Assignment 5

Group 2

2023-11-13

Task 1

Task 2

Task 3

Task 4

Task 5

Task 6

Task 7

```
suppressMessages(if(!requireNamespace("ElemStatLearn")) {
  URL <- "https://cran.r-project.org/src/contrib/Archive/ElemStatLearn"
  install.packages(file.path(URL, "ElemStatLearn_2015.6.26.2.tar.gz"))})
suppressMessages(if(!require(mboost)) install.packages("mboost"))
suppressMessages(if(!require(caret)) install.packages("caret"))
suppressMessages(if(!require(pROC)) install.packages("pROC"))

data("SAheart", package="ElemStatLearn")
set.seed(123)
df=SAheart

#Train-test split (75%-25%)
df$id = 1:nrow(df)
train = df %>% dplyr::sample_frac(0.75)
test = dplyr::anti_join(df, train, by = 'id')
train = train[-c(11)]
test = test[-c(11)]
train$chd = as.factor(train$chd)
test$chd = as.factor(test$chd)


#Logistic regression model with backward-stepwise regression
train_x = train[, -which(names(train) == "chd")]
train_y = train$chd
test_x = test[, -which(names(test) == "chd")]
```

```r
test_y = test$chd

lin_model = step(glm(chd ~ ., data = train, family = binomial()), direction="backward")
```

```
## Start:  AIC=349.41
## chd ~ sbp + tobacco + ldl + adiposity + famhist + typea + obesity +
##     alcohol + age
##
##             Df Deviance    AIC
## - adiposity  1   330.14 348.14
## - alcohol    1   330.25 348.25
## - sbp        1   330.73 348.73
## - obesity    1   331.09 349.09
## <none>           329.41 349.41
## - age        1   333.48 351.48
## - famhist    1   336.88 354.88
## - typea      1   341.59 359.59
## - tobacco    1   347.19 365.19
## - ldl        1   347.73 365.73
##
## Step:  AIC=348.14
## chd ~ sbp + tobacco + ldl + famhist + typea + obesity + alcohol +
##     age
##
##           Df Deviance    AIC
## - alcohol  1   331.03 347.03
## - obesity  1   331.11 347.11
## - sbp      1   331.56 347.56
## <none>         330.14 348.14
## - famhist  1   337.86 353.86
## - age      1   338.29 354.29
## - typea    1   342.27 358.27
## - tobacco  1   347.69 363.69
## - ldl      1   350.84 366.84
##
## Step:  AIC=347.03
## chd ~ sbp + tobacco + ldl + famhist + typea + obesity + age
##
##           Df Deviance    AIC
## - obesity  1   332.10 346.10
## - sbp      1   332.88 346.88
## <none>         331.03 347.03
## - age      1   338.91 352.91
## - famhist  1   339.24 353.24
## - typea    1   343.32 357.32
## - tobacco  1   350.60 364.60
## - ldl      1   350.99 364.99
##
## Step:  AIC=346.1
## chd ~ sbp + tobacco + ldl + famhist + typea + age
##
##           Df Deviance    AIC
## - sbp      1   333.55 345.55
```

```
## <none>         332.10 346.10
## - age      1   339.55 351.55
## - famhist  1   340.07 352.07
## - typea    1   343.81 355.81
## - ldl      1   351.04 363.04
## - tobacco  1   351.66 363.66
##
## Step:  AIC=345.55
## chd ~ tobacco + ldl + famhist + typea + age
##
##           Df Deviance    AIC
## <none>         333.55 345.55
## - famhist  1   341.53 351.53
## - age      1   343.72 353.72
## - typea    1   345.38 355.38
## - tobacco  1   352.91 362.91
## - ldl      1   353.75 363.75
```

```
summary(lin_model)
```

```
##
## Call:
## glm(formula = chd ~ tobacco + ldl + famhist + typea + age, family = binomial(),
##     data = train)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -7.41093    1.14942  -6.448 1.14e-10 ***
## tobacco         0.14181    0.03543   4.003 6.26e-05 ***
## ldl             0.30444    0.07150   4.258 2.06e-05 ***
## famhistPresent  0.77215    0.27386   2.819 0.004810 **
## typea           0.04899    0.01486   3.298 0.000975 ***
## age             0.03766    0.01209   3.116 0.001834 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 445.38  on 345  degrees of freedom
## Residual deviance: 333.55  on 340  degrees of freedom
## AIC: 345.55
##
## Number of Fisher Scoring iterations: 5
```

```
boost_model = gamboost(chd ~ sbp+tobacco+ldl+adiposity+typea+obesity+alcohol+famhist+age, family=Binomia
```

```
## Warning in bbs(as.data.frame(list(...)), df = dfbase): cannot compute 'bbs' for
## non-numeric variables; used 'bols' instead.
```

```
summary(boost_model)
```

```
##
```

```
##   Model-based Boosting
##
## Call:
## gamboost(formula = chd ~ sbp + tobacco + ldl + adiposity + typea +     obesity + alcohol + famhist +
##
##
##   Negative Binomial Likelihood (logit link)
##
## Loss function: {
##      f <- pmin(abs(f), 36) * sign(f)
##      p <- exp(f)/(exp(f) + exp(-f))
##      y <- (y + 1)/2
##      -y * log(p) - (1 - y) * log(1 - p)
##  }
##
##
## Number of boosting iterations: mstop = 100
## Step size:  0.1
## Offset:  -0.3229133
## Number of baselearners:  9
##
## Selection frequencies:
##              bbs(typea, df = dfbase)                 bbs(age, df = dfbase)
##                                 0.19                                  0.18
##            bbs(tobacco, df = dfbase)                 bbs(ldl, df = dfbase)
##                                 0.17                                  0.15
##                bbs(sbp, df = dfbase)             bbs(obesity, df = dfbase)
##                                 0.11                                  0.09
## bols(famhist, by = by, index = index)             bbs(alcohol, df = dfbase)
##                                 0.09                                  0.02
```

```r
preds_lin = ifelse(predict(lin_model, newdata = test, type = "response") > 0.5, 1, 0)
preds_boost = ifelse(predict(boost_model, newdata = test, type = "response") > 0.5, 1, 0)
```

```
## Warning in bsplines(mf[[i]], knots = args$knots[[i]]$knots, boundary.knots =
## args$knots[[i]]$boundary.knots, : Some 'x' values are beyond 'boundary.knots';
## Linear extrapolation used.

## Warning in bsplines(mf[[i]], knots = args$knots[[i]]$knots, boundary.knots =
## args$knots[[i]]$boundary.knots, : Some 'x' values are beyond 'boundary.knots';
## Linear extrapolation used.

## Warning in bsplines(mf[[i]], knots = args$knots[[i]]$knots, boundary.knots =
## args$knots[[i]]$boundary.knots, : Some 'x' values are beyond 'boundary.knots';
## Linear extrapolation used.
```

```r
#Predictive performance of the logistic regression model
##Confusion Matrix
confusionMatrix(table(preds_lin, test_y))
```

