# DATA7440: Final Project

Aleksei Luchinsky, Mostafa MR

04/25/2023

# Instruction

Buskirk, DATA 7440

Final Assignment, Spring 2023

- This assignment can be completed in groups of 2 to 3 students and you are encouraged to submit your assignment in groups.
- The assignment is worth 100 points and accounts for 40% of your final grade (taking the place of individual quizzes and a group project).
- Your group should submit only one final assignment and you should plan to use R for your submissions via an Rmarkdown file that generates a pdf document.
- Be sure to include your code chunks as part of the solution.
- The assignment is due via canvas submission Tuesday April 25, 2023 by 2pm.

# Problem 1

## Libraries

```
rm(list = ls())

suppressPackageStartupMessages(library(glmnet))
suppressPackageStartupMessages(library(RWeka))
suppressPackageStartupMessages(library(pre))
suppressPackageStartupMessages(library(randomForest))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(magrittr))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(iml))

# I need to attach these libraries to suppress messages
 from Interaction$new
suppressPackageStartupMessages(library(fs))
suppressPackageStartupMessages(library(xml2))
suppressPackageStartupMessages(library(jsonlite))
```

```
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(dbplyr))
suppressPackageStartupMessages(library(readr))
suppressPackageStartupMessages(library(httr))
suppressPackageStartupMessages(library(gargle))
suppressPackageStartupMessages(library(rvest))
suppressPackageStartupMessages(library(googlesheets4))
suppressPackageStartupMessages(library(hms))
```

# Datset

```
load("./data/external/apipop1.RData")
str(apipop_train1)
```

```
'data.frame':   4474 obs. of  16 variables:
 $ awards  : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1
 2 2 2 2 ...
 $ stype   : Factor w/ 3 levels "E","H","M": 1 1 2 2 2
2 1 1 1 1 ...
 $ meals   : int  88 70 31 4 37 9 17 21 42 65 ...
 $ ell     : int  13 32 18 3 29 2 3 12 16 48 ...
 $ mobility: int  13 37 13 7 9 6 14 12 16 20 ...
 $ pct.resp: int  83 90 75 90 95 89 89 99 58 57 ...
 $ not.hsg : int  27 21 39 1 41 1 7 10 14 10 ...
 $ hsg     : int  36 37 20 8 19 13 36 35 16 26 ...
 $ some.col: int  30 33 19 19 19 29 25 18 34 27 ...
 $ col.grad: int  4 9 20 44 17 39 27 26 22 25 ...
 $ grad.sch: int  2 1 2 29 4 17 5 11 14 12 ...
 $ avg.ed  : num  2.18 2.32 2.25 3.91 2.25 ...
 $ full    : int  79 79 89 100 91 94 89 100 95 63 ...
 $ emer    : int  14 21 11 4 9 6 11 0 5 37 ...
 $ enroll  : int  275 325 990 2306 2132 1215 262 377 63
9 509 ...
 $ api.stu : int  216 278 809 2115 1205 1144 226 337 57
0 435 ...
```

# Functions

```
fill.table <- function(model_name, pred_func, y_train=a
pipop_train1$awards, y_test = y.test,
                       trainX = apipop_train1,
                       testX = apipop_test1) {
  train.predictions <- pred_func(trainX)
  cm.train = confusionMatrix(train.predictions, y_train
)
  test.predictions <- pred_func(testX)
  cm.test = confusionMatrix(test.predictions, y_test)
```

```
  return(data.frame(
    model = model_name,
    test.accuracy = cm.test$overall[1],
    test.kappa = cm.test$overall[2],
    train.accuracy = cm.train$overall[1],
    train.kappa = cm.train$overall[2]
  ))
}
```

# Part 1: Generating the Models

Preparing train data

```
y.train <- apipop_train1$awards
y.test <- apipop_test1$awards
```

## a) LASSO

> construct a LASSO model that predicts whether or not a school qualified for it's award program. Use 5-fold cross validation on the apipop_train1 dataset to determine the penalty parameter and specify the seed 7440 prior to running the model.

```
set.seed(7440)
cv.lasso.model <<- cv.glmnet(
  x = data.matrix(apipop_train1[,-1]),
  y = apipop_train1[,1],
  alpha = 1, standardize = TRUE, nfolds = 5, family = "
binomial")
```

The best penalty parameter is

```
cv.lasso.model$lambda.min
```

```
[1] 0.0005778871
```

```
lasso.best.lambda <- cv.lasso.model$lambda.min
```

Here is the modal trained with this value of parameter

```
lasso.best.model <- glmnet(
  x = data.matrix(apipop_train1[,-1]),
  y = apipop_train1[,1],
  alpha = 1, lambda = lasso.best.lambda, family = "bino
mial")
```

## b) oneR

> Now apply a one rule algorithm to determine the feature that most explains the award outcomes.

```
oneR.model <- OneR::OneR(awards ~ ., data = apipop_trai
n1)
oneR.model
```

```
Call:
OneR.formula(formula = awards ~ ., data = apipop_train1
)

Rules:
If stype = E then awards = Yes
If stype = H then awards = No
If stype = M then awards = Yes

Accuracy:
3143 of 4474 instances classified correctly (70.25%)
```

As you can see, the feature, that is most useful to explain the outcome is **stype**

## c) Sequential Covering

> Now apply a sequential covering algorithm to expand the rule set derived in B.

```
seq.model <- JRip(awards ~ ., data = apipop_test1)
seq.model
```

```
JRIP rules:
===========
```

```
(enroll >= 687) and (stype = H) => awards=No (154.0/48.
0)
(grad.sch <= 9) and (enroll >= 1082) => awards=No (42.0
/10.0)
(enroll >= 600) and (not.hsg >= 24) and (stype = M) =>
awards=No (48.0/19.0)
(stype = H) => awards=No (35.0/14.0)
 => awards=Yes (1220.0/314.0)

Number of Rules : 5
```

