Lab 9

Tziporah Horowitz

11:59PM May 2, 2020

Set a seed and load the `adult` dataset and remove missingness. We also drop the education variable as it's linearly dependent with the education_num variable and will complicate the interactions further on.

```
set.seed(1)
pacman::p_load_gh("coatless/ucidata")
data(adult)
adult = na.omit(adult)
adult$education = NULL
```

We had problems with the features "occupation" and "native_country". Go through these two features and and identify levels with too few examples and wrap them into a level called "other". This is standard practice.

```
sort(table(adult$occupation))
```

```
##
##      Armed-Forces   Priv-house-serv   Protective-serv      Tech-support
##                 9               143               644               912
##   Farming-fishing Handlers-cleaners  Transport-moving Machine-op-inspct
##               989              1350              1572              1966
##     Other-service             Sales      Adm-clerical   Exec-managerial
##              3212              3584              3720              3992
##      Craft-repair    Prof-specialty
##              4030              4038
```

```
adult$occupation = as.character(adult$occupation)
adult$other = adult$occupation %in% c("Armed-Forces", "Priv-house-serv", "Protective-serv")
table(adult$other)
```

```
##
## FALSE  TRUE
## 29365   796
```

```
adult$occupation[adult$other] = "other"
adult$other = NULL
adult$occupation = as.factor(adult$occupation)
sort(table(adult$occupation))
```

```
##
```

```
##            other      Tech-support   Farming-fishing Handlers-cleaners
##              796               912               989              1350
##  Transport-moving Machine-op-inspct     Other-service             Sales
##             1572              1966              3212              3584
##      Adm-clerical   Exec-managerial      Craft-repair    Prof-specialty
##             3720              3992              4030              4038
```

```r
sort(table(adult$native_country))
```

```
## 
##         Holand-Netherlands                   Scotland 
##                          1                         11 
##                   Honduras                    Hungary 
##                         12                         13 
## Outlying-US(Guam-USVI-etc)                 Yugoslavia 
##                         14                         16 
##                       Laos                   Thailand 
##                         17                         17 
##                   Cambodia            Trinadad&Tobago 
##                         18                         18 
##                       Hong                    Ireland 
##                         19                         24 
##                    Ecuador                     France 
##                         27                         27 
##                     Greece                       Peru 
##                         29                         30 
##                  Nicaragua                   Portugal 
##                         33                         34 
##                      Haiti                       Iran 
##                         42                         42 
##                     Taiwan                   Columbia 
##                         42                         56 
##                     Poland                      Japan 
##                         56                         59 
##                  Guatemala                    Vietnam 
##                         63                         64 
##         Dominican-Republic                      China 
##                         67                         68 
##                      Italy                      South 
##                         68                         71 
##                    Jamaica                    England 
##                         80                         86 
##                       Cuba                El-Salvador 
##                         92                        100 
##                      India                     Canada 
##                        100                        107 
##                Puerto-Rico                    Germany 
##                        109                        128 
##                Philippines                     Mexico 
##                        188                        610 
##              United-States 
##                      27503
```

```r
adult$domestic = ifelse(adult$native_country == "United-States", 1, 0)
table(adult$domestic)
```

```
##
##     0     1
##  2658 27503
```

```r
adult$native_country = NULL
```

We will be doing model selection. We will split the dataset into 3 distinct subsets. Set the size of our splits here. For simplicitiy, all three splits will be identically sized. We are making it small so the stepwise algorithm can compute quickly. If you have a faster machine, feel free to increase this.

```r
Nsplitsize = 1000
```

Now create the following variables: `Xtrain`, `ytrain`, `Xselect`, `yselect`, `Xtest`, `ytest` with `Nsplitsize` observations:

```r
adult = adult[sample(1 : nrow(adult)), ]

Xtrain = adult[1 : Nsplitsize, ]
Xtrain$income = NULL
ytrain = ifelse(adult[1 : Nsplitsize, "income"] == ">50K", 1, 0)
Xselect = adult[(Nsplitsize + 1) : (2 * Nsplitsize), ]
Xselect$income = NULL
yselect = ifelse(adult[(Nsplitsize + 1) : (2 * Nsplitsize), "income"] ==">50K", 1, 0)
Xtest = adult[(2 * Nsplitsize + 1) : (3 * Nsplitsize), ]
Xtest$income = NULL
ytest = ifelse(adult[(2 * Nsplitsize + 1) : (3 * Nsplitsize), "income"] == ">50K", 1, 0)
```

Fit a vanilla logistic regression on the training set.

```r
logistic_mod = glm(ytrain ~ ., Xtrain, family = "binomial")
```

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

and report the log scoring rule, the Brier scoring rule.

```r
phat_train = predict(logistic_mod, Xtrain, type = 'response')
mean(ytrain * log(phat_train) + (1 - ytrain) * log(1 - phat_train)) #log score computation
```

```
## [1] -0.2885109
```

```r
mean(-(ytrain - phat_train)^2) #brier score computations
```

```
## [1] -0.09365104
```