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

```
##          test_y
## preds_lin  0  1
##         0 61 26
##         1 14 15
##
##                Accuracy : 0.6552
##                  95% CI : (0.5612, 0.741)
##     No Information Rate : 0.6466
##     P-Value [Acc > NIR] : 0.46510
##
##                   Kappa : 0.1919
##
##  Mcnemar's Test P-Value : 0.08199
##
##             Sensitivity : 0.8133
##             Specificity : 0.3659
##          Pos Pred Value : 0.7011
##          Neg Pred Value : 0.5172
##              Prevalence : 0.6466
##          Detection Rate : 0.5259
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.5896
##
##        'Positive' Class : 0
##
```

```r
##AUC
auc_lin <- auc(roc(test_y, preds_lin))
```

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

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

```r
cat("AUC score of the logistic regression model: ", auc_lin)
```

```
## AUC score of the logistic regression model:  0.5895935
```

```r
#Predictive performance of the boosted logistic regression model
##Confusion Matrix
confusionMatrix(table(preds_boost, test_y))
```

```
## Confusion Matrix and Statistics
##
##            test_y
## preds_boost  0  1
##           0 60 22
##           1 15 19
##
##                Accuracy : 0.681
##                  95% CI : (0.5881, 0.7645)
##     No Information Rate : 0.6466
```

```
##     P-Value [Acc > NIR] : 0.2500
##
##                   Kappa : 0.274
##
##  Mcnemar's Test P-Value : 0.3239
##
##             Sensitivity : 0.8000
##             Specificity : 0.4634
##          Pos Pred Value : 0.7317
##          Neg Pred Value : 0.5588
##              Prevalence : 0.6466
##          Detection Rate : 0.5172
##    Detection Prevalence : 0.7069
##       Balanced Accuracy : 0.6317
##
##        'Positive' Class : 0
##
```

```r
##AUC
auc_boost <- auc(roc(test_y, preds_boost))
```

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

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

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

```r
cat("AUC score of the boosted regression model: ", auc_boost)
```

```
## AUC score of the boosted regression model:  0.6317073
```

Comparing the confusion matrices, we can see that the boosted logistic regression model performs better with a higher accuracy. In addition to that, AUC score for boosted model is higher meaning that the boosted model's discriminatory power is higher.

