

```

set.seed(10)
library(readxl)
creditcard <- read_excel("C:/Users/taeho/Documents/bankruptcy_data.xlsx")
# View(creditcard)

# number of predictors and observations
k=length(creditcard[,])-2
n=nrow(creditcard)

crx = sapply(creditcard[,c(-1,-95)], function(x) (x - min(x, na.rm = T)) / (max(x, na.rm = T)
- min(x, na.rm=T)))
df = cbind(creditcard[,1], crx)
names(df) <- c("Class", paste("V", 1:94, sep=""))
attach(df)

#now feed it to glm:

attach(df)

```

```

## The following objects are masked from df (pos = 3):
##
##      Class, V1, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V2,
##      V20, V21, V22, V23, V24, V25, V26, V27, V28, V29, V3, V30, V31,
##      V32, V33, V34, V35, V36, V37, V38, V39, V4, V40, V41, V42, V43,
##      V44, V45, V46, V47, V48, V49, V5, V50, V51, V52, V53, V54, V55,
##      V56, V57, V58, V59, V6, V60, V61, V62, V63, V64, V65, V66, V67,
##      V68, V69, V7, V70, V71, V72, V73, V74, V75, V76, V77, V78, V79, V8,
##      V80, V81, V82, V83, V84, V85, V86, V87, V88, V89, V9, V90, V91,
##      V92, V93, V94

```

```

glmAll =glm(Class~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+
V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28+V29+
V30+V31+V32+V33+V34+V35+V36+V37+V38+V39+V40+V41+V42+V43+
V44+V45+V46+V47+V48+V49+V50+
V51+V52+V53+V54+V55+V56+V57+V58+V59+V60+
V61+V62+V63+V64+V65+V66+V67+V68+V69+V70+
V71+V72+V73+V74+V75+V76+V77+V78+V79+V80+
V81+V82+V83+V84+V85+V86+V87+V88+V89+V90+
V91+V92+V93+V94
,data=df,family="binomial")

```

```
## Warning: glm.fit: 알고리즘이 수렴하지 않았습니다
```

```
## Warning: glm.fit: 적합된 확률값들이 0 또는 1 입니다
```

```
summary(glmAll)
```

```
##
## Call:
## glm(formula = Class ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +
##      V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 +
##      V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 + V28 +
##      V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 + V38 +
##      V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 + V48 +
##      V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 + V57 + V58 +
##      V59 + V60 + V61 + V62 + V63 + V64 + V65 + V66 + V67 + V68 +
##      V69 + V70 + V71 + V72 + V73 + V74 + V75 + V76 + V77 + V78 +
##      V79 + V80 + V81 + V82 + V83 + V84 + V85 + V86 + V87 + V88 +
##      V89 + V90 + V91 + V92 + V93 + V94, family = "binomial", data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q        Max
## -8.49      0.00      0.00      0.00      8.49
##
## Coefficients: (3 not defined because of singularities)
##              Estimate Std. Error   z value Pr(>|z|)
## (Intercept) -1.700e+22  4.542e+15  -3741815  <2e-16 ***
## V1           -1.455e+16  1.114e+08 -130580970  <2e-16 ***
## V2           -2.153e+15  8.205e+07 -26241622  <2e-16 ***
## V3            1.540e+16  1.339e+08  115018310  <2e-16 ***
## V4            3.856e+18  3.942e+11   9782398  <2e-16 ***
## V5           -2.728e+16  1.577e+09 -17303753  <2e-16 ***
## V6           -8.283e+19  1.414e+13  -5858048  <2e-16 ***
## V7            6.936e+19  1.181e+13   5873020  <2e-16 ***
## V8           -1.391e+17  2.552e+09 -54528903  <2e-16 ***
## V9           -3.963e+19  6.760e+12  -5862490  <2e-16 ***
## V10          -2.619e+16  1.179e+09 -22209751  <2e-16 ***
## V11           1.249e+13  2.817e+06   4433459  <2e-16 ***
## V12           1.067e+14  3.263e+06   32703803  <2e-16 ***
## V13          -1.411e+16  1.316e+08 -107202662  <2e-16 ***
## V14          -1.035e+15  7.589e+06 -136408001  <2e-16 ***
## V15           3.224e+14  6.757e+06   47712956  <2e-16 ***
## V16          -3.141e+16  6.809e+08  -46132450  <2e-16 ***
## V17           1.057e+16  1.519e+09   6960400  <2e-16 ***
## V18           1.713e+16  1.358e+09   12618115  <2e-16 ***
## V19          -1.358e+16  1.384e+08  -98097662  <2e-16 ***
## V20          -2.011e+15  9.505e+07 -21159436  <2e-16 ***
## V21           5.409e+15  9.933e+07  54454447  <2e-16 ***
## V22           1.290e+16  7.349e+08  17549516  <2e-16 ***
## V23           7.131e+15  1.142e+08  62456578  <2e-16 ***
## V24          -1.546e+14  6.861e+07  -2253546  <2e-16 ***
## V25           1.564e+15  1.051e+08  14879447  <2e-16 ***
## V26           2.639e+16  6.781e+08  38924057  <2e-16 ***
## V27          -2.554e+16  6.733e+08 -37934978  <2e-16 ***
## V28          -1.578e+15  8.171e+07  -19317417  <2e-16 ***
## V29          -5.942e+13  2.973e+06  -19987750  <2e-16 ***
## V30           3.394e+15  7.668e+07   44266010  <2e-16 ***
## V31          -1.973e+15  1.141e+08  -17297161  <2e-16 ***
## V32           2.742e+15  7.586e+07   36148605  <2e-16 ***
## V33          -2.413e+15  7.828e+07  -30821996  <2e-16 ***
## V34          -5.340e+14  3.157e+07  -16915624  <2e-16 ***
## V35          -3.104e+15  7.267e+07  -42718912  <2e-16 ***
```