Then use this probability estimation model to do classification by thresholding at 0.5. Tabulate a confusion matrix and compute the misclassification error.

```
y_hat_train = ifelse(phat_train >= 0.5, 1, 0)
table(ytrain, y_hat_train)
```

```
##       y_hat_train
## ytrain   0   1
##      0 712  50
##      1  89 149
```

```
mean(ytrain != y_hat_train)
```

```
## [1] 0.139
```

We will be doing model selection using a basis of linear features consisting of all interactions of the 14 raw features. Create a model matrix from the training data containing all these features. Make sure it has an intercept column too (the one vector is usually an important feature). Cast it as a data frame so we can use it more easily for modeling later on.

```
Xmm_train = data.frame(model.matrix(~ . * ., Xtrain))
dim(Xmm_train)
```

```
## [1] 1000  755
```

We're going to need those model matrices (as data frames) for both the select and test sets. So make them here:

```
Xmm_select = data.frame(model.matrix(~ . * . , Xselect))
dim(Xmm_select)
```

```
## [1] 1000  755
```

```
Xmm_test = data.frame(model.matrix(~ . * . , Xtest))
dim(Xmm_test)
```

```
## [1] 1000  755
```

Write code that will fit a model stepwise. You can refer to the chunk of line 83 in practice lecture 12. Use the Brier score to do the selection. Run the code and hit "stop" when you begin to the see the Brier score degrade appreciably oos. Be patient as it will wobble.

```
pacman::p_load(Matrix)
p_plus_one = ncol(Xmm_train)
predictor_by_iteration = c() #keep a growing list of predictors by iteration
in_sample_brier_by_iteration = c() #keep a growing list of se's by iteration
oos_brier_by_iteration = c() #keep a growing list of se's by iteration
i = 1

repeat {

  #get all predictors left to try
```

```r
  all_brier = array(NA, p_plus_one) #record all possibilities
  for (j_try in 1 : p_plus_one){
    if (!(j_try %in% predictor_by_iteration)){
      Xmm_sub = Xmm_train[, c(predictor_by_iteration, j_try), drop = FALSE]
      #we need a check here to ensure the matrix is full rank
      if (ncol(Xmm_sub) > rankMatrix(Xmm_sub)){
        next
      }
      #use suppressWarnings to get this to run without blasting the console
      logistic_mod = suppressWarnings(glm(ytrain ~ ., data.frame(Xmm_sub), family = "binomial"))
      phatTrain = suppressWarnings(predict(logistic_mod, data.frame(Xmm_sub), type = 'response'))
      all_brier[j_try] = mean(-(ytrain - phatTrain)^2)
    }
  }
  j_star = which.max(all_brier) #We didn't catch this in lab... it has to be max Brier.
  predictor_by_iteration = c(predictor_by_iteration, j_star)
  in_sample_brier_by_iteration = c(in_sample_brier_by_iteration, all_brier[j_star])

  #now let's look at oos
  Xmm_sub = Xmm_train[, predictor_by_iteration, drop = FALSE]
  logistic_mod = suppressWarnings(glm(ytrain ~ ., data.frame(Xmm_sub), family = "binomial"))
  phatTrain = suppressWarnings(predict(logistic_mod, data.frame(Xmm_sub), type = 'response'))
  all_brier[j_try] = mean(-(ytrain - phatTrain)^2)

  phat_select = suppressWarnings(predict(logistic_mod, data.frame(Xmm_select[, predictor_by_iteration, 

  oos_brier = mean(-(yselect - phat_select)^2)
  oos_brier_by_iteration = c(oos_brier_by_iteration, oos_brier)

  cat("i =", i, "in sample brier = ", all_brier[j_star], "oos brier =", oos_brier, "\n   predictor adde

  i = i + 1

  if (i > 5000 || i > p_plus_one){
    break #why??
  }
}
```

