Lab 8

Marin Azhar

11:59PM April 29, 2021

I want to make some use of my CART package. Everyone please try to run the following:

```
#if (!pacman::p_isinstalled(YARF)){
#  pacman::p_install_gh("kapelner/YARF/YARFJARs", ref = "dev")
 # pacman::p_install_gh("kapelner/YARF/YARF", ref = "dev", force = TRUE)
#}
#options(java.parameters = "-Xmx4000m")
#pacman::p_load(YARF)
```

For many of you it will not work. That's okay.

Throughout this 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.

```
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".

```
#TO-DO
data(storms)
storms2 = storms%>%
  filter(!is.na(ts_diameter )& !is.na(hu_diameter ) & ts_diameter> 0 &
hu_diameter >0)

storms2

## # A tibble: 1,022 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 Alex   2004     8     3     6  33   -77.4 hurricane 1           70
983
##  2 Alex   2004     8     3    12  34.2 -76.4 hurricane 2           85
974
##  3 Alex   2004     8     3    18  35.3 -75.2 hurricane 2           85
972
##  4 Alex   2004     8     4     0  36   -73.7 hurricane 1           80
974
##  5 Alex   2004     8     4     6  36.8 -72.1 hurricane 1           80
```

```
973
##   6 Alex   2004     8     4    12  37.3 -70.2 hurricane 2            85
973
##   7 Alex   2004     8     4    18  37.8 -68.3 hurricane 2            95
965
##   8 Alex   2004     8     5     0  38.5 -66   hurricane 3           105
957
##   9 Alex   2004     8     5     6  39.5 -63.1 hurricane 3           105
957
##  10 Alex   2004     8     5    12  40.8 -59.6 hurricane 3           100
962
## # … with 1,012 more rows, and 2 more variables: ts_diameter <dbl>,
## #   hu_diameter <dbl>
```

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

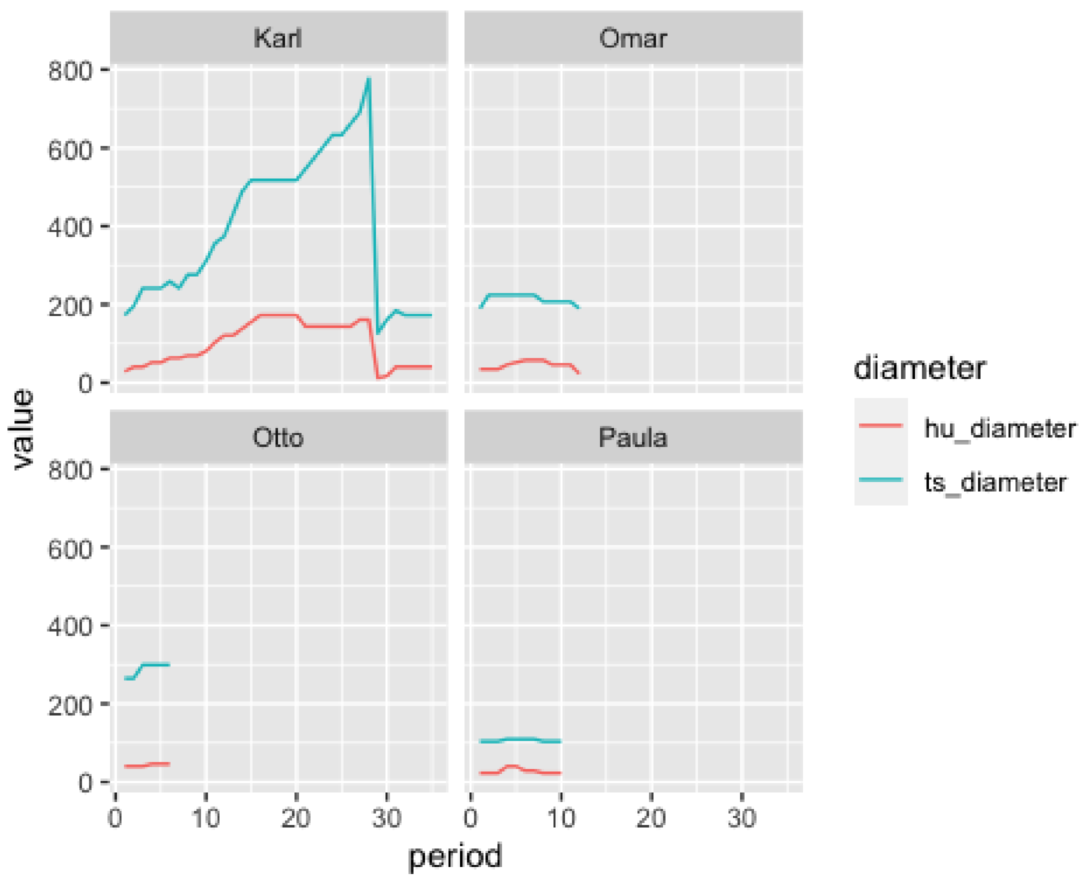
```
#TO-DO