| | | | | |
|--------|------------|-----------|------------|------------|
| ## V36 | 7.972e+15 | 6.018e+07 | 132470618 | <2e-16 *** |
| ## V37 | 7.680e+14 | 7.990e+07 | 9611885 | <2e-16 *** |
| ## V38 | NA | NA | NA | NA |
| ## V39 | -1.684e+15 | 3.503e+07 | -48080453 | <2e-16 *** |
| ## V40 | 5.835e+14 | 3.055e+08 | 1909832 | <2e-16 *** |
| ## V41 | -4.260e+15 | 2.097e+08 | -20315806 | <2e-16 *** |
| ## V42 | -9.165e+15 | 7.395e+08 | -12393888 | <2e-16 *** |
| ## V43 | 3.335e+15 | 1.449e+08 | 23018514 | <2e-16 *** |
| ## V44 | 9.443e+15 | 3.677e+08 | 25682502 | <2e-16 *** |
| ## V45 | -1.248e+15 | 1.973e+07 | -63245489 | <2e-16 *** |
| ## V46 | -6.615e+15 | 3.567e+07 | -185421745 | <2e-16 *** |
| ## V47 | -4.549e+15 | 3.529e+07 | -128921985 | <2e-16 *** |
| ## V48 | 7.260e+13 | 2.671e+06 | 27182094 | <2e-16 *** |
| ## V49 | 2.394e+13 | 3.800e+06 | 6299205 | <2e-16 *** |
| ## V50 | -3.959e+14 | 5.654e+07 | -7001849 | <2e-16 *** |
| ## V51 | -2.965e+15 | 9.582e+07 | -30948164 | <2e-16 *** |
| ## V52 | 7.500e+14 | 3.143e+07 | 23861024 | <2e-16 *** |
| ## V53 | -4.240e+14 | 2.834e+07 | -14957937 | <2e-16 *** |
| ## V54 | 2.266e+22 | 6.041e+15 | 3750556 | <2e-16 *** |
| ## V55 | 5.106e+14 | 8.885e+06 | 57471881 | <2e-16 *** |
| ## V56 | -6.205e+21 | 1.654e+15 | -3750556 | <2e-16 *** |
| ## V57 | -1.672e+15 | 1.034e+07 | -161759353 | <2e-16 *** |
| ## V58 | -1.932e+15 | 4.683e+07 | -41252795 | <2e-16 *** |
| ## V59 | -1.158e+14 | 1.587e+07 | -7295806 | <2e-16 *** |
| ## V60 | 2.021e+22 | 5.388e+15 | 3750556 | <2e-16 *** |
| ## V61 | 4.249e+15 | 6.797e+07 | 62507804 | <2e-16 *** |
| ## V62 | 1.238e+15 | 8.107e+07 | 15265286 | <2e-16 *** |
| ## V63 | -3.800e+14 | 1.434e+07 | -26488072 | <2e-16 *** |
| ## V64 | 1.076e+14 | 1.155e+07 | 9314621 | <2e-16 *** |
| ## V65 | -1.781e+16 | 4.566e+08 | -39014723 | <2e-16 *** |
| ## V66 | 1.101e+16 | 1.030e+09 | 10683080 | <2e-16 *** |
| ## V67 | -2.472e+14 | 1.380e+07 | -17913905 | <2e-16 *** |
| ## V68 | 6.806e+13 | 6.200e+07 | 1097755 | <2e-16 *** |
| ## V69 | 1.476e+15 | 7.788e+07 | 18950230 | <2e-16 *** |
| ## V70 | -6.024e+15 | 5.524e+07 | -109044202 | <2e-16 *** |
| ## V71 | 3.521e+13 | 3.548e+06 | 9922468 | <2e-16 *** |
| ## V72 | 5.569e+13 | 2.891e+06 | 19263642 | <2e-16 *** |
| ## V73 | 2.030e+16 | 8.448e+08 | 24029406 | <2e-16 *** |
| ## V74 | -1.948e+14 | 2.969e+06 | -65630206 | <2e-16 *** |
| ## V75 | 2.433e+16 | 4.489e+08 | 54186768 | <2e-16 *** |
| ## V76 | 2.307e+15 | 7.403e+07 | 31162336 | <2e-16 *** |
| ## V77 | NA | NA | NA | NA |
| ## V78 | NA | NA | NA | NA |
| ## V79 | 7.854e+15 | 2.803e+08 | 28023378 | <2e-16 *** |
| ## V80 | 1.239e+14 | 3.716e+07 | 3335713 | <2e-16 *** |
| ## V81 | -4.300e+15 | 4.561e+07 | -94272748 | <2e-16 *** |
| ## V82 | -1.430e+14 | 3.800e+07 | -3763565 | <2e-16 *** |
| ## V83 | -5.231e+14 | 1.155e+08 | -4530376 | <2e-16 *** |
| ## V84 | -6.571e+14 | 4.654e+07 | -14118467 | <2e-16 *** |
| ## V85 | 1.965e+15 | 3.563e+07 | 55146713 | <2e-16 *** |
| ## V86 | -3.767e+15 | 1.092e+08 | -34494464 | <2e-16 *** |
| ## V87 | -2.204e+14 | 2.216e+07 | -9944961 | <2e-16 *** |
| ## V88 | -1.253e+13 | 6.668e+07 | -187881 | <2e-16 *** |
| ## V89 | -3.823e+18 | 3.941e+11 | -9700396 | <2e-16 *** |
| ## V90 | -1.809e+15 | 1.665e+08 | -10860941 | <2e-16 *** |
| ## V91 | -3.106e+16 | 1.219e+09 | -25475937 | <2e-16 *** |

```
## V92          1.630e+15  5.200e+07  31353485  <2e-16 ***
## V93          5.417e+15  6.176e+07  87701515  <2e-16 ***
## V94         -4.936e+15  2.885e+07 -171080270  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1943.7  on 6818  degrees of freedom
## Residual deviance: 15498.8  on 6727  degrees of freedom
## AIC: 15683
##
## Number of Fisher Scoring iterations: 25
```

```
library(caret)
```

```
## 필요한 패키지를 로딩중입니다: ggplot2
```

```
## 필요한 패키지를 로딩중입니다: lattice
```

```
library(InformationValue)
```

```
##
## 다음의 패키지를 부착합니다: 'InformationValue'
```

```
## The following objects are masked from 'package:caret':
##
##      confusionMatrix, precision, sensitivity, specificity
```

```
library(ISLR)
```

```
confusionMatrix(predict(glmAll, type="response") >= 0.5, df$class)->tt
(tt[1,1]+tt[2,2])/sum(tt)*100
```

```
## [1] 96.84705
```

```
tt[1,1]/(tt[1,1]+tt[1,2])*100
```

```
## [1] 99.78785
```

```
tt[2,2]/(tt[2,1]+tt[2,2])*100
```

```
## [1] 8.636364
```

```

if (tensorflow::tf$executing_eagerly())
  tensorflow::tf$compat$v1$disable_eager_execution()

library(keras)
K <- keras::backend()
# training parameters
vae_batch_size = 160L
fnn_batch_size = 160L
epochs = 1L
vae_ep = 1L
fnn_ep = 7L
vae_flag = 0L

sel_pr_up = 1.0 # upper bound probability for VAE
sel_pr_dw = 0.0 # lower bound probability for VAE
sel_rate = 0.6

vae_w1 = 0.0
vae_w2 = 0.0
vae_w3 = 1.0

# latent and intermediate dimension
latent_dim = 2L
intermediate_dim = 10L
epsilon_std <- 0.1

# input image dimensions
input_shape = c(k+1)

