AEs

```r
library(keras)
library(RColorBrewer)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(readr)
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
setwd("/home/ajo/gitRepos/DNN/task1")
clinical <- read_delim("clinical.csv","\t", escape_double = FALSE, trim_ws = TRUE)
```

```
## Rows: 847 Columns: 19
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (19): Sample, Histology, PAM50Call, ajcc_cancer_metastasis_stage_code, a...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#gene_expression <- read_delim("gene_expression.csv","\t", escape_double = FALSE, trim_ws = TRUE)
protein_abundance <- read_delim("protein_abundance.csv","\t", escape_double = FALSE, trim_ws = TRUE)
```

```
## Rows: 410 Columns: 143
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr   (1): Sample
## dbl (142): 14-3-3_epsilon, 4E-BP1, 4E-BP1_pS65, 4E-BP1_pT37, 4E-BP1_pT70, 53...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#copy_number <- read_delim("copy_number.csv","\t", escape_double = FALSE, trim_ws = TRUE)

set1<-intersect(protein_abundance$Sample,clinical$Sample)

xclinical<-clinical[clinical$Sample%in%set1,]
xprotein<-protein_abundance[protein_abundance$Sample%in%set1,]
```

```r
xclinical<-xclinical[,c(1,9)]
sel1<-which(xclinical$breast_carcinoma_estrogen_receptor_status!="Positive")
sel2<-which(xclinical$breast_carcinoma_estrogen_receptor_status!="Negative")

sel<-intersect(sel1,sel2)
xclinical<-xclinical[-sel,]

xclinical<-xclinical[-which(is.na(xclinical$breast_carcinoma_estrogen_receptor_status)),]


mprotein<-merge(xclinical,xprotein,by.x="Sample",by.y="Sample")


# pprotein
sel<-complete.cases(t(mprotein))
```

```r
set.seed(111)
training<-sample(1:nrow(mprotein),2*nrow(mprotein)/3)


xtrain<-mprotein[training,-c(1,2)]
xtest<-mprotein[-training,-c(1,2)]

xtrain<-scale(xtrain)
xtest<-scale(xtest)

ytrain<-mprotein[training,2]
ytest<-mprotein[-training,2]

ylabels<-vector()
ylabels[ytrain=="Positive"]<-1
ylabels[ytrain=="Negative"]<-0


ytestlabels<-vector()
ytestlabels[ytest=="Positive"]<-1
ytestlabels[ytest=="Negative"]<-0
```

Autoencoder

```r
model<-keras_model_sequential() %>%
  layer_dense(units=50,activation="relu",input_shape=c(142)) %>%
  layer_dense(units=20,activation="relu") %>%
  layer_dense(units=50,activation="relu") %>%
  layer_dense(units=142,activation="linear")
```

```
## Loaded Tensorflow version 2.7.1
```

```r
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                       Param #
## ================================================================================
```

