Code Challenge in ML: Replicating Baby Names Gender Prediction

Yongjun Zhang, Ph.D.

Data Preparation

You can download ssa baby names via here https://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-national-data

I have also used the following codes to preprocess the data. I have generated aggregated name counts by gender and then define our target outcome as "female if over 50 of women use that name." I have also created four variables: first letter, first two letters, last letter, and laster two letters. You can directly use the RData file, https://yongjunzhang.com/files/ssa_baby_names.RData.

```
require(pacman)
```

```
## Loading required package: pacman
```

```
p_load(tidyverse,glue)

files <- list.files(path = "./names/",pattern = ".txt",full.names = TRUE)
files
```

```
##   [1] "./names//yob1880.txt" "./names//yob1881.txt" "./names//yob1882.txt"
##   [4] "./names//yob1883.txt" "./names//yob1884.txt" "./names//yob1885.txt"
##   [7] "./names//yob1886.txt" "./names//yob1887.txt" "./names//yob1888.txt"
##  [10] "./names//yob1889.txt" "./names//yob1890.txt" "./names//yob1891.txt"
##  [13] "./names//yob1892.txt" "./names//yob1893.txt" "./names//yob1894.txt"
##  [16] "./names//yob1895.txt" "./names//yob1896.txt" "./names//yob1897.txt"
##  [19] "./names//yob1898.txt" "./names//yob1899.txt" "./names//yob1900.txt"
##  [22] "./names//yob1901.txt" "./names//yob1902.txt" "./names//yob1903.txt"
##  [25] "./names//yob1904.txt" "./names//yob1905.txt" "./names//yob1906.txt"
##  [28] "./names//yob1907.txt" "./names//yob1908.txt" "./names//yob1909.txt"
##  [31] "./names//yob1910.txt" "./names//yob1911.txt" "./names//yob1912.txt"
##  [34] "./names//yob1913.txt" "./names//yob1914.txt" "./names//yob1915.txt"
##  [37] "./names//yob1916.txt" "./names//yob1917.txt" "./names//yob1918.txt"
##  [40] "./names//yob1919.txt" "./names//yob1920.txt" "./names//yob1921.txt"
##  [43] "./names//yob1922.txt" "./names//yob1923.txt" "./names//yob1924.txt"
##  [46] "./names//yob1925.txt" "./names//yob1926.txt" "./names//yob1927.txt"
##  [49] "./names//yob1928.txt" "./names//yob1929.txt" "./names//yob1930.txt"
##  [52] "./names//yob1931.txt" "./names//yob1932.txt" "./names//yob1933.txt"
##  [55] "./names//yob1934.txt" "./names//yob1935.txt" "./names//yob1936.txt"
##  [58] "./names//yob1937.txt" "./names//yob1938.txt" "./names//yob1939.txt"
##  [61] "./names//yob1940.txt" "./names//yob1941.txt" "./names//yob1942.txt"
##  [64] "./names//yob1943.txt" "./names//yob1944.txt" "./names//yob1945.txt"
##  [67] "./names//yob1946.txt" "./names//yob1947.txt" "./names//yob1948.txt"
##  [70] "./names//yob1949.txt" "./names//yob1950.txt" "./names//yob1951.txt"
##  [73] "./names//yob1952.txt" "./names//yob1953.txt" "./names//yob1954.txt"
##  [76] "./names//yob1955.txt" "./names//yob1956.txt" "./names//yob1957.txt"
##  [79] "./names//yob1958.txt" "./names//yob1959.txt" "./names//yob1960.txt"
```

```
##  [82] "./names//yob1961.txt" "./names//yob1962.txt" "./names//yob1963.txt"
##  [85] "./names//yob1964.txt" "./names//yob1965.txt" "./names//yob1966.txt"
##  [88] "./names//yob1967.txt" "./names//yob1968.txt" "./names//yob1969.txt"
##  [91] "./names//yob1970.txt" "./names//yob1971.txt" "./names//yob1972.txt"
##  [94] "./names//yob1973.txt" "./names//yob1974.txt" "./names//yob1975.txt"
##  [97] "./names//yob1976.txt" "./names//yob1977.txt" "./names//yob1978.txt"
## [100] "./names//yob1979.txt" "./names//yob1980.txt" "./names//yob1981.txt"
## [103] "./names//yob1982.txt" "./names//yob1983.txt" "./names//yob1984.txt"
## [106] "./names//yob1985.txt" "./names//yob1986.txt" "./names//yob1987.txt"
## [109] "./names//yob1988.txt" "./names//yob1989.txt" "./names//yob1990.txt"
## [112] "./names//yob1991.txt" "./names//yob1992.txt" "./names//yob1993.txt"
## [115] "./names//yob1994.txt" "./names//yob1995.txt" "./names//yob1996.txt"
## [118] "./names//yob1997.txt" "./names//yob1998.txt" "./names//yob1999.txt"
## [121] "./names//yob2000.txt" "./names//yob2001.txt" "./names//yob2002.txt"
## [124] "./names//yob2003.txt" "./names//yob2004.txt" "./names//yob2005.txt"
## [127] "./names//yob2006.txt" "./names//yob2007.txt" "./names//yob2008.txt"
## [130] "./names//yob2009.txt" "./names//yob2010.txt" "./names//yob2011.txt"
## [133] "./names//yob2012.txt" "./names//yob2013.txt" "./names//yob2014.txt"
## [136] "./names//yob2015.txt" "./names//yob2016.txt" "./names//yob2017.txt"
## [139] "./names//yob2018.txt" "./names//yob2019.txt" "./names//yob2020.txt"
```

```r
# let us say we want to define a read_us_baby function
readSsaBabyNames <- function(file,...){
  require(tidyverse)
  data <- read_csv(file,col_names = FALSE,show_col_types = FALSE) %>%
    mutate(year=str_replace_all(file,"[^0-9]",""))
  colnames(data) <- c("names","sex","count","year")
  return(data)
}

dat_year <- map_dfr(files, readSsaBabyNames)

# we only use names after 1970 and used by at least 10
dat_all <- dat_year %>%
  filter(year>1970) %>%
  group_by(names,sex) %>%
  summarise(count=sum(count)) %>%
  # we only keep names used by at least 10
  filter(count>10) %>%
  pivot_wider(names_from = "sex",values_from="count") %>%
  replace_na(list(F=0,M=0)) %>%
  mutate(female=(F/(F+M)>.5)*1) %>%
  mutate(flt1= str_extract(names,"^.")%>% tolower,
         flt2= str_extract(names,"^.{2}") %>% tolower,
         llt1= str_extract(names,".$")%>% tolower,
         llt2= str_extract(names,".{2}$")%>% tolower
         )
```

```
## `summarise()` has grouped output by 'names'. You can override using the
## `.groups` argument.
```

```r
save(dat_year,dat_all,file="./ssa_baby_names.RData")
```

