Lab 8

Elizabeth McHugh

11:59PM April 29, 2021

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

```
Sys.setenv(JAVA_HOME = '/usr/lib/jvm/jdk1.8.0_65')

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)
```

```
## YARF can now make use of 7 cores.
```

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".

```
data(storms)

storms2 = storms %>% filter(!is.na(ts_diameter) & !is.na(hu_diameter) & ts_diameter > 0 & hu_diameter >

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.

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

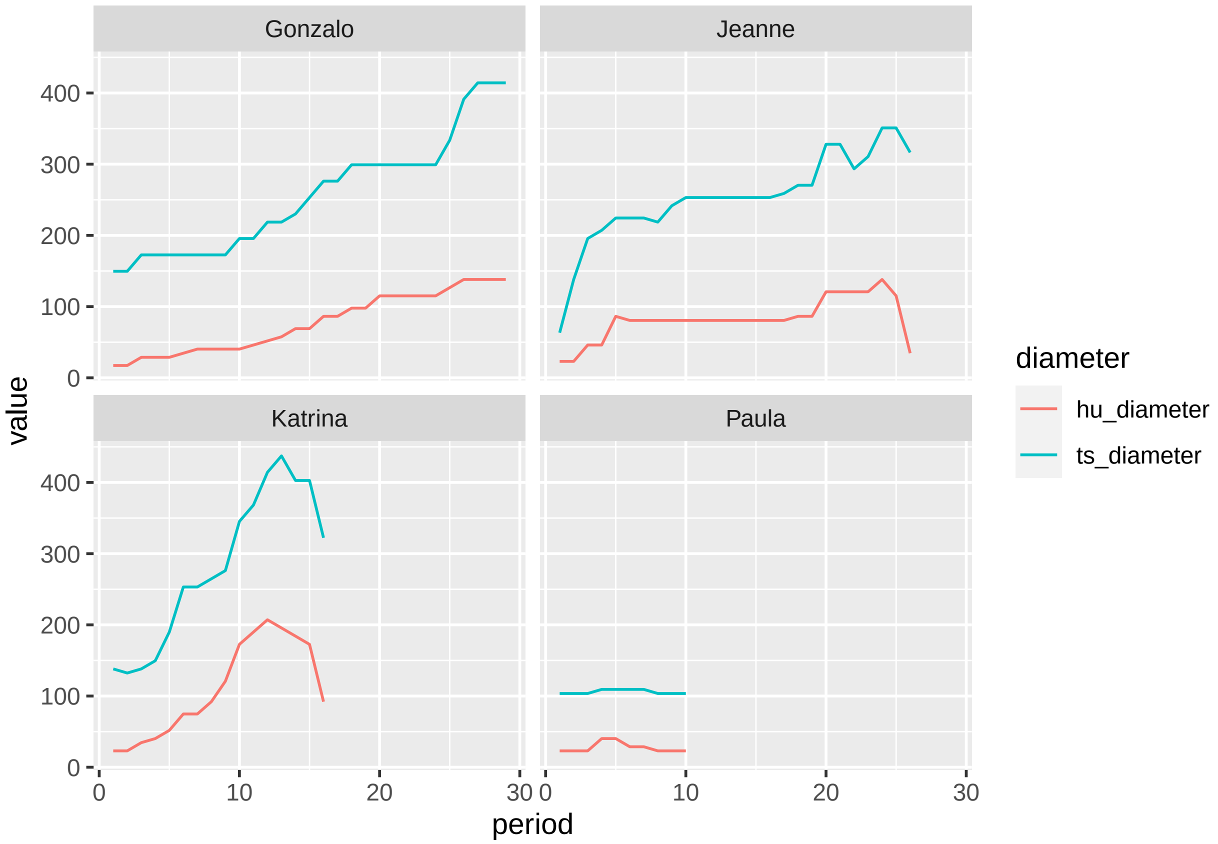
Create a data frame in long format with columns "diameter" for the measurement and "diameter_type" which will be categorical taking on the values "hu" or "ts".

```r
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
```

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.