storms2 = storms2%>%
  select(name, ts_diameter, hu_diameter)%>%
  group_by(name)%>%
  mutate(period = row_number())
```

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".

```
#TO-DO
storms_long =pivot_longer(storms2, cols =matches("diameter"), names_to =
"diameter")
storms_long

## # A tibble: 2,044 x 4
## # Groups:   name [63]
##    name  period diameter    value
##    <chr>  <int> <chr>       <dbl>
##  1 Alex       1 ts_diameter 150.
##  2 Alex       1 hu_diameter  46.0
##  3 Alex       2 ts_diameter 150.
##  4 Alex       2 hu_diameter  46.0
##  5 Alex       3 ts_diameter 190.
##  6 Alex       3 hu_diameter  57.5
##  7 Alex       4 ts_diameter 178.
##  8 Alex       4 hu_diameter  63.3
##  9 Alex       5 ts_diameter 224.
## 10 Alex       5 hu_diameter  74.8
## # … with 2,034 more rows

#the cols argument tells the pivot longer which of the cols to consolidate to
one long col
```

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.

```
#TO-DO
storms_sample = sample(unique(storms2$name), 4)

ggplot(storms_long%>% filter(name %in% storms_sample))+
  geom_line(aes(x=period, y=value, col = diameter))+         #we squished
all four storms....
  facet_wrap(name ~., nrow =2)
```



In this next first part of this lab, we will be joining three datasets in an effort to make a design matrix that predicts if a bill will be paid on time. Clean up and load up the three files. Then I’ll rename a few features and then we can examine the data frames:

```
rm(list = ls())
pacman::p_load(tidyverse, magrittr, data.table, R.utils)
bills =
fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/b
ills_dataset/bills.csv.bz2")
payments =
```

```
fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/payments.csv.bz2")
discounts =
fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/discounts.csv.bz2")
setnames(bills, "amount", "tot_amount")
setnames(payments, "amount", "paid_amount")
head(bills)

##          id   due_date invoice_date tot_amount customer_id discount_id
## 1: 15163811 2017-02-12   2017-01-13   99490.77    14290629     5693147
## 2: 17244832 2016-03-22   2016-02-21   99475.73    14663516     5693147
## 3: 16072776 2016-08-31   2016-07-17   99477.03    14569622     7302585
## 4: 15446684 2017-05-29   2017-05-29   99478.60    14488427     5693147
## 5: 16257142 2017-06-09   2017-05-10   99678.17    14497172     5693147
## 6: 17244880 2017-01-24   2017-01-24   99475.04    14663516     5693147

head(payments)

##          id paid_amount transaction_date  bill_id
## 1: 15272980    99165.60       2017-01-16 16571185
## 2: 15246935    99148.12       2017-01-03 16660000
## 3: 16596393    99158.06       2017-06-19 16985407
## 4: 16596651    99175.03       2017-06-19 17062491
## 5: 16687702    99148.20       2017-02-15 17184583
## 6: 16593510    99153.94       2017-06-11 16686215

head(discounts)