Split our data into train and test data

Before we further split our data, let us take a look at data first. Since the code challenge asks us to select top 5 most frequent features, we take a look and see which letters are most frequent in the database.

```
top_letters <- dat_all %>%
  group_by(flt1) %>%
  summarise(fl1=n()) %>%
  top_n(n=5) %>%
  bind_cols(
    dat_all %>%
      group_by(flt2) %>%
      summarise(fl2=n()) %>%
      top_n(n=5)
  ) %>%
  bind_cols(
    dat_all %>%
      group_by(llt1) %>%
      summarise(ll1=n()) %>%
      top_n(n=5)
  ) %>%
  bind_cols(
    dat_all %>%
      group_by(llt2) %>%
      summarise(ll2=n()) %>%
      top_n(n=5)
  )
```

```
## Selecting by fl1
## Selecting by fl2
## Selecting by ll1
## Selecting by ll2
```

```
knitr::kable(top_letters)
```

flt1	fl1	flt2	fl2	llt1	ll1	llt2	ll2
a	7685	da	1701	a	19482	ah	3136
j	5755	ja	2904	e	9336	an	2552
k	5694	ka	2183	h	4387	ia	3242
m	5173	ma	2884	i	4014	na	3915
s	5659	sh	2352	n	11613	on	2741

Let us create these features

including, flt1.a, flt1.j, flt1.k, flt1.m, flt1.s,flt2.da, flt2.ja, flt2.ka,flt2.ma, flt2.sh, ll1.a, ll1.e, ll1.h, ll1.i, ll1.n, etc..