```r
storms_sample = sample(unique(storms2$name), 4)

ggplot(storms_long %>% filter(name %in% storms_sample))+
  geom_line(aes(x = period, y = value, col = diameter)) +
  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/bills_dataset/bills
payments = fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/pa
discounts = fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/d
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
```

```
bills = as.tibble(bills)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
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"))
head(bills_with_payments)
```

```
## # A tibble: 6 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
## # ... with 2 more variables: paid_amount <dbl>, transaction_date <date>
```

```
bills_with_payments_with_discounts = left_join(bills_with_payments, discounts, by = c("discount_id" = "i
head(bills_with_payments_with_discounts)
```

```
## # A tibble: 6 x 12
##         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
## # ... with 5 more variables: paid_amount <dbl>, transaction_date <date>,
## #   num_days <int>, pct_off <dbl>, days_until_discount <int>
```

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!

```
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>
## 113064 113370      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.

```
pacman::p_load(lubridate)

bills_data = bills_data %>%
  select(-id, -id.y, -num_days, -transaction_date, -pct_off, -days_until_discount, -sum_of_payment_amour
    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, ]

head(bills_data_train)
```

```
## # A tibble: 6 x 5
##   tot_amount discount_id num_days_to_pay bill_num paid_in_full
##        <dbl> <fct>                 <int>    <int>        <dbl>
## 1     99480. 7397895                  45        1            0
## 2     99529. 7397895                  30        1            0
## 3     99477. 7397895                  11        1            0
## 4     99477. 7397895                  30        3            0
## 5     99477. 7397895                  30        1            0
## 6     99477. 7397895                   0        1            0
```

```
head(bills_data)
```

```
## # A tibble: 6 x 5
##   tot_amount discount_id num_days_to_pay bill_num paid_in_full
##        <dbl> <fct>                 <int>    <int>        <dbl>
## 1     99480. 7397895                  45        1            0
## 2     99529. 7397895                  30        1            0
## 3     99477. 7397895                  11        1            0
## 4     99479. 7397895                   0        2            0
## 5     99477. 7397895                  30        3            0
## 6     99477. 7397895                  30        1            0
```

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.

```
##tree_mod_paid_in_full = YARFCART(Xy = bills_data_train, calculate_oob_error = FALSE)
#Doesn't work properly.

X_train = bills_data_train %>%
  select(-paid_in_full)
y_train = bills_data_train$paid_in_full

tree_mod = YARFCART(X_train, y_train, calculate_oob_error = FALSE)
```

```
## YARF initializing with a fixed 1 trees...
## YARF factors created...
## YARF after data preprocessed... 36 total features...
## Beginning YARF regression model construction...done.
```

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

```
get_tree_num_nodes_leaves_max_depths(tree_mod)
```

```
## $num_nodes
## [1] 53337
##
## $num_leaves
```

```
## [1] 26669
##
## $max_depth
## [1] 39
```

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

```
illustrate_trees(tree_mod, max_depth = 6, open_file = TRUE)
```

Predict on the test set and compute a confusion matrix.

```
X_test = bills_data_test %>%
  select(-paid_in_full)
y_test = bills_data_test$paid_in_full

y_hat = predict(tree_mod, X_test)

y_hats_test = factor(ifelse(y_hat >= 0.5, "1", "0")) #factorize predicted full payment from y_hat

bills_data_conf = table(y_test, y_hats_test)     #create confusion matrix for factorized predictions
```

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

```
n = sum(bills_data_conf)
fp = bills_data_conf[1, 2]
fn = bills_data_conf[2, 1]
tp = bills_data_conf[2, 2]
tn = bills_data_conf[1, 1]
num_pred_pos = sum(bills_data_conf[, 2])
num_pred_neg = sum(bills_data_conf[, 1])
num_pos = sum(bills_data_conf[2, ])
num_neg = sum(bills_data_conf[1, ])

ME = (fp + fn) / n
cat("misclassification error", round(ME * 100, 2), "%\n")
```

```
## misclassification error 23.79 %
```

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

```
## precision 74.65 %
```

```
recall = tp / num_pos
cat("recall", round(recall * 100, 2), "%\n")
```

```
## recall 79.28 %
```

```
F_1 = 2 / ((1 / recall) + (1 / precision))
cat("F1", round(F_1 * 100, 2), "%\n")
```

```
## F1 76.89 %
```

```
FDR = 1 - precision
cat("false discovery rate", round(FDR * 100, 2), "%\n")
```

```
## false discovery rate 25.35 %
```

```
FOR = fn / num_pred_neg
cat("false omission rate", round(FOR * 100, 2), "%\n")
```

```
## false omission rate 22.02 %
```

Is this a good model? (yes/no and explain).