```

```

model1 <- keras_model_sequential() %>%
  layer_dense(units = 1, activation = "sigmoid", input_shape = c(94))
# %>%      layer_dense(units = 1, activation = "sigmoid")

model1 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

```

```

history2 <- model1 %>% fit(
  as.matrix(df[, -1]), as.matrix(df[, 1]),
  shuffle = TRUE,
  epochs = fnn_ep, batch_size = fnn_batch_size,
  verbose = 0
)

temp3 <- predict(model1, as.matrix(df[, -1]))
confusionMatrix(temp3 >= 0.5, df[, 1]) -> tt3
tt3

```

```
## FALSE
## 0 6599
## 1 220
```

```
(tt3[1,1]+tt3[2,2])/(sum(tt3))*100
```

```
## numeric(0)
```

```
tt3[1,1]/(tt3[1,1]+tt3[1,2])*100
```

```
## numeric(0)
```

```
tt3[2,2]/(tt3[2,1]+tt3[2,2])*100
```

```
## numeric(0)
```

```
# 0의 범주를 갖는 행과 1의 범주를 갖는 행을 분리
df_class_0 <- df[df$Class == 0, ]
df_class_1 <- df[df$Class == 1, ]
nrow(df_class_0)
```

```
## [1] 6599
```

```
nrow(df_class_1)
```

```
## [1] 220
```

```
df_class_1 <- df_class_1[sample(nrow(df_class_1),nrow(df_class_0),replace=TRUE),]
oversample <- rbind(df_class_0, df_class_1)
oversample <- oversample[sample(nrow(oversample),nrow(oversample),replace=FALSE),]
```

```
glmOVER =glm(Class~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+
V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28+V29+
V30+V31+V32+V33+V34+V35+V36+V37+V38+V39+V40+V41+V42+V43+
V44+V45+V46+V47+V48+V49+V50+
V51+V52+V53+V54+V55+V56+V57+V58+V59+V60+
V61+V62+V63+V64+V65+V66+V67+V68+V69+V70+
V71+V72+V73+V74+V75+V76+V77+V78+V79+V80+
V81+V82+V83+V84+V85+V86+V87+V88+V89+V90+
V91+V92+V93+V94
,data=oversample,family="binomial")
```

```
## Warning: glm.fit: 알고리즘이 수렴하지 않았습니다
```

```
## Warning: glm.fit: 적합된 확률값들이 0 또는 1 입니다
```

```
summary(glmOVER)
```

```
##
## Call:
## glm(formula = Class ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +
##      V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 +
##      V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 + V28 +
##      V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 + V38 +
##      V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 + V48 +
##      V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 + V57 + V58 +
##      V59 + V60 + V61 + V62 + V63 + V64 + V65 + V66 + V67 + V68 +
##      V69 + V70 + V71 + V72 + V73 + V74 + V75 + V76 + V77 + V78 +
##      V79 + V80 + V81 + V82 + V83 + V84 + V85 + V86 + V87 + V88 +
##      V89 + V90 + V91 + V92 + V93 + V94, family = "binomial", data = oversample)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q        Max
## -8.49      0.00      0.00      0.00      8.49
##
## Coefficients: (3 not defined because of singularities)
##              Estimate Std. Error   z value Pr(>|z|)
## (Intercept)  6.047e+23  3.489e+15  173331495  <2e-16 ***
## V1           -9.288e+15  6.795e+07 -136692401  <2e-16 ***
## V2            6.341e+15  5.103e+07  124263426  <2e-16 ***
## V3            3.920e+15  8.849e+07   44306116  <2e-16 ***
## V4           -4.188e+19  2.943e+11 -142306838  <2e-16 ***
## V5            2.321e+17  1.261e+09  184145517  <2e-16 ***
## V6           -7.594e+20  1.080e+13  -70285695  <2e-16 ***
## V7            6.345e+20  9.024e+12   70314906  <2e-16 ***
## V8           -2.775e+17  2.348e+09 -118170277  <2e-16 ***
## V9           -3.630e+20  5.165e+12  -70289166  <2e-16 ***
## V10           2.211e+16  1.065e+09   20756937  <2e-16 ***
## V11           2.145e+14  2.057e+06  104299846  <2e-16 ***
## V12           1.762e+14  2.513e+06   70108015  <2e-16 ***
## V13          -5.326e+15  1.180e+08  -45141687  <2e-16 ***
## V14           9.124e+13  7.341e+06   12429049  <2e-16 ***
## V15           6.903e+14  5.194e+06  132919262  <2e-16 ***
## V16          -5.311e+16  5.204e+08 -102055800  <2e-16 ***
## V17           6.718e+16  8.160e+08   82326487  <2e-16 ***
## V18          -1.554e+16  6.282e+08  -24734550  <2e-16 ***
## V19          -2.340e+15  1.149e+08  -20355146  <2e-16 ***
## V20           8.233e+13  7.349e+07   1120194  <2e-16 ***
## V21          -4.926e+15  9.481e+07  -51954561  <2e-16 ***
## V22          -1.596e+15  6.130e+08  -2603746  <2e-16 ***
## V23          -8.935e+15  1.055e+08  -84677212  <2e-16 ***
## V24           1.285e+15  6.446e+07   19935639  <2e-16 ***
## V25          -4.018e+15  9.251e+07  -43433440  <2e-16 ***
## V26           2.571e+16  5.657e+08  45444773  <2e-16 ***
## V27          -1.865e+16  5.640e+08  -33068281  <2e-16 ***
## V28          -4.931e+14  7.751e+07   -6361911  <2e-16 ***
## V29           1.794e+14  2.418e+06   74206715  <2e-16 ***
## V30           2.420e+15  1.669e+07  145014924  <2e-16 ***
## V31          -9.874e+14  1.053e+08  -9376572  <2e-16 ***
## V32          -3.786e+14  2.498e+07  -15158243  <2e-16 ***
## V33          -1.015e+15  7.121e+07  -14253296  <2e-16 ***
## V34          -2.481e+15  1.768e+07 -140380411  <2e-16 ***
## V35           3.243e+15  4.600e+07   70510511  <2e-16 ***
```


