SS4850 Final Project

Rui Zhu

4/6/2021

Explanatory analysis

```
HCVdata <- read.csv(file = 'hcvdat0.csv')
suppressMessages(library(dplyr))
drop <- c("X")
HCVdata = HCVdata[,!(names(HCVdata) %in% drop)]
colnames(HCVdata) = c("Category","Age","Sex","ALB","ALP","ALT","AST",
                      "BIL","CHE","CHOL","CREA","GGT","PROT")

# 1 and 2 for blood donor and suspect blood donor, 2,3,4 for hepatitis, fibrosis,cirrhosis respectively
HCVdata$Category <- factor(HCVdata$Category,labels=c(1,2,3,4,5),
                           levels=c('0=Blood Donor',
                                    '0s=suspect Blood Donor',
                                    '1=Hepatitis',
                                    '2=Fibrosis',
                                    '3=Cirrhosis'))


# 0 for male, 1 for female
HCVdata$Sex <- factor(HCVdata$Sex,levels=c("m","f"),labels=c("0","1"))
HCVdata=na.omit(HCVdata[c("Category","Age","Sex","ALB","ALP","ALT","AST",
                          "BIL","CHE","CHOL","CREA","GGT","PROT")])
for(var in 1:13)
{
  HCVdata[,var]=as.numeric(HCVdata[,var])
}

tail(HCVdata)
```

```
##     Category Age Sex ALB   ALP  ALT   AST BIL  CHE CHOL  CREA   GGT PROT
## 608        5  52   2  39  37.0  1.3  30.4  21 6.33 3.78 158.2 142.5 82.7
## 609        5  58   2  34  46.4 15.0 150.0   8 6.26 3.98  56.0  49.7 80.6
## 610        5  59   2  39  51.3 19.6 285.8  40 5.77 4.51 136.1 101.1 70.5
## 611        5  62   2  32 416.6  5.9 110.3  50 5.57 6.30  55.7 650.9 68.5
## 612        5  64   2  24 102.8  2.9  44.4  20 1.54 3.02  63.0  35.9 71.3
## 613        5  64   2  29  87.3  3.5  99.0  48 1.66 3.63  66.7  64.2 82.0
```

EDA!!!

```
#class information, distribution of "category"
library(dplyr)
HCVdata %>%
  group_by(Category) %>%
  summarise(no_rows = length(Category))
```

```
## # A tibble: 5 x 2
##   Category no_rows
##      <dbl>   <int>
## 1        1     526
## 2        2       7
## 3        3      20
## 4        4      12
## 5        5      24
```

```
#mean of each variable
colMeans(HCVdata)
```

```
##  Category       Age       Sex       ALB       ALP       ALT       AST       BIL
##  1.303905 47.417657  1.383701 41.624278 68.123090 26.575382 33.772835 11.018166
##       CHE      CHOL      CREA       GGT      PROT
##  8.203633  5.391341 81.669100 38.198472 71.890153
```

```
#median of each variable
apply(HCVdata,2,median)
```

```
## Category      Age      Sex      ALB      ALP      ALT      AST      BIL
##     1.00    47.00     1.00    41.90    66.20    22.70    25.70     7.10
##      CHE     CHOL     CREA      GGT     PROT
##     8.26     5.31    77.00    22.80    72.10
```

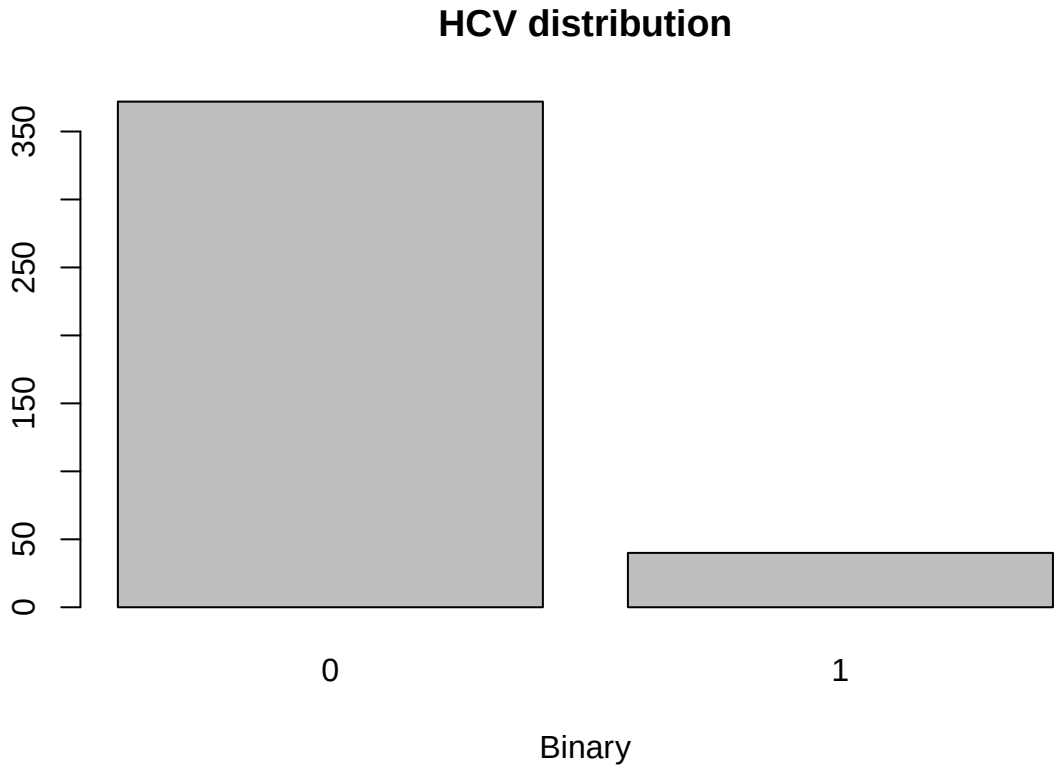
train and test split 7:3

```
## train and test set 7:3
HCVdata<-HCVdata[complete.cases(HCVdata),]
set.seed(4850)
sample <- sample.int(n = nrow(HCVdata), size = floor(.70*nrow(HCVdata)), replace = F)
train <- HCVdata[sample, ]
test <- HCVdata[-sample, ]
```