This doesn't seem to be a very good model upon inspection, as precision and recall leave a lot of room for improvement, in general. However, considering that we didn't expect there to be much of a signal within the given data, this "not greatness" might be more a reflection of the data which we were given than the model itself.

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

```
C_FP =  mean(bills_with_payments$tot_amount) + 1200     #loss of average bill amount plus $1,200 collec
C_FN =  mean(bills_with_payments$tot_amount)            #loss of average bill (given, there is a potent

C = (C_FP * fp) + (C_FN * fn)

C
```

```
## [1] 1365921812
```

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

```
logistic_mod = glm(paid_in_full ~ ., bills_data_train, family = "binomial")

p_hat_train = predict(logistic_mod, bills_data_train, type = "response")
p_hats_test = predict(logistic_mod, bills_data_test, 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",
```

```r
    "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
}

performance_metrics_in_sample = as.data.table(compute_metrics_prob_classifier(p_hat_train, y_train))
performance_metrics_oos = as.data.table(compute_metrics_prob_classifier(p_hats_test, y_test))

performance_metrics_in_sample
```

```
##       p_th    TN    FP    FN    TP miscl_err precision       recall       FDR
```

```
##   1: 0.001 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   2: 0.002 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   3: 0.003 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   4: 0.004 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   5: 0.005 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##  ---
## 995: 0.995 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 996: 0.996 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 997: 0.997 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 998: 0.998 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 999: 0.999 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
##            FPR          FOR    miss_rate
##   1: 0.8669062 8.967001e-05 1.175309e-05
##   2: 0.8669062 8.967001e-05 1.175309e-05
##   3: 0.8669062 8.967001e-05 1.175309e-05
##   4: 0.8669062 8.967001e-05 1.175309e-05
##   5: 0.8669062 8.967001e-05 1.175309e-05
##  ---
## 995: 0.0000000 5.038492e-01 9.999882e-01
## 996: 0.0000000 5.038492e-01 9.999882e-01
## 997: 0.0000000 5.038492e-01 9.999882e-01
## 998: 0.0000000 5.038492e-01 9.999882e-01
## 999: 0.0000000 5.038492e-01 9.999882e-01
```

```
performance_metrics_oos
```

```
##       p_th    TN    FP    FN    TP miscl_err precision    recall       FDR
##   1: 0.001  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   2: 0.002  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   3: 0.003  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   4: 0.004  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   5: 0.005  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##  ---
## 995: 0.995 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 996: 0.996 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 997: 0.997 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 998: 0.998 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 999: 0.999 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
##            FPR          FOR    miss_rate
##   1: 0.8667095 0.0002678811 3.539197e-05
##   2: 0.8667095 0.0002678811 3.539197e-05
##   3: 0.8667095 0.0002678811 3.539197e-05
##   4: 0.8667095 0.0002678811 3.539197e-05
##   5: 0.8667095 0.0002678811 3.539197e-05
##  ---
## 995: 0.0000000 0.5022753937 1.000000e+00
## 996: 0.0000000 0.5022753937 1.000000e+00
## 997: 0.0000000 0.5022753937 1.000000e+00
## 998: 0.0000000 0.5022753937 1.000000e+00
## 999: 0.0000000 0.5022753937 1.000000e+00
```