| | | | | |
|--------|------------|-----------|------------|------------|
| ## V36 | 4.668e+15 | 5.059e+07 | 92272468 | <2e-16 *** |
| ## V37 | 1.111e+16 | 4.311e+07 | 257770505 | <2e-16 *** |
| ## V38 | NA | NA | NA | NA |
| ## V39 | 1.831e+14 | 3.022e+07 | 6057141 | <2e-16 *** |
| ## V40 | 3.255e+16 | 1.239e+08 | 262815124 | <2e-16 *** |
| ## V41 | 8.720e+15 | 8.906e+07 | 97912235 | <2e-16 *** |
| ## V42 | 1.375e+16 | 6.202e+08 | 22170860 | <2e-16 *** |
| ## V43 | -1.334e+15 | 1.340e+08 | -9956813 | <2e-16 *** |
| ## V44 | -1.071e+16 | 2.034e+08 | -52638098 | <2e-16 *** |
| ## V45 | -9.725e+14 | 1.488e+07 | -65356519 | <2e-16 *** |
| ## V46 | -8.429e+15 | 3.403e+07 | -247706063 | <2e-16 *** |
| ## V47 | -3.705e+15 | 3.502e+07 | -105784141 | <2e-16 *** |
| ## V48 | -1.976e+14 | 2.007e+06 | -98412020 | <2e-16 *** |
| ## V49 | -1.703e+14 | 2.494e+06 | -68266959 | <2e-16 *** |
| ## V50 | -3.911e+15 | 3.965e+07 | -98652910 | <2e-16 *** |
| ## V51 | -3.925e+15 | 6.732e+07 | -58302835 | <2e-16 *** |
| ## V52 | -2.178e+15 | 2.633e+07 | -82685507 | <2e-16 *** |
| ## V53 | -1.777e+15 | 2.791e+07 | -63670522 | <2e-16 *** |
| ## V54 | -8.037e+23 | 4.639e+15 | -173232412 | <2e-16 *** |
| ## V55 | -4.168e+14 | 6.289e+06 | -66278627 | <2e-16 *** |
| ## V56 | 2.201e+23 | 1.271e+15 | 173232413 | <2e-16 *** |
| ## V57 | -2.632e+15 | 8.269e+06 | -318347622 | <2e-16 *** |
| ## V58 | -3.778e+15 | 4.421e+07 | -85442843 | <2e-16 *** |
| ## V59 | -6.706e+14 | 6.709e+06 | -99965962 | <2e-16 *** |
| ## V60 | -7.169e+23 | 4.138e+15 | -173232413 | <2e-16 *** |
| ## V61 | 4.540e+15 | 5.635e+07 | 80565834 | <2e-16 *** |
| ## V62 | 1.381e+15 | 7.620e+07 | 18119174 | <2e-16 *** |
| ## V63 | -1.345e+15 | 1.111e+07 | -121090963 | <2e-16 *** |
| ## V64 | 1.036e+15 | 8.031e+06 | 129032044 | <2e-16 *** |
| ## V65 | 3.763e+15 | 2.522e+08 | 14920984 | <2e-16 *** |
| ## V66 | 5.262e+16 | 4.463e+08 | 117907972 | <2e-16 *** |
| ## V67 | -4.981e+14 | 9.653e+06 | -51600535 | <2e-16 *** |
| ## V68 | -1.448e+15 | 4.377e+07 | -33090259 | <2e-16 *** |
| ## V69 | 4.188e+15 | 7.676e+07 | 54561054 | <2e-16 *** |
| ## V70 | -2.783e+15 | 3.596e+07 | -77385726 | <2e-16 *** |
| ## V71 | 5.431e+14 | 2.535e+06 | 214220520 | <2e-16 *** |
| ## V72 | -4.421e+14 | 2.097e+06 | -210827850 | <2e-16 *** |
| ## V73 | 5.839e+16 | 7.926e+08 | 73673643 | <2e-16 *** |
| ## V74 | -5.544e+14 | 2.349e+06 | -235999462 | <2e-16 *** |
| ## V75 | 3.149e+16 | 4.329e+08 | 72747072 | <2e-16 *** |
| ## V76 | 3.063e+15 | 3.003e+07 | 101971947 | <2e-16 *** |
| ## V77 | NA | NA | NA | NA |
| ## V78 | NA | NA | NA | NA |
| ## V79 | 1.480e+16 | 1.378e+08 | 107406467 | <2e-16 *** |
| ## V80 | 2.671e+15 | 2.342e+07 | 114029537 | <2e-16 *** |
| ## V81 | -5.589e+15 | 3.438e+07 | -162564287 | <2e-16 *** |
| ## V82 | -2.927e+15 | 2.391e+07 | -122442674 | <2e-16 *** |
| ## V83 | -3.556e+15 | 6.017e+07 | -59101847 | <2e-16 *** |
| ## V84 | 2.504e+15 | 2.733e+07 | 91631349 | <2e-16 *** |
| ## V85 | 2.779e+14 | 1.098e+07 | 25301660 | <2e-16 *** |
| ## V86 | -8.091e+15 | 5.392e+07 | -150044073 | <2e-16 *** |
| ## V87 | 1.182e+15 | 1.113e+07 | 106241932 | <2e-16 *** |
| ## V88 | -1.600e+14 | 6.247e+07 | -2561270 | <2e-16 *** |
| ## V89 | 4.165e+19 | 2.943e+11 | 141532934 | <2e-16 *** |
| ## V90 | 9.163e+14 | 6.147e+07 | 14907418 | <2e-16 *** |
| ## V91 | -9.331e+16 | 5.912e+08 | -157831453 | <2e-16 *** |

```
## V92          1.857e+15  3.883e+07  47813605  <2e-16 ***
## V93          -3.101e+15  5.647e+07 -54906686  <2e-16 ***
## V94          -3.185e+15  2.548e+07 -124984168  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 18296  on 13197  degrees of freedom
## Residual deviance: 161115  on 13106  degrees of freedom
## AIC: 161299
##
## Number of Fisher Scoring iterations: 25
```

```
confusionMatrix(predict(glmOVER, type="response") >= 0.5, oversample$class)->tt
(tt[1,1]+tt[2,2])/sum(tt)*100
```

```
## [1] 83.06562
```

```
tt[1,1]/(tt[1,1]+tt[1,2])*100
```

```
## [1] 78.05728
```

```
tt[2,2]/(tt[2,1]+tt[2,2])*100
```

```
## [1] 88.07395
```

```
model1 <- keras_model_sequential() %>%
  layer_dense(units = 1, activation = "sigmoid", input_shape = c(94))
# %>%      layer_dense(units = 1, activation = "sigmoid")

model1 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

```
history2 <- model1 %>% fit(
  as.matrix(oversample[, -1]), as.matrix(oversample[, 1]),
  shuffle = TRUE,
  epochs = fnn_ep, batch_size = fnn_batch_size,
  verbose = 0
)
```

```
temp3<-predict(model1,as.matrix(oversample[, -1]))
confusionMatrix(temp3>=0.5, oversample[, 1]) -> tt3
tt3
```

```
## FALSE TRUE
## 0 5585 1014
## 1 1693 4906
```

```
(tt3[1,1]+tt3[2,2])/(sum(tt3))*100
```

```
## [1] 79.48932
```

```
tt3[1,1]/(tt3[1,1]+tt3[1,2])*100
```

```
## [1] 84.63404
```

```
tt3[2,2]/(tt3[2,1]+tt3[2,2])*100
```

```
## [1] 74.3446
```

```
##data partition##
df = df[sample(nrow(df),nrow(df),replace=FALSE),]
# 0의 범주를 갖는 행과 1의 범주를 갖는 행을 분리
df_class_0 <- df[df$Class == 0, ]
df_class_1 <- df[df$Class == 1, ]