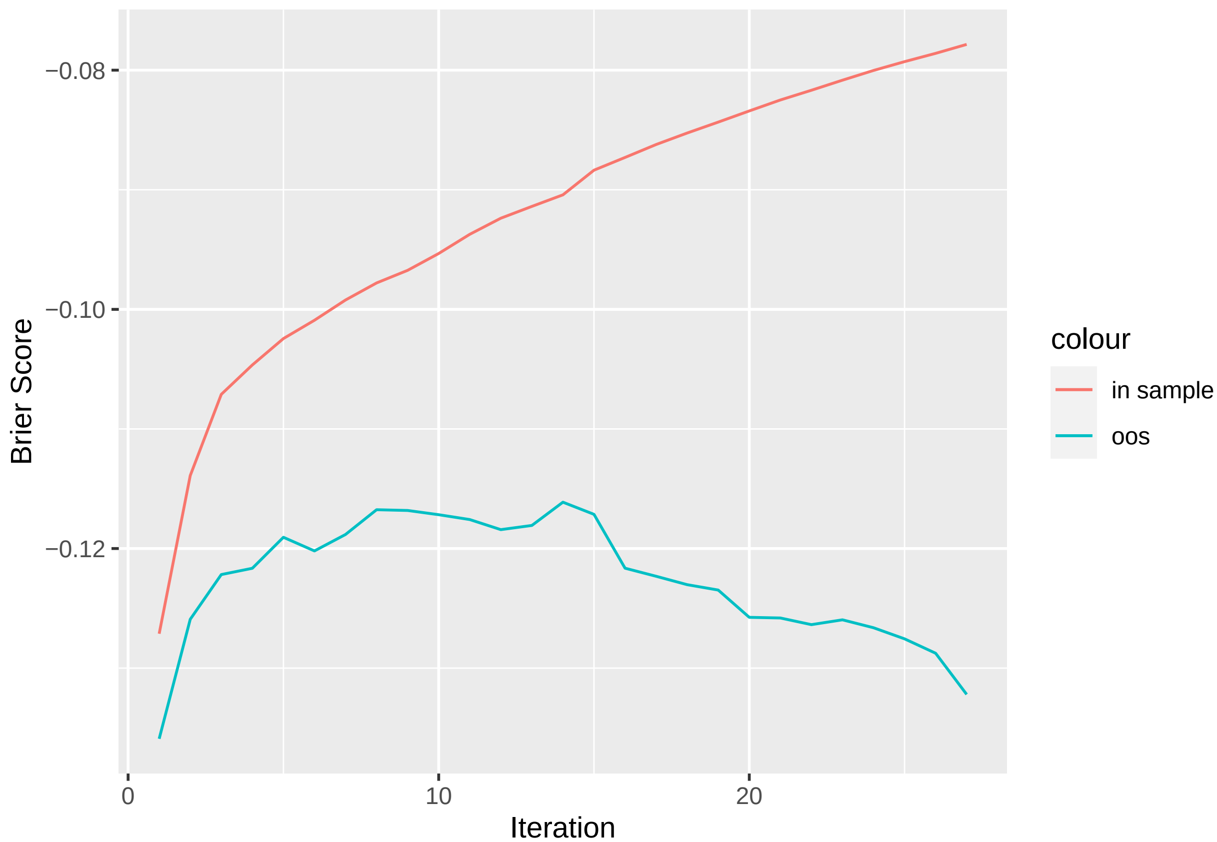
```
## i = 1 in sample brier =  -0.1271238 oos brier = -0.1359097
##    predictor added: education_num.marital_statusMarried.civ.spouse
## i = 2 in sample brier =  -0.1139004 oos brier = -0.1259166
##    predictor added: age.capital_gain
## i = 3 in sample brier =  -0.1071079 oos brier = -0.1221774
##    predictor added: age.education_num
## i = 4 in sample brier =  -0.1046498 oos brier = -0.1216508
##    predictor added: education_num.capital_loss
## i = 5 in sample brier =  -0.1024441 oos brier = -0.1190587
##    predictor added: fnlwgt.occupationExec.managerial
## i = 6 in sample brier =  -0.1009124 oos brier = -0.1202012
##    predictor added: relationshipWife.raceWhite
## i = 7 in sample brier =  -0.09921886 oos brier = -0.1188251
##    predictor added: marital_statusMarried.civ.spouse.hours_per_week
## i = 8 in sample brier =  -0.09779324 oos brier = -0.1167559
```

```
##    predictor added: age.workclassSelf.emp.not.inc
## i = 9 in sample brier =  -0.09673786 oos brier = -0.1168182
##    predictor added: relationshipWife.raceAsian.Pac.Islander
## i = 10 in sample brier =  -0.09533004 oos brier = -0.1171725
##    predictor added: sexMale
## i = 11 in sample brier =  -0.09372471 oos brier = -0.1175782
##    predictor added: sexMale.capital_loss
## i = 12 in sample brier =  -0.09237827 oos brier = -0.1184199
##    predictor added: occupationProf.specialty.relationshipOwn.child
## i = 13 in sample brier =  -0.09139228 oos brier = -0.1180696
##    predictor added: marital_statusNever.married.capital_gain
## i = 14 in sample brier =  -0.09042432 oos brier = -0.1161255
##    predictor added: occupationOther.service.sexMale
## i = 15 in sample brier =  -0.08836353 oos brier = -0.1171408
##    predictor added: workclassSelf.emp.inc.occupationOther.service
## i = 16 in sample brier =  -0.08729914 oos brier = -0.1216441
##    predictor added: workclassLocal.gov.marital_statusMarried.civ.spouse
## i = 17 in sample brier =  -0.08621779 oos brier = -0.1223202
##    predictor added: age.occupationTransport.moving
## i = 18 in sample brier =  -0.08525636 oos brier = -0.1230216
##    predictor added: marital_statusWidowed.raceAsian.Pac.Islander
## i = 19 in sample brier =  -0.08434343 oos brier = -0.1234723
##    predictor added: workclassSelf.emp.not.inc.occupationProf.specialty
## i = 20 in sample brier =  -0.08340996 oos brier = -0.1257542
##    predictor added: workclassLocal.gov.occupationExec.managerial
## i = 21 in sample brier =  -0.08249394 oos brier = -0.1258082
##    predictor added: relationshipWife.capital_gain
## i = 22 in sample brier =  -0.08167932 oos brier = -0.1263641
##    predictor added: marital_statusSeparated.occupationMachine.op.inspct
## i = 23 in sample brier =  -0.08083868 oos brier = -0.1259613
##    predictor added: fnlwgt.capital_loss
## i = 24 in sample brier =  -0.08002525 oos brier = -0.126624
##    predictor added: occupationFarming.fishing.capital_gain
## i = 25 in sample brier =  -0.07928562 oos brier = -0.1275523
##    predictor added: workclassLocal.gov.occupationTransport.moving
## i = 26 in sample brier =  -0.0785932 oos brier = -0.1287562
##    predictor added: occupationTech.support.relationshipNot.in.family
## i = 27 in sample brier =  -0.0778442 oos brier = -0.1321986
##    predictor added: sexMale.hours_per_week
```

