Homework 4: Binary Classification

Thomas Zwiller

There are four questions (30 total points) in this assignment. The minimum increment is 1 point. Please type in your answers directly in the R Markdown file. After completion, **successfully** knitr it as an html file. Submit **both** the html file and the R Markdown file via Canvas. Please name the R Markdown file in the following format: LastName_FirstName_HW4.Rmd, e.g. Zhao_Zifeng_HW4.Rmd.

Adult Income Dataset [30 points]

The adult income dataset contains information about 9755 adults from the 2010 U.S. Census database. The data is stored in `Adult_Income.csv`. It contains 8 variables, `age`, `workclass`, `education_num`, `marital_status`, `capital_gain`, `capital_loss`, `hours_per_week`, and `income_50k`. We would like to build several statistical models to predict `income_50k` (i.e. whether income is higher than 50k or not) of a person with given personal information. The data description is as follows.

- `age`: Age in years
- `workclass`: Type of employment the person has
- `education_num`: Education in years
- `marital_status`: Marital status
- `capital_gain`: Capital gain in the past year
- `capital_loss`: Capital loss in the past year
- `hours_per_week`: Average working hours per week
- `income_50k`: A factor with levels Yes and No indicating whether the income higher than 50k or not

Q1 [3 points] Data Partition

Q1(a) [1 points] Let's correctly read in the data in `Adult_Income.csv` and name it as `total_data`.

```
## Write code solution to Q1(a) here
rm(list=ls())

#reading in the data from the csv file
total_data = read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 2/Advanced Stats/Homework 4/Adult_Income
                      #setting any strings to factors
                      stringsAsFactors = TRUE,
                      #and headers as true
                      header = TRUE)
```

Q1(b) [2 points] Let's partition the data in `total_data` into training **(60%)** and test data **(40%)** and store them as `R` objects `train_data` and `test_data` respectively. Use random seed **`set.seed(7)`**!

```
#setting random seed
set.seed(7)
#the number of rows as a variable
```

```r
num_row <-  nrow(total_data)
#getting a random sample of the indices
indi <- sample(1:num_row, 0.6*num_row)
#getting training data
train_data <- total_data[indi , ]
#and then test data
test_data <- total_data[-indi , ]
```

Q2 [8 points] Logistic Regression and GAM

Q2(a) [**3 points**] Fit a logistic regression model of `income_50k` w.r.t. all 7 predictors using the **training data**, name it `lm1`.

```r
#and making a log model using all 7 predictors
lm1 <- glm(income_50k ~ age + workclass + education_num + marital_status + capital_gain
            + capital_loss + hours_per_week,
           data = train_data,
            family = "binomial")
```

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

Q2(b) [**5 points**] Fit a GAM of `income_50k` w.r.t. all 7 predictors using the **training data**, name it `gam1`. Let's use splines with **df=4** for all 5 numerical predictors, which include `age`, `education_num`, `capital_gain`, `capital_loss` and `hours_per_week`.

```r
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.4.1

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.22-5
```

```r
#and then the gam model with splines on the 5 numerical predictors
gam1 <- gam(income_50k ~ s(age) + workclass + s(education_num) + marital_status +                  s(ca
           data = train_data, family = "binomial")
```

Q3 [9 points] Neural Networks

Fit an NN of **standardized** `income_50k` w.r.t. all 7 predictors using the **training data**, name it `nn1`. For the architecture of NN, let's use one hidden layer with 6 hidden units.

Q3(a) [**2 points**] Let's generate the **training dataset** that are needed for the estimation of NN using the function `model.matrix()` and store it in `x_train_nn`. In addition, use the `scale()` function to standardize the predictors by centering with mean and scaling with sd.

```r
#making the train data into the neural net train format
x_train_nn <- model.matrix( ~ ., data = train_data, na.rm = TRUE)[ , -17]
#getting the mean from the train data
x_mean <- apply(x_train_nn, MARGIN = 2, FUN = mean)
#getting the sd from the train data
x_sd <- apply(x_train_nn, MARGIN = 2, FUN = sd)
#then scaling the data on the mean and sd
x_train_nn <- scale(x_train_nn, center = x_mean, scale = x_sd)
#dropping the intercept column
x_train_nn <- x_train_nn[ , -1]
```

Q3(b) [**1 points**] Let's further combine the dependent variable `income_50k` with the standardized predictors `x_train_nn` generated in Q3(a).

```r
#adding in the response variable
x_train_nn <- cbind.data.frame(train_data$income_50k, x_train_nn)

#renaming the response variable
colnames(x_train_nn)[1] <- 'income_50k'
```