# 0의 범주를 8대2 비율로 train과 test로 나눔
train_class_0_rows <- round(0.8 * nrow(df_class_0))
train_class_0 <- df_class_0[1:train_class_0_rows, ]
test_class_0 <- df_class_0[(train_class_0_rows + 1):nrow(df_class_0), ]

nrow(train_class_0)
```

```
## [1] 5279
```

```
# 1의 범주를 8대2 비율로 train과 test로 나눔

train_class_1_rows <- round(0.8 * nrow(df_class_1))
train_class_1 <- df_class_1[1:train_class_1_rows, ]
train_class_1 <- train_class_1[sample(nrow(train_class_1),nrow(train_class_0),replace=TRUE),]
test_class_1 <- df_class_1[(train_class_1_rows + 1):nrow(df_class_1), ]

# train과 test를 합쳐 최종 train_df와 test_df 생성
df <- rbind(train_class_0, train_class_1)
dfTS <- rbind(test_class_0, test_class_1)
dfTR0 = df[df$Class==0,]
overDF1 = df[df$Class==1,]
table(df$Class)
```

```
##
## 0 1
## 5279 5279
```

```
table(dfTS$Class)
```

```
##  
##      0      1  
## 1320    44
```

```
glmOver =glm(Class~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+  
              V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28+V29+  
              V30+V31+V32+V33+V34+V35+V36+V37+V38+V39+V40+V41+V42+V43+  
              V44+V45+V46+V47+V48+V49+V50+  
              V51+V52+V53+V54+V55+V56+V57+V58+V59+V60+  
              V61+V62+V63+V64+V65+V66+V67+V68+V69+V70+  
              V71+V72+V73+V74+V75+V76+V77+V78+V79+V80+  
              V81+V82+V83+V84+V85+V86+V87+V88+V89+V90+  
              V91+V92+V93+V94  
              ,data=df,family="binomial")
```