As you can see, the **stype** variable is part of almost all of the rules, so it is important

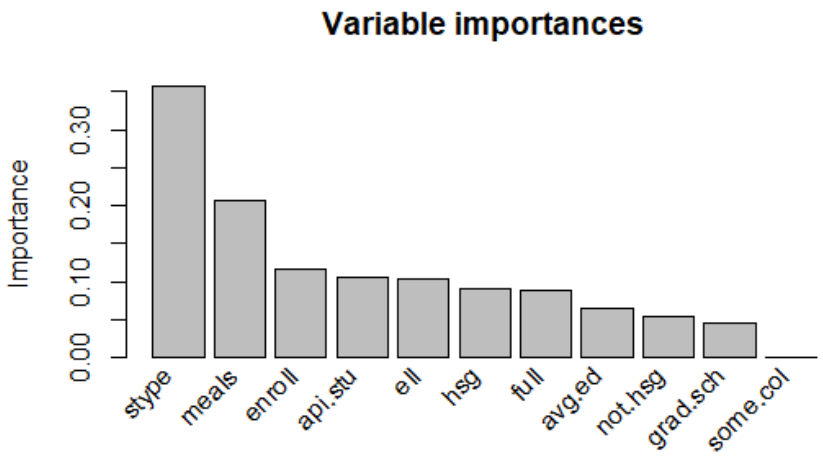
## d) Rule-Fit

> Now apply a rule-fit algorithm to determine features and possible interaction effects that are important for explaining award qualification. For this model set the seed to be 2021 and also use 5 folds.

```
set.seed(2021)
rule.fit.model <- pre(awards ~ .,
                      data = apipop_train1,
                      family = "binomial", nfolds = 5
                      )
```

This model also tells that **stype** is the most important variable:

```
rule.fit.imps <- pre::importance(rule.fit.model)
```




```
rule.fit.imps$baseimps %>% head(10)
```

| | rule<br><chr> |
|---|---|
| 1 | rule285 |
| 2 | rule592 |
| 3 | rule685 |
| 4 | rule703 |
| 5 | rule788 |
| 6 | rule452 |
| 7 | rule573 |
| 8 | rule676 |
| 9 | rule10 |
| 10 | rule692 |

1-10 of 10 rows | 1-2 of 5 columns

Most important rules here are

1. rule285: stype = “H”, grad.sch <= 38, ell>0, coeff = -0.389. This rule votes for negative answer and it agrees with oneR rule, default rule of the Sequential Covering model
2. rule592: enroll <= 864 & api.stu > 271, coeff = 0.18 Coefficient is positive, to this rule votes for “Yes”. It agrees, with the 1st rule of the Sequential Covering model: rule592 tends to “Yes” for small enroll, while Seq.Cov wants “No” for large enroll

3. rule685: stype = c("E") & meals <= 96, coeff = 0.17 Coefficient is positive, to this rule votes for "Yes". This agrees with the 1st rule of the oneR model.

# Part 2: Comparing The Models

> Using the respective models create a table that provides overall accuracy, sensitivity and specificity for predicting if a school qualified for its awards program using the apipop_test1 as the test set for each of the models you derived in parts a-d of Part 1.

```
y.test <- apipop_test1$awards
cm.table <- data.frame()
```

I will use this function to collect statistics about model's accuracy and kappa on training and test sets

```
fill.table <- function(model_name, pred_func, y_train=a
pipop_train1$awards, y_test = y.test,
                       trainX = apipop_train1,
                       testX = apipop_test1) {
  train.predictions <- pred_func(trainX)
  cm.train = confusionMatrix(train.predictions, y_train
)
  test.predictions <- pred_func(testX)
  cm.test = confusionMatrix(test.predictions, y_test)
  return(data.frame(
    model = model_name,
    test.accuracy = cm.test$overall[1],
    test.kappa = cm.test$overall[2],
    train.accuracy = cm.train$overall[1],
    train.kappa = cm.train$overall[2]
  ))
}
```

## a) LASSO model

```
cm.table <- rbind(cm.table,
  fill.table("LASSO",
             pred_func = function(X) {
               probs = predict(lasso.best.model, newx = d
```

```
ata.matrix(X), type = "response")
            as.factor(as.vector(ifelse( probs> 0.5, "Y
es", "No")))
}, trainX = apipop_train1[,-1], testX = apipop_test1[,-
1])
)
```

## b) OneR

```
cm.table <- rbind(cm.table,
  fill.table("OneR",
          pred_func = function(X) {
            as.factor(predict(oneR.model, X))
})
)
```

## c) Sequential model

```
cm.table <- rbind(cm.table,
  fill.table("SeqCover",
          pred_func = function(X) {
            as.factor(predict(seq.model, X))
})
)
```

## d) Rule Fit

```
cm.table <- rbind(cm.table,
  fill.table("RuleFit",
          pred_func = function(X) {
            probs = predict(rule.fit.model, newdata =
X, type = "response")
            as.factor(as.vector(ifelse( probs> 0.5, "Y
es", "No")))
})
)
```

## e) Results

```
rownames(cm.table) <- 1:nrow(cm.table)
cm.table %>% mutate_if(is.numeric, round, digits = 3) %
>% arrange(test.accuracy)
```

| model <chr> | test.accuracy <dbl> | test.kappa <dbl> | train.accuracy <dbl> |
|---|---|---|---|
| LASSO | 0.705 | 0.217 | 0.700 |
| OneR | 0.708 | 0.226 | 0.703 |
| RuleFit | 0.712 | 0.256 | 0.719 |
| SeqCover | 0.730 | 0.318 | 0.701 |

4 rows | 1-4 of 5 columns

As you can see, sequential covering is the best model both in terms of test accuracy and test kappa.

# Part 3: Important Variables

> For every model for which it is appropriate, derive the variable importance measures (feature specific importances, not rule specific importances) and create another table that lists the top 4 most important features for predicting whether a school qualified for it's award program.