##         id num_days pct_off days_until_discount
## 1: 5000000       20      NA                  NA
## 2: 5693147       NA       2                  NA
## 3: 6098612       20      NA                  NA
## 4: 6386294      120      NA                  NA
## 5: 6609438       NA       1                   7
## 6: 6791759       31       1                  NA

#the data is fake data
#bill data and payment data has a relation

bills =as_tibble(bills) #data.table to
payments =as_tibble(payments)
discounts =as_tibble(discounts)
```

The unit we care about is the bill. The y metric we care about will be "paid in full" which is 1 if the company paid their total amount (we will generate this y metric later).

Since this is the response, we would like to construct the very best design matrix in order to predict y.

I will create the basic steps for you guys. First, join the three datasets in an intelligent way. You will need to examine the datasets beforehand.

```
bills_with_payments = left_join(bills, payments, by = c("id" = "bill_id"),
all.x = TRUE)
#id payments = id.y
bills_with_payments

## # A tibble: 279,118 x 9
##          id due_date   invoice_date tot_amount customer_id discount_id
id.y
##       <dbl> <date>     <date>            <dbl>       <int>       <dbl>
<dbl>
##  1 15163811 2017-02-12 2017-01-13       99491.    14290629     5693147
14670862
##  2 17244832 2016-03-22 2016-02-21       99476.    14663516     5693147
16691206
##  3 16072776 2016-08-31 2016-07-17       99477.    14569622     7302585
NA
##  4 15446684 2017-05-29 2017-05-29       99479.    14488427     5693147
16591210
##  5 16257142 2017-06-09 2017-05-10       99678.    14497172     5693147
16538398
##  6 17244880 2017-01-24 2017-01-24       99475.    14663516     5693147
16691231
##  7 16214048 2017-03-08 2017-02-06       99475.    14679281     5693147
16845763
##  8 15579946 2016-06-13 2016-04-14       99476.    14450223     5693147
16593380
##  9 15264234 2014-06-06 2014-05-07       99480.    14532786     7708050
16957842
## 10 17031731 2017-01-12 2016-12-13       99476.    14658929     5693147
NA
## # … with 279,108 more rows, and 2 more variables: paid_amount <dbl>,
## #   transaction_date <date>

  bills_with_payments_with_discounts = left_join(bills_with_payments,
discounts, by = c("discount_id" ="id" ), all.x = TRUE)
```

Now create the binary response metric `paid_in_full` as the last column and create the beginnings of a design matrix `bills_data`. Ensure the unit / observation is bill i.e. each row should be one bill!

```
#TO-DO
bills_data = bills_with_payments_with_discounts%>%
  mutate(tot_amount = if_else(is.na(pct_off), tot_amount, tot_amount*(1-
pct_off/100)))%>%
  group_by(id)%>%
  mutate(sum_of_payment_amount = sum(paid_amount))%>%
  mutate(paid_in_full = if_else(sum_of_payment_amount >= tot_amount, 1,0,
missing =0  ))%>%
```

```
  slice(1)%>%
  ungroup()
table(bills_data$paid_in_full, useNA = "always")
```

```
##
##      0      1   <NA>
## 112664 113770      0
```

How should you add features from transformations (called "featurization")? What data type(s) should they be? Make some features below if you think of any useful ones. Name the columns appropriately so another data scientist can easily understand what information is in your variables.

```
#TO-DO



  pacman::p_load("lubridate")
  bills_data = bills_data %>%
  select(-id, -id.y, -num_days, -transaction_date, -pct_off, -
days_until_discount, -sum_of_payment_amount, -paid_amount) %>%
  mutate(num_days_to_pay = as.integer(ymd(due_date) - ymd(invoice_date))) %>%
  select(-due_date, -invoice_date) %>%
  mutate(discount_id = as.factor(discount_id)) %>%
  group_by(customer_id) %>%
  mutate(bill_num = row_number()) %>%
  ungroup() %>%
  select(-customer_id) %>%
  relocate(paid_in_full, .after = last_col())
```

Now let's do this exercise. Let's retain 25% of our data for test.