```
## Warning: glm.fit: 알고리즘이 수렴하지 않았습니다
```

```
## Warning: glm.fit: 적합된 확률값들이 0 또는 1 입니다
```

```
summary(glmOver)
```

```
##
## Call:
## glm(formula = Class ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +
##      V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 +
##      V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 + V28 +
##      V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 + V38 +
##      V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 + V48 +
##      V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 + V57 + V58 +
##      V59 + V60 + V61 + V62 + V63 + V64 + V65 + V66 + V67 + V68 +
##      V69 + V70 + V71 + V72 + V73 + V74 + V75 + V76 + V77 + V78 +
##      V79 + V80 + V81 + V82 + V83 + V84 + V85 + V86 + V87 + V88 +
##      V89 + V90 + V91 + V92 + V93 + V94, family = "binomial", data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q        Max
## -8.49      0.00      0.00      0.00      8.49
##
## Coefficients: (3 not defined because of singularities)
##              Estimate Std. Error   z value Pr(>|z|)
## (Intercept)  3.280e+23  3.960e+15   82847592  <2e-16 ***
## V1           -2.153e+16  9.277e+07  -232069100  <2e-16 ***
## V2           -4.131e+15  6.016e+07  -68672975   <2e-16 ***
## V3            2.330e+16  1.153e+08  202051933   <2e-16 ***
## V4           -6.644e+19  3.337e+11  -199108656  <2e-16 ***
## V5            6.701e+16  1.361e+09   49252961   <2e-16 ***
## V6           -5.183e+18  1.247e+13   -415540    <2e-16 ***
## V7            3.742e+18  1.042e+13    359227     <2e-16 ***
## V8            7.207e+17  4.218e+09  170850125   <2e-16 ***
## V9           -2.299e+18  5.963e+12   -385508    <2e-16 ***
## V10          -1.465e+17  2.633e+09  -55661875   <2e-16 ***
## V11          -2.717e+14  2.340e+06  -116113008  <2e-16 ***
## V12           2.223e+13  2.827e+06    7865128    <2e-16 ***
## V13          -1.761e+16  1.302e+08  -135245835  <2e-16 ***
## V14           4.134e+13  8.333e+06    4961096    <2e-16 ***
## V15           5.502e+14  5.938e+06   92663752   <2e-16 ***
## V16          -4.785e+16  5.759e+08  -83090783   <2e-16 ***
## V17           1.338e+16  8.944e+08   14957805   <2e-16 ***
## V18           3.149e+16  6.904e+08   45610095   <2e-16 ***
## V19          -2.423e+16  1.298e+08  -186687542  <2e-16 ***
## V20           3.747e+15  8.133e+07   46065078   <2e-16 ***
## V21          -1.335e+16  1.599e+08  -83510934   <2e-16 ***
## V22           1.523e+16  6.533e+08   23308984   <2e-16 ***
## V23           5.546e+15  1.174e+08   47246580   <2e-16 ***
## V24           4.228e+15  6.570e+07   64359179   <2e-16 ***
## V25           1.968e+15  1.012e+08   19439548   <2e-16 ***
## V26           1.844e+14  6.131e+08    300757    <2e-16 ***
## V27           9.829e+14  6.118e+08   1606628    <2e-16 ***
## V28           1.143e+15  7.785e+07   14680346   <2e-16 ***
## V29           2.609e+14  2.710e+06   96255878   <2e-16 ***
## V30           3.152e+15  1.886e+07   167135802  <2e-16 ***
## V31          -1.504e+15  1.083e+08  -13885563   <2e-16 ***
## V32          -6.460e+15  5.774e+07  -111880424  <2e-16 ***
## V33          -6.996e+25  1.858e+17  -376596471  <2e-16 ***
## V34          -5.929e+15  3.674e+07  -161374512  <2e-16 ***
## V35           2.170e+13  6.255e+07    346850    <2e-16 ***
```

| | | | | |
|--------|------------|-----------|------------|------------|
| ## V36 | 2.253e+16 | 5.522e+07 | 408062235 | <2e-16 *** |
| ## V37 | 5.771e+15 | 4.957e+07 | 116418980 | <2e-16 *** |
| ## V38 | NA | NA | NA | NA |
| ## V39 | 4.138e+14 | 3.105e+07 | 13326495 | <2e-16 *** |
| ## V40 | 3.269e+16 | 1.573e+08 | 207841551 | <2e-16 *** |
| ## V41 | 1.586e+16 | 1.188e+08 | 133495829 | <2e-16 *** |
| ## V42 | -2.889e+15 | 6.602e+08 | -4376163 | <2e-16 *** |
| ## V43 | 9.165e+15 | 1.438e+08 | 63720140 | <2e-16 *** |
| ## V44 | -1.898e+16 | 2.355e+08 | -80578044 | <2e-16 *** |
| ## V45 | 2.503e+15 | 1.752e+07 | 142858336 | <2e-16 *** |
| ## V46 | -1.398e+16 | 5.367e+07 | -260375225 | <2e-16 *** |
| ## V47 | -8.624e+15 | 3.945e+07 | -218568478 | <2e-16 *** |
| ## V48 | 2.801e+14 | 2.271e+06 | 123346836 | <2e-16 *** |
| ## V49 | -4.301e+14 | 2.877e+06 | -149514517 | <2e-16 *** |
| ## V50 | -1.402e+16 | 4.950e+07 | -283200258 | <2e-16 *** |
| ## V51 | 6.753e+16 | 3.229e+08 | 209121370 | <2e-16 *** |
| ## V52 | -1.741e+15 | 2.958e+07 | -58864562 | <2e-16 *** |
| ## V53 | -2.886e+15 | 3.498e+07 | -82505235 | <2e-16 *** |
| ## V54 | -4.362e+23 | 5.265e+15 | -82849026 | <2e-16 *** |
| ## V55 | -5.578e+14 | 7.142e+06 | -78099309 | <2e-16 *** |
| ## V56 | 1.195e+23 | 1.442e+15 | 82849027 | <2e-16 *** |
| ## V57 | -1.934e+15 | 9.260e+06 | -208855016 | <2e-16 *** |
| ## V58 | -4.882e+15 | 5.701e+07 | -85627887 | <2e-16 *** |
| ## V59 | -1.287e+14 | 8.280e+06 | -15542987 | <2e-16 *** |
| ## V60 | -3.891e+23 | 4.697e+15 | -82849025 | <2e-16 *** |
| ## V61 | 1.763e+15 | 6.171e+07 | 28570440 | <2e-16 *** |
| ## V62 | -2.706e+15 | 8.723e+07 | -31024708 | <2e-16 *** |
| ## V63 | 8.392e+14 | 1.229e+07 | 68311031 | <2e-16 *** |
| ## V64 | -6.013e+14 | 9.763e+06 | -61586712 | <2e-16 *** |
| ## V65 | 1.899e+16 | 2.977e+08 | 63797095 | <2e-16 *** |
| ## V66 | 1.024e+17 | 5.481e+08 | 186771253 | <2e-16 *** |
| ## V67 | 4.829e+14 | 1.074e+07 | 44978539 | <2e-16 *** |
| ## V68 | -1.354e+15 | 4.845e+07 | -27941008 | <2e-16 *** |
| ## V69 | 8.528e+15 | 7.911e+07 | 107798492 | <2e-16 *** |
| ## V70 | -6.024e+15 | 4.006e+07 | -150365318 | <2e-16 *** |
| ## V71 | 2.303e+14 | 2.874e+06 | 80124621 | <2e-16 *** |
| ## V72 | -4.041e+12 | 2.358e+06 | -1713941 | <2e-16 *** |
| ## V73 | 1.388e+17 | 1.543e+09 | 89951856 | <2e-16 *** |
| ## V74 | -7.375e+14 | 2.632e+06 | -280223522 | <2e-16 *** |
| ## V75 | 6.118e+17 | 1.132e+09 | 540566411 | <2e-16 *** |
| ## V76 | -1.140e+15 | 3.000e+07 | -38002315 | <2e-16 *** |
| ## V77 | NA | NA | NA | NA |
| ## V78 | NA | NA | NA | NA |
| ## V79 | 2.592e+16 | 1.688e+08 | 153557057 | <2e-16 *** |
| ## V80 | 1.021e+15 | 2.653e+07 | 38500629 | <2e-16 *** |
| ## V81 | -8.126e+15 | 3.785e+07 | -214674854 | <2e-16 *** |
| ## V82 | 2.445e+15 | 3.166e+07 | 77235853 | <2e-16 *** |
| ## V83 | 7.969e+14 | 6.884e+07 | 11576676 | <2e-16 *** |
| ## V84 | -1.745e+15 | 3.073e+07 | -56773139 | <2e-16 *** |
| ## V85 | 1.088e+15 | 1.280e+07 | 85038937 | <2e-16 *** |
| ## V86 | -6.677e+15 | 6.208e+07 | -107566151 | <2e-16 *** |
| ## V87 | -4.280e+14 | 1.559e+07 | -27459868 | <2e-16 *** |
| ## V88 | 2.271e+15 | 7.029e+07 | 32306263 | <2e-16 *** |
| ## V89 | 6.637e+19 | 3.337e+11 | 198881841 | <2e-16 *** |
| ## V90 | -4.711e+15 | 7.891e+07 | -59696608 | <2e-16 *** |
| ## V91 | -1.419e+17 | 7.359e+08 | -192784647 | <2e-16 *** |

```
## V92          -1.742e+15  3.952e+07  -44075752  <2e-16 ***
## V93          -1.346e+15  7.622e+07  -17653604  <2e-16 ***
## V94          -3.582e+15  3.409e+07  -105074226  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 14636  on 10557  degrees of freedom
## Residual deviance: 124855  on 10466  degrees of freedom
## AIC: 125039
##
## Number of Fisher Scoring iterations: 25
```

```
confusionMatrix(predict(glmOver, type="response") >= 0.5, df[,1])->tt
tt
```

```
##      FALSE TRUE
## 0    3943 1336
## 1     396 4883
```

```
(tt[1,1]+tt[2,2])/sum(tt)*100
```

```
## [1] 83.59538
```

```
tt[1,1]/(tt[1,1]+tt[1,2])*100
```

```
## [1] 74.69218
```

```
tt[2,2]/(tt[2,1]+tt[2,2])*100
```

```
## [1] 92.49858
```

```
confusionMatrix(predict(glmOver, as.data.frame(dfTS),type="response") >= 0.5, dfTS[,1])->tt1
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
tt1
```

```
##      FALSE TRUE
## 0     970   350
## 1        8    36
```

```
(tt1[1,1]+tt1[2,2])/sum(tt1)*100
```

```
## [1] 73.75367
```

```
tt1[1,1]/(tt1[1,1]+tt1[1,2])*100
```

```
## [1] 73.48485
```

```
tt1[2,2]/(tt1[2,1]+tt1[2,2])*100
```

```
## [1] 81.81818
```

```
table(dfTS$Class)
```

```
##  
##      0      1  
## 1320    44
```



```
#####VAE fitting#####

if (tensorflow::tf$executing_eagerly())
  tensorflow::tf$compat$v1$disable_eager_execution()

library(keras)
K <- keras::backend()