## a) LASSO

```
lasso.imps <- lasso.best.model %>% coef %>% as.matrix %>% as.data.frame %>%
  slice(-1) %>% abs %>% arrange(desc(s0))
head(lasso.imps)
```

| | s0 <dbl> |
|---|---|
| stype | 0.375712635 |
| grad.sch | 0.015816183 |
| full | 0.015248647 |
| emer | 0.013521232 |
| meals | 0.007729355 |
| ell | 0.005204630 |

6 rows

**stype** is the most important variable

# b) One R

```
oneR.model
```

```
Call:
OneR.formula(formula = awards ~ ., data = apipop_train1
)

Rules:
If stype = E then awards = Yes
If stype = H then awards = No
If stype = M then awards = Yes

Accuracy:
3143 of 4474 instances classified correctly (70.25%)
```

As you can see, only **stype** variable is important in oneR model

# c) Sequential Covering

Here is the list of the rules

```
seq.model
```

```
JRIP rules:
===========

(enroll >= 687) and (stype = H) => awards=No (154.0/48.
0)
(grad.sch <= 9) and (enroll >= 1082) => awards=No (42.0
/10.0)
(enroll >= 600) and (not.hsg >= 24) and (stype = M) =>
awards=No (48.0/19.0)
(stype = H) => awards=No (35.0/14.0)
 => awards=Yes (1220.0/314.0)

Number of Rules : 5
```

The following models enter these rules and are important for making the decisions:

- stype
- enroll
- grad.sch
- not.hsg

I can convert it to string and try to extract the rules using regexp

## d) Rule Fit

```
rule.fit.imps <- pre::importance(rule.fit.model, plot =
 FALSE)
rfi4 <- rule.fit.imps$varimps[1:4, ]
rfi4
```

| | varname<br><chr> | imp<br><dbl> |
|---|---|---|
| 1 | stype | 0.3570190 |
| 2 | meals | 0.2064637 |
| 3 | enroll | 0.1170933 |
| 4 | api.stu | 0.1064105 |

4 rows

```
NA
```

## e) Results

To summarize, all models tell that **stype** is the most important variable. Such variables as **enroll**, **grad.sch**, **meals** are also often present in the list of important predictors.

Here is the table of the most important variables:

```
cbind(
  data.frame(Lasso = rownames(lasso.imps)[1:4], lasso.i
mps = lasso.imps[1:4,1]),
  data.frame(oneR = c("stype", NA, NA, NA), oneR.imps =
 rep(NA, 4)),
  data.frame(SeqCover = c("stype", "enroll", "grad.sch"
, "not.hsg"), seq.imps = rep(NA, 4)),
  data.frame(RuleFit = rfi4$varname, rule.imps = rfi4$i
mp)
) %>% select(Lasso, oneR, SeqCover, RuleFit)
```

| Lasso<br><chr> | oneR<br><chr> | SeqCover<br><chr> | RuleFit<br><chr> |
|---|---|---|---|
| stype | stype | stype | stype |
| grad.sch | *NA* | enroll | meals |

| | | | |
|---|---|---|---|
| full | *NA* | grad.sch | enroll |
| emer | *NA* | not.hsg | api.stu |

4 rows

```
NA
```

# Part 4: Random Forest

> Now compute a random forest model to predict whether or not a school qualifies for its awards program (using the training data: apipop_train1) with 750 trees and default value of the mtry parameter. Prior to running the model, set the seed to be 429. Using your model:

```
set.seed(429)
rf.model <- randomForest(awards ~ ., data = apipop_trai
n1, ntree = 750)
```

Train accuracy and kappa of this model is perfect. By accuracy on the test subset it is not as good:

```
 cm.table <- rbind(cm.table,
  fill.table("RF",
             pred_func = function(X) {
               predict(rf.model, newdata = X)
})
)
rownames(cm.table) <- 1:nrow(cm.table)
cm.table <- unique(cm.table)
cm.table %>% mutate_if(is.numeric, round, digits = 3) %
>% arrange(test.accuracy)
```

| model <chr> | test.accuracy <dbl> | test.kappa <dbl> | train.accuracy <dbl> |
|---|---|---|---|
| LASSO | 0.705 | 0.217 | 0.700 |
| RF | 0.706 | 0.261 | 1.000 |
| OneR | 0.708 | 0.226 | 0.703 |

| | | | |
|---|---|---|---|
| RuleFit | 0.712 | 0.256 | 0.719 |
| SeqCover | 0.730 | 0.318 | 0.701 |

5 rows | 1-4 of 5 columns

# a) Overall Interaction

> create an overall interaction plot that displays the overall H statistic for each of the features.

Here is the code to calculate one-feature H-statistics

```
set.seed(429)
suppressPackageStartupMessages(
  predictor1 <-  Predictor$new(rf.model, dat = apipop_test1, y = apipop_test1$awards)
)
suppressPackageStartupMessages(
  rf.interaction1 <- Interaction$new(predictor1)
)
```

```
rf.interaction1$results %>% head(8)
```

| .feature <chr> | .class <fctr> | .interaction <dbl> |
|---|---|---|
| awards | No | 0.0007475464 |
| awards | Yes | 0.0007475464 |
| stype | No | 0.4971598252 |
| stype | Yes | 0.4971598252 |
| meals | No | 0.3046433023 |
| meals | Yes | 0.3046433023 |
| ell | No | 0.5526520105 |
| ell | Yes | 0.5526520105 |

8 rows

As you can see, the number of h-statistics values is doubled (two values for each feature, corresponding to award = “Yes” and “No”). It turns out, however, that these values are exactly the same

Hide

```
as.data.frame(rf.interaction1$results) %>%
  tidyr::pivot_wider(names_from = .class, values_from =
 .interaction) %>%
  mutate(ratio = Yes/No) %>%
  head(10)
```