```
K = 4
test_indices = sample(1 : nrow(bills_data), round(nrow(bills_data) / K))
train_indices = setdiff(1 : nrow(bills_data), test_indices)
bills_data_test = bills_data[test_indices, ]
bills_data_train = bills_data[train_indices, ]
```

Now try to build a classification tree model for `paid_in_full` with the features (use the `Xy` parameter in `YARF`). If you cannot get `YARF` to install, use the package `rpart` (the standard R tree package) instead. You will need to install it and read through some documentation to find the correct syntax.

Warning: this data is highly anonymized and there is likely zero signal! So don't expect to get predictive accuracy. The value of the exercise is in the practice. I think this exercise (with the joining exercise above) may be one of the most useful exercises in the entire semester.

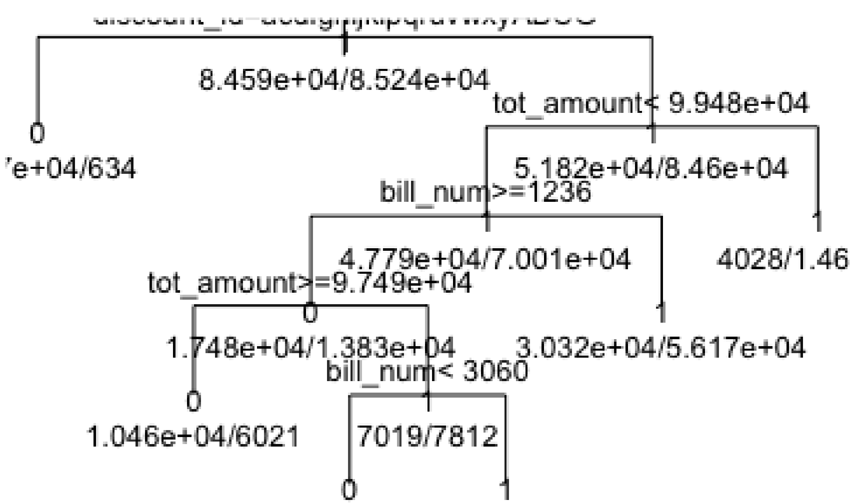
```
#TO-DO
pacman::p_load(rpart)
mod1 = rpart(paid_in_full ~., data = bills_data_train, method = "class")
```

For those of you who installed YARF, what are the number of nodes and depth of the tree?

```
#TO-DO
nrow(mod1$frame) ##number of nodes

## [1] 11
```

For those of you who installed YARF, print out an image of the tree.

```
#TO-DO
plot(mod1, uniform=TRUE)
text(mod1, use.n=TRUE, all=TRUE, cex=.8)
```



Predict on the test set and compute a confusion matrix.

```
#TO-DO
yhat = predict(mod1, bills_data_test, type = c("class"), na.action = na.pass)
oos_conf_table=table( bills_data_test$paid_in_full,yhat)
#yhat is top most row
```

Report the following error metrics: misclassifcation error, precision, recall, F1, FDR, FOR.

```
#TO-DO
n = sum(oos_conf_table)
fp = oos_conf_table[1, 2]
```

```r
fn = oos_conf_table[2, 1]
tp = oos_conf_table[2, 2]
tn = oos_conf_table[1, 1]
num_pred_pos = sum(oos_conf_table[, 2])
num_pred_neg = sum(oos_conf_table[, 1])
num_pos = sum(oos_conf_table[2, ])
num_neg = sum(oos_conf_table[1, ])
me= (fn+fp)/n
cat("misclassifcation error", round(me*100,2), "%\n")

## misclassifcation error 27.94 %

precision = tp / num_pred_pos
cat("precision", round(precision * 100, 2), "%\n")

## precision 67.13 %

recall = tp / num_pos
cat("recall", round(recall * 100, 2), "%\n") #true positive in relation to
false negatives

## recall 87.33 %

false_discovery_rate = 1 - precision
cat("false_discovery_rate", round(false_discovery_rate * 100, 2), "%\n")
#false pos???????