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

```r
performance_metrics_in_sample %>%
    mutate(total_cost = C_FP * FP + C_FN * FN)
```

```
##       p_th    TN    FP    FN    TP miscl_err precision       recall       FDR
##   1: 0.001 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   2: 0.002 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   3: 0.003 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   4: 0.004 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##   5: 0.005 11151 72632     1 85083 0.4276907 0.5394731 9.999882e-01 0.4605269
##  ---
## 995: 0.995 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 996: 0.996 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 997: 0.997 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 998: 0.998 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
## 999: 0.999 83783     0 85083     1 0.5010010 1.0000000 1.175309e-05 0.0000000
##            FPR          FOR    miss_rate total_cost
##   1: 0.8669062 8.967001e-05 1.175309e-05 7404876855
##   2: 0.8669062 8.967001e-05 1.175309e-05 7404876855
##   3: 0.8669062 8.967001e-05 1.175309e-05 7404876855
##   4: 0.8669062 8.967001e-05 1.175309e-05 7404876855
##   5: 0.8669062 8.967001e-05 1.175309e-05 7404876855
##  ---
## 995: 0.0000000 5.038492e-01 9.999882e-01 8572046306
## 996: 0.0000000 5.038492e-01 9.999882e-01 8572046306
## 997: 0.0000000 5.038492e-01 9.999882e-01 8572046306
## 998: 0.0000000 5.038492e-01 9.999882e-01 8572046306
## 999: 0.0000000 5.038492e-01 9.999882e-01 8572046306
```

```r
performance_metrics_oos %>%
    mutate(total_cost = C_FP * FP + C_FN * FN)
```

```
##       p_th    TN    FP    FN    TP miscl_err precision    recall       FDR
##   1: 0.001  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   2: 0.002  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   3: 0.003  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   4: 0.004  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##   5: 0.005  3732 24267     1 28254 0.4287027 0.5379562 0.9999646 0.4620438
##  ---
## 995: 0.995 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 996: 0.996 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 997: 0.997 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 998: 0.998 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
## 999: 0.999 27999     0 28255     0 0.4991344       NaN 0.0000000       NaN
##            FPR          FOR    miss_rate total_cost
##   1: 0.8667095 0.0002678811 3.539197e-05 2474102591
##   2: 0.8667095 0.0002678811 3.539197e-05 2474102591
##   3: 0.8667095 0.0002678811 3.539197e-05 2474102591
##   4: 0.8667095 0.0002678811 3.539197e-05 2474102591
##   5: 0.8667095 0.0002678811 3.539197e-05 2474102591
##  ---
## 995: 0.0000000 0.5022753937 1.000000e+00 2846669351
## 996: 0.0000000 0.5022753937 1.000000e+00 2846669351
## 997: 0.0000000 0.5022753937 1.000000e+00 2846669351
```

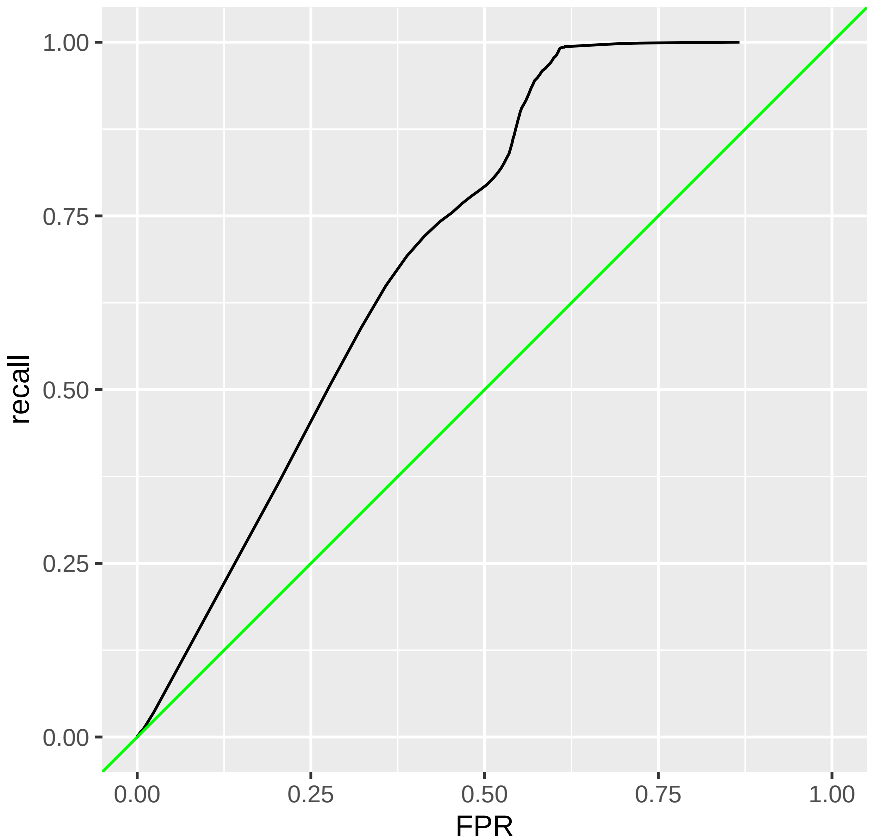
```
## 998: 0.0000000 0.5022753937 1.000000e+00 2846669351
## 999: 0.0000000 0.5022753937 1.000000e+00 2846669351
```