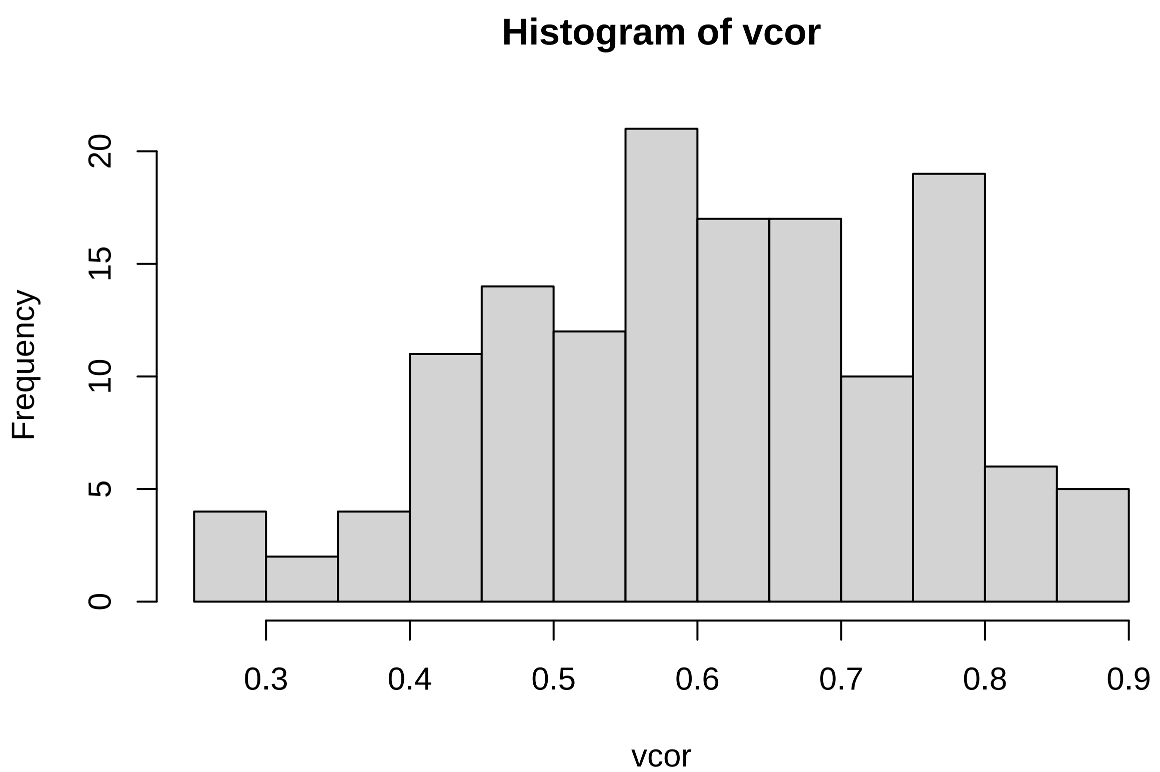
```
##  dense_3 (Dense)                    (None, 50)                    7150
##
##  dense_2 (Dense)                    (None, 20)                    1020
##
##  dense_1 (Dense)                    (None, 50)                    1050
##
##  dense (Dense)                      (None, 142)                   7242
##
## ================================================================================
## Total params: 16,462
## Trainable params: 16,462
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model %>% compile(
  optimizer = "rmsprop",
  loss = "mse"
)
```

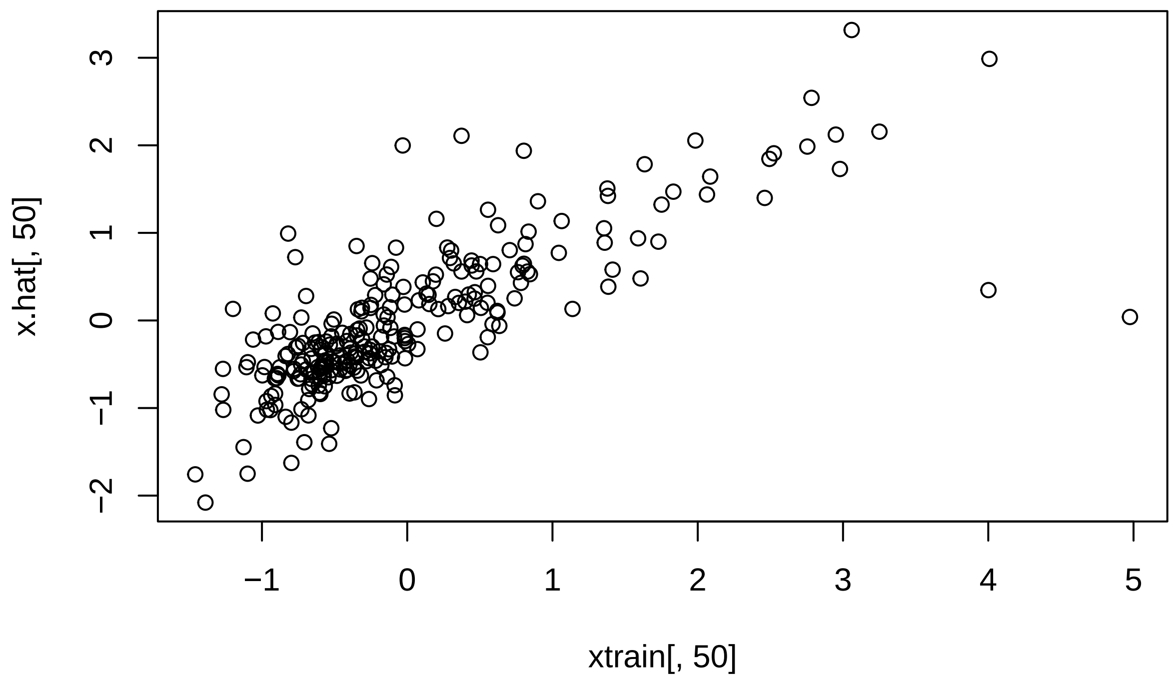
```
model %>% fit(
  x=as.matrix(xtrain),
  y=as.matrix(xtrain),
  epochs = 25,
  batch_size=64,
  validation_split = 0.2
  )
```

```
x.hat<-predict(model,as.matrix(xtrain))
```

```
vcor<-diag(cor(x.hat,xtrain))
hist(vcor)
```