## false_discovery_rate 32.87 %

false_omission_rate = fn / num_pred_neg
cat("false_omission_rate", round(false_omission_rate * 100, 2), "%\n") #
false negative

## false_omission_rate 18.55 %

#lec 21
```

Is this a good model? (yes/no and explain). #true positives over all postives #TO-DO no, it is not a good model the error rates are higher than what we want because the fdr is 1/3 which is bad it means 2/3 of the ppl are not paying

There are probability asymmetric costs to the two types of errors. Assign the costs below and calculate oos total cost.

```r
#TO-DO
c_fp =45
c_fn =2
cost =c_fp*fp + c_fn*fn
cost

## [1] 556277
```

We now wish to do asymmetric cost classification. Fit a logistic regression model to this data.

```
log_mod =glm(paid_in_full~., bills_data_train, family = binomial(link
="logit"))
#p_hat_train = predict(log_mod, bills_data_train, type ="response")
```

Use the function from class to calculate all the error metrics for the values of the probability threshold being 0.001, 0.002, ..., 0.999 in a data frame.

```
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",
    "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]
    y_hats = factor(ifelse(p_hats >= p_th, 1, 0))
    confusion_table = table(
      factor(y_true, levels = c(0, 1)),
      factor(y_hats, levels = c(0, 1))
    )

    fp = confusion_table[1, 2]
    fn = confusion_table[2, 1]
    tp = confusion_table[2, 2]
    tn = confusion_table[1, 1]
    npp = sum(confusion_table[, 2])
    npn = sum(confusion_table[, 1])
    np = sum(confusion_table[2, ])
```

```
    nn = sum(confusion_table[1, ])

    performance_metrics[i, ] = c(
      p_th,
      tn,
      fp,
      fn,
      tp,
      (fp + fn) / n,
      tp / npp, #precision
      tp / np,  #recall
      fp / npp, #false discovery rate (FDR)
      fp / nn,  #false positive rate (FPR)
      fn / npn, #false omission rate (FOR)
      fn / np   #miss rate
    )
  }

  #finally return the matrix
  performance_metrics
}

p_hat_train = predict(log_mod, bills_data_train, type ="response")



p_hat_test = predict(log_mod, bills_data_test, type = "response")

performance_metrics_in_sample =
as_tibble(compute_metrics_prob_classifier(p_hat_train,
bills_data_train$paid_in_full))
performance_metrics_in_sample

## # A tibble: 999 x 12
##      p_th    TN    FP    FN    TP miscl_err precision recall   FDR   FPR
FOR
##     <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl> <dbl>
<dbl>
##   1 0.001 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##   2 0.002 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##   3 0.003 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##   4 0.004 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##   5 0.005 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##   6 0.006 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
```

```
0.000186
## 7 0.007 10762 72871 2 85210 0.429 0.539 1.00 0.461 0.871
0.000186
## 8 0.008 10762 72871 2 85210 0.429 0.539 1.00 0.461 0.871
0.000186
## 9 0.009 10762 72871 2 85210 0.429 0.539 1.00 0.461 0.871
0.000186
## 10 0.01 14393 69240 6 85206 0.408 0.552 1.00 0.448 0.828
0.000417
## # … with 989 more rows, and 1 more variable: miss_rate <dbl>
```

```
performance_metrics_oos =
as_tibble(compute_metrics_prob_classifier(p_hat_test,
bills_data_test$paid_in_full))
performance_metrics_oos
```