Task 8

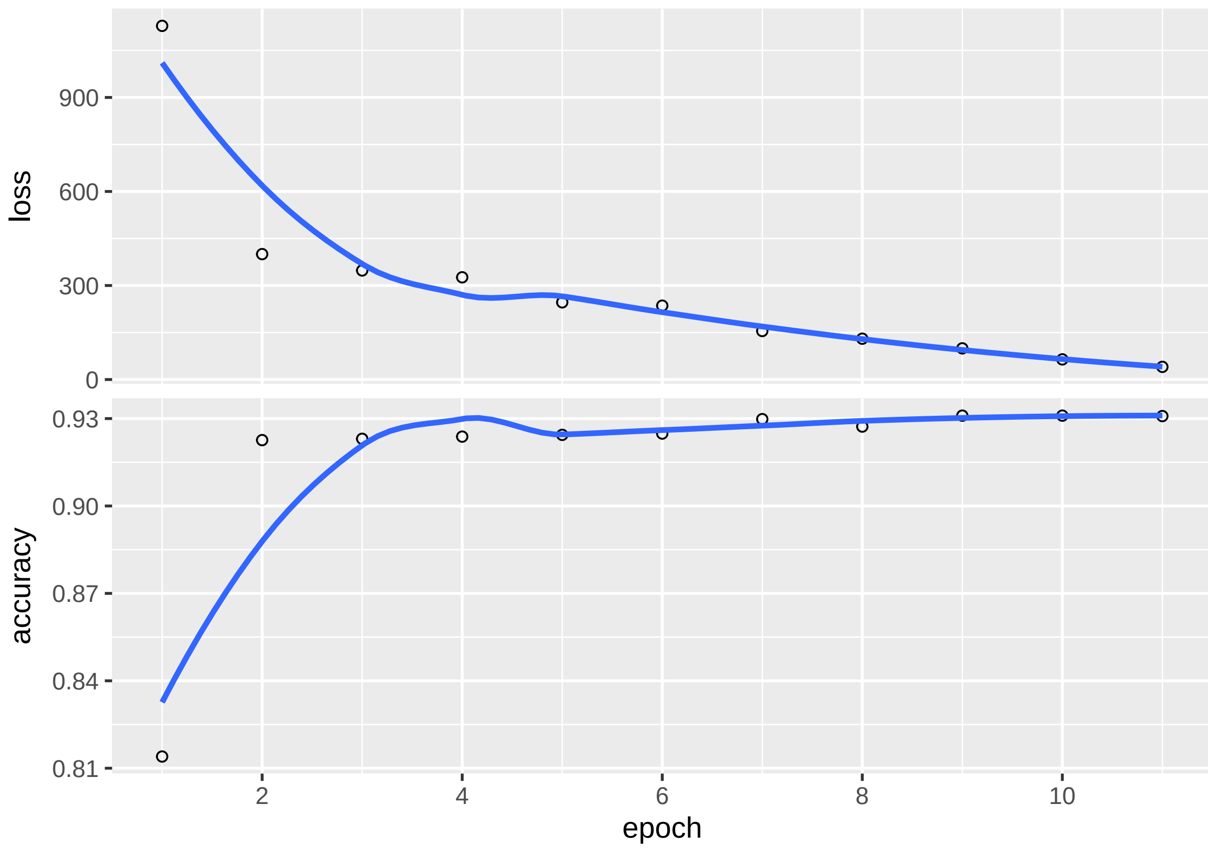
Task 9

```r
suppressMessages(if(!require(ISLR2)) install.packages("ISLR2"))
suppressMessages(if(!require(keras)) install.packages("keras"))
suppressMessages(if(!require(tensorflow)) install.packages("tensorflow"))
suppressMessages(if(!require(glmnet)) install.packages("glmnet"))

data("Default", package = "ISLR")
df=Default

#Train-test split
```

```r
set.seed(123)
n = nrow(Default)
ntest = trunc(n/3)
testid = sample(1:n, ntest)

x = model.matrix(default ~. -1, data=df)

train_x = x[-testid,]
test_x = x[testid,]

train_y = df$default[-testid]=='Yes'
test_y = df$default[testid] == 'Yes'

#Fit a neural network
nn_model = keras_model_sequential() %>%
  layer_dense(units=10, activation='relu', input_shape=ncol(x)) %>%
  layer_dropout(rate=0.4) %>%
  layer_dense(units = 1, activation='sigmoid')

nn_model %>% compile(
  optimizer=optimizer_rmsprop(),
  loss='binary_crossentropy',
  metrics='accuracy')

nn_fit = nn_model %>% fit(
  x = train_x,
  y = train_y,
  epochs=11,
  batch_size=32)
```

```
## Epoch 1/11
## 209/209 - 2s - loss: 1128.3599 - accuracy: 0.8140 - 2s/epoch - 10ms/step
## Epoch 2/11
## 209/209 - 2s - loss: 400.1223 - accuracy: 0.9226 - 2s/epoch - 8ms/step
## Epoch 3/11
## 209/209 - 2s - loss: 347.9744 - accuracy: 0.9231 - 2s/epoch - 7ms/step
## Epoch 4/11
## 209/209 - 2s - loss: 326.1497 - accuracy: 0.9238 - 2s/epoch - 8ms/step
## Epoch 5/11
## 209/209 - 2s - loss: 246.2815 - accuracy: 0.9244 - 2s/epoch - 8ms/step
## Epoch 6/11
## 209/209 - 2s - loss: 235.6524 - accuracy: 0.9249 - 2s/epoch - 8ms/step
## Epoch 7/11
## 209/209 - 2s - loss: 154.7975 - accuracy: 0.9298 - 2s/epoch - 8ms/step
## Epoch 8/11
## 209/209 - 2s - loss: 130.2855 - accuracy: 0.9273 - 2s/epoch - 8ms/step
## Epoch 9/11
## 209/209 - 2s - loss: 99.4745 - accuracy: 0.9310 - 2s/epoch - 8ms/step
## Epoch 10/11
## 209/209 - 2s - loss: 64.3028 - accuracy: 0.9310 - 2s/epoch - 7ms/step
## Epoch 11/11
## 209/209 - 2s - loss: 40.4479 - accuracy: 0.9309 - 2s/epoch - 7ms/step
```