roc with random forest

```
#binary category
#1 for HCV, 0 for donor(regular/suspect)
set.seed(4850)
```

```r
train$binary <- ifelse(as.numeric(train$Category)>2,1,0)
test$binary <- ifelse(as.numeric(test$Category)>2,1,0)
#barplot on binary
counts <- table(train$binary)
barplot(counts, main="HCV distribution",
   xlab="Binary")
```




```r
#load libraries
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
rfmod <- randomForest(formula = binary ~ .-Category, data = train, ntree = 10, maxnodes= 100, norm.votes
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
suppressMessages(library(dplyr))
testm <- mutate(test,
                rf_pred = predict(rfmod,newdata = test,
                                  type="response"))

suppressMessages(library(pROC))

roc1 <- roc(testm$binary,testm$rf_pred)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
AUC1 <- auc(roc1)
rs <- roc1[['rocs']]
```

Mixed model GLMM

binary random: AST

```
suppressMessages(library(ggpubr))
library(ggplot2)
agePlot=ggplot(train, aes(x=Age,y=binary))+geom_point()
ALPPlot=ggplot(train, aes(x=ALP,y=binary))+geom_point()
ALBPlot=ggplot(train, aes(x=ALB,y=binary))+geom_point()

# regular logistic model
bmod <- glm(binary ~.-Category,binomial,data=,train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(bmod)
```

```
##
## Call:
## glm(formula = binary ~ . - Category, family = binomial, data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.62415  -0.00004   0.00000   0.00000   2.23866
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  8.13762   21.51779   0.378   0.7053
## Age         -0.44341    0.29817  -1.487   0.1370
## Sex          9.46074    5.14238   1.840   0.0658 .
```

```
## ALB         -0.64847    0.51808  -1.252   0.2107
## ALP         -0.46248    0.25763  -1.795   0.0726 .
## ALT         -0.56161    0.33256  -1.689   0.0913 .
## AST          0.15499    0.07922   1.956   0.0504 .
## BIL          0.68036    0.39872   1.706   0.0879 .
## CHE          1.09459    0.79950   1.369   0.1710
## CHOL        -1.98504    2.19451  -0.905   0.3657
## CREA         0.11206    0.06064   1.848   0.0646 .
## GGT          0.22884    0.12947   1.767   0.0771 .
## PROT         0.29164    0.43242   0.674   0.5000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 262.556  on 411  degrees of freedom
## Residual deviance:  12.735  on 399  degrees of freedom
## AIC: 38.735
##
## Number of Fisher Scoring iterations: 14
```

```
#GLMM model random effect:Age
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
set.seed(4850)
suppressMessages(modpql <- glmmPQL(binary ~Sex+ALB+Age+ALT+ALT+BIL+CHE+CHOL+CREA+GGT+PROT,
                  data=train, random= ~ 1|AST, family=binomial))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modpql$rsquared
```

```
## NULL
```

```
typeof(modpql)
```

```
## [1] "list"
```

```
suppressMessages(library(dplyr))
testm <- mutate(test,
                glmm_pred = predict(modpql,newdata = test,
                                    type="response"))

suppressMessages(library(pROC))

roc2 <- roc(testm$binary,testm$glmm_pred)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
AUC <- auc(roc2)
rs <- roc2[['rocs']]
```

Random effect:ALT

```
#GLMM model random effect:ALT
library(MASS)
set.seed(4850)
suppressMessages(modpql <- glmmPQL(binary ~Sex+ALB+Age+ALP+AST+BIL+CHE+CHOL+CREA+GGT+PROT,
                  data=train, random= ~ 1|ALT, family=binomial))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modpql$rsquared
```

```
## NULL
```

```
typeof(modpql)
```

```
## [1] "list"
```

```
suppressMessages(library(dplyr))
testm <- mutate(test,
                glmm_pred = predict(modpql,newdata = test,
                                    type="response"))

suppressMessages(library(pROC))

roc22 <- roc(testm$binary,testm$glmm_pred)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
AUC <- auc(roc22)
rs <- roc22[['rocs']]
```