Plot the in-sample and oos (select set) Brier score by p. Does this look like what's expected?

```r
pacman::p_load(ggplot2)

simulation_results = data.frame(
  iteration = 1:length(in_sample_brier_by_iteration),
  in_sample_brier_by_iteration = in_sample_brier_by_iteration,
  oos_brier_by_iteration = oos_brier_by_iteration
)

ggplot(data = simulation_results) +
  geom_line(aes(x = iteration, y = oos_brier_by_iteration, color = "oos")) +
  geom_line(aes(x = iteration, y = in_sample_brier_by_iteration, color = "in sample")) +
  xlab("Iteration") +
```

```r
  ylab("Brier Score")
```



Print out the coefficients of the model selection procedure's guess as to the locally optimal probability estimation model and interpret the five largest (in abolute value) coefficients. Do the signs make sense on these coefficients?

```r
p_optimal = which.max(oos_brier_by_iteration)

optimal_model = glm(ytrain ~ ., Xmm_train[predictor_by_iteration[1:p_optimal]], family = "binomial")

five_largest = sort(abs(optimal_model$coefficients), decreasing = TRUE)[1:5]

five_largest_coef = c()
for (i in 1:(p_optimal + 1)) {
  for (j in 1:5){
    if (abs(optimal_model$coefficients[i]) == five_largest[j]){
      five_largest_coef = c(five_largest_coef, optimal_model$coefficients[i])
    }
  }
}
five_largest_coef
```

```
##                                     (Intercept)
##                                       -7.742027
##                         relationshipWife.raceWhite
```

```
##                                   3.334433
##        relationshipWife.raceAsian.Pac.Islander
##                                  18.747663
##                                    sexMale
##                                   2.126989
## occupationProf.specialty.relationshipOwn.child
##                                   3.695270
```

Use this locally optimal probability estimation model to make predictions in all three data sets: train, select test. Compare to the Brier scores across all three sets. Is this expected?

```r
phatTrain = predict(optimal_model, Xmm_train[predictor_by_iteration[1:p_optimal]], type = 'response')
mean(-(ytrain - phatTrain)^2)
```

```
## [1] -0.09042432
```

```r
phatSelect = predict(optimal_model, Xmm_select[predictor_by_iteration[1:p_optimal]], type = 'response')
mean(-(yselect - phatSelect)^2)
```

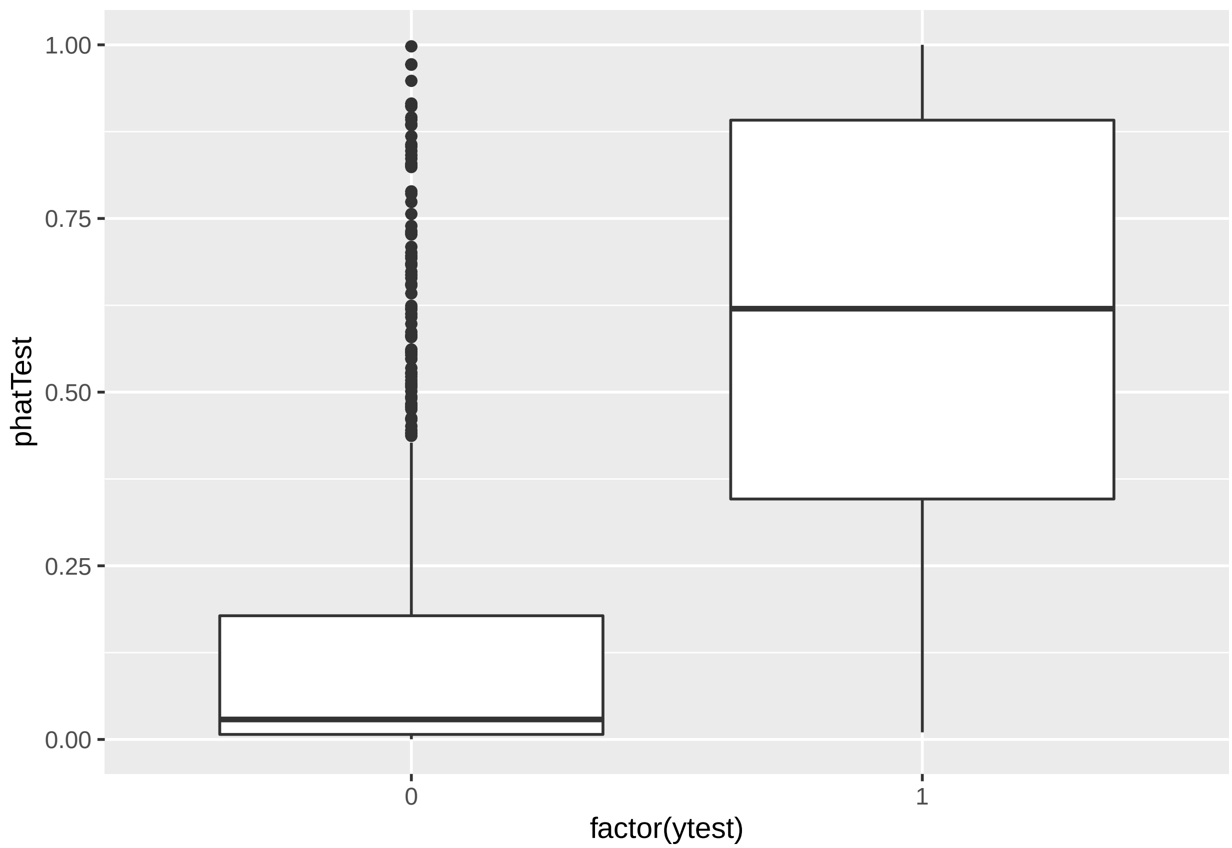
```
## [1] -0.1161255
```

```r
phatTest = predict(optimal_model, Xmm_test[predictor_by_iteration[1:p_optimal]], type = 'response')
mean(-(ytest - phatTest)^2)
```

```
## [1] -0.110359
```