```
## # A tibble: 999 x 12
##      p_th    TN    FP    FN    TP miscl_err precision recall   FDR   FPR
FOR
##     <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl> <dbl>
<dbl>
## 1 0.001 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 2 0.002 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 3 0.003 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 4 0.004 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 5 0.005 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 6 0.006 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 7 0.007 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 8 0.008 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 9 0.009 3540 24209 2 28525 0.428 0.541 1.00 0.459 0.872
0.000565
## 10 0.01 4710 23039 3 28524 0.407 0.553 1.00 0.447 0.830
0.000637
## # … with 989 more rows, and 1 more variable: miss_rate <dbl>
```

```
#metrics_prob_classifier = compute_metrics_prob_classifier(p_hat_train,
y_true)
#metrics_prob_classifier2=as_tibble(metrics_prob_classifier)
#metrics_prob_classifier2
```

Calculate the column `total_cost` and append it to this data frame.

```
c_fp =45
c_fn =2
#metrics_prob_classifier2= metrics_prob_classifier2%>%
performance_metrics_oos=performance_metrics_oos%>%
 mutate(total_cost =c_fp*FP + c_fn*FN)
performance_metrics_oos
```

```
## # A tibble: 999 x 13
##      p_th    TN    FP    FN    TP miscl_err precision recall   FDR   FPR
FOR
##     <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl> <dbl>
<dbl>
##  1 0.001  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  2 0.002  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  3 0.003  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  4 0.004  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  5 0.005  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  6 0.006  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  7 0.007  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  8 0.008  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
##  9 0.009  3540 24209     2 28525     0.428     0.541   1.00 0.459 0.872
0.000565
## 10 0.01   4710 23039     3 28524     0.407     0.553   1.00 0.447 0.830
0.000637
## # … with 989 more rows, and 2 more variables: miss_rate <dbl>, total_cost
<dbl>
```

```
performance_metrics_in_sample=performance_metrics_in_sample%>%
 mutate(total_cost =c_fp*FP + c_fn*FN)
performance_metrics_in_sample
```

```
## # A tibble: 999 x 13
##      p_th    TN    FP    FN    TP miscl_err precision recall   FDR   FPR
FOR
##     <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl> <dbl>
<dbl>
##  1 0.001 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##  2 0.002 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
##  3 0.003 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
```

```
## 4 0.004 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 5 0.005 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 6 0.006 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 7 0.007 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 8 0.008 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 9 0.009 10762 72871     2 85210     0.429     0.539   1.00 0.461 0.871
0.000186
## 10 0.01  14393 69240     6 85206     0.408     0.552   1.00 0.448 0.828
0.000417
## # … with 989 more rows, and 2 more variables: miss_rate <dbl>, total_cost
<dbl>