Random effect:ALP

```r
#GLMM model random effect:ALT
library(MASS)
set.seed(4850)
suppressMessages(modpql <- glmmPQL(binary ~Sex+ALB+Age+ALP+AST+BIL+CHE+CHOL+CREA+GGT+PROT,
                  data=train, random= ~ 1|ALP, family=binomial))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
suppressMessages(drop1(modpql, test="Chi"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Single term deletions
##
## Model:
## binary ~ Sex + ALB + Age + ALP + AST + BIL + CHE + CHOL + CREA +
##     GGT + PROT
##        Df AIC LRT Pr(>Chi)
## <none>
## Sex     1
## ALB     1
## Age     1
## ALP     1
## AST     1
## BIL     1
## CHE     1
## CHOL    1
## CREA    1
## GGT     1
## PROT    1
```

```r
summary(modpql)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: train
##   AIC BIC logLik
##    NA  NA     NA
```

```
##
## Random effects:
##  Formula: ~1 | ALP
##         (Intercept) Residual
## StdDev:     7996511 11797543
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects:  binary ~ Sex + ALB + Age + ALP + AST + BIL + CHE + CHOL + CREA +      GGT + PROT
##                     Value    Std.Error  DF    t-value p-value
## (Intercept) -3.623745e+15 6.385179e+14 317  -5.675245  0.0000
## Sex          5.262527e+14 8.938216e+13  83   5.887671  0.0000
## ALB         -6.214548e+13 8.871369e+12  83  -7.005173  0.0000
## Age          3.411670e+12 4.194092e+12  83   0.813447  0.4183
## ALP         -1.950967e+13 1.768267e+12 317 -11.033212  0.0000
## AST          1.389197e+13 1.459312e+12  83   9.519529  0.0000
## BIL          4.411631e+13 2.572143e+12  83  17.151577  0.0000
## CHE          5.755256e+13 2.320404e+13  83   2.480282  0.0151
## CHOL        -6.191268e+13 4.026513e+13  83  -1.537625  0.1279
## CREA         7.730453e+12 6.950193e+11  83  11.122644  0.0000
## GGT          9.665966e+12 9.851343e+11  83   9.811826  0.0000
## PROT         5.013084e+13 9.665602e+12  83   5.186521  0.0000
##  Correlation:
##      (Intr) Sex    ALB    Age    ALP    AST    BIL    CHE    CHOL   CREA
## Sex  -0.234
## ALB  -0.067  0.165
## Age  -0.470  0.038  0.081
## ALP  -0.129 -0.055  0.083 -0.072
## AST  -0.064  0.123  0.181 -0.017  0.207
## BIL  -0.114  0.150  0.096  0.058 -0.005 -0.127
## CHE  -0.046  0.278 -0.050  0.070 -0.068  0.034  0.282
## CHOL  0.030 -0.127  0.004 -0.173 -0.081  0.182 -0.007 -0.333
## CREA -0.167  0.159  0.012  0.056 -0.147  0.097  0.023  0.045  0.072
## GGT   0.086  0.052 -0.046 -0.054 -0.482 -0.466 -0.127  0.067 -0.093 -0.065
## PROT -0.661 -0.118 -0.541  0.142 -0.001 -0.207 -0.090 -0.178 -0.182  0.006
##      GGT
## Sex
## ALB
## Age
## ALP
## AST
## BIL
## CHE
## CHOL
## CREA
## GGT
## PROT  0.037
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -5.53673167 -0.27104498 -0.11545497  0.05928468  5.47370115
##
## Number of Observations: 412
```

```
## Number of Groups: 319
```

```
modpql$rsquared
```

```
## NULL
```

```
typeof(modpql)
```

```
## [1] "list"
```

```
suppressMessages(library(dplyr))
testm <- mutate(test,
                glmm_pred = predict(modpql,newdata = test,
                                    type="response"))

suppressMessages(library(pROC))

roc23 <- roc(testm$binary,testm$glmm_pred)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
AUC <- auc(roc23)
rs <- roc23[['rocs']]
coords(roc23,"best")
```

```
##   threshold specificity sensitivity
## 1       0.5   0.9714286           1
```

