## 27 9.322860e-01  
## 28 4.778909e-01  
## 29 4.284113e-01  
## 30 4.706118e-01  
## 31 3.047425e-01  
## 32 1.223234e-01  
## 33 2.267684e-03  
## 34 5.226830e-01  
## 35 2.842728e-01  
## 36 1.017081e-03  
## 37 2.593824e-01  
## 38 1.019385e-01  
## 39 1.687830e-01  
## 40 6.274077e-03  
## 41 1.018969e-02  
## 42 7.086549e-02  
## 43 3.054273e-02  
## 44 9.044710e-01  
## 45 6.349606e-01  
## 46 7.478516e-01  
## 47 6.391170e-02  
## 48 9.667444e-01  
## 49 1.083125e-02  
## 50 2.458783e-01  
## 51 3.370299e-01  
## 52 6.267749e-01  
## 53 6.368284e-01  
## 54 9.339471e-01  
## 55 4.440892e-16  
## 56 9.303199e-05  
## 57 1.439130e-02  
## 58 8.721501e-01  
## 59 5.613138e-01  
## 60 4.722292e-01  
## 61 6.357867e-04  
## 62 8.563315e-01  
## 63 9.720948e-03  
## 64 1.305338e-01  
## 65 7.881011e-01  
## 66 6.940836e-01  
## 67 8.916763e-06  
## 68 2.532988e-01  
## 69 2.796112e-01  
## 70 9.733118e-01  
## 71 2.452897e-06  
## 72 1.711495e-01  
## 73 3.971942e-01  
## 74 2.190198e-07  
## 75 6.889112e-04  
## 76 9.407519e-01  
## 77 8.279207e-01  
## 78 0.000000e+00

#The results of the corrleation table shows that correlation is significant at the .005 level between Death\_Event and age, ejection\_fraction, serum\_creatinine and serum\_sodium.

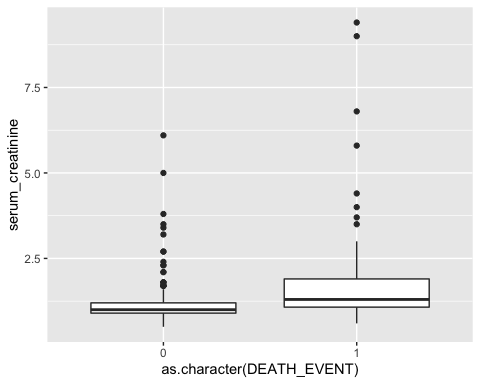
#Building Box Plots  
de\_age <- ggplot(data = cvd, aes(x = as.character(DEATH\_EVENT), y = age)) +   
 geom\_boxplot()  
  
de\_ejection <- ggplot(data = cvd, aes(x = as.character(DEATH\_EVENT), y = ejection\_fraction)) +   
 geom\_boxplot()  
  
de\_serum\_cre <- ggplot(data = cvd, aes(x = as.character(DEATH\_EVENT), y = serum\_creatinine)) +   
 geom\_boxplot()  
  
de\_serum\_sod <- ggplot(data = cvd, aes(x = as.character(DEATH\_EVENT), y = serum\_sodium)) +   
 geom\_boxplot()  
  