Q3(c) [**2 points**] Let's generate the **test dataset** that are needed for the out-of-sample prediction evaluation of NN using the function `model.matrix` and store it in `x_test_nn`. Use the `scale()` function to standardize the predictors by centering with mean and scaling with sd as in Q3(a).

```r
#making the neural net test sample from the data
x_test_nn <- model.matrix( ~ ., data = test_data, na.rm = TRUE)[ , -17]
#scaling based on the train mean and sd
x_test_nn <- scale(x_test_nn, center = x_mean, scale = x_sd)
#adding in the response variable
x_test_nn <- cbind.data.frame(test_data$income_50k, x_test_nn)
#renaming the response variable
colnames(x_test_nn)[1] <- 'income_50k'

#dropping the intercept column
x_test_nn <- x_test_nn[ , -2]
```

Q3(d) [**4 points**] Let's fit an NN that has one hidden layer with 6 hidden units. Make sure to use random seed **`set.seed(7)`**! Note that since some categorical variables have a number of different levels, for convenience, let's use the shortcut formula `Y~.` in the function `neuralnet()`. (You can also use the `Y~X1+X2...+Xp` formula if you want.)

```r
#setting the seed at 7 for consistency
set.seed(7)
#importing neural net
library(neuralnet)

#making the neural net model with our output set to 50k yes.
nn1 <- neuralnet(income_50k == 'Yes' ~ .,
                 #6 hidden units in one layer
                 hidden = c(6),
                 #using train data
                 data = x_train_nn,
                 #non-linear data
```

```r
                  linear.output = FALSE)

plot(nn1, type = 'best')
```

Q4 [10 points] Model Evaluation (Prediction)

Q4(a) [2 points] Use `lm1`, `gam1` and `nn1` to generate probability predictions for `income_50k` on the test data and store the predicted probability in `lm1_pred`, `gam1_pred` and `nn1_pred` respectively.

```r
#making predictions from the log model
lm1_pred <- predict(lm1, newdata = test_data, type = "response")

#making predictions from the gam model
gam1_pred <- predict(gam1, newdata = test_data, type = "response")

#making predictions from the neural net model
nn1_pred <- predict(nn1, newdata = x_test_nn, type = "response")
```

Q4(b) [3 points] Use the `R` package `caret` to evaluate the prediction performance of `lm1`, `gam1` and `nn1`. What are the TP for `lm1`, `gam1` and `nn1`?

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
#getting a confusion matrix from the log predictions
lm_full_acc <- confusionMatrix(factor(ifelse(lm1_pred > 0.5, 'Yes', 'No')), test_data$income_50k, positi

print(lm_full_acc)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2770  387
##        Yes  209  536
##
##                Accuracy : 0.8473
##                  95% CI : (0.8356, 0.8584)
##     No Information Rate : 0.7635
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.547
##
##  Mcnemar's Test P-Value : 4.161e-13
##
##             Sensitivity : 0.5807
##             Specificity : 0.9298
##          Pos Pred Value : 0.7195
```

```
##          Neg Pred Value : 0.8774
##              Prevalence : 0.2365
##          Detection Rate : 0.1374
##    Detection Prevalence : 0.1909
##       Balanced Accuracy : 0.7553
##
##        'Positive' Class : Yes
##
```

```r
#getting a confusion matrix from the gam predictions
gam_1_acc <- confusionMatrix(factor(ifelse(gam1_pred > 0.5, 'Yes', 'No')), test_data$income_50k, positi

print(gam_1_acc)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2799  391
##        Yes  180  532
##
##                Accuracy : 0.8537
##                  95% CI : (0.8422, 0.8646)
##     No Information Rate : 0.7635
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5601
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5764
##             Specificity : 0.9396
##          Pos Pred Value : 0.7472
##          Neg Pred Value : 0.8774
##              Prevalence : 0.2365
##          Detection Rate : 0.1363
##    Detection Prevalence : 0.1825
##       Balanced Accuracy : 0.7580
##
##        'Positive' Class : Yes
##
```

```r
#getting a confusion matrix from the nn predictions
nn_1_acc <- confusionMatrix(factor(ifelse(nn1_pred > 0.5, 'Yes', 'No')), x_test_nn$income_50k, positive

print(nn_1_acc)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2756  382
##        Yes  223  541
```