Plot the probability predictions in the test set by `y`. Does this plot look good?

```r
ggplot() +
  geom_boxplot(aes(x = factor(ytest), y = phatTest))
```



Calculate misclassification error, sensitivity (recall), specificity (true negative rate, TN / N), FDR, FOR for this model if you threshold at phat = 0.5. Interpret these metrics.

```
classifi = rep(NA, length(ytrain))
TN = rep(NA, length(ytrain))
FN = rep(NA, length(ytrain))
TP = rep(NA, length(ytrain))
FP = rep(NA, length(ytrain))

for (i in 1:length(ytrain)) {
  classifi[i] = ifelse(phatTrain[i] >= 0.5, 1, 0)

  TN[i] = ifelse(classifi[i] == 0 & ytrain[i] == 0, 1, 0)
  FN[i] = ifelse(classifi[i] == 0 & ytrain[i] == 1, 1, 0)
  TP[i] = ifelse(classifi[i] == 1 & ytrain[i] == 1, 1, 0)
  FP[i] = ifelse(classifi[i] == 1 & ytrain[i] == 0, 1, 0)
}

PN = sum(TN) + sum(FN)
PP = sum(FP) + sum(TP)
N = sum(TN) + sum(FP)
P = sum(FN) + sum(TP)
n = PN + PP

err = (sum(FP) + sum(FN)) / n
sensitivity = sum(TP) / P
```

```r
specificity = sum(TN) / N
FDR = sum(FP) / sum(PP)
FOR = sum(FN) / sum(PN)

err             # false prediction 13.5% of the time
```

```
## [1] 0.135
```

```r
sensitivity     # 61.8% of positives are predicted positive
```

```
## [1] 0.6176471
```

```r
specificity     # 94.2% of negatives are predicted negative
```

```
## [1] 0.9422572
```

```r
FDR             # 23% of predicted positives are false
```

```
## [1] 0.2303665
```

```r
FOR             # 11.2% of predicted negatives are false
```

```
## [1] 0.1124845
```