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
```

```
##     lift
```

```
#create the full datasets with dummies
top_features <- c(paste0("flt1",c("a","j","k","m","s")),
                  paste0("flt2",c("da","ja","ka","ma","sh")),
                  paste0("llt1",c("a","e","h","i","n")),
                  paste0("llt2",c("ah","an","ia","na","on"))
                  )
fllt_d=predict(dummyVars(~llt1+llt2+flt1+flt2,data=dat_all),newdata=dat_all)%>%as.data.frame()

df=dat_all %>%
  ungroup %>%
  select(names,female) %>%
  filter(!is.na(female)) %>%
  mutate(female=ifelse(female==1,"Y","N") %>% as.factor()) %>%
  bind_cols(fllt_d %>%
              select(all_of(top_features)))
```

Let us finally split our data into train and test

```
inTrain <- createDataPartition(
  y = df$female,
  ## the outcome data are needed
  p = .75,
  ## The percentage of data in the
  ## training set
  list = FALSE
)

train <- df[ inTrain,]
test  <- df[-inTrain,]

nrow(train)
```

```
## [1] 50610
```

```
#> [1] 157
nrow(test)
```

```
## [1] 16869
```

```
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:glue':
##
##     trim

## The following objects are masked from 'package:dplyr':
##
##     combine, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     keep
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

```
## The following object is masked from 'package:utils':
##
##     object.size
```

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```
keep(dat_all,df,train,test,sure=TRUE)
```

Let us train a nb model to predict gender

You need naivebayes package to be installed, here is the documentation: https://cran.r-project.org/web/packages/naivebayes/naivebayes.pdf

```
# K folds cross validation
# try parallel computing
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
cl <- makePSOCKcluster(3)
registerDoParallel(cl)

# Define tuning grid
grid_nb <- expand.grid(usekernel = c(TRUE, FALSE),
                       laplace = c(0, 0.5, 1),
                       adjust = c(0.75, 1, 1.25, 1.5))

train_control <- caret::trainControl(
  method = "cv",
  number = 3,
  classProbs=T,savePredictions = T,
  verboseIter = FALSE,
  allowParallel = TRUE
)
```

```r
nb_base <- caret::train(
  female~.,
  data=train %>%  dplyr::select(-c(names)),
  trControl = train_control,
  tuneGrid = grid_nb,
  method = "naive_bayes",
  verbose = TRUE
)

stopCluster(cl)

save(nb_base,file="./nb_base.RData")
```

Let us check model performance, get the comfusion matrix firrst