# training parameters
vae_batch_size = 160L
fnn_batch_size = 160L
epochs = 1L
vae_ep = 1L
fnn_ep = 7L
vae_flag = 0L

sel_pr_up = 1.0 # upper bound probability for VAE
sel_pr_dw = 0.0 # lower bound probability for VAE
sel_rate = 0.6

vae_w1 = 0.0
vae_w2 = 0.0
vae_w3 = 1.0

# latent and intermediate dimension
latent_dim = 2L
intermediate_dim = 10L
epsilon_std <- 0.1

# input image dimensions
input_shape = c(k+1)

# encoder
original_input_size = c(k+1)
inp <- layer_input(shape = original_input_size)
x <- layer_lambda(inp, f=function(x) {x[,2:(k+1)]})
y <- layer_lambda(inp, f=function(x) {x[,1:1]})

hidden_1 <- layer_dense(x, units=intermediate_dim, activation="relu")
dropout_1 <- layer_dropout(hidden_1, rate = 0.5)
hidden_2 <- layer_dense(dropout_1, units=intermediate_dim, activation="relu")
dropout_2 <- layer_dropout(hidden_2, rate = 0.5)

z_mean = layer_dense(dropout_2, units = latent_dim)
z_log_var <- layer_dense(hidden_2, units = latent_dim)

# sampling part
sampling <- function(args) {
  z_mean <- args[, 1:(latent_dim)]
  z_log_var <- args[, (latent_dim + 1):(2 * latent_dim)]

  epsilon <- k_random_normal(
```

```

    shape = c(k_shape(z_mean)[[1]]),
    mean = 0.,
    stddev = epsilon_std
  )
  z_mean + k_exp(z_log_var) * epsilon
}

z <- layer_concatenate(list(z_mean, z_log_var)) %>% layer_lambda(sampling)

# decoder + prediction model
output_shape = c(vae_batch_size, k)

decoder_hidden = layer_dense(units=intermediate_dim, activation="relu")
decoder_upsample = layer_dense(units = intermediate_dim, activation="relu")
decoder_reshape <- layer_reshape(target_shape = intermediate_dim)
decoder_hidden1 = layer_dense(units=k, activation="sigmoid")

pred_layer = layer_dense(units = 1, activation = "sigmoid")

hidden_decoded = decoder_hidden(z)
up_decoded = decoder_upsample(hidden_decoded)
reshape_decoded <- decoder_reshape(up_decoded)
hidden1_decoded = decoder_hidden1(reshape_decoded)

y_pred =pred_layer(hidden1_decoded)

vae_loss <- function(y, y_pred) {
  x <- k_flatten(x)
  x_decoded_mean_squash <- k_flatten(hidden1_decoded)
  xent_loss <- 1.0 * # initial weight = 1
    loss_mean_squared_error(x, x_decoded_mean_squash) # loss_categorical_crossentropy도 시도해
볼 것
  kl_loss <- -0.5 * k_mean(1 + z_log_var - k_square(z_mean) - # initial weight = -0.5
    k_exp(z_log_var), axis = -1L)
  p_loss <- 1.0 * loss_binary_crossentropy(y, y_pred) # initial weight = 0 * 12000

  k_mean(xent_loss*vae_w1 + kl_loss*vae_w2 + p_loss*vae_w3)
}

vae <- keras_model(inp, y_pred)
optimizers <- keras::keras$optimizers
vae %>% compile(optimizer = optimizers$legacy$RMSprop(learning_rate=0.0001), loss = vae_loss,
  metrics = c("accuracy"))
# summary(vae)

## encoder: model to project inputs on the latent space
# encoder <- keras_model(inp, list(z_mean, z_log_var))

## build a digit generator that can sample from the learned distribution
# gen_decoder_input <- layer_input(shape = latent_dim)
# gen_hidden_decoded <- decoder_hidden(gen_decoder_input)
# gen_up_decoded <- decoder_upsample(gen_hidden_decoded)
# gen_hidden1_decoded <- decoder_hidden1(gen_up_decoded)

# generator <- keras_model(gen_decoder_input, gen_hidden1_decoded)

```

```
vae1 <- keras_model(inp, hidden1_decoded) # can be used for generating synthetic samples for case 0 and 1
```

```
model1 <- keras_model_sequential() %>%  
  layer_dense(units = 1, activation = "sigmoid", input_shape = c(94))  
# %>%      layer_dense(units = 1, activation = "sigmoid")  
  
model1 %>% compile(  
  optimizer = "rmsprop",  
  loss = "binary_crossentropy",  
  metrics = c("accuracy")  
)
```

```
history2 <- model1 %>% fit(  
  as.matrix(df[, -1]), as.matrix(df[, 1]),  
  shuffle = TRUE,  
  epochs = fnn_ep, batch_size = fnn_batch_size,  
  validation_data = list(as.matrix(dfTS[, -1]), as.matrix(dfTS[, 1]))  
  , verbose = 0  
)  
  
temp3 <- predict(model1, as.matrix(df[, -1]))  
confusionMatrix(temp3 >= 0.5, df[, 1]) -> tt3  
tt3
```

```
##      FALSE TRUE  
## 0   4334   945  
## 1    972 4307
```

```
(tt3[1,1]+tt3[2,2])/(sum(tt3))*100
```

```
## [1] 81.84315
```

```
tt3[1,1]/(tt3[1,1]+tt3[1,2])*100
```

```
## [1] 82.09888
```

```
tt3[2,2]/(tt3[2,1]+tt3[2,2])*100
```

```
## [1] 81.58742
```

```
temp3 <- predict(model1, as.matrix(dfTS[, -1]))  
confusionMatrix(temp3 >= 0.5, dfTS[, 1]) -> tt3  
tt3
```

```
## FALSE TRUE
## 0 1073 247
## 1 6 38
```

```
(tt3[1,1]+tt3[2,2])/(sum(tt3))*100
```

```
## [1] 81.45161
```

```
tt3[1,1]/(tt3[1,1]+tt3[1,2])*100
```

```
## [1] 81.28788
```

```
tt3[2,2]/(tt3[2,1]+tt3[2,2])*100
```

```
## [1] 86.36364
```

```

# j : number of epoch
# i : number of batchs for one epoch

for (j in 1:epochs) {

# FNN MODEL FITTING

model1 <- keras_model_sequential() %>%
  layer_dense(units = 1, activation = "sigmoid", input_shape = c(94))
  # %>%      layer_dense(units = 1, activation = "sigmoid")

model1 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

# Insert VAE part here if needed

if(vae_flag == 1){
history = vae %>% fit(
  as.matrix(df), as.matrix(df[,1]),
  shuffle = TRUE,
  epochs = vae_ep,
  batch_size = vae_batch_size,
  validation_data = list(as.matrix(dfTS), as.matrix(dfTS[,1])),
  verbose = 0
)
}