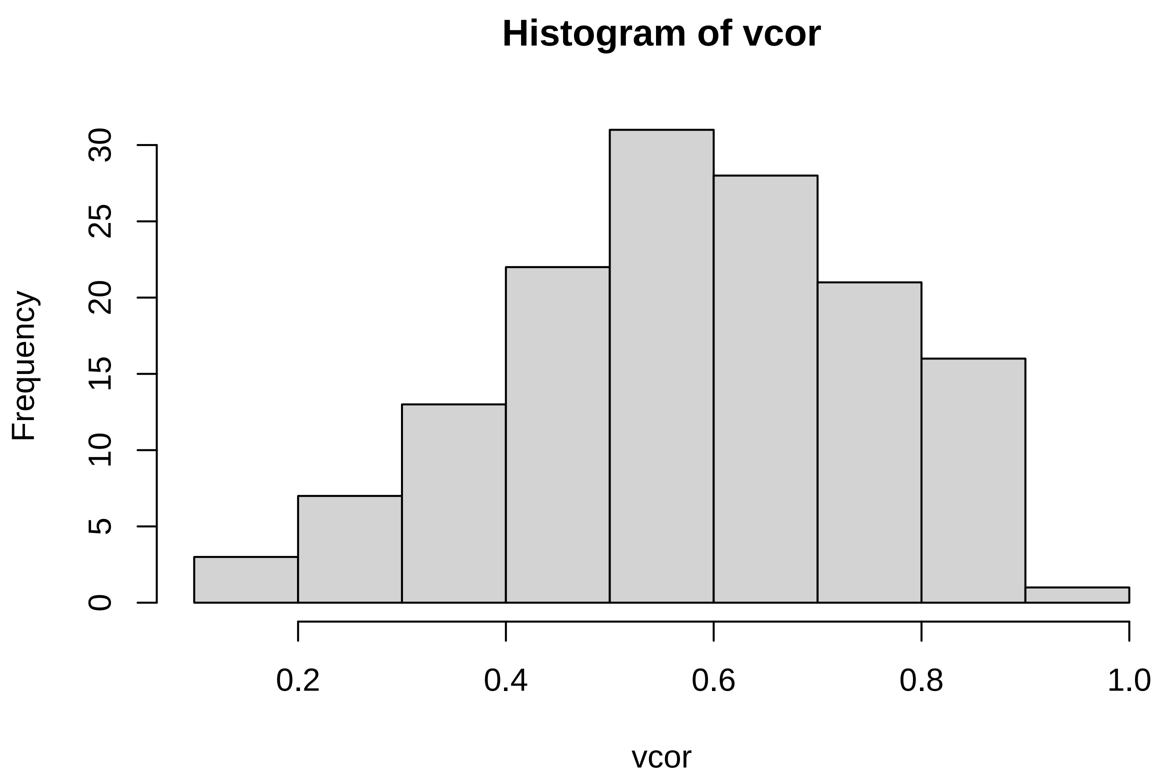


```r
# visual inspection
plot(x.hat[,50]~xtrain[,50])
```