```r
#load("./nb_base.RData")
# check cf matrix
nb_base
```

```
## Naive Bayes
##
## 50610 samples
##    20 predictor
##     2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 33740, 33740, 33740
## Resampling results across tuning parameters:
##
##   usekernel  laplace  adjust  Accuracy   Kappa
##   FALSE      0.0      0.75    0.7259237  0.4632047
##   FALSE      0.0      1.00    0.7259237  0.4632047
##   FALSE      0.0      1.25    0.7259237  0.4632047
##   FALSE      0.0      1.50    0.7259237  0.4632047
##   FALSE      0.5      0.75    0.7259237  0.4632047
##   FALSE      0.5      1.00    0.7259237  0.4632047
##   FALSE      0.5      1.25    0.7259237  0.4632047
##   FALSE      0.5      1.50    0.7259237  0.4632047
##   FALSE      1.0      0.75    0.7259237  0.4632047
##   FALSE      1.0      1.00    0.7259237  0.4632047
##   FALSE      1.0      1.25    0.7259237  0.4632047
##   FALSE      1.0      1.50    0.7259237  0.4632047
##    TRUE      0.0      0.75    0.6712310  0.3962739
##    TRUE      0.0      1.00    0.6650069  0.3870472
##    TRUE      0.0      1.25    0.6650069  0.3870472
##    TRUE      0.0      1.50    0.6650069  0.3870472
##    TRUE      0.5      0.75    0.6712310  0.3962739
##    TRUE      0.5      1.00    0.6650069  0.3870472
##    TRUE      0.5      1.25    0.6650069  0.3870472
##    TRUE      0.5      1.50    0.6650069  0.3870472
##    TRUE      1.0      0.75    0.6712310  0.3962739
##    TRUE      1.0      1.00    0.6650069  0.3870472
```

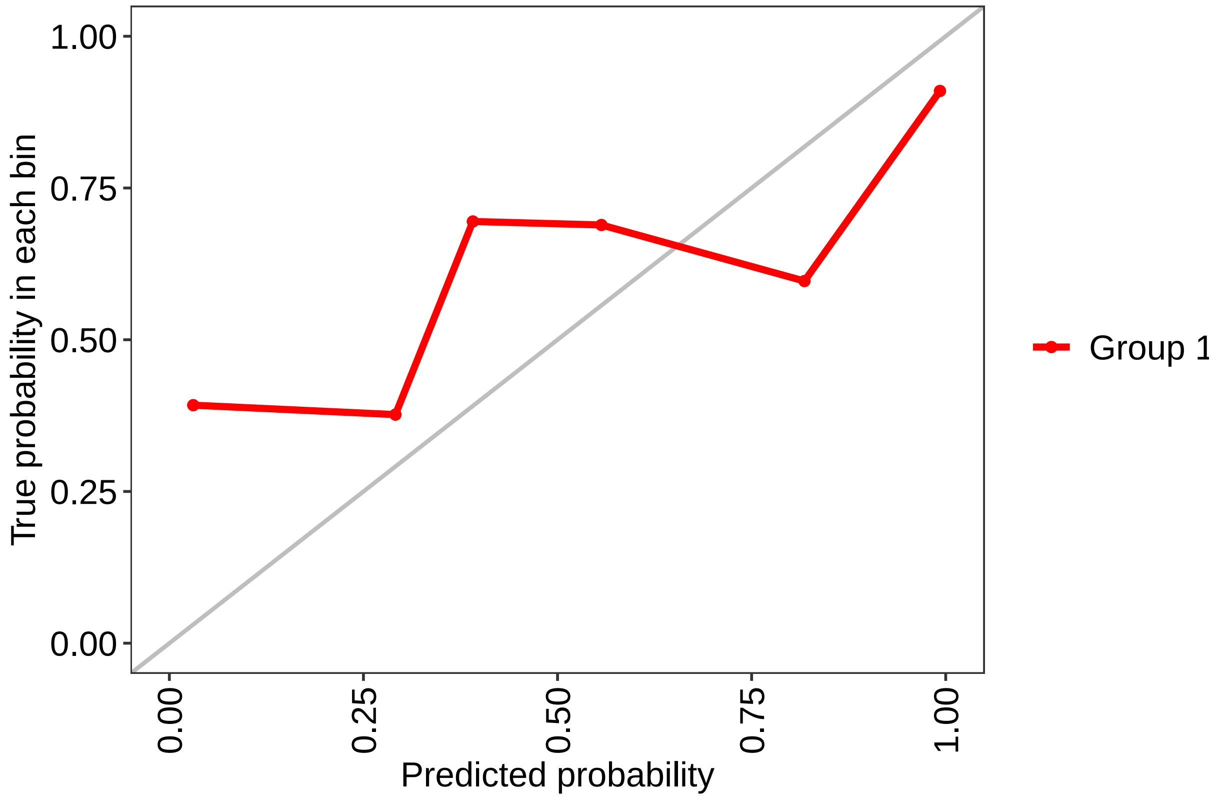
```
##    TRUE      1.0      1.25    0.6650069  0.3870472
##    TRUE      1.0      1.50    0.6650069  0.3870472
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 0.75.
```

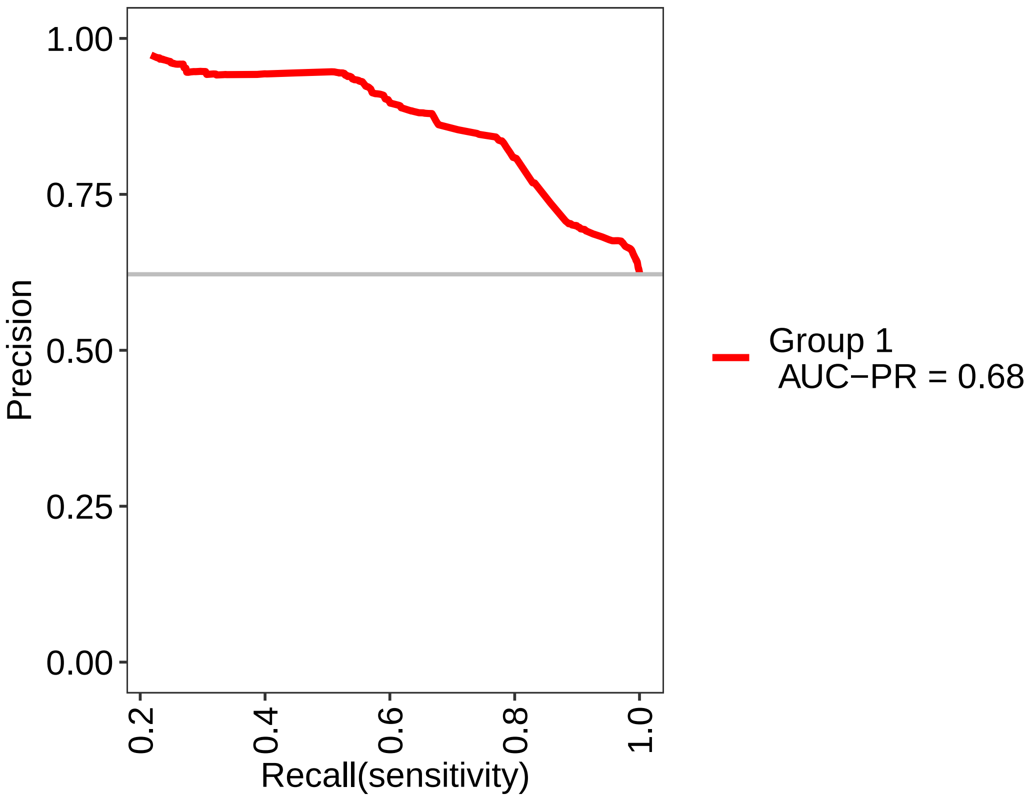
```
confusionMatrix(nb_base)
```

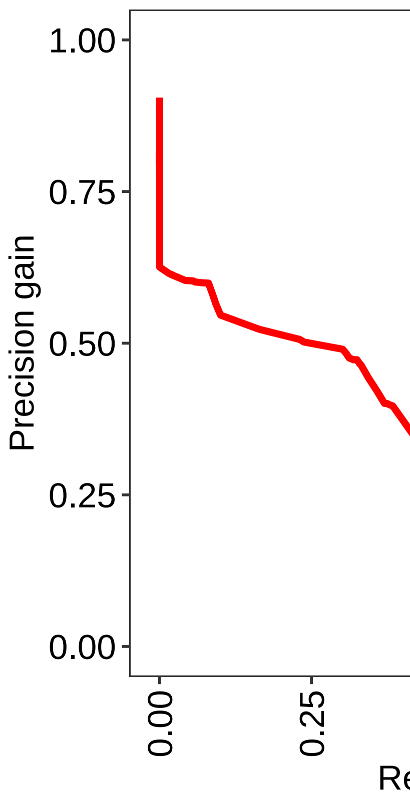
```
## Cross-Validated (3 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    N    Y
##          N 32.4 22.0
##          Y  5.4 40.2
##
##  Accuracy (average) : 0.7259
```