Now, consider an asymmetric costs scenario. Let's say you're trying to sell people luxury products and want to advertise with only high-salaried individuals. Since your advertising is expensive, you want to not waste money on people who do not make a high salary. Thus your cost of predicting >50K when it truly is <=50K, i.e. a false positive (FP), is higher than predicting <=50K when the person truly makes >50K, i.e. a false negative (FN). Set the cost of FP to 3x more than the cost of FN. Use a grid of 0.001 to step through thresholds for the locally optimal probability estimation model (source the function from practice lecture 15). Do this in the selection dataset.

```r
#' Computes performance metrics for a binary probabilistic classifer
#'
#' Each row of the result will represent one of the many models and its elements record the performance
#'
#' @param p_hats  The probability estimates for n predictions
#' @param y_true  The true observed responses
#' @param res     The resolution to use for the grid of threshold values (defaults to 1e-3)
#'
#' @return        The matrix of all performance results
compute_metrics_prob_classifier = function(p_hats, y_true, res = 0.001){
  #we first make the grid of all prob thresholds
  p_thresholds = seq(0 + res, 1 - res, by = res) #values of 0 or 1 are trivial

  #now we create a matrix which will house all of our results
  performance_metrics = matrix(NA, nrow = length(p_thresholds), ncol = 12)
  colnames(performance_metrics) = c(
    "p_th",
```

```r
  "TN",
  "FP",
  "FN",
  "TP",
  "miscl_err",
  "precision",
  "recall",
  "FDR",
  "FPR",
  "FOR",
  "miss_rate"
)

#now we iterate through each p_th and calculate all metrics about the classifier and save
n = length(y_true)
for (i in 1 : length(p_thresholds)){
  p_th = p_thresholds[i]
  # yhats
  classifi[i] = ifelse(p_hats[i] >= p_th, 1, 0)

  TN[i] = ifelse(classifi[i] == 0 & y_true[i] == 0, 1, 0)
  FN[i] = ifelse(classifi[i] == 0 & y_true[i] == 1, 1, 0)
  TP[i] = ifelse(classifi[i] == 1 & y_true[i] == 1, 1, 0)
  FP[i] = ifelse(classifi[i] == 1 & y_true[i] == 0, 1, 0)

  PN = sum(TN) + sum(FN)
  PP = sum(FP) + sum(TP)
  N = sum(TN) + sum(FP)
  P = sum(FN) + sum(TP)
  n = PN + PP

  t = c(
    p_th,
    sum(TN), #TN
    sum(FP), #FP
    sum(FN), #FN
    sum(TP), #TP
    (sum(FP) + sum(FN)) / n,
    sum(TP) / PP, #precision
    sum(TP) / P,  #recall
    sum(FP) / sum(PP), #false discovery rate (FDR)
    sum(FP) / N,  #false positive rate (FPR)
    sum(FN) / sum(PN), #false omission rate (FOR)
    sum(FN) / P   #miss rate
  )

  for (j in 1:12) {
    performance_metrics[i, j] = t[j]
  }

}

#finally return the data frame
```

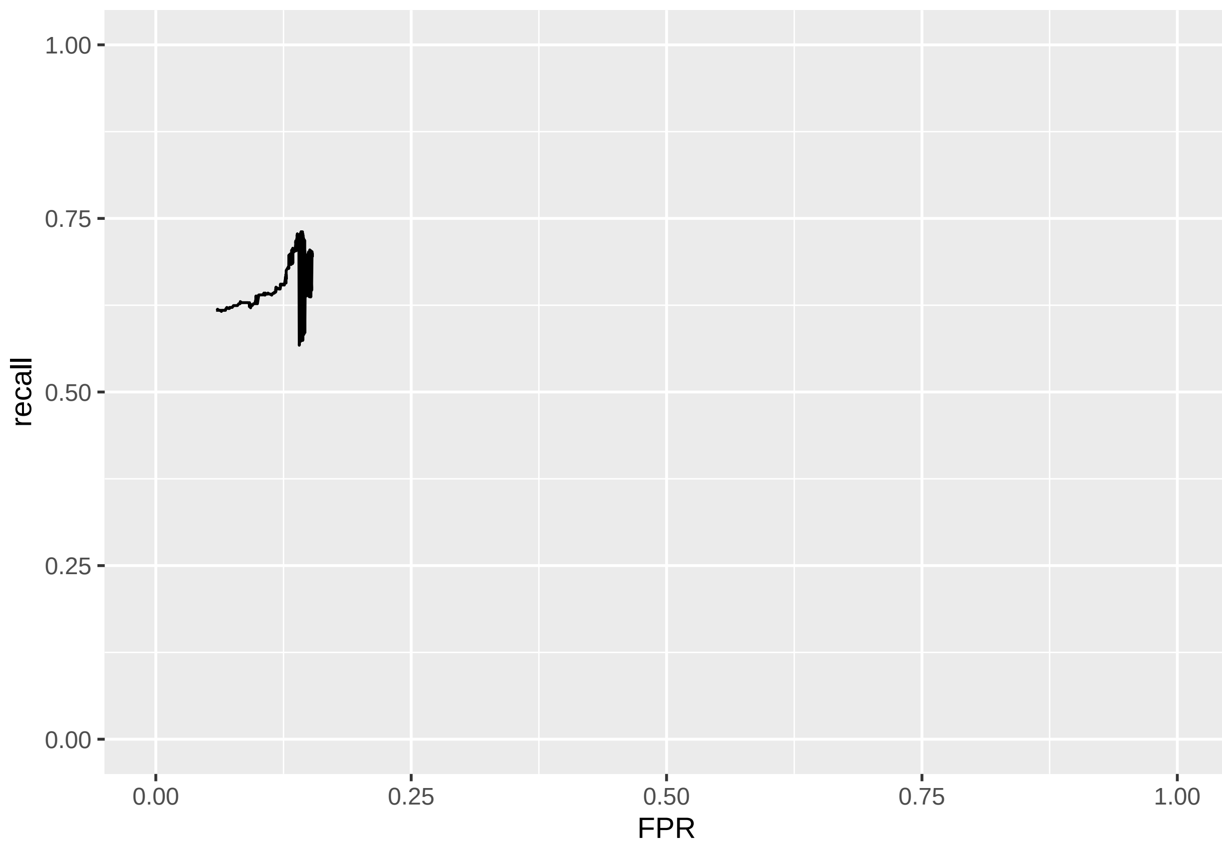
```r
  data.frame(performance_metrics)
}


performance = compute_metrics_prob_classifier(phatSelect, yselect)
head(performance)
```

```
##    p_th  TN FP FN  TP miscl_err precision    recall       FDR        FPR
## 1 0.001 717 45 91 147     0.136 0.7656250 0.6176471 0.2343750 0.05905512
## 2 0.002 717 45 91 147     0.136 0.7656250 0.6176471 0.2343750 0.05905512
## 3 0.003 716 46 91 147     0.137 0.7616580 0.6176471 0.2383420 0.06036745
## 4 0.004 715 46 91 148     0.137 0.7628866 0.6192469 0.2371134 0.06044678
## 5 0.005 715 47 91 147     0.138 0.7577320 0.6176471 0.2422680 0.06167979
## 6 0.006 714 48 91 147     0.139 0.7538462 0.6176471 0.2461538 0.06299213
##         FOR miss_rate
## 1 0.1126238 0.3823529
## 2 0.1126238 0.3823529
## 3 0.1127633 0.3823529
## 4 0.1129032 0.3807531
## 5 0.1129032 0.3823529
## 6 0.1130435 0.3823529
```

```r
c_FP = -1
c_FN = 3 * c_FP
```