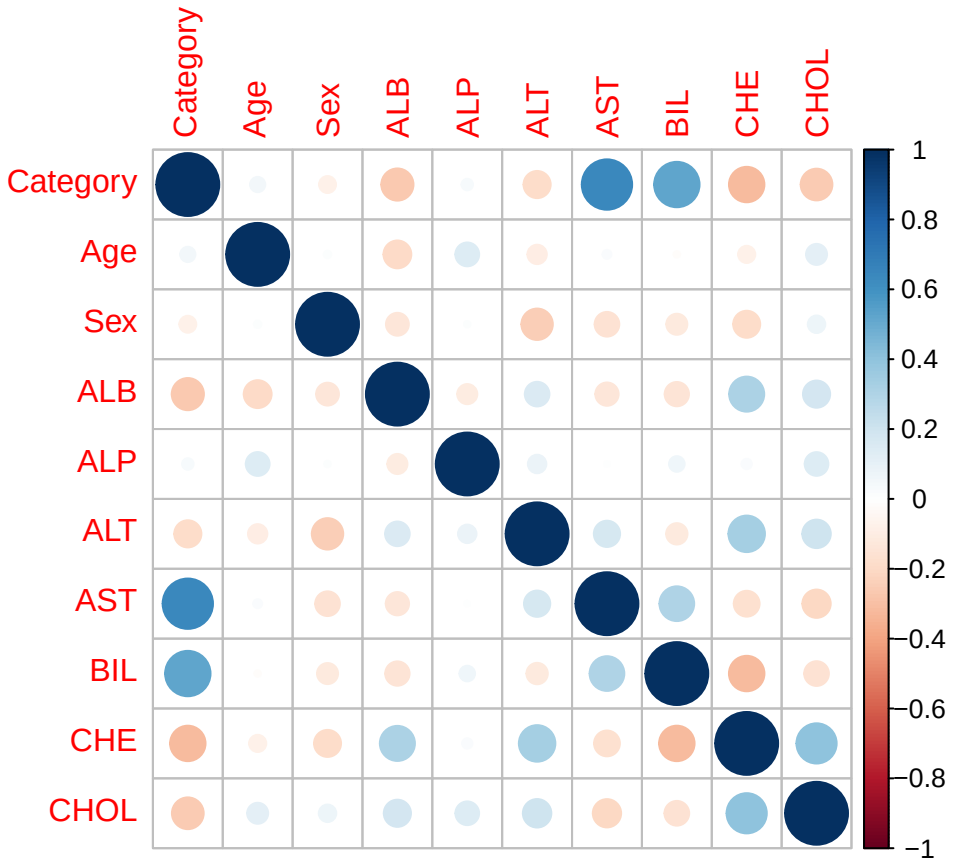
Deep learning

binary

```
library(dplyr)
library(keras)
library(tensorflow)
library(tfdatasets)



set.seed(4850)
# Store the overall correlation in `M`
M <- cor(train[,1:10])

# Plot the correlation plot with `M`
library(corrplot)
```

```
## corrplot 0.84 loaded
```

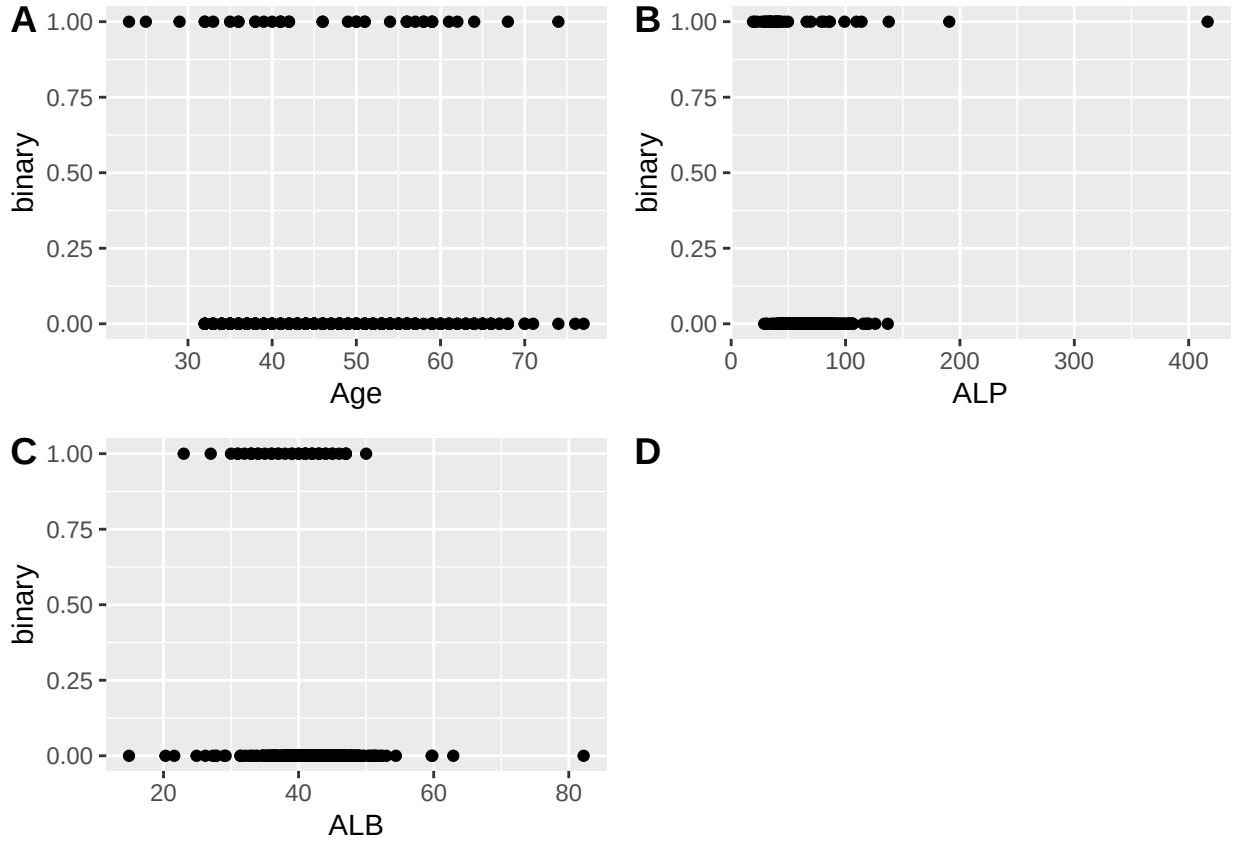
```
predictorsCorr=corrplot(M, method="circle")
```




```
figure <- ggarrange(agePlot, ALPPlot, ALBPlot,predictorsCorr,
                    labels = c("A", "B", "C","D"),
                    ncol = 2, nrow = 2)
```

```
## Warning in as_grob.default(plot): Cannot convert object of class matrixarray
## into a grob.
```

```
figure
```




```
#normalize
spec <- feature_spec(train, binary ~ .-Category ) %>%
  step_numeric_column(all_numeric(), normalizer_fn = scaler_standard()) %>%
  fit()
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\Admin\.conda\envs\test-env/python.exe": The system cannot find the file
## specified
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\Admin\.conda\envs\test-env/python.exe": The system cannot find the file
## specified
```

```
spec
```

```
## -- Feature Spec ----------------------------------------------------------
## A feature_spec with 12 steps.
## Fitted: TRUE
## -- Steps -----------------------------------------------------------------
## The feature_spec has 1 dense features.
## StepNumericColumn: Age, Sex, ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT, PROT
## -- Dense features --------------------------------------------------------
```

```
layer <- layer_dense_features(
  feature_columns = dense_features(spec),
  dtype = tf$float32
)
suppressMessages(layer(train))
```

```
## tf.Tensor(
## [[ 0.653347   -0.6670947  -0.568868   ... -0.4312038   0.7929118
##   -0.76765865]
##  [-0.44780117 -0.22397378 -0.13247223 ... -0.32766175 -0.35917082
##   -0.76765865]
##  [-0.05207561  2.561358    0.41447717 ...  1.2864846   0.079717
##    1.2995006 ]
##  ...
##  [-0.9811694  -0.68571323 -1.2147336  ...  0.36570016  0.55518115
##   -0.76765865]
##  [-1.1188128   0.05530416 -0.1441094  ... -0.14276527 -2.059866
##    1.2995006 ]
##  [ 0.08556774 -0.71922666  0.8683287  ...  0.6338001  -0.03000497
##   -0.76765865]], shape=(412, 12), dtype=float32)
```

```
input <- layer_input_from_dataset(train[,2:13])


output <- input %>%
  layer_dense_features(dense_features(spec)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 1)

dpmod <- keras_model(input, output)
```