```
# PLOT ROC CURVE
library(MLeval)
## run MLeval
res <- evalm(nb_base)
```

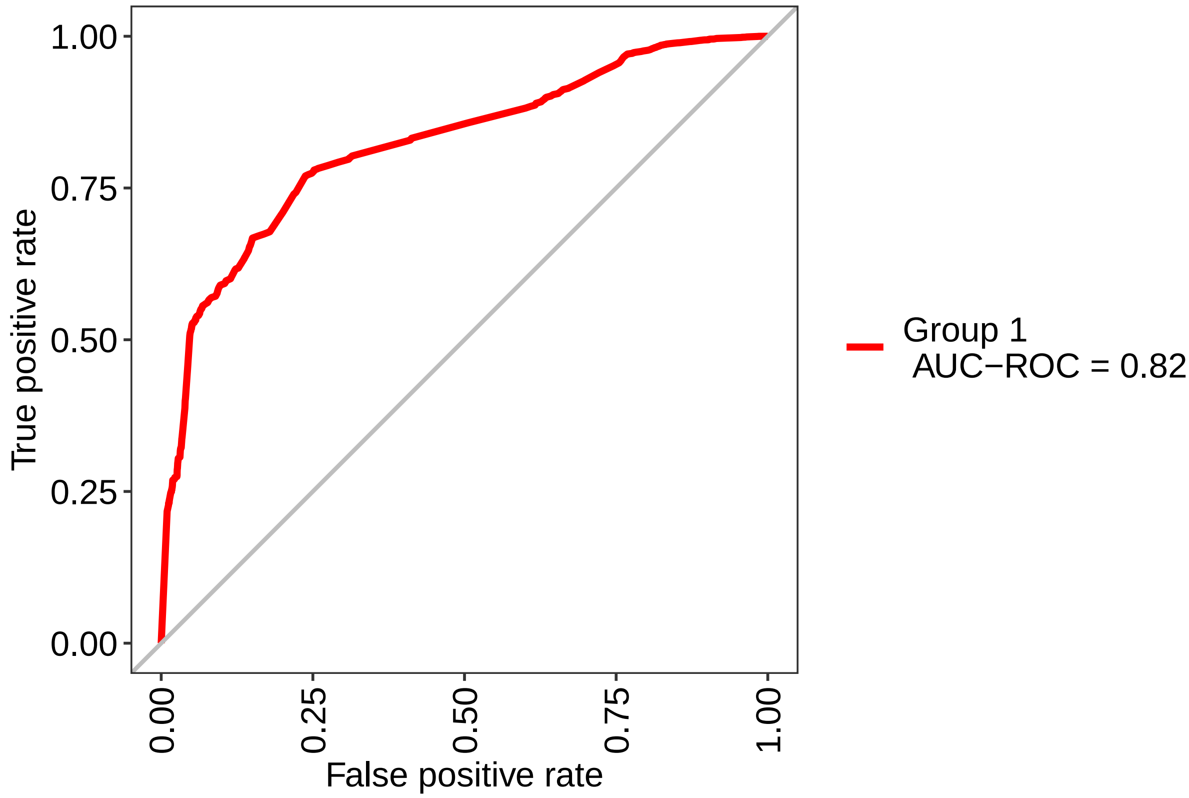
```
## ***MLeval: Machine Learning Model Evaluation***