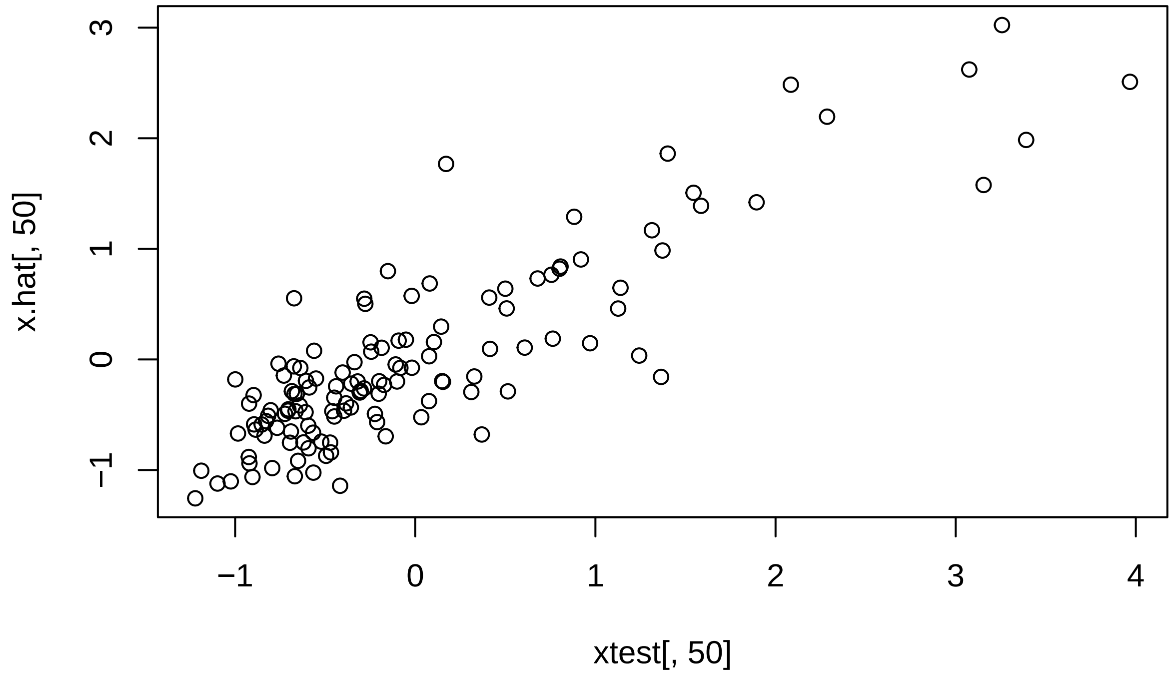



```r
x.hat<-predict(model,as.matrix(xtest))
```

```r
vcor<-diag(cor(x.hat,xtest))
hist(vcor)
```



```r
# visual inspection
plot(x.hat[,50]~xtest[,50])
```



Defining Encoder and Decoder submodels

```r
input_enc<-layer_input(shape = 142)
output_enc<-input_enc %>%
  layer_dense(units=50,activation="relu") %>%
  layer_dense(units=20,activation="relu")

encoder = keras_model(input_enc, output_enc)
summary(encoder)
```

```
## Model: "model"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_1 (InputLayer)               [(None, 142)]                     0
##
##  dense_5 (Dense)                    (None, 50)                        7150
##
##  dense_4 (Dense)                    (None, 20)                        1020
##
## ================================================================================
## Total params: 8,170
## Trainable params: 8,170
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
input_dec = layer_input(shape = 20)
output_dec<-input_dec %>%
  layer_dense(units=50,activation="relu") %>%
```

```
  layer_dense(units=142,activation="linear")

decoder = keras_model(input_dec, output_dec)

summary(decoder)
```

```
## Model: "model_1"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_2 (InputLayer)               [(None, 20)]                      0
##
##  dense_7 (Dense)                    (None, 50)                        1050
##
##  dense_6 (Dense)                    (None, 142)                       7242
##
## ================================================================================
## Total params: 8,292
## Trainable params: 8,292
## Non-trainable params: 0
## ________________________________________________________________________________
```