#Displaying Box Plots  
de\_age



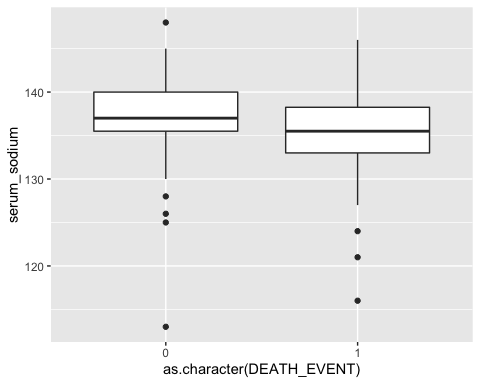
de\_ejection



de\_serum\_cre



de\_serum\_sod



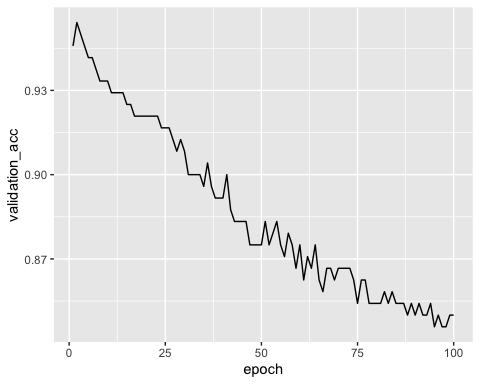
#Model Creation - One Layer - 64 Units  
model\_1 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 64, activation = 'relu', input\_shape = c(12)) %>% #input shape = features  
 layer\_dense( units = 1, activation = 'sigmoid')

## Loaded Tensorflow version 2.6.2

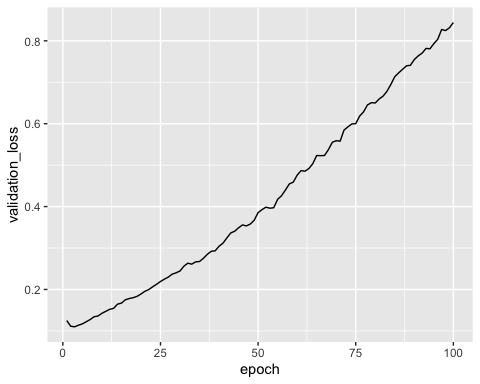
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_1 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_1 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 100, batch\_size = 1, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2   
## processing fold # 3   
## processing fold # 4

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()

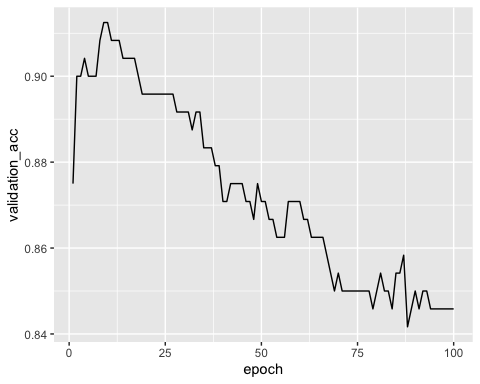


# Overfitting starts occuring after the second epoch as evidence by validation loss and validation accuracy. 64 units may be too many. Lets adjust with one layer and fewer units.

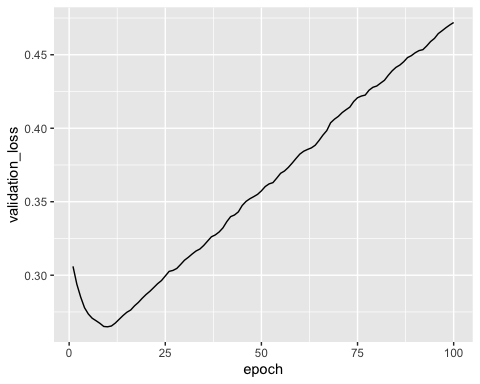
#Model Creation - One layer - 8 Units  
model\_2 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 8, activation = 'relu', input\_shape = c(12)) %>% #input shape = features  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_2 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
   
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
   
 history <- model\_2 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 100, batch\_size = 1, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2   
## processing fold # 3   
## processing fold # 4

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()

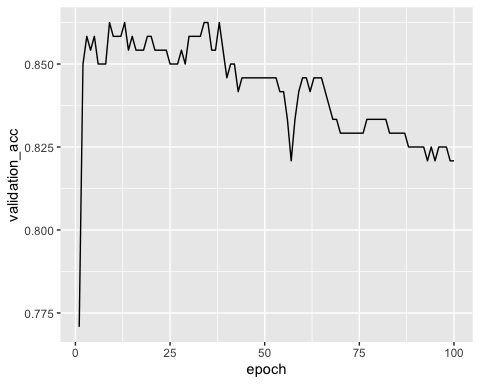


#Still overfitting - will add a second layer of 8 units and along with one dropout layer to reduce overfitting