## Input: caret train function object

## Not averaging probs.

## Group 1 type: cv

## Observations: 50610

## Number of groups: 1

## Observations per group: 50610

## Positive: Y

## Negative: N

## Group: Group 1

## Positive: 31473

## Negative: 19137

## ***Performance Metrics***
```



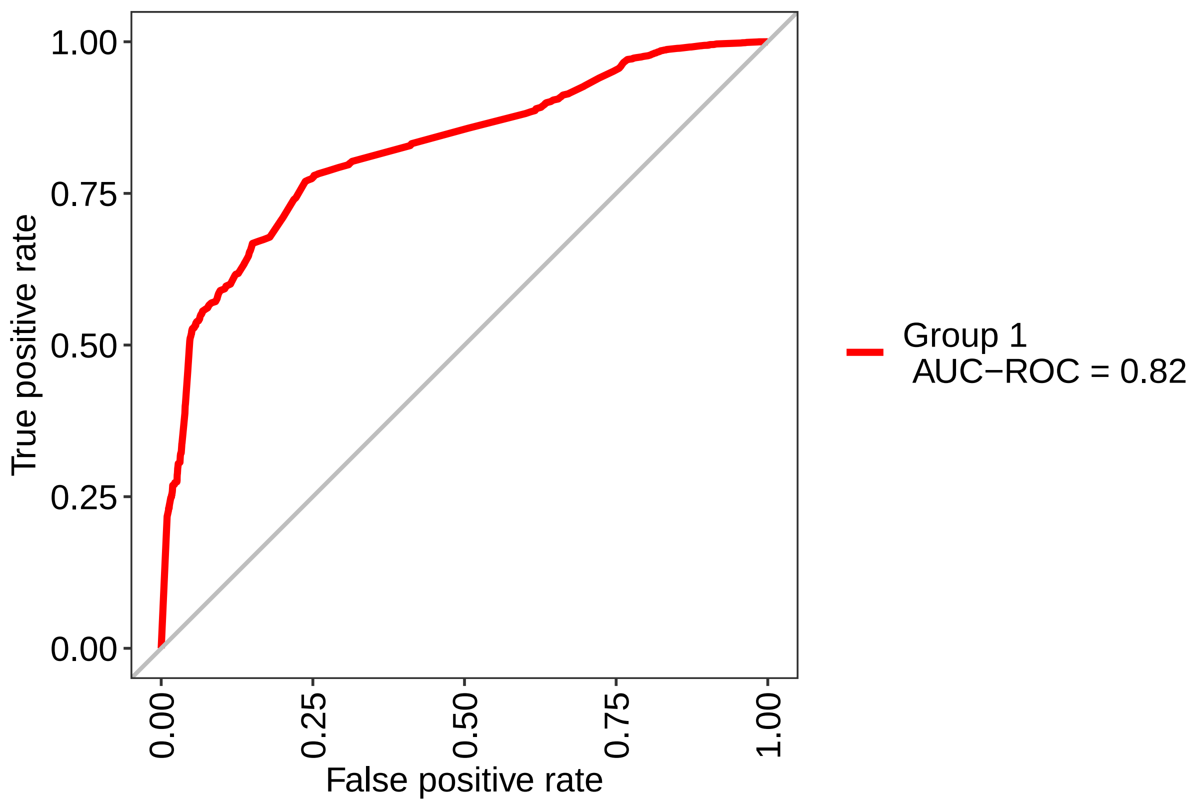





```
## Group 1 Optimal Informedness = 0.532211429721906
## Group 1 AUC-ROC = 0.82
```




```
## get ROC
res$roc
```



Test model performance on test set

```r
# predict test names
test_df <- test %>% dplyr::select(-c(names,female))
test_pred <- predict(nb_base$finalModel,newdata=test_df) %>% as.data.frame()
colnames(test_pred) <- "nb_female"
# check confusion matrix
confusionMatrix(test_pred$nb_female,test$female)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N 5463 3698
##          Y  915 6793
##
##                Accuracy : 0.7265
##                  95% CI : (0.7197, 0.7333)
##     No Information Rate : 0.6219
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4643
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8565
##             Specificity : 0.6475
##          Pos Pred Value : 0.5963
##          Neg Pred Value : 0.8813
##              Prevalence : 0.3781
##          Detection Rate : 0.3238
##    Detection Prevalence : 0.5431
##       Balanced Accuracy : 0.7520
##
##        'Positive' Class : N
##
```

Let us try penalized logit models

we will use glmnet -combination of lasso and ridge regression -Can fit a mix of the two models -alpha [0, 1]: pure lasso to pure ridge -lambda (0, infinity): size of the penalty