```r
plot(nn_fit)
```




```r
preds_nn = predict(nn_model, test_x)
```

```
## 105/105 - 0s - 173ms/epoch - 2ms/step
```

```r
#Linear logistic regression
lin_model = glm(default~.,family="binomial", data=df[-testid,])
preds_lin = predict(lin_model,newdata=df[testid,])


#Comparison - Accuracy
##Linear Regression
acc_lin <- mean(as.numeric(preds_lin > 0.5) == test_y)
cat("Accuracy of the linear model: ", acc_lin)
```

```
## Accuracy of the linear model:  0.9714971
```

```r
##Neural Network
acc_nn <- mean(as.numeric(preds_nn > 0.5) == test_y)
cat("Accuracy of the neural network model: ", acc_nn)
```

```
## Accuracy of the neural network model:  0.9573957
```

```r
#Comparison - Brier Scores
##Linear Regression
brier_lin <- data.frame(test_y ,preds_lin=as.numeric(preds_lin > 0.5))
brier_lin$sq_difference <- (brier_lin$preds_lin-brier_lin$test_y)^2
brier_score_lin <- mean(brier_lin$sq_difference)
cat("Brier score of the linear model: ", brier_score_lin)
```

```
## Brier score of the linear model:  0.02850285
```

```r
##Neural Network
brier_nn <- data.frame(test_y ,preds_nn=as.numeric(preds_nn > 0.5))
brier_nn$sq_difference <- (brier_nn$preds_nn-brier_nn$test_y)^2
brier_score_nn <- mean(brier_nn$sq_difference)
cat("Brier score of the neural network model: ", brier_score_nn)
```

```
## Brier score of the neural network model:  0.04260426
```

Comparing the accuracy of the two models, we can see that the linear regression model performs better with a higher accuracy. In addition to that, Brier score of the linear regression model is lower. One explanation for that is, the datasets that we used are imbalanced datasets and we do have an imbalanced dataset as can be seen from the tables below.

```r
table(df$default)
```

```
##
##   No  Yes
## 9667  333
```

```r
table(train_y)
```

```
## train_y
## FALSE  TRUE
##  6452   215
```

```r
table(test_y)
```

```
## test_y
## FALSE  TRUE
##  3215   118
```

Task 10