Defining Autoencoder

```
aen_input = layer_input(shape = 142)
aen_output = aen_input %>%
  encoder() %>%
  decoder()

aen = keras_model(aen_input, aen_output)
summary(aen)
```

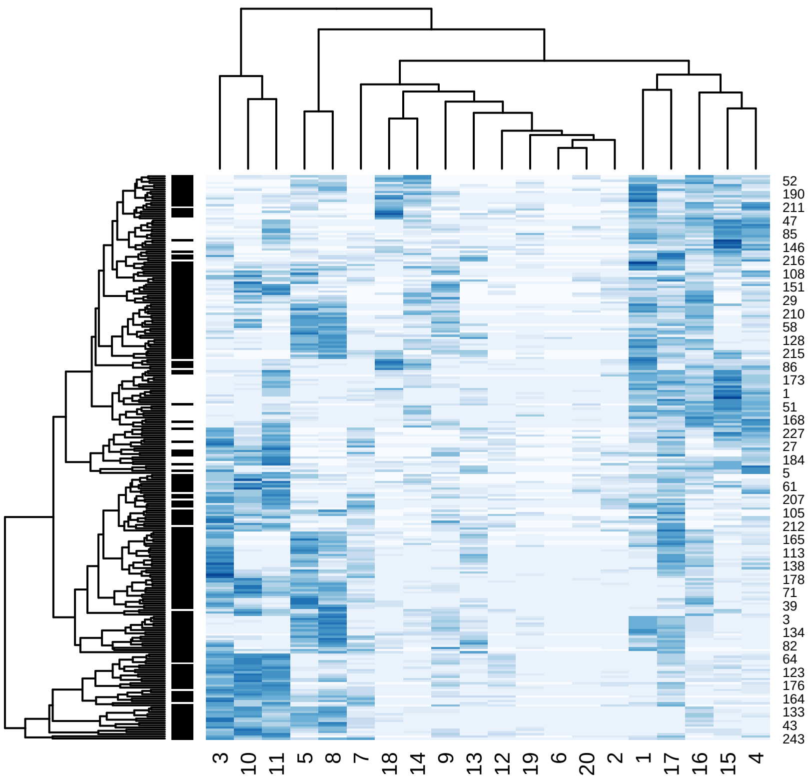
```
## Model: "model_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_3 (InputLayer)               [(None, 142)]                     0
##
##  model (Functional)                 (None, 20)                        8170
##
##  model_1 (Functional)               (None, 142)                       8292
##
## ================================================================================
## Total params: 16,462
## Trainable params: 16,462
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
aen %>% compile(
  optimizer = "rmsprop",
  loss = "mse"
)
```

```
aen %>% fit(
  x=as.matrix(xtrain),
  y=as.matrix(xtrain),
  epochs = 25,
  batch_size=64,
  validation_split = 0.2
  )
```

```
#Generating with Autoencoder
encoded_expression <- encoder %>% predict(as.matrix(xtrain))
decoded_expression <- decoder %>% predict(encoded_expression)
```

```
colMain <- colorRampPalette(brewer.pal(8, "Blues"))(15)
heatmap(encoded_expression,  RowSideColors=as.character(ylabels) , col=colMain, scale="row" )
```