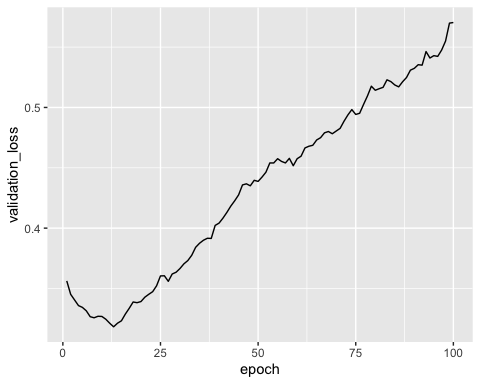
#Model Creation - Two Layer - 8 Units - dropout  
model\_3 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 8, activation = 'relu', input\_shape = c(12)) %>% #input shape = features  
 layer\_dropout(rate = 0.5) %>%  
 layer\_dense(units = 8, activation = 'relu') %>%  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_3 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_3 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 100, batch\_size = 1, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2   
## processing fold # 3   
## processing fold # 4

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()



#Model Creation - Two Layer - 8 Units - one dropout = call backs   
model\_4 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 8, activation = 'relu', input\_shape = c(12)) %>%#input shape = features  
 layer\_dropout(rate = 0.5) %>%  
 layer\_dense(units = 8, activation = 'relu') %>%  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
callbacks\_list <- list(  
 callback\_early\_stopping(  
 monitor = "val\_accuracy",  
 patience = 1  
),  
  
callback\_model\_checkpoint(  
 filepath = "my\_model.h4",  
 monitor = "val\_loss",  
 save\_best\_only = TRUE  
) )  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_4 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_4 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 100, batch\_size = 1, callbacks = callbacks\_list, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

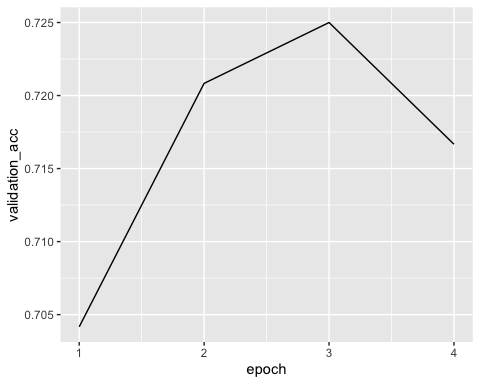
## processing fold # 1   
## processing fold # 2

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

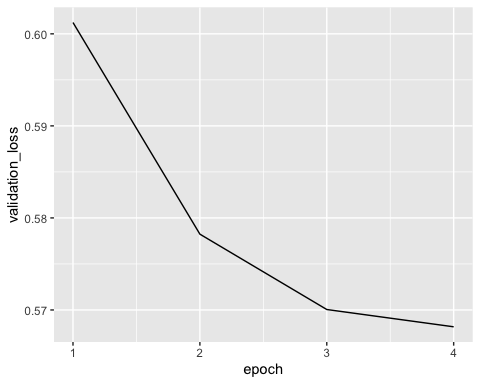
## Warning in rbind(all\_loss\_histories, loss\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

## processing fold # 3   
## processing fold # 4

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()



#Model Creation - Two Layer - 16/8 Units - one dropout = call backs   
model\_5 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 16, activation = 'relu', input\_shape = c(12)) %>%#input shape = features  
 layer\_dropout(rate = 0.5) %>%  
 layer\_dense(units = 8, activation = 'relu') %>%  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
callbacks\_list <- list(  
 callback\_early\_stopping(  
 monitor = "val\_accuracy",  
 patience = 1  
),  
  
callback\_model\_checkpoint(  
 filepath = "my\_model.h4",  
 monitor = "val\_loss",  
 save\_best\_only = TRUE  
) )  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_5 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_5 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 100, batch\_size = 1, callbacks = callbacks\_list, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2

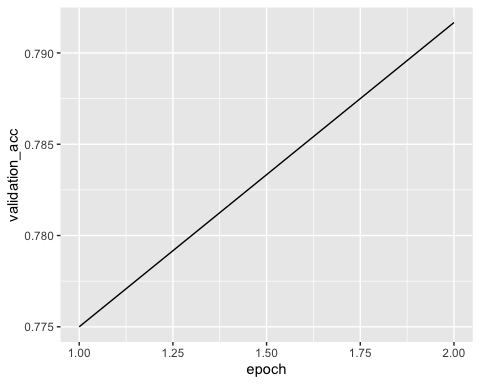
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