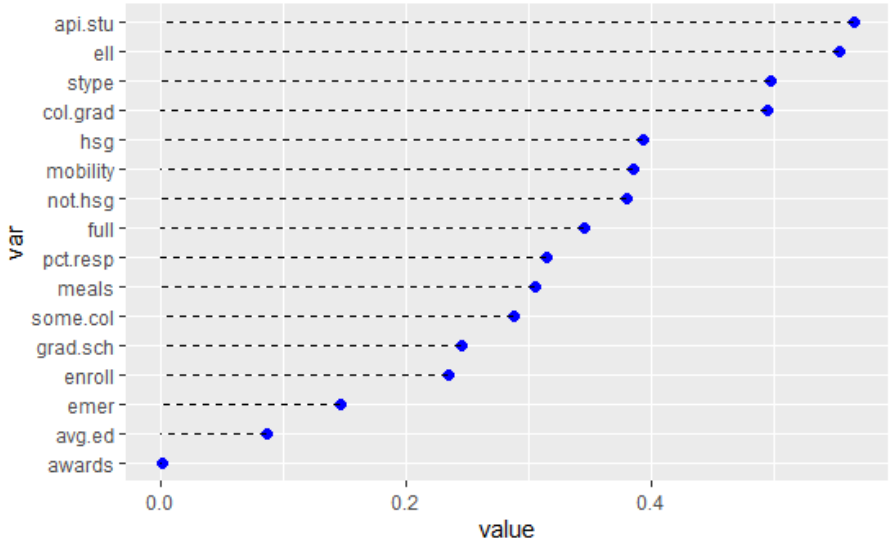
| .feature <chr> | No <dbl> | Yes <dbl> | ratio <dbl> |
|---|---|---|---|
| awards | 0.0007475464 | 0.0007475464 | 1 |
| stype | 0.4971598252 | 0.4971598252 | 1 |
| meals | 0.3046433023 | 0.3046433023 | 1 |
| ell | 0.5526520105 | 0.5526520105 | 1 |
| mobility | 0.3851307362 | 0.3851307362 | 1 |
| pct.resp | 0.3140788948 | 0.3140788948 | 1 |
| not.hsg | 0.3794202484 | 0.3794202484 | 1 |
| hsg | 0.3933387624 | 0.3933387624 | 1 |
| some.col | 0.2871200287 | 0.2871200287 | 1 |
| col.grad | 0.4939960623 | 0.4939960623 | 1 |

1-10 of 10 rows

For this reason I will be interested only in .class = “Yes” part

Here is the plot

```
h.res <- data.frame(rf.interaction1$results) %>%
  filter(.class=="Yes") %>%
  transmute(var=.feature, value=.interaction) %>%
  arrange(value) %>%
  mutate(var = factor(var, levels = var))
ggplot(h.res, aes(y=var, yend=var, xend = 0, x=value))
+ geom_point(color = "blue", cex = 2) +
  geom_segment(linetype = 2)
```




As you can see, **stype** has the largest H-score. This agrees with our previous results

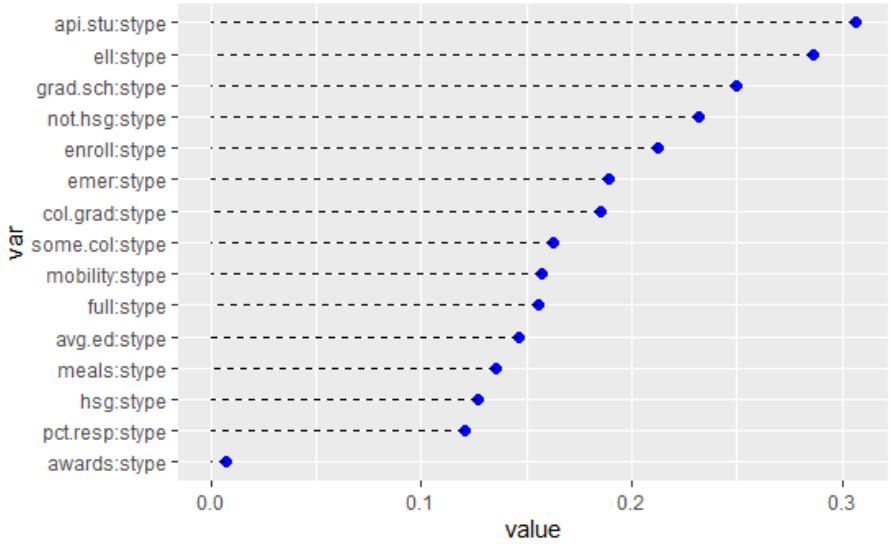
# b) STYPE interactions

> Using the feature with the highest overall H statistic, create an additional plot that shows the H statistics related to this specific feature

```
set.seed(456)
interactSTYPE <- Interaction$new(predictor1, feature =
"stype")
```

Again, we have to identical sets of the statistics, here is ordered plot of one of them:

```
h2.res <- data.frame(interactSTYPE$results) %>%
  filter(.class=="Yes") %>%
  transmute(var=.feature, value=.interaction) %>%
  arrange(value) %>%
  mutate(var = factor(var, levels = var))
ggplot(h2.res, aes(y=var, yend=var, xend = 0, x=value))
 + geom_point(color = "blue", cex = 2) +
  geom_segment(linetype = 2)
```




## c) Comparison with Rule-Fit

> Is the information in the second plot consistent with the some of the rules the Rule Fit model is reporting? Why or why not?