Stacked autoencoder

AE1

```
input_enc1<-layer_input(shape = 142)
output_enc1<-input_enc1 %>%
  layer_dense(units=50,activation="relu")
encoder1 = keras_model(input_enc1, output_enc1)
summary(encoder1)
```

```
## Model: "model_3"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_4 (InputLayer)               [(None, 142)]                     0
##
```

```
##  dense_8 (Dense)                   (None, 50)                    7150
##
## ================================================================================
## Total params: 7,150
## Trainable params: 7,150
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
input_dec1 = layer_input(shape = 50)
output_dec1<-input_dec1 %>%
  layer_dense(units=142,activation="linear")

decoder1 = keras_model(input_dec1, output_dec1)

summary(decoder1)
```

```
## Model: "model_4"
## ________________________________________________________________________________
##  Layer (type)                      Output Shape                  Param #
## ================================================================================
##  input_5 (InputLayer)              [(None, 50)]                  0
##
##  dense_9 (Dense)                   (None, 142)                   7242
##
## ================================================================================
## Total params: 7,242
## Trainable params: 7,242
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
aen_input1 = layer_input(shape = 142)
aen_output1 = aen_input1 %>%
  encoder1() %>%
  decoder1()

sae1 = keras_model(aen_input1, aen_output1)
summary(sae1)
```

```
## Model: "model_5"
## ________________________________________________________________________________
##  Layer (type)                      Output Shape                  Param #
## ================================================================================
##  input_6 (InputLayer)              [(None, 142)]                 0
##
##  model_3 (Functional)              (None, 50)                    7150
##
##  model_4 (Functional)              (None, 142)                   7242
##
## ================================================================================
## Total params: 14,392
## Trainable params: 14,392
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
sae1 %>% compile(
  optimizer = "rmsprop",
```

```
  loss = "mse"
)
```

```
sae1 %>% fit(
  x=as.matrix(xtrain),
  y=as.matrix(xtrain),
  epochs = 25,
  batch_size=64,
  validation_split = 0.2
  )
```

```
#Generating with Autoencoder
encoded_expression1 <- encoder1 %>% predict(as.matrix(xtrain))
```

AE2

```
input_enc2<-layer_input(shape = 50)
output_enc2<-input_enc2 %>%
  layer_dense(units=20,activation="relu")
encoder2 = keras_model(input_enc2, output_enc2)
summary(encoder2)
```

```
## Model: "model_6"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  input_7 (InputLayer)               [(None, 50)]                    0
##
##  dense_10 (Dense)                   (None, 20)                      1020
##
## ================================================================================
## Total params: 1,020
## Trainable params: 1,020
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
input_dec2 = layer_input(shape = 20)
output_dec2<-input_dec2 %>%
  layer_dense(units=50,activation="linear")

decoder2 = keras_model(input_dec2, output_dec2)

summary(decoder2)
```

```
## Model: "model_7"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  input_8 (InputLayer)               [(None, 20)]                    0
##
##  dense_11 (Dense)                   (None, 50)                      1050
##
## ================================================================================
## Total params: 1,050
## Trainable params: 1,050
```

```
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
aen_input2 = layer_input(shape = 50)
aen_output2 = aen_input2 %>%
  encoder2() %>%
  decoder2()

sae2 = keras_model(aen_input2, aen_output2)
summary(sae2)
```

```
## Model: "model_8"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  input_9 (InputLayer)               [(None, 50)]                     0
##
##  model_6 (Functional)               (None, 20)                       1020
##
##  model_7 (Functional)               (None, 50)                       1050
##
## ================================================================================
## Total params: 2,070
## Trainable params: 2,070
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
sae2 %>% compile(
  optimizer = "rmsprop",
  loss = "mse"
)
```

```
sae2 %>% fit(
  x=as.matrix(encoded_expression1),
  y=as.matrix(encoded_expression1),
  epochs = 25,
  batch_size=64,
  validation_split = 0.2
  )
```

```
#Generating with Autoencoder
encoded_expression2 <- encoder2 %>% predict(as.matrix(encoded_expression1))
```