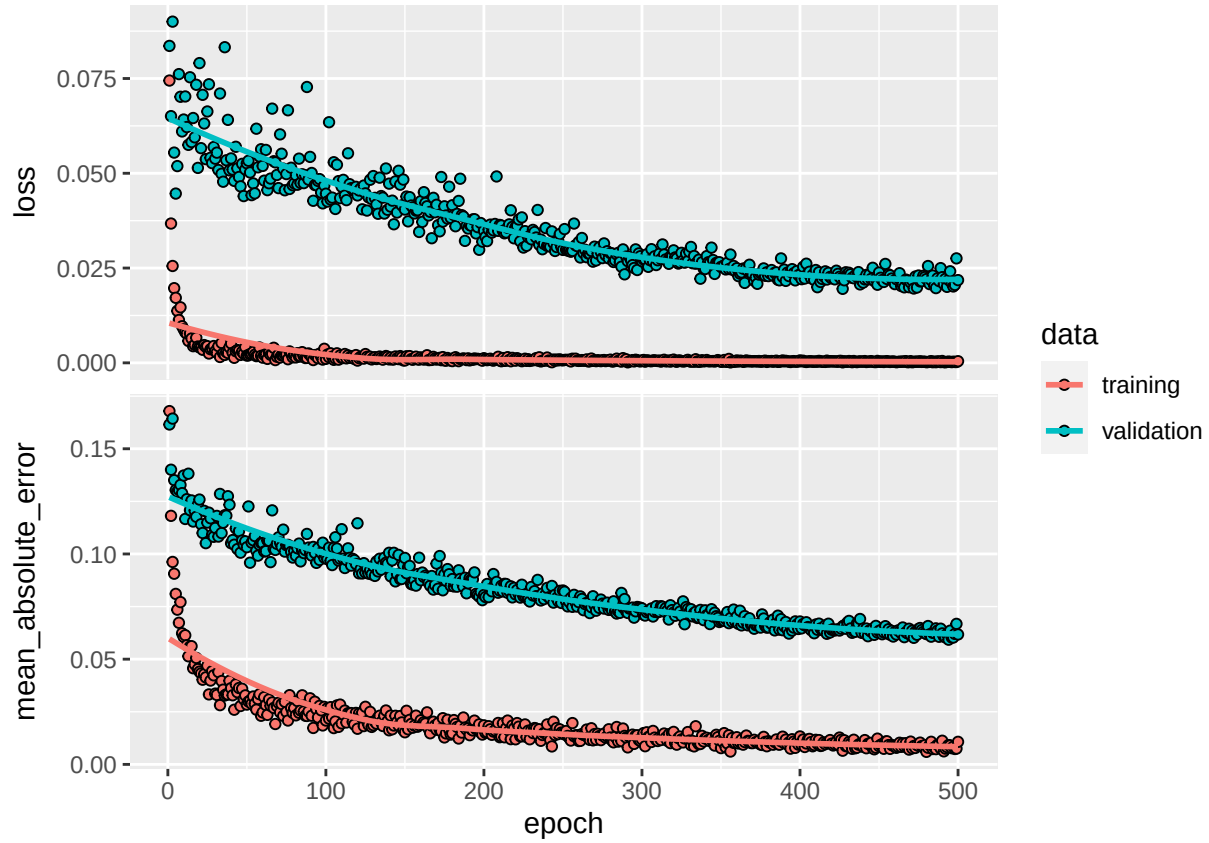
```
dpmod %>%
  compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_absolute_error")
  )

build_model <- function() {
  input <- layer_input_from_dataset(train[,2:13])

  output <- input %>%
    layer_dense_features(dense_features(spec)) %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 1)

  dpmod <- keras_model(input, output)
```

```
  dpmod %>%
    compile(
      loss = "mse",
      optimizer = optimizer_rmsprop(),
      metrics = list("mean_absolute_error")
    )

  dpmod
}


# Display training progress by printing a single dot for each completed epoch.
print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)

dpmod <- build_model()

history <- dpmod %>% fit(
  x = train[2:13],
  y = train$binary,
  epochs = 500,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(print_dot_callback)
)
```

```
##
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ..................
```