Plot an ROC curve for the selection dataset.

```r
ggplot(data = performance) +
  geom_line(aes(x = FPR, y = recall)) +
  xlim(0, 1) +
  ylim(0, 1)
```



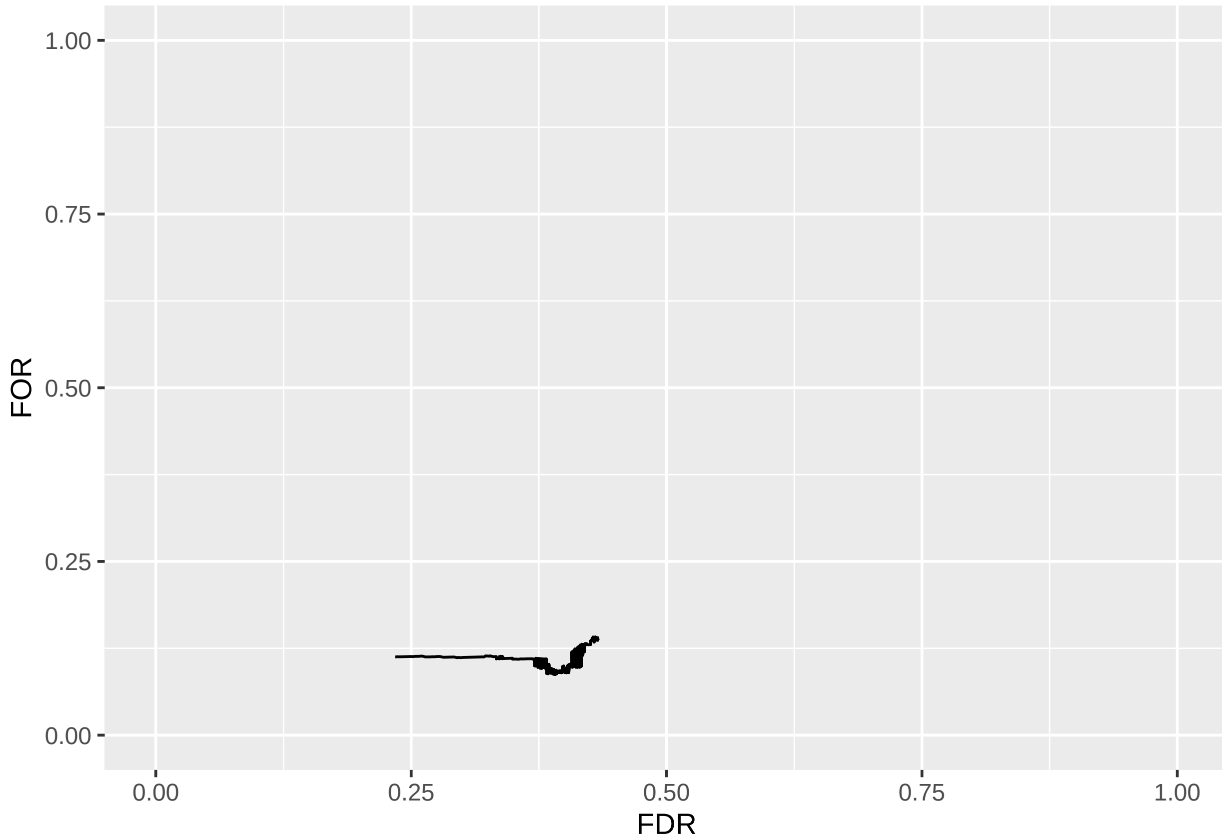
Calculate AUC and interpret.

```
pacman::p_load(pracma)
trapz(performance$FPR, performance$recall)
```

```
## [1] 0.05327928
```