As you can see, the highest h-statistic value here is for api.stu : stype interaction. For Rule Fit model there is only one rule, containing both of these variables, its importance is not very large

```
rule.fit.imps$baseimps %>% filter(grepl("stype", description) & grepl("api.stu", description))
```

| rule | description |
|---|---|
| <chr> | <chr> |
| rule617 | stype %in% c("E") & meals <= 78 & api.stu > 224 |

1 row | 1-2 of 5 columns

Second most important interaction according to H statistics is ell:stype. We have this pair in Rule Fit model, but it does not seem to be important

```
rule.fit.imps$baseimps %>% filter(grepl("stype", description) & grepl("ell", description))
```

| rule | description |
|---|---|
| <chr> | <chr> |

| | |
|---|---|
| rule285 | stype %in% c("H") & grad.sch <= 38 & ell > 0 |
| rule567 | stype %in% c("E") & hsg <= 20 & ell <= 24 |
| rule722 | stype %in% c("H", "M") & ell > 12 |

3 rows | 1-2 of 5 columns

## d) H-Statistics

> What is the H-statistic value corresponding to the variables included in the top- most important rule in the rule fit model that involves only two predictors? (Be sure to indicate what the top-most important rule with two features is from your rule fit model, along with it's importance) as well as the plot of that rule from the RuleFit model.

```
rule.fit.imps$baseimps %>% arrange(desc(imp)) %>% head(10)
```

| | rule <chr> |
|---|---|
| 1 | rule285 |
| 2 | rule592 |
| 3 | rule685 |
| 4 | rule703 |
| 5 | rule788 |
| 6 | rule452 |
| 7 | rule573 |
| 8 | rule676 |
| 9 | rule10 |
| 10 | rule692 |

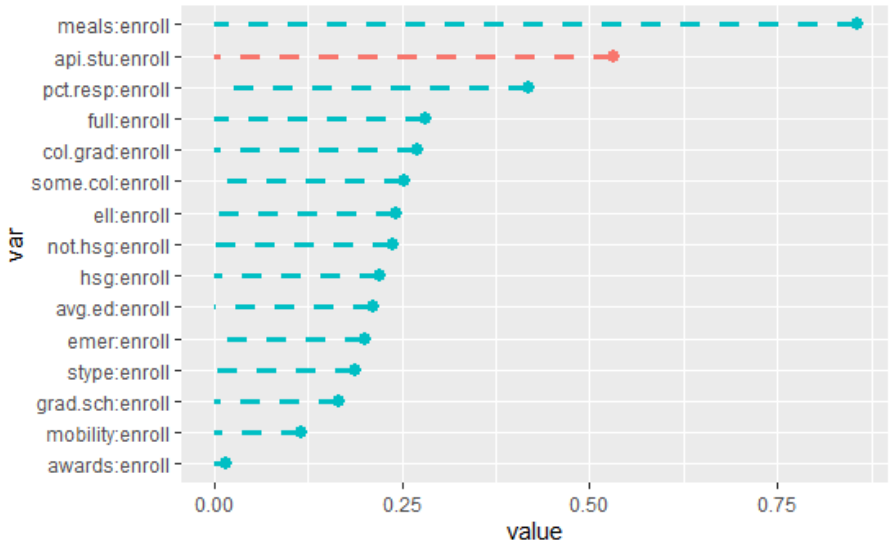
1-10 of 10 rows | 1-2 of 5 columns

The most important rule, that includes two features is rule592: enroll <= 864 & api.stu > 271

```
set.seed(456)
interactEnroll <- Interaction$new(predictor1, feature =
 "enroll")
```

In the plot below it is marked as red

```
h2.res <- data.frame(interactEnroll$results) %>%
  filter(.class=="Yes") %>%
  transmute(var=.feature, value=.interaction) %>%
  arrange(value) %>%
  mutate(var = factor(var, levels = var))
ggplot(h2.res, aes(y=var, yend=var, xend = 0, x=value,
color = !grepl("api.stu", var))) + geom_point(cex = 2.5
) +
  geom_segment(linetype = 2, lwd = 1.2) + theme(legend.
position = "none")
```




As you can see, this interaction is quite important, its H-value is 0.53

```
interactEnroll$results %>% filter(grepl("api.stu", .fea
ture))
```

| .feature<br><chr> | .class<br><fctr> | .interaction<br><dbl> |
|---|---|---|
| api.stu:enroll | No | 0.5312866 |
| api.stu:enroll | Yes | 0.5312866 |

2 rows

It is not the maximal H-value for interactions, though.

# Problem 2

[50 points]

We will use the api data again for this problem, but this time we are interested in predicting the api test scores for 2000 (i.e. api00) based on a battery of school, teacher and parent information. This time the data and models are available in the apipopProb2.RData R workspace. Note the data sets here are different than for Problem 1.

For this problem, you are given three models that aim at predicting **api00** (R objects model1 – model3), using different methods (i.e., CART, XGBoost, OLS regression). Utilize the following interpretation techniques with the training data to learn more about how those models were trained. Specifically for each of the models you should:

# Libraries

```
rm(list = ls())

suppressPackageStartupMessages(library(randomForest)) # Random Forest
suppressPackageStartupMessages(library(pdp)) # pdp
suppressPackageStartupMessages(library(ggplot2)) # ggplot2
suppressPackageStartupMessages(library(iml)) # iml
suppressPackageStartupMessages(library(kableExtra)) # kableExtra
suppressPackageStartupMessages(library(knitr)) # needed for the kable function
suppressPackageStartupMessages(library(ICEbox))

suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(grid))
suppressPackageStartupMessages(library(gridExtra))
suppressPackageStartupMessages(library(lattice))
suppressPackageStartupMessages(library(xgboost))
suppressPackageStartupMessages(library(glmnet))
```

# Dataset

```
load("./data/external/apipopProb2.RData")
```

# Functions

## F1. PDP1var

```
PDP1var <- function(model1, var1) {

    pd1 <- partial(model1,
                   pred.var = var1,
                   plot = TRUE,
                   rug = TRUE
                   )

    pd1a <- partial(model1,
                    pred.var = var1,
                    plot = TRUE,
                    plot.engine="ggplot2"
                    )

    pd1a <- pd1a + ggtitle("ggplot2-based PDP")

    grid.arrange(pd1,
                 pd1a,
                 ncol=2
                 )

}
```