AE3

```
input_enc3<-layer_input(shape = 20)
output_enc3<-input_enc3 %>%
  layer_dense(units=10,activation="relu")
encoder3 = keras_model(input_enc3, output_enc3)
summary(encoder3)
```

```
## Model: "model_9"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  input_10 (InputLayer)              [(None, 20)]                     0
```

```
##
##  dense_12 (Dense)                    (None, 10)                        210
##
## ================================================================================
## Total params: 210
## Trainable params: 210
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
input_dec3 = layer_input(shape = 10)
output_dec3<-input_dec3 %>%
  layer_dense(units=20,activation="linear")

decoder3 = keras_model(input_dec3, output_dec3)

summary(decoder3)
```

```
## Model: "model_10"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_11 (InputLayer)              [(None, 10)]                      0
##
##  dense_13 (Dense)                   (None, 20)                        220
##
## ================================================================================
## Total params: 220
## Trainable params: 220
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
aen_input3 = layer_input(shape = 20)
aen_output3 = aen_input3 %>%
  encoder3() %>%
  decoder3()

sae3 = keras_model(aen_input3, aen_output3)
summary(sae3)
```

```
## Model: "model_11"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_12 (InputLayer)              [(None, 20)]                      0
##
##  model_9 (Functional)               (None, 10)                        210
##
##  model_10 (Functional)              (None, 20)                        220
##
## ================================================================================
## Total params: 430
## Trainable params: 430
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
sae3 %>% compile(
  optimizer = "rmsprop",
  loss = "mse"
)
```

```
sae3 %>% fit(
  x=as.matrix(encoded_expression2),
  y=as.matrix(encoded_expression2),
  epochs = 25,
  batch_size=64,
  validation_split = 0.2
  )
```

```
#Generating with Autoencoder
encoded_expression3 <- encoder3 %>% predict(as.matrix(encoded_expression2))
```

Final model

```
sae_input = layer_input(shape = 142)
sae_output = sae_input %>%
  encoder1() %>%
  encoder2()  %>%
  encoder3() %>%
  layer_dense(5,activation = "relu")%>%
  layer_dense(1,activation = "sigmoid")

sae = keras_model(sae_input, sae_output)
summary(sae)
```

```
## Model: "model_12"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  input_13 (InputLayer)              [(None, 142)]                   0
##
##  model_3 (Functional)               (None, 50)                      7150
##
##  model_6 (Functional)               (None, 20)                      1020
##
##  model_9 (Functional)               (None, 10)                      210
##
##  dense_15 (Dense)                   (None, 5)                       55
##
##  dense_14 (Dense)                   (None, 1)                       6
##
## ================================================================================
## Total params: 8,441
## Trainable params: 8,441
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
freeze_weights(sae,from=1,to=3)
```

```
summary(sae)
```

```
## Model: "model_12"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  input_13 (InputLayer)              [(None, 142)]                     0
##
##  model_3 (Functional)               (None, 50)                        7150
##
##  model_6 (Functional)               (None, 20)                        1020
##
##  model_9 (Functional)               (None, 10)                        210
##
##  dense_15 (Dense)                   (None, 5)                         55
##
##  dense_14 (Dense)                   (None, 1)                         6
##
## ================================================================================
## Total params: 8,441
## Trainable params: 271
## Non-trainable params: 8,170
## ________________________________________________________________________________
```

```r
sae %>% compile(
  optimizer = "rmsprop",
  loss = 'binary_crossentropy',
  metric = "acc"
  )
```

```r
sae %>% fit(
  x=xtrain,
  y=ylabels,
  epochs = 30,
  batch_size=64,
  validation_split = 0.2
  )
```

```r
sae %>%
  evaluate(as.matrix(xtest), ytestlabels)
```

```
##      loss       acc
## 0.6690835 0.7187500
```

```r
yhat <- predict(sae,as.matrix(xtest))
```

```r
yhatclass<-as.factor(ifelse(yhat<0.5,0,1))
table(yhatclass,  ytestlabels)
```

```
##          ytestlabels
## yhatclass  0  1
##         0 10 13
##         1 23 82
```

```r
confusionMatrix(yhatclass,as.factor(ytestlabels))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  0  1
##          0 10 13
##          1 23 82
##
##                Accuracy : 0.7188
##                  95% CI : (0.6325, 0.7946)
##     No Information Rate : 0.7422
##     P-Value [Acc > NIR] : 0.7628
##
##                   Kappa : 0.1844
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.30303
##             Specificity : 0.86316
##          Pos Pred Value : 0.43478
##          Neg Pred Value : 0.78095
##              Prevalence : 0.25781
##          Detection Rate : 0.07812
##    Detection Prevalence : 0.17969
##       Balanced Accuracy : 0.58309
##
##        'Positive' Class : 0
##
```

