---

title: "Lab 8"

author: "Feiyi Ma"

output: pdf_document

date: "11:59PM April 29, 2021"

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

```{r}
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.

```{r}
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".

```{r}
pacman::p_load(dplyr)

data(storms)

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

storms2
```

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())

storms2
```

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
```

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:

````
```{r}

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
.csv.bz2")

payments =
fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/pay
ments.csv.bz2")

discounts =
fread("https://github.com/kapelner/QC_MATH_342W_Spring_2021/raw/master/labs/bills_dataset/disc
ounts.csv.bz2")

setnames(bills, "amount", "tot_amount")

setnames(payments, "amount", "paid_amount")

head(bills)

head(payments)
````
````

head(discounts)

bill = as_tibble(bills)

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.

```{r}

bills_with_payments = left_join(bills, payments, by = c("id" = "bill_id"))

bills_with_payments

#id payment = id.y

bills_with_payments_with_discounts = left_join(bills_with_payments, discounts, by = c("discount_id" = "id"))

bills_with_payments_with_discounts

```

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!

```{r}

#if their payment amount is greater or equal to invoice amount, sum of payments

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")
```

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.

```{r}
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.

```{r}
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.

```{r}
x_train = factor(bills_data_train$y)

x_train = bills_data_train

x_train$y = NULL


y_train = factor(bills_data_test$y)

y_train = bills_data_test

y_train$y = NULL


paid_in_full_tree = YARFCART(X= x_train, y= y_train, calculate_oob_error = FALSE)
```

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

```{r}
get_tree_num_nodes_leaces_max_depths(paid_in_full_tree)
```

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

```{r}
illustrate_trees(paid_in_full_tree, max_depth = 4, open_file = TRUE)
y_hat = predict(paid_in_full_tree, x_predict)
ggplot(data.frame(x = x_train, y = y_train), aes(x_train, y_train)) +
  geom_point(lwd = 0.6) +
  geom_point(aes(x, y), data.frame(x = x_predict, y = yhat), col = "blue")
```

Predict on the test set and compute a confusion matrix.

```{r}
y_hats_test = factor(ifelse(p_hats_test >= 0.5, ">1", "<=1"))
mean(y_hats_train != y_hat)
oos_conf_table = table(y_hat, y_hats_train)
oos_conf_table
```

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

```{r}
n = sum(oos_conf_table)
fp = oos_conf_table[1, 2]
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, ])

precision = tp / num_pred_pos

cat("precision", round(precision * 100, 2), "%\n")

recall = tp / num_pos

cat("recall", round(recall * 100, 2), "%\n")

false_discovery_rate = 1 - precision

cat("false_discovery_rate", round(false_discovery_rate * 100, 2), "%\n")

false_omission_rate = fn / num_pred_neg

cat("false_omission_rate", round(false_omission_rate * 100, 2), "%\n")
```

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

#TO-DO

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

```{r}
y_hats_test = factor(ifelse(p_hats_test >= 0.9, ">1", "<=1"))

mean(y_hats_train != y_hat)

oos_conf_table = table(y_hat, y_hats_train)

oos_conf_table
```

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

```{r}
n = sum(oos_conf_table)
fp = oos_conf_table[1, 2]
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, ])
precision = tp / num_pred_pos
cat("precision", round(precision * 100, 2), "%\n")
recall = tp / num_pos
cat("recall", round(recall * 100, 2), "%\n")
false_discovery_rate = 1 - precision
cat("false_discovery_rate", round(false_discovery_rate * 100, 2), "%\n")
false_omission_rate = fn / num_pred_neg
cat("false_omission_rate", round(false_omission_rate * 100, 2), "%\n")
```

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.

```{r}
#TO-DO
```

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

```{r}
#TO-DO
```

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

```{r}
#TO-DO
```

Plot an ROC curve and interpret.

```{r}
#TO-DO
```

#TO-DO interpretation

Calculate AUC and interpret.

```{r}
#TO-DO
```

#TO-DO interpretation

Plot a DET curve and interpret.

```{r}
#TO-DO
```

#TO-DO interpretation