# F2. PDP2var

```
PDP2var <- function(model1, var1, var2) {

    pdp12 <-  partial(model1,
                      pred.var = c(var1, var2),
                      plot = TRUE,
                      chull = TRUE,
                      palette = "magma"
                      )
    pdp12
}
```

# F3. ALE

```
ALE <- function(model1, feature1, data1, response1){

    predictor1 <- Predictor$new(model = model1,
                                data = data1,
                                y = response1
                                )

    pdp1  <- FeatureEffect$new(predictor1,
                               feature=feature1,
```

```
                                method="pdp"
                                )

    ale1  <- FeatureEffect$new(predictor1,
                                feature=feature1
                                )

    grid.arrange(ale1$plot(),
                 pdp1$plot(),
                 ncol=2
                 )
}
```

# F4. Interaction overall

```
Interaction_overall <- function(model1, data1, response
1) {

    predictor1 <- Predictor$new(model1,
                                data = data1,
                                y = response1
                                )
    interact <- Interaction$new(predictor1)
    results_head <- head(interact$results, nrow(interac
t$results))
    plot_output <- plot(interact)

    output_list <- list(results_head = results_head,
                        plot_output = plot_output
                        )
    return(output_list)
```

# F5. Interaction feature spec

```
Interaction_feature_specific <- function(model1, data1,
 response1, feature1) {

    predictor1 <- Predictor$new(model1,
                                data = data1,
                                y = response1
                                )
    interact <- Interaction$new(predictor1,
                                feature = feature1
                                )
    results_head <- head(interact$results, nrow(interac
t$results))
    plot_output <- plot(interact)
```

```
    output_list <- list(results_head = results_head,
                        plot_output = plot_output
                        )
    return(output_list)
}
```

# Important predictor

## model 1 VarImp

```
varImp(model1, scale = FALSE)
```

```
model1 variable importance

  only 20 most important variables shown (out of 72)
```

| | Overall <dbl> |
|---|---|
| meals | 36.034925 |
| stypeH | 23.873086 |
| ell | 15.095446 |
| stypeM | 14.695049 |
| full | 11.710786 |
| `cnameSan Diego` | 11.050500 |
| pct.resp | 9.787145 |
| cnameOrange | 9.553330 |
| `cnameLos Angeles` | 9.042844 |
| enroll | 8.369579 |

1-10 of 20 rows Previous 1 2 Next

## model 2 VarImp

```
varImp(model2, scale = FALSE)
```

```
model2 variable importance

  only 20 most important variables shown (out of 72)
```

| | Overall |
|---|---|
| | <dbl> |
| not.hsg | 1.9447936 |
| avg.ed | 1.8619536 |
| meals | 1.3359520 |
| grad.sch | 1.0332473 |
| ell | 0.7319306 |
| hsg | 0.5961552 |
| some.col | 0.4525073 |
| stypeH | 0.4171871 |
| enroll | 0.3597957 |
| api.stu | 0.1921287 |

1-10 of 20 rows Previous 1 2 Next

# model 3 VarImp

```
varImp(model3, scale = FALSE)
```

```
Warning: The model had been generated by XGBoost versio
n 1.0.0 or earlier and was loaded from a RDS file. We s
trongly ADVISE AGAINST using saveRDS() function, to ens
ure that your model can be read in current and upcoming
 XGBoost releases. Please use xgb.save() instead to pre
serve models for the long term. For more details and ex
planation, see https://xgboost.readthedocs.io/en/latest
/tutorials/saving_model.html
```

```
[12:23:44] WARNING: src/objective/regression_obj.cu:213
: reg:linear is now deprecated in favor of reg:squarede
rror.
model3 variable importance

  only 20 most important variables shown (out of 72)
```

| | Overall |
|---|---|
| | <dbl> |
| meals | 0.390101223 |
| avg.ed | 0.214796542 |
| not.hsg | 0.189276865 |

| | |
|---|---|
| full | 0.025878181 |
| stypeH | 0.023359827 |
| enroll | 0.022527774 |
| ell | 0.019530776 |
| pct.resp | 0.017719716 |
| api.stu | 0.011752570 |
| emer | 0.011314750 |

1-10 of 20 rows Previous 1 2 Next

# a. PDP 1-var

PDP 1-var: Partial Dependence plots 1-var

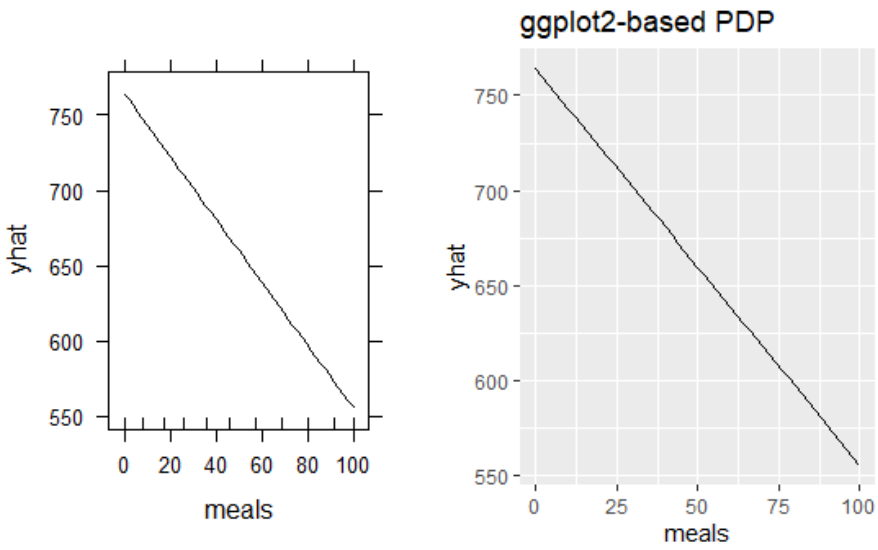
Produce Partial Dependence plots for the most important predictor of each model.