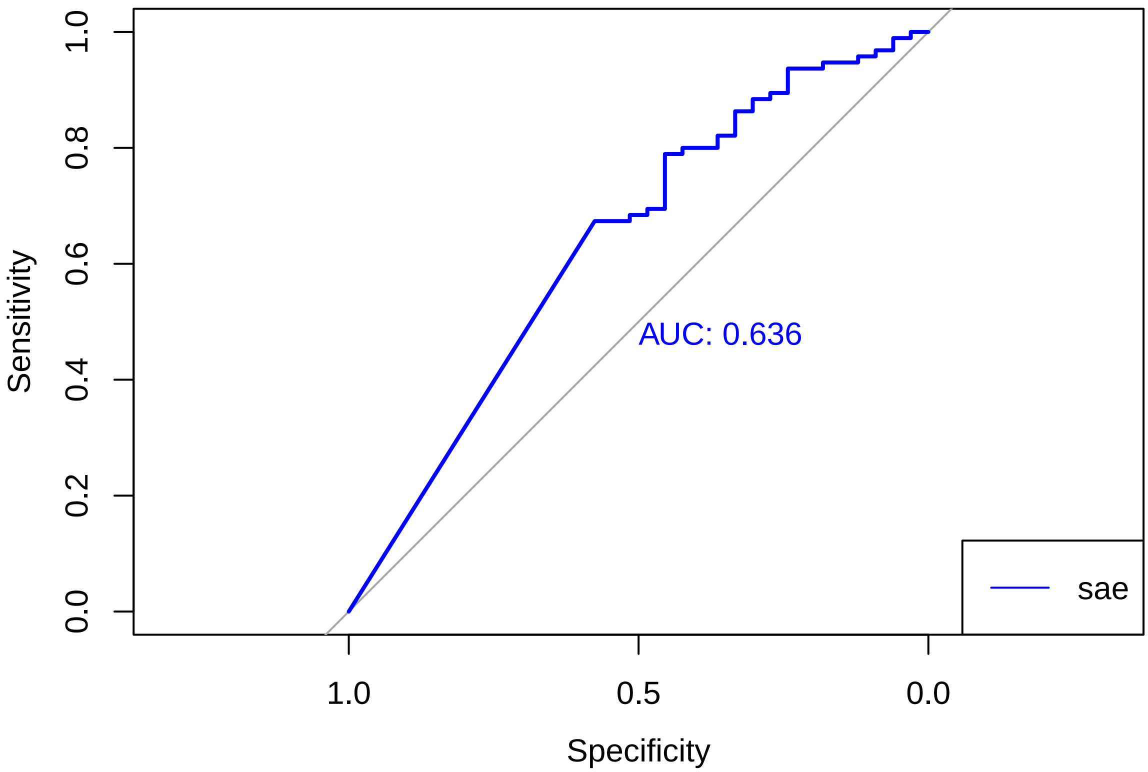
```r
roc_sae_test <- roc(response = ytestlabels, predictor =yhat)
```

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

```
## Warning in roc.default(response = ytestlabels, predictor = yhat): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.
```

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

```r
plot(roc_sae_test, col = "blue", print.auc=TRUE)
legend("bottomright", legend = c("sae"), lty = c(1), col = c("blue"))
```


Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
AUC: 0.636
sae
1.0
0.5
0.0
Specificity