# whole train and test data preparation

library(dplyr)

temp0 <- predict(vae1, as.matrix(df))
temp <- predict(vae, as.matrix(df))
temp1 <- as.data.frame(cbind(c(1), temp0[temp<=quantile(temp, sel_pr_up) &
temp>=quantile(temp, sel_pr_dw),]))
names(temp1) = names(dfTRO)
samp_ind = sample(1:nrow(temp), size = round(nrow(temp)*sel_rate))
temp1 <- temp1[samp_ind,]

temp2 <- dfTRO %>% sample_frac(nrow(temp1)/nrow(dfTRO), replace = TRUE)
train_df <-rbind(temp2, dfTRO, overDF1, temp1)
train_df <- train_df[sample(1:nrow(train_df)),]

# print(i)

## ---- Fitting -----

history2 <- model1 %>% fit(
  as.matrix(train_df[,,-1]), as.matrix(train_df[,1]),
  shuffle = TRUE,
  epochs = fnn_ep, batch_size = fnn_batch_size,

```

```
validation_data = list(as.matrix(dfTS[,-1]), as.matrix(dfTS[,1]))
, verbose = 0
)

print("FNN")
print(history2)
# plot(history2)
print("VAE")
if(vae_flag == 1){
  print(history)}
#plot(history)
print(j)
}
```

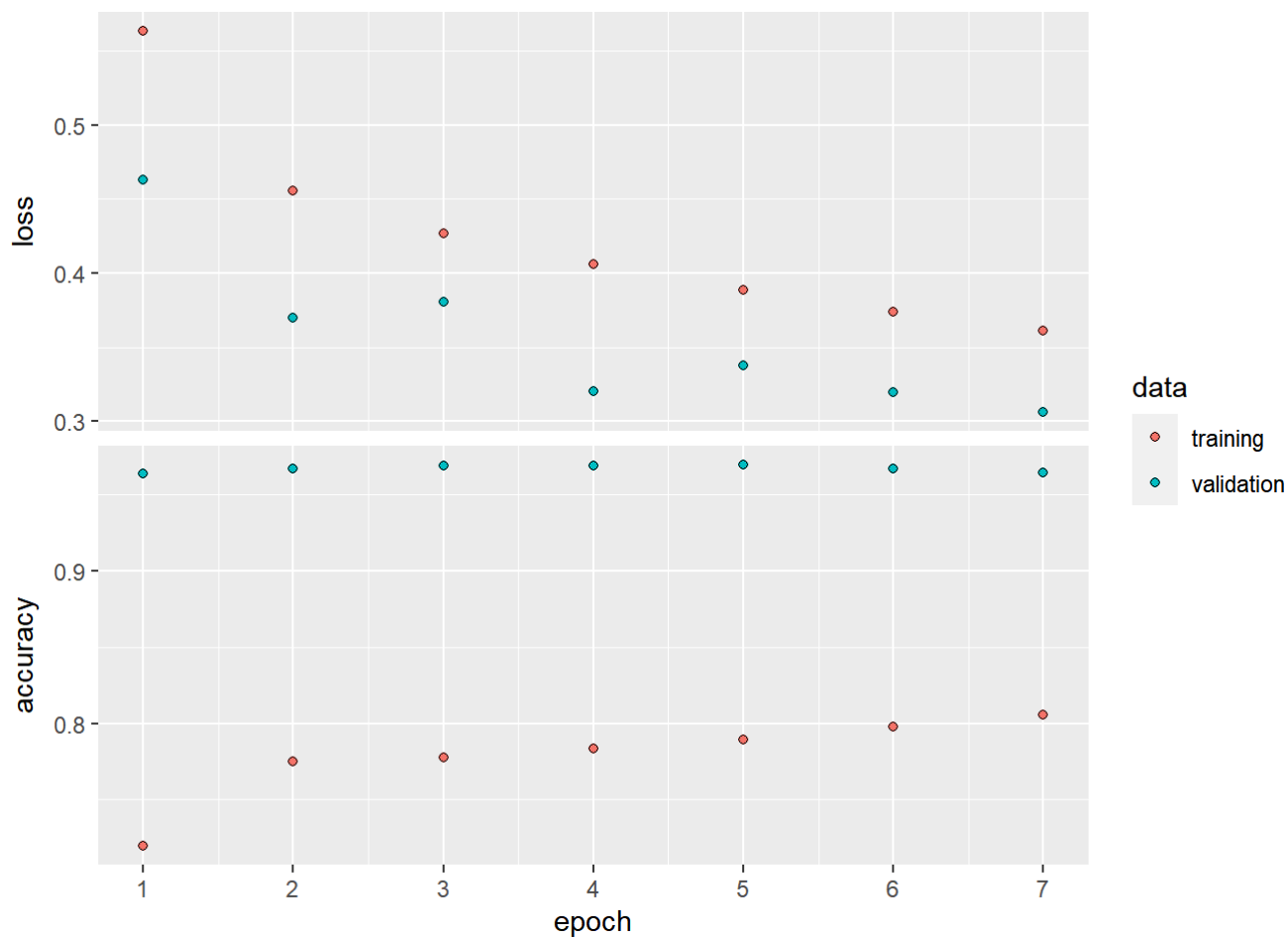
```
##
## 다음의 패키지를 부착합니다: 'dplyr'
```

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

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

```
## [1] "FNN"
## Trained on 23,228 samples (batch_size=160, epochs=7)
## Final epoch (plot to see history):
##      loss: 0.3613
##   accuracy: 0.8057
##   val_loss: 0.306
## val_accuracy: 0.9641
## [1] "VAE"
## [1] 1
```

```
if(vae_flag == 1){
  plot(history)
}
plot(history2)
```



```
temp3 <- predict(vae, as.matrix(dfTS))
confusionMatrix(temp3>=0.5, dfTS[,1]) -> tt
tt
```

```
## FALSE
## 0 1320
## 1 44
```

```
(tt[1,1]+tt[2,2])/(sum(tt))*100
```

```
## numeric(0)
```

```
tt[1,1]/(tt[1,1]+tt[1,2])*100
```

```
## numeric(0)
```

```
tt[2,2]/(tt[2,1]+tt[2,2])*100
```

```
## numeric(0)
```

```
temp3 <- predict(model1, as.matrix(dfTS[,,-1]))
confusionMatrix(temp3>=0.5, dfTS[,1]) -> tt3
tt3
```

```
## FALSE TRUE
## 0 1305 15
## 1 34 10
```

```
(tt3[1,1]+tt3[2,2])/(sum(tt3))*100
```

```
## [1] 96.40762
```

accuracy for case 0

```
tt3[1,1]/(tt3[1,1]+tt3[1,2])*100
```

```
## [1] 98.86364
```

accuracy for case 1

```
tt3[2,2]/(tt3[2,1]+tt3[2,2])*100
```

```
## [1] 22.72727
```