## model1: GLM

GLM: Generalized Linear Model

**Interpretation**:
A steep straight decline which means a strong negative relationship between “yhat” and “meals”.

```
PDP1var(model1, "meals")
```



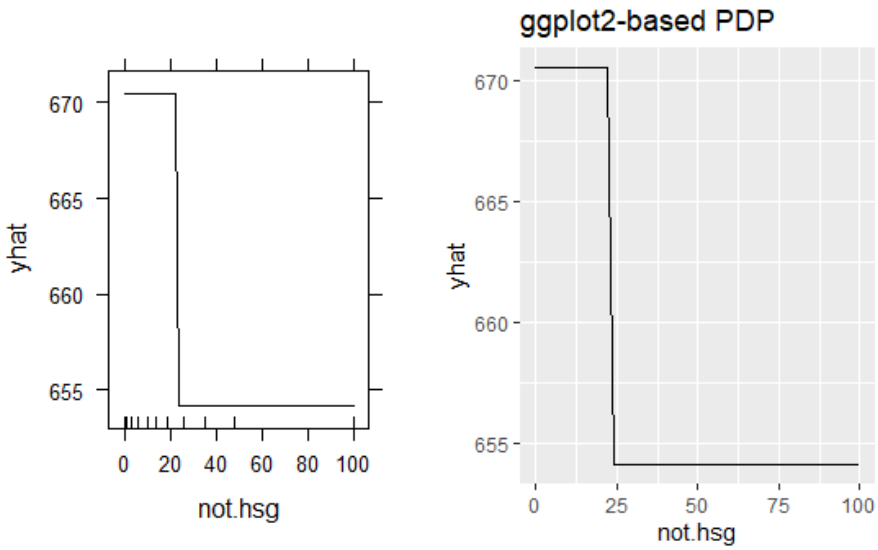

## model2: CART

CART: Classification And Regression Tree

**Interpretation**:

The curve on the Partial Dependence Plot (PDP) appears as a step function, with flat regions separated by sharp jumps downward, it suggests that there may be specific values of the predictor variable (not.hsg) that are particularly important in determining the predicted outcome (yhat).

```
PDP1var(model2, "not.hsg")
```




# model3: XGBoost

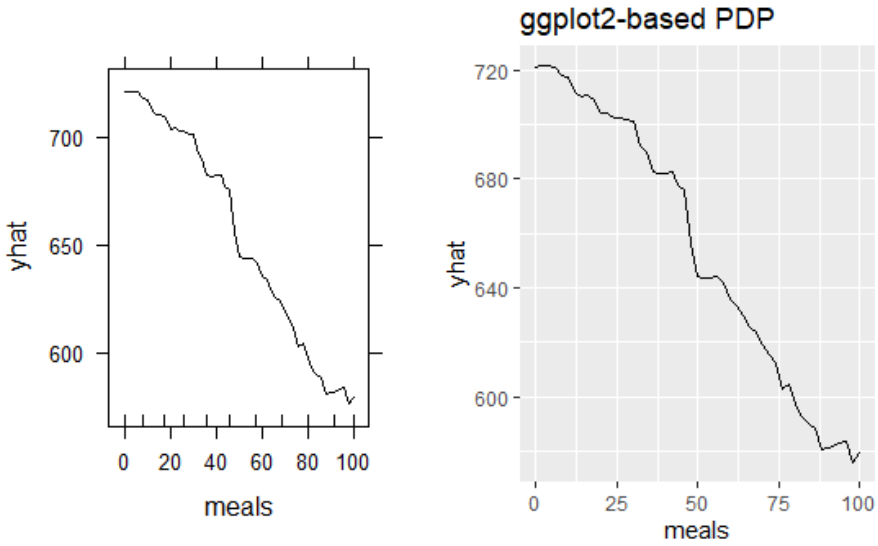
XGBoost: eXtreme Gradient Boosting

**Interpretation**:

The curve on a Partial Dependence Plot (PDP) is declining overall but has local ups and downs, it suggests:

1. There may be a non-linear relationship between the predictor variable and the predicted outcome. These local deviations could represent interactions between the predictor variable and other features in the model, or they could suggest that there are specific ranges of the predictor variable where it has a stronger or weaker impact on the predicted outcome.
2. It's also possible that the local ups and downs in the curve are simply due to noise in the data or limitations of the model. In this case, it's important to carefully evaluate the quality of the data and the appropriateness of the modeling approach to ensure that the PDP accurately represents the relationship between the predictor variable and the predicted outcome.

```
PDP1var(model3, "meals")
```




# b. PDP 2-var

PDP 2-var: Partial Dependence Plots 2-var

Produce Partial Dependence plots for the two most important predictors of each model (i.e., 3D plots or heatmaps or contour plots).

# model1: GLM

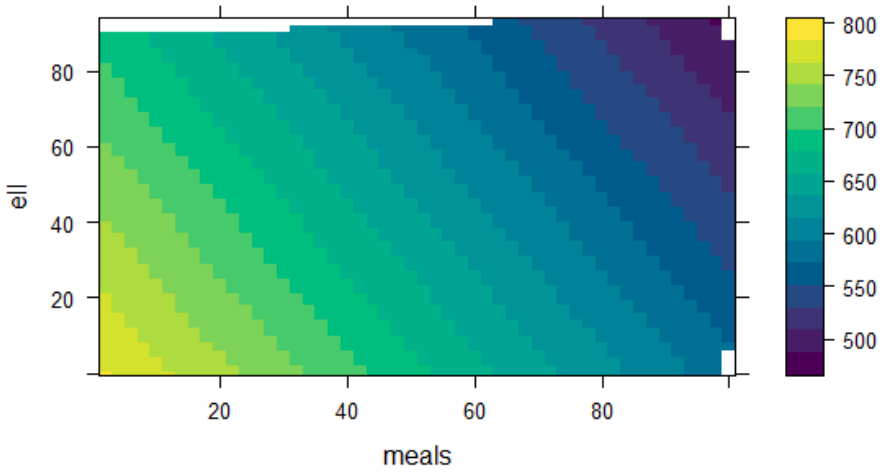
GLM: Generalized Linear Model

**Interpretation**:

This type of relationship is known as a U-shaped relationship, where the predicted outcome is highest when both variables are at low levels, decreases as one or both variables increase, and then may increase again at higher levels of both variables. It suggests that there may be a nonlinear relationship between the two predictor variables and the target variable.