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

```
#win_thresh_prob =
win_thresh_cost = which.min(performance_metrics_oos$total_cost)
```

Plot an ROC curve and interpret.

```
pacman::p_load(ggplot2)
ggplot(performance_metrics_in_sample) +
  geom_line(aes(x = FPR, y = recall)) +
  geom_abline(intercept = 0, slope = 1, col = "green") +
  coord_fixed() + xlim(0, 1) + ylim(0, 1)
```



#TO-DO interpretation

Calculate AUC and interpret.

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

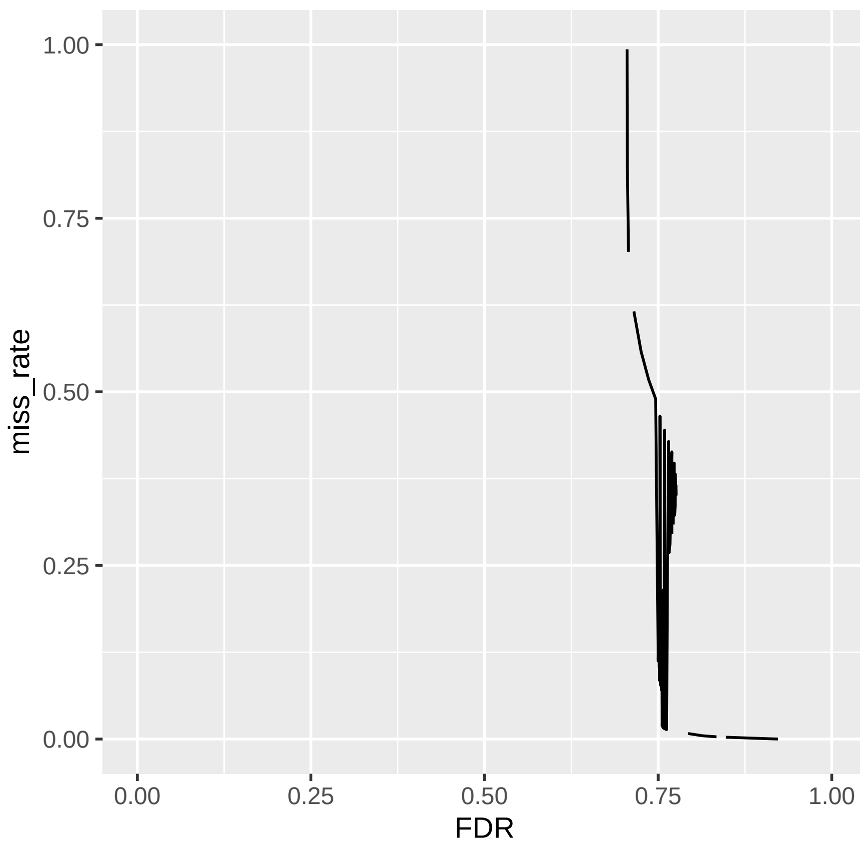
```
## [1] 0.5738784
```

Interpretation: I still couldn't get things to work out from previous chunks.

Plot a DET curve and interpret.

```
performance_metrics_in_and_oos = performance_metrics_in_sample + performance_metrics_oos

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

```
## Warning: Removed 365 row(s) containing missing values (geom_path).
```



Interpretation: I'm not at all sure as to what this is, though the plot seems... interesting?