```r
# K folds cross validation
# try parallel computing
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.0.5
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```r
cl <- makePSOCKcluster(3)
registerDoParallel(cl)

# Define tuning grid
grid_plr <- expand.grid( alpha = c(0,1),
                         lambda = c(1e-4, 1e-2, 1))

train_control <- caret::trainControl(
  method = "cv",
  number = 3,
  classProbs=T,savePredictions = T,
  verboseIter = FALSE,
  allowParallel = TRUE
)

plr_base <- caret::train(
  female~.,
  data=train %>%dplyr::select(-c(names)),
  trControl = train_control,
  tuneGrid = grid_plr,
  method = "glmnet",
  verbose = TRUE
)

stopCluster(cl)
save(plr_base,file="./plr_base.RData")
# check cf matrix
plr_base
```

```
## glmnet
##
## 50610 samples
##    20 predictor
##     2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 33740, 33740, 33740
## Resampling results across tuning parameters:
##
##   alpha  lambda  Accuracy   Kappa
##   0      1e-04   0.7867220  0.5558511
##   0      1e-02   0.7867220  0.5558511
##   0      1e+00   0.6795100  0.1922293
##   1      1e-04   0.7859909  0.5567345
##   1      1e-02   0.7821379  0.5560635
##   1      1e+00   0.6218731  0.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 0.01.
```

```
confusionMatrix(plr_base)
```

```
## Cross-Validated (3 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    N    Y
##          N 29.2 12.7
##          Y  8.6 49.5
##
##  Accuracy (average) : 0.7867
```

Let us try random forest

Random Forest

-method = 'ranger' -Type: Classification, Regression

Tuning parameters:

-mtry (#Randomly Selected Predictors) -splitrule (Splitting Rule) -min.node.size (Minimal Node Size) -Required packages: e1071, ranger, dplyr

here is the documentation: https://cran.r-project.org/web/packages/ranger/ranger.pdf

```
# K folds cross validation
# try parallel computing
require(pacman)
p_load(ranger,e1071)
cl <- makePSOCKcluster(3)
registerDoParallel(cl)

# Define tuning grid
grid_rf <- expand.grid(splitrule= c("gini", "extratrees","hellinger"),
                      mtry = c(1,2,4,10),
                      min.node.size=c(1))

train_control <- caret::trainControl(
  method = "cv",
  number = 3,
  classProbs=T,savePredictions = T,
  verboseIter = FALSE,
  allowParallel = TRUE
)

rf_base <- caret::train(
  female~.,
  data=train %>%dplyr::select(-c(names)),
  trControl = train_control,
  tuneGrid = grid_rf,
  method ='ranger',
  verbose = TRUE
)

stopCluster(cl)
```

```r
save(rf_base,file="./rf_base.RData")
# check cf matrix
rf_base
```

```
## Random Forest
##
## 50610 samples
##    20 predictor
##     2 classes: 'N', 'Y'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 33740, 33740, 33740
## Resampling results across tuning parameters:
##
##   splitrule   mtry  Accuracy   Kappa
##   gini         1    0.6809721  0.1993943
##   gini         2    0.7383916  0.4170149
##   gini         4    0.7845880  0.5553652
##   gini        10    0.7884805  0.5610431
##   extratrees   1    0.6810907  0.2020840
##   extratrees   2    0.7177633  0.3530887
##   extratrees   4    0.7847659  0.5558603
##   extratrees  10    0.7884805  0.5610431
##   hellinger    1    0.6806955  0.1996275
##   hellinger    2    0.7386090  0.4173106
##   hellinger    4    0.7845090  0.5550081
##   hellinger   10    0.7884805  0.5610431
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 10, splitrule = gini
##  and min.node.size = 1.
```

```r
confusionMatrix(rf_base)
```

```
## Cross-Validated (3 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    N    Y
##          N 29.6 12.9
##          Y  8.2 49.2
##
##  Accuracy (average) : 0.7885
```