```
##
##                Accuracy : 0.845
##                  95% CI : (0.8332, 0.8562)
##     No Information Rate : 0.7635
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5436
##
##  Mcnemar's Test P-Value : 1.331e-10
##
##             Sensitivity : 0.5861
##             Specificity : 0.9251
##          Pos Pred Value : 0.7081
##          Neg Pred Value : 0.8783
##              Prevalence : 0.2365
##          Detection Rate : 0.1386
##    Detection Prevalence : 0.1958
##       Balanced Accuracy : 0.7556
##
##        'Positive' Class : Yes
##
```

Answer:

lm1 correctly identified 2770 'No' observations and 536 'Yes' observations, for a total of 3,306 true positive predictions and a total accuracy of 84.73%.

gam1 correctly identified 2799 'No' observations and 532 'Yes' observations for a total of 3,331 true positive predictions and a total accuracy of 85.37%.

nn1 correctly identified 2756 'No' observations and 541 'Yes' observations for a total of 3,297 true positive predictions and a total accuracy of 84.50%.

Q4(c) [**1 points**] Which statistical model has the best sensitivity, `lm1` or `gam1` or `nn1`? Give the model and the corresponding sensitivity value.

Answer:

lm1: 0.5807

gam1: 0.5764

nn1: 0.5861

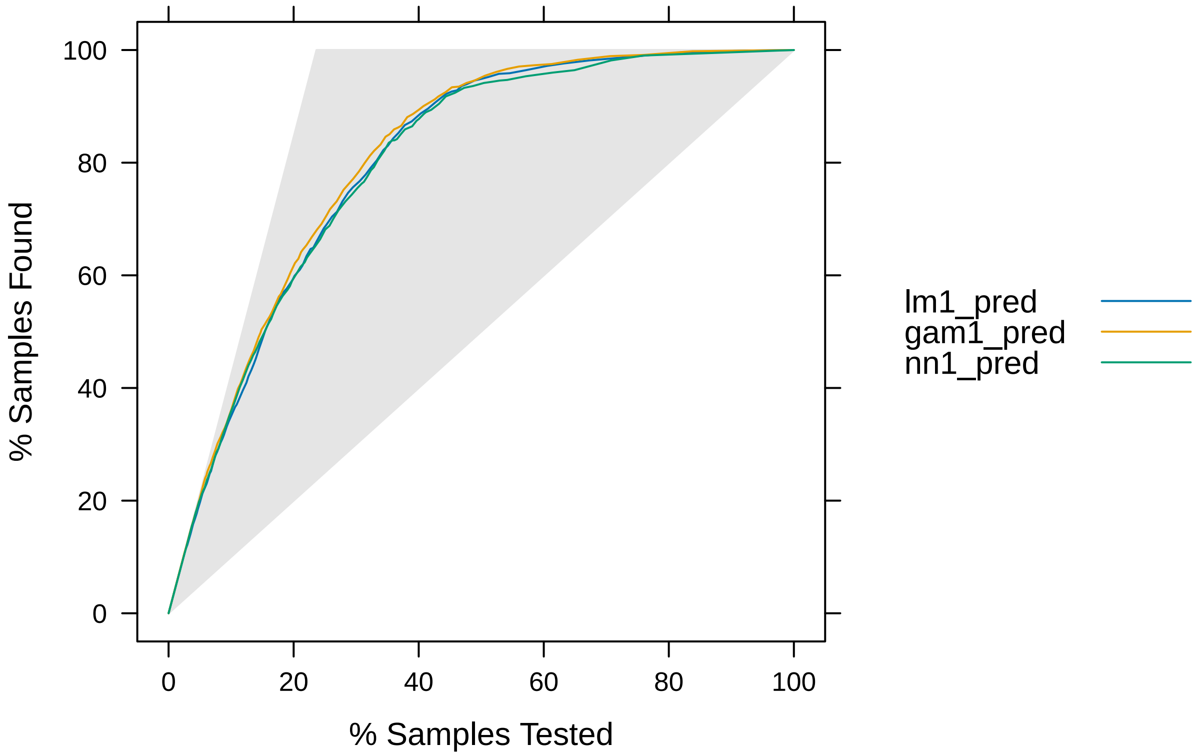
The neural network model (nn1) had the best sensitivity with a sensitivity of 0.5861.

Q4(d) [**2 points**] Use the `R` package `caret` to generate the lift charts of `lm1`, `gam1` and `nn1`. Make sure to set the `cuts` argument in the `lift()` function as `cuts=100` to save computational time.

```
#making a lift chart of the three models
lift_chart <- lift(test_data$income_50k ~ lm1_pred + gam1_pred + nn1_pred,
                   class = 'Yes', cuts = 100)

#making an xyplot of the lift chart
final_plot <- xyplot(lift_chart, auto.key = T)

#printing the plot
final_plot
```



Q4(e) [**2 points**] Based on the lift chart, which statistical model performs the best in terms of identifying people with income > 50k?

Answer:

Based on the lift chart, the best performing model is the gam model because it has the most area underneath the curve, and had the most TP predictions.