#metrics_prob_classifier2
#TO-DO
```

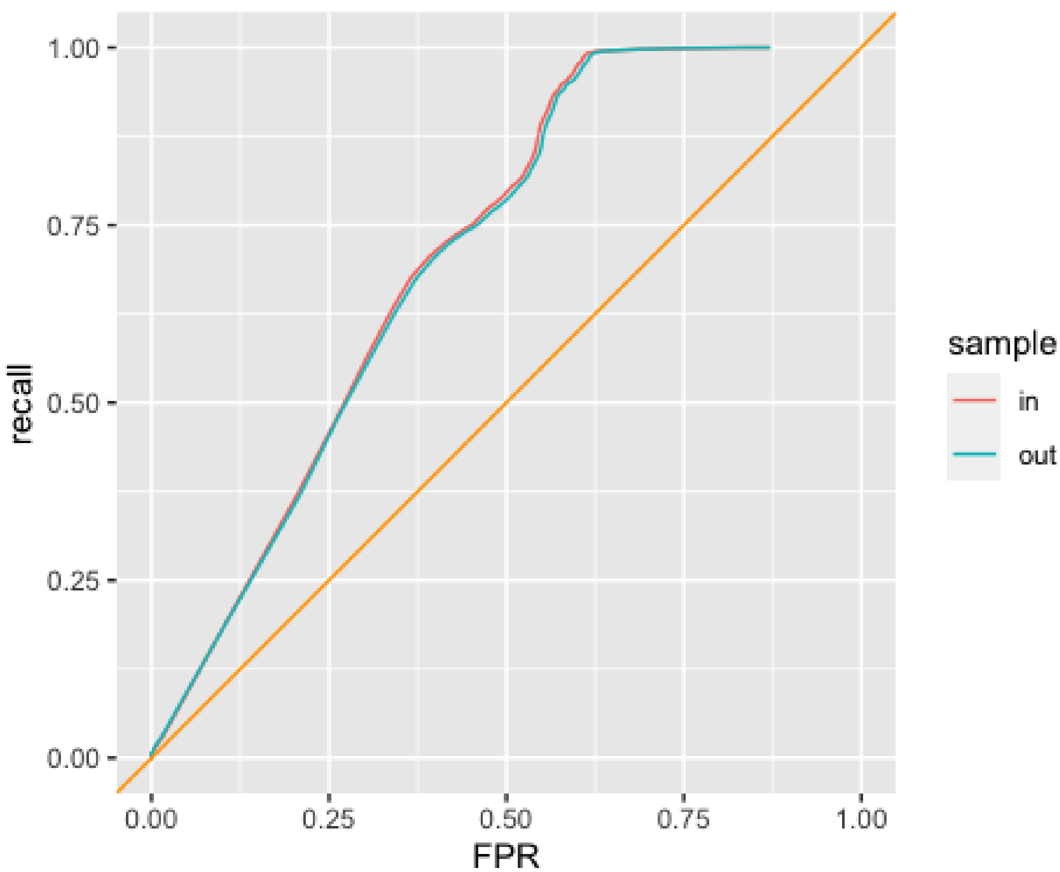
Which is the winning probability threshold value and the total cost at that threshold?

```
#TO-DO
best_prob_thsehold_index2 = which.min(performance_metrics_oos$total_cost)
best_prob_thsehold_index2

## [1] 925

best_prob_thsehold_matrix2 =
performance_metrics_oos[best_prob_thsehold_index2,]
best_prob_thsehold_matrix2

## # A tibble: 1 x 13
##    p_th    TN    FP    FN    TP miscl_err precision  recall    FDR     FPR
FOR
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>   <dbl>  <dbl>   <dbl>
<dbl>
## 1 0.925 27748     1 28497    30     0.503     0.968 0.00105 0.0323 3.60e-5
0.507
## # … with 2 more variables: miss_rate <dbl>, total_cost <dbl>

cat("total cost of win probs oos",
min(best_prob_thsehold_matrix2$total_cost))

## total cost of win probs oos 57039
```

Plot an ROC curve and interpret.

```
#TO-DO

pacman::p_load(ggplot2)
#ggplot(performance_metrics_in_sample) +
```

```r
#  geom_line(aes(x = FPR, y = recall)) +
# geom_abline(intercept = 0, slope = 1, col = "orange") +
# coord_fixed() + xlim(0, 1) + ylim(0, 1)

#now do for the oos + in sample

performance_metrics_in_and_oos= rbind(
    cbind(performance_metrics_in_sample, tibble(sample = "in")),
   cbind(performance_metrics_oos, tibble(sample = "out"))
)




ggplot(performance_metrics_in_and_oos) +
  geom_line(aes(x = FPR, y = recall, col = sample)) +
  geom_abline(intercept = 0, slope = 1, col = "orange") +
  coord_fixed() + xlim(0, 1) + ylim(0, 1)
```