## Warning in rbind(all\_loss\_histories, loss\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

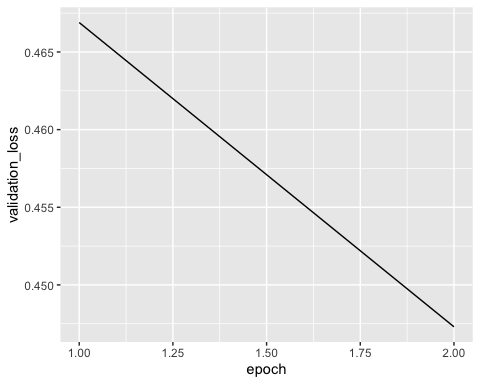
## processing fold # 3   
## processing fold # 4

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)  
  
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()



#Model Creation - Two Layer - 16 Units - one dropout = call backs   
model\_6 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 16, activation = 'relu', input\_shape = c(12)) %>%#input shape = features  
 layer\_dropout(rate = 0.5) %>%  
 layer\_dense(units = 16, activation = 'relu') %>%  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
callbacks\_list <- list(  
 callback\_early\_stopping(  
 monitor = "val\_accuracy",  
 patience = 1  
),  
  
callback\_model\_checkpoint(  
 filepath = "my\_model.h4",  
 monitor = "val\_loss",  
 save\_best\_only = TRUE  
) )  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_6 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_6 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 25, batch\_size = 1, callbacks = callbacks\_list, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

## Warning in rbind(all\_loss\_histories, loss\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

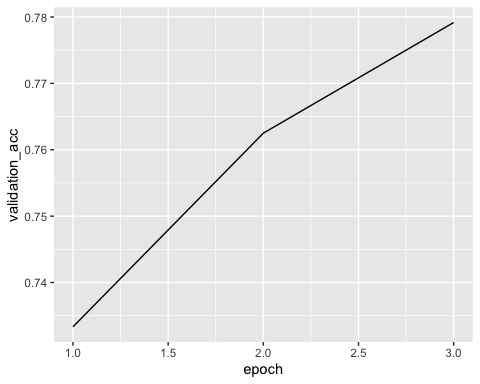
## processing fold # 3

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)  
  
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

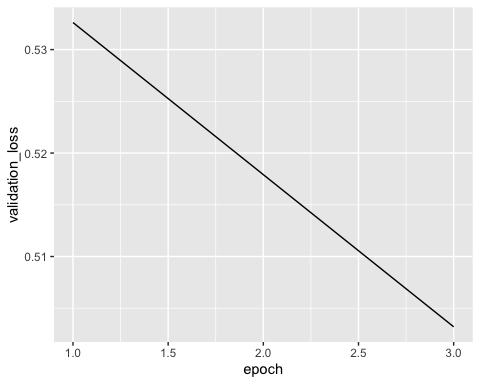
## processing fold # 4

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)  
  
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()



#Model Creation - Two Layer - 32/8 Units - one dropout = call backs   
model\_7 <- keras\_model\_sequential() %>%  
 layer\_dense(units = 32, activation = 'relu', input\_shape = c(12)) %>%#input shape = features  
 layer\_dropout(rate = 0.5) %>%  
 layer\_dense(units = 8, activation = 'relu') %>%  
 layer\_dense( units = 1, activation = 'sigmoid')  
  
callbacks\_list <- list(  
 callback\_early\_stopping(  
 monitor = "val\_accuracy",  
 patience = 3  
),  
  
callback\_model\_checkpoint(  
 filepath = "my\_model.h4",  
 monitor = "val\_loss",  
 save\_best\_only = TRUE  
) )  
  
#I will use relu for the activation function  
#I will use sigmoid activation since this is binary classification problem.   
model\_7 %>% compile(  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = c("accuracy")  
)  
  
#Setting up empty vectors to hold results from folds  
all\_acc\_histories <- NULL  
all\_loss\_histories <- NULL  
  