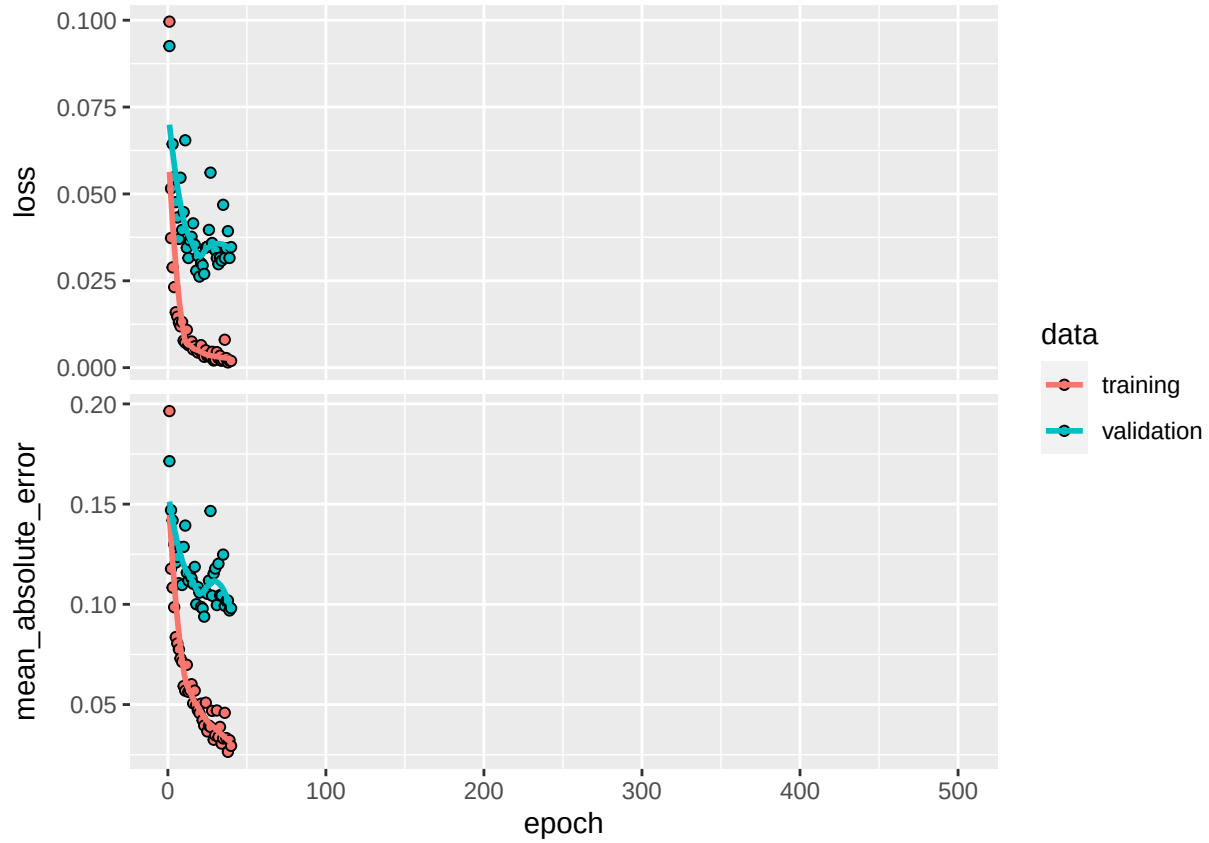
```
library(ggplot2)

plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# it will stop when no more improvement
early_stop <- callback_early_stopping(monitor = "val_loss", patience = 20)

dpmod <- build_model()

history <- dpmod %>% fit(
  x = train[2:13],
  y = train$binary,
  epochs = 500,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(early_stop)
)

plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
suppressMessages(library(dplyr))
testm <- mutate(test,
                dp_pred = dpmod %>% predict(test[,2:13]), type="response")



roc3 <- roc(testm$binary,testm$dp_pred)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(testm$binary, testm$dp_pred): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```
AUC3 <- auc(roc3)
rs <- roc3[['rocs']]
suppressMessages(summary(dpmod))
```

```
## Model: "model_2"
## _________________________________________________________________________________
## Layer (type)                 Output Shape              Param #   Connected to
## =================================================================================
## ALB (InputLayer)             [(None,)]                 0
```

```
## ____________________________________________________________________________
## ALP (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## ALT (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## AST (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## Age (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## BIL (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## CHE (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## CHOL (InputLayer)         [(None,)]         0
## ____________________________________________________________________________
## CREA (InputLayer)         [(None,)]         0
## ____________________________________________________________________________
## GGT (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## PROT (InputLayer)         [(None,)]         0
## ____________________________________________________________________________
## Sex (InputLayer)          [(None,)]         0
## ____________________________________________________________________________
## dense_features_3 (DenseFe (None, 12)        0         ALB[0][0]
##                                                       ALP[0][0]
##                                                       ALT[0][0]
##                                                       AST[0][0]
##                                                       Age[0][0]
##                                                       BIL[0][0]
##                                                       CHE[0][0]
##                                                       CHOL[0][0]
##                                                       CREA[0][0]
##                                                       GGT[0][0]
##                                                       PROT[0][0]
##                                                       Sex[0][0]
## ____________________________________________________________________________
## dense_8 (Dense)           (None, 64)        832       dense_features_3[0][0]
## ____________________________________________________________________________
## dense_7 (Dense)           (None, 64)        4160      dense_8[0][0]
## ____________________________________________________________________________
## dense_6 (Dense)           (None, 1)         65        dense_7[0][0]
## ============================================================================
## Total params: 5,057
## Trainable params: 5,057
## Non-trainable params: 0
## ____________________________________________________________________________
```

```
#RM,(AST ALT ALP) glmm, nn
coords(roc1,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))##the random forest model
```

```
##           threshold specificity sensitivity  accuracy precision
## threshold 0.2833333   0.9689441           1 0.9717514 0.7619048
```

```
coords(roc2,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the AST GLMM model
```

```
##              threshold specificity sensitivity  accuracy precision
## threshold 8.117251e-06    0.952381   0.6666667 0.9444444 0.2857143
```

```
coords(roc22,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))#the ALT GLMM model
```

```
##            threshold specificity sensitivity  accuracy precision
## threshold 0.04422787   0.9292929           1 0.9326923 0.4166667
```

```
coords(roc23,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the ALP GLMM model
```

```
##           threshold specificity sensitivity accuracy precision
## threshold       0.5   0.9714286           1 0.972973 0.6666667
```