Plot a DET curve for the selection dataset.

```
ggplot(data = performance) +
  geom_line(aes(x = FDR, y = FOR)) +
  xlim(0, 1) +
  ylim(0, 1)
```



Calculate total cost for each classification model defined by each threshold.

```
performance$cost = (performance$FP * c_FP) + (performance$FN * c_FN)
```

Find the probability estimate threshold for the locally optimal asymmetric cost model for your FP and FN costs. Use this optimal probability estimate threshold and classify the test set. Print out its confusion matrix in the test set and calculate average cost per future observation, future FDR and future FOR and interpret these metrics in the context of this scenario. Is this model successful in internalizing your asymmetric costs?

```
opt_threshold = performance$p_th[which.max(performance$cost)]

for (i in 1:length(ytest)) {
  classifi[i] = ifelse(phatTest[i] >= opt_threshold, 1, 0)
}

pacman::p_load(e1071)
c_matrix = t(matrix(caret::confusionMatrix(table(phat = classifi, ytest))$table, nrow = 2, ncol = 2))
c_matrix = rbind(c_matrix, c(sum(c_matrix[, 1]), sum(c_matrix[, 2])))
c_matrix = cbind(c_matrix, c(sum(c_matrix[1, ]), sum(c_matrix[2, ]), length(ytest)))
rownames(c_matrix) = c("y=0", "y=1", "Total")
colnames(c_matrix) = c("yhat=0", "yhat=1", "Total")
c_matrix
```

```
##       yhat=0 yhat=1 Total
## y=0      633    126   759
```

```
## y=1        56    185   241
## Total     689    311  1000
```

```
avg_cost = ((c_matrix[1, 2] * c_FP) + (c_matrix[2, 1] * c_FN)) / length(ytest)
avg_cost
```

```
## [1] -0.294
```

```
future_FDR = c_matrix[1, 2] / c_matrix[3, 2]
future_FDR      # 40.5% of predicted positives are false
```

```
## [1] 0.4051447
```

```
future_FOR = c_matrix[2, 1] / c_matrix[3, 1]
future_FOR      # 8.1% of predicted negatives are false
```

```
## [1] 0.08127721
```

Throughout the next part of this assignment you can use either the `tidyverse` package suite or `data.table` to answer but not base R. You can mix `data.table` with `magrittr` piping if you wish but don't go back and forth between `tbl_df`'s and `data.table` objects.

```
rm(list = ls())
pacman::p_load(tidyverse, magrittr, data.table)
```

We will be using the `storms` dataset from the `dplyr` package. Filter this dataset on all storms that have no missing measurements for the two diameter variables, "ts_diameter" and "hu_diameter".

```
data(storms)
storms
```

```
## # A tibble: 10,010 x 13
##    name   year month   day  hour   lat  long status category  wind pressure
##    <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
##  1 Amy    1975     6    27     0  27.5 -79   tropi~ -1          25     1013
##  2 Amy    1975     6    27     6  28.5 -79   tropi~ -1          25     1013
##  3 Amy    1975     6    27    12  29.5 -79   tropi~ -1          25     1013
##  4 Amy    1975     6    27    18  30.5 -79   tropi~ -1          25     1013
##  5 Amy    1975     6    28     0  31.5 -78.8 tropi~ -1          25     1012
##  6 Amy    1975     6    28     6  32.4 -78.7 tropi~ -1          25     1012
##  7 Amy    1975     6    28    12  33.3 -78   tropi~ -1          25     1011
##  8 Amy    1975     6    28    18  34   -77   tropi~ -1          30     1006
##  9 Amy    1975     6    29     0  34.4 -75.8 tropi~ 0           35     1004
## 10 Amy    1975     6    29     6  34   -74.8 tropi~ 0           40     1002
## # ... with 10,000 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

```
storms %<>%
  filter(!is.na(ts_diameter) & !is.array(hu_diameter)) %>%
  group_by(name) %>%
  mutate(obs_period = row_number())
```

From this subset, create a data frame that only has storm, observation period number (i.e., 1, 2, . . . , T) and the "ts_diameter" and "hu_diameter" metrics.

```
storms %<>%
  select(name, obs_period, ts_diameter, hu_diameter)
```

Create a data frame in long format with columns "diameter" for the measurement and "diameter_type" which will be categorical taking on the values "hu" or "ts".

```
storms_long = storms %>%
  gather(diameter_type, diameter, ts_diameter:hu_diameter) %>%
  mutate(diameter_type = ifelse(diameter_type == "ts_diameter", "ts", "hu"))
```

Using this long-formatted data frame, use a line plot to illustrate both "ts_diameter" and "hu_diameter" metrics by observation period for four random storms using a 2x2 faceting. The two diameters should appear in two different colors and there should be an appropriate legend.

```
random_storms = sample(storms$name, 4)

storms_long %>%
  filter(name %in% random_storms) %>%
  ggplot() +
  geom_line(aes(x = obs_period, y = diameter, color = diameter_type)) +
  facet_wrap(vars(name)) +
  xlab("Observation Period") +
  ylab("Diameter")
```