```r
#out of sample has higher recall compared to in sample but there is a trade off because the insample has higher fpr while the out of sample has lower fpr
```

#TO-DO interpretation

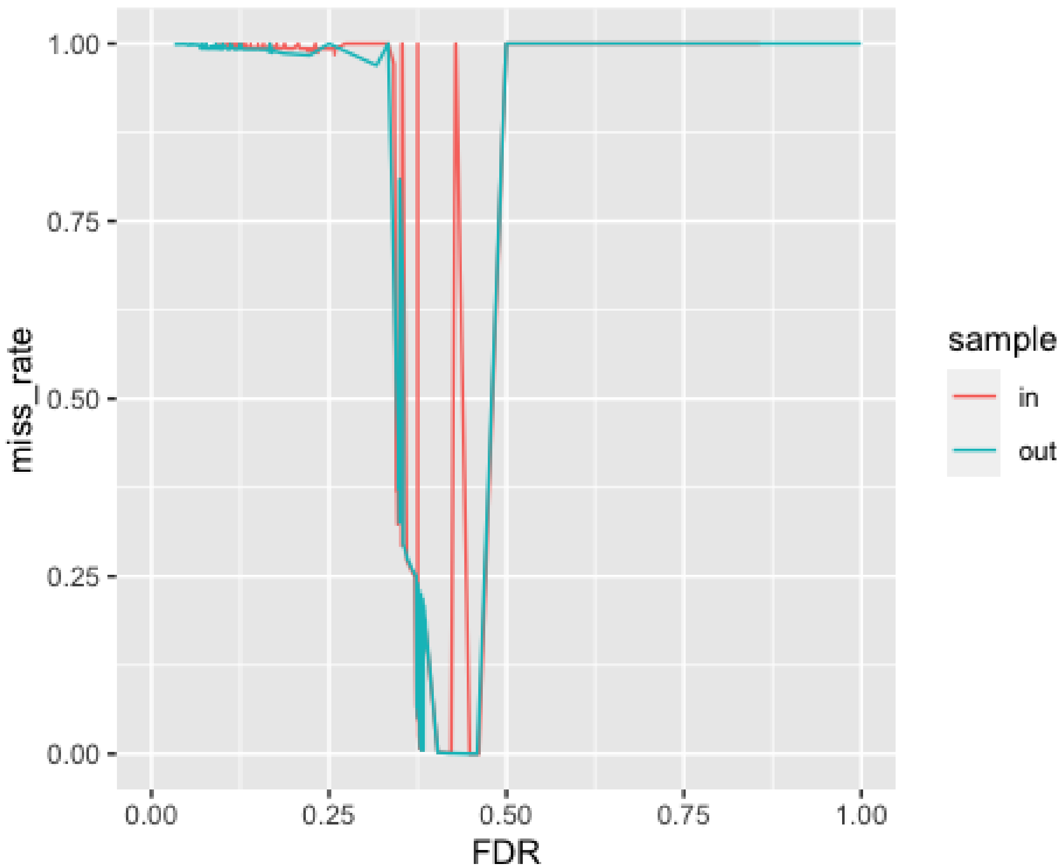
Calculate AUC and interpret.

```
#TO-DO

pacman::p_load(pracma)
-trapz(performance_metrics_in_and_oos$FPR,
performance_metrics_in_and_oos$recall)

## [1] 0.7229933

#the AUC is closer to 1 which means the model is able distinguish/ predict
when a person will pay or not pay their bills. Thus this is a good model.
```

#TO-DO interpretation

Plot a DET curve and interpret.

```
#TO-DO
table(bills_data_test$paid_in_full) / length(bills_data_test$paid_in_full)

##
##         0         1
## 0.4959723 0.5040277

table(bills_data_train$paid_in_full) / length(bills_data_train$paid_in_full)

##
##         0         1
## 0.4980863 0.5019137

#this is funky

ggplot(performance_metrics_in_and_oos) +
  geom_line(aes(x = FDR, y = miss_rate, col = sample)) +
  coord_fixed() + xlim(0, 1) + ylim(0, 1)
```



#TO-DO interpretation

Here we are calculation for the error rates for the binary classification for false positives and flase negatives it's suppose to look weird but it's helpful.