#Specifications for k-fold validation  
k <- 4  
indices <- sample(1:nrow(train\_X.norm))  
folds <- cut(indices, breaks = k, labels = FALSE)  
  
#Model Function  
  
for (i in 1:k) {  
 cat("processing fold #", i, "\n")  
 val\_indices <- which(folds == i, arr.ind = TRUE)  
 val\_data <- as.matrix(train\_X.norm[val\_indices,])  
 val\_targets <- as.matrix(train\_y[val\_indices,])  
 partial\_train\_data <- as.matrix(train\_X.norm[-val\_indices,])  
 partial\_train\_targets <- as.matrix(train\_y[-val\_indices,])  
  
 history <- model\_7 %>% fit(  
 partial\_train\_data, partial\_train\_targets,  
 validation\_data = list(val\_data, val\_targets),  
 epochs = 25, batch\_size = 1, callbacks = callbacks\_list, verbose = 0)  
   
 acc\_history <- history$metrics$val\_accuracy  
 all\_acc\_histories <- rbind(all\_acc\_histories, acc\_history)  
   
 loss\_history <-history$metrics$val\_loss  
 all\_loss\_histories <- rbind(all\_loss\_histories, loss\_history)  
  
}

## processing fold # 1   
## processing fold # 2

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

## Warning in rbind(all\_loss\_histories, loss\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

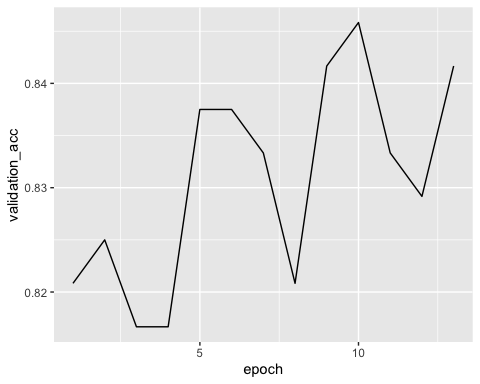
## processing fold # 3

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)  
  
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

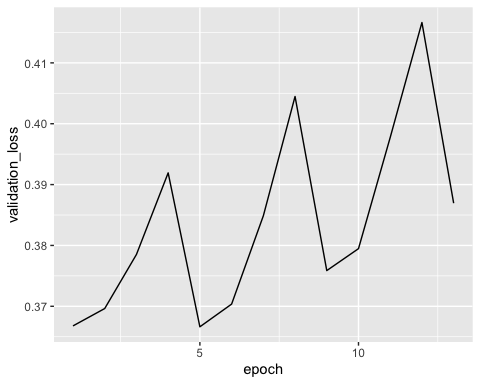
## processing fold # 4

## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)  
  
## Warning in rbind(all\_acc\_histories, acc\_history): number of columns of  
## result is not a multiple of vector length (arg 2)

#Compiling Validationa Accuracy and Loss from folds  
average\_acc\_history <- data.frame(  
 epoch = seq(1:ncol(all\_acc\_histories)),  
 validation\_acc = apply(all\_acc\_histories, 2, mean)  
)  
  
  
average\_loss\_history <- data.frame(  
 epoch = seq(1:ncol(all\_loss\_histories)),  
 validation\_loss = apply(all\_loss\_histories, 2, mean)  
)  
  
#Plotting and Printing results  
ggplot(average\_acc\_history, aes(x = epoch, y = validation\_acc)) + geom\_line()



ggplot(average\_loss\_history, aes(x = epoch, y = validation\_loss)) + geom\_line()



#Evaluate Model Performance on Test Data  
results <- model\_7 %>% evaluate(as.matrix(test\_X.norm), as.matrix(test\_y))  
  
  
#Prediction Values   
predict\_prob <- model\_7 %>% predict(as.matrix(test\_X.norm)) %>% as.data.frame()  
  