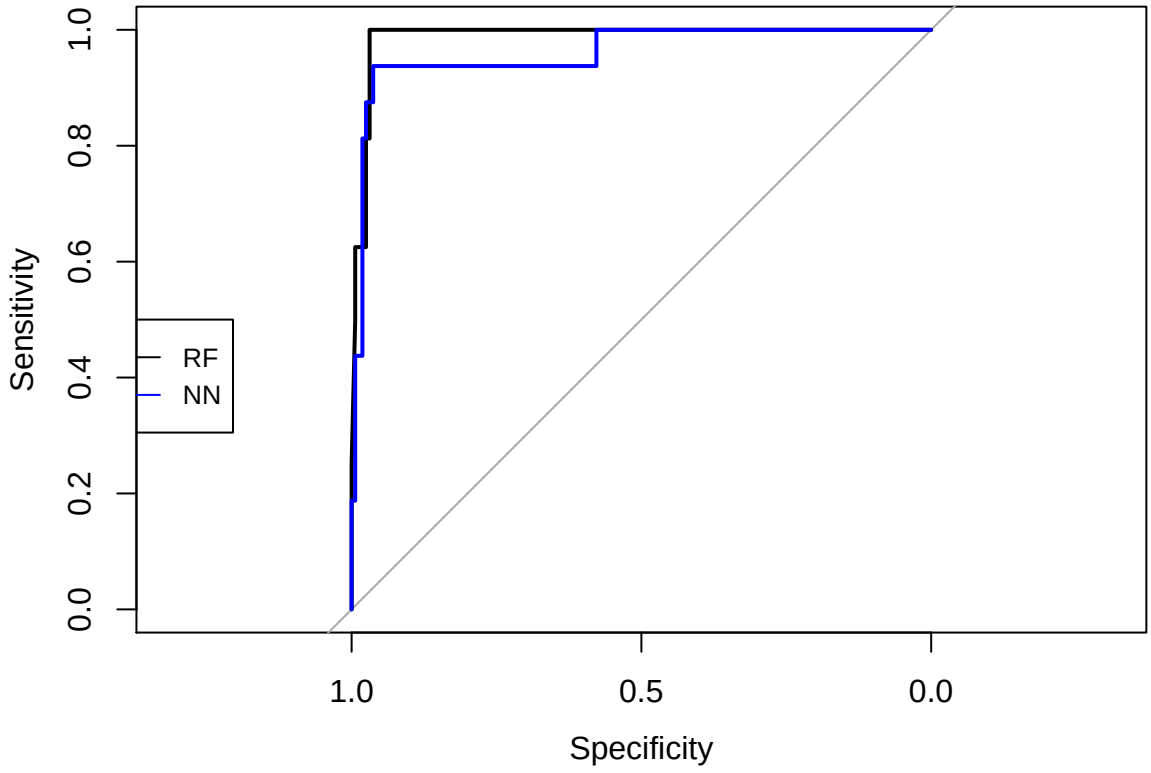
```
coords(roc3,"best",ret=c("threshold",
"specificity", "sensitivity", "accuracy","precision"))# the nn GLMM model
```

```
##           threshold specificity sensitivity accuracy precision
## threshold 0.1863039   0.9627329      0.9375 0.960452 0.7142857
```