# Create Lables based on various thrsholds  
predict\_prob\_tb <- predict\_prob %>% mutate(Death\_Event\_.2 = ifelse(predict\_prob$V1 >= .2,1,0),  
 Death\_Event\_.3 = ifelse(predict\_prob$V1 >= .3,1,0),  
 Death\_Event\_.4 = ifelse(predict\_prob$V1 >= .4,1,0),  
 Death\_Event\_.5 = ifelse(predict\_prob$V1 >= .5,1,0)  
)  
  
  
#preparing Data and Constructing Confusion Matrix  
predict\_prob\_tb$Death\_Event\_.2 <- as.factor(predict\_prob\_tb$Death\_Event\_.2)  
predict\_prob\_tb$Death\_Event\_.3 <- as.factor(predict\_prob\_tb$Death\_Event\_.3)  
predict\_prob\_tb$Death\_Event\_.4 <- as.factor(predict\_prob\_tb$Death\_Event\_.4)  
predict\_prob\_tb$Death\_Event\_.5 <- as.factor(predict\_prob\_tb$Death\_Event\_.5)  
test\_y.factor <- as.factor(test\_y$DEATH\_EVENT)  
  
confusionMatrix(predict\_prob\_tb$Death\_Event\_.5, test\_y.factor)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 31 7  
## 1 9 12  
##   
## Accuracy : 0.7288   
## 95% CI : (0.5973, 0.8364)  
## No Information Rate : 0.678   
## P-Value [Acc > NIR] : 0.2459   
##   
## Kappa : 0.3956   
##   
## Mcnemar's Test P-Value : 0.8026   
##   
## Sensitivity : 0.7750   
## Specificity : 0.6316   
## Pos Pred Value : 0.8158   
## Neg Pred Value : 0.5714   
## Prevalence : 0.6780   
## Detection Rate : 0.5254   
## Detection Prevalence : 0.6441   
## Balanced Accuracy : 0.7033   
##   
## 'Positive' Class : 0   
##

confusionMatrix(predict\_prob\_tb$Death\_Event\_.4, test\_y.factor)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 30 6  
## 1 10 13  
##   
## Accuracy : 0.7288   
## 95% CI : (0.5973, 0.8364)  
## No Information Rate : 0.678   
## P-Value [Acc > NIR] : 0.2459   
##   
## Kappa : 0.4115   
##   
## Mcnemar's Test P-Value : 0.4533   
##   
## Sensitivity : 0.7500   
## Specificity : 0.6842   
## Pos Pred Value : 0.8333   
## Neg Pred Value : 0.5652   
## Prevalence : 0.6780   
## Detection Rate : 0.5085   
## Detection Prevalence : 0.6102   
## Balanced Accuracy : 0.7171   
##   
## 'Positive' Class : 0   
##

confusionMatrix(predict\_prob\_tb$Death\_Event\_.3, test\_y.factor)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 28 5  
## 1 12 14  
##   
## Accuracy : 0.7119   
## 95% CI : (0.5792, 0.8224)  
## No Information Rate : 0.678   
## P-Value [Acc > NIR] : 0.3433   
##   
## Kappa : 0.3983   
##   
## Mcnemar's Test P-Value : 0.1456   
##   
## Sensitivity : 0.7000   
## Specificity : 0.7368   
## Pos Pred Value : 0.8485   
## Neg Pred Value : 0.5385   
## Prevalence : 0.6780   
## Detection Rate : 0.4746   
## Detection Prevalence : 0.5593   
## Balanced Accuracy : 0.7184   
##   
## 'Positive' Class : 0   
##

confusionMatrix(predict\_prob\_tb$Death\_Event\_.2, test\_y.factor)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 25 5  
## 1 15 14  
##   
## Accuracy : 0.661   
## 95% CI : (0.5261, 0.7792)  
## No Information Rate : 0.678   
## P-Value [Acc > NIR] : 0.66682   
##   
## Kappa : 0.3179   
##   
## Mcnemar's Test P-Value : 0.04417   
##   
## Sensitivity : 0.6250   
## Specificity : 0.7368   
## Pos Pred Value : 0.8333   
## Neg Pred Value : 0.4828   
## Prevalence : 0.6780   
## Detection Rate : 0.4237   
## Detection Prevalence : 0.5085   
## Balanced Accuracy : 0.6809   
##   
## 'Positive' Class : 0   
##