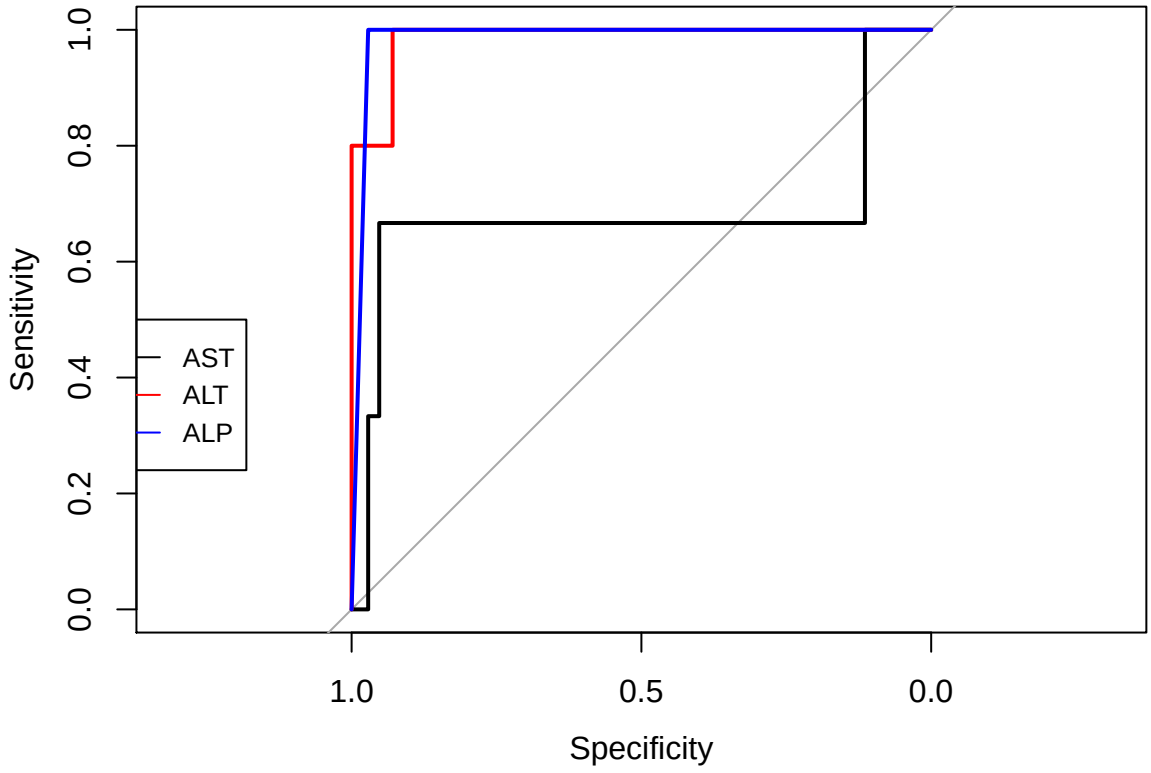
```
#importance: impurity gini importance
importance(rfmod)
```

```
##      IncNodePurity
## Age   3.610843e-01
## Sex   2.775558e-18
## ALB   4.248468e-01
## ALP   3.345247e+00
## ALT   7.598513e+00
## AST   1.973172e+01
## BIL   1.716077e+00
## CHE   1.254098e+00
## CHOL  3.677537e-01
## CREA  4.640634e-01
## GGT   2.672794e+00
## PROT  6.762646e-15
```

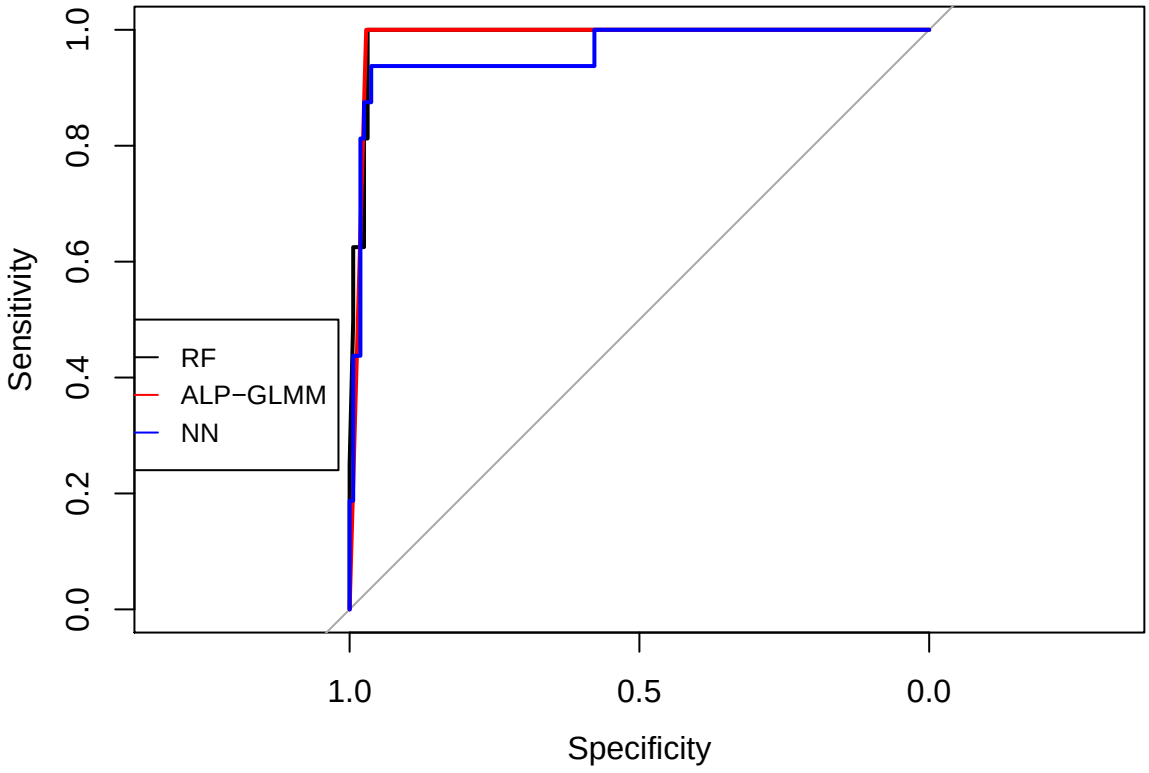
```
#neural network and random forest
plot(roc1)
plot(roc3, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("RF", "NN"),
       col=c( "black","blue"), lty=1, cex=0.8)
```




```
#three glmm roc curve
plot(roc2)
plot(roc22, add=TRUE, col='red')
plot(roc23, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("AST", "ALT", "ALP"),
       col=c( "black","red", "blue"), lty=1, cex=0.8)
```




```
#randomforest best glmm and cnn roc curve
plot(roc1)
plot(roc23, add=TRUE, col='red')
plot(roc3, add=TRUE, col='blue')
legend(1.45, 0.5, legend=c("RF", "ALP-GLMM", "NN"),
       col=c( "black","red", "blue"), lty=1, cex=0.8)
```


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
RF
ALP−GLMM
NN
1.0
0.5
0.0
Specificity