The slightly increasing slope of the plot suggests that the effect of the predictor variables on the predicted outcome is becoming slightly stronger as the values of the predictor variables increase. However, the U-shaped relationship suggests that there may be other factors at play that are causing the decrease in the predicted outcome as the values of the predictor variables increase.

```
PDP2var(model1, "meals", "ell")
```




# model2: CART

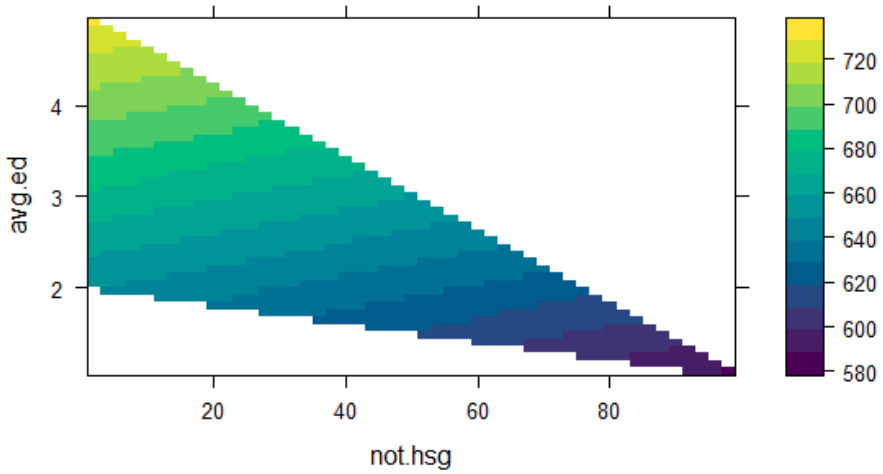
CART: Classification And Regression Tree

**Interpretation**:

The decreasing slope suggests that as one predictor variable increases, the effect of the other predictor variable on the predicted outcome is becoming weaker. This is indicative of an interaction effect, where the effect of one predictor variable depends on the value of the other predictor variable.

The fact that the z-axis is highest when the y-axis is high and the x-axis is low, and lowest when the y-axis is low and the x-axis is high, suggests that the two predictor variables are working in opposite directions. This means that when one predictor variable is high and the other is low, the predicted outcome is high, while the predicted outcome is low when the values are reversed.

```
PDP2var(model1, "not.hsg", "avg.ed")
```




# model3: XGBoost

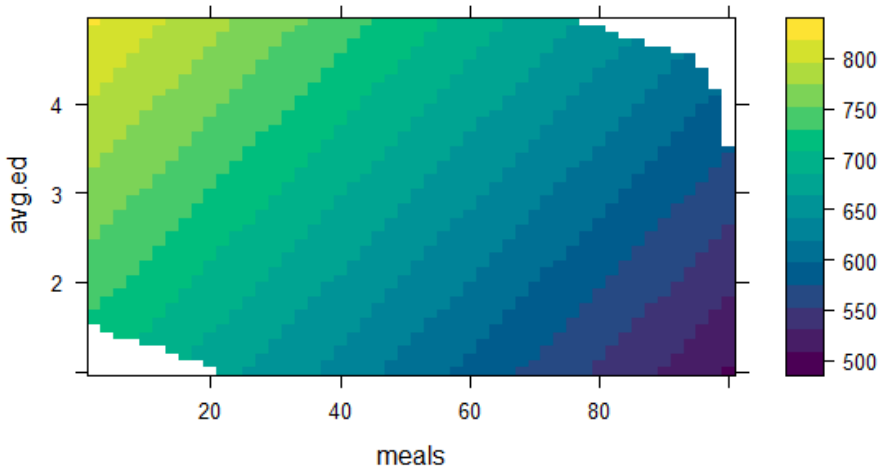
XGBoost: eXtreme Gradient Boosting

**Interpretation**:

The slightly decreasing slope suggests that the interaction effect is weaker than in the case where the slope is strictly decreasing. However, it still suggests that as one predictor variable increases, the effect of the other predictor variable on the predicted outcome is becoming weaker.

The fact that the z-axis is highest when the y-axis is high and the x-axis is low, and lowest when the y-axis is low and the x-axis is high, suggests that the two predictor variables are working in opposite directions. This means that when one predictor variable is high and the other is low, the predicted outcome is high, while the predicted outcome is low when the values are reversed.

```
PDP2var(model1, "meals", "avg.ed")
```



# c. ALE plots

ALE plots: Accumulated Local Effects plots

- Produce Accumulated Local Effects plots for the most important predictor in each model.
- How do these plots vary from those produced in part b? Given the correlation structure in the apipop_train data is it surprising or not that these plots differ or are similar to each other.
    - **Answer**:
        - **model1 (GLM)**: The both interpretations are describing different types of relationships between predictor variables and a target variable. The PDP 2-var interpretation describes a U-shaped relationship, where the predicted outcome initially increases as both predictor variables increase, but then decreases again at higher levels of both variables. This suggests a nonlinear relationship between the predictor variables and the target variable. In contrast, the ALE plot interpretation describes a negative relationship between the predictor variable and the predicted outcome. As the value of the predictor variable increases, the predicted outcome decreases. This suggests a linear relationship between the predictor variable and the target variable. Additionally, the PDP 2-var interpretation mentions that there may be other factors at play causing the decrease in the predicted outcome as the values of the predictor variables increase, while the ALE plot interpretation does not mention any potential confounding factors.
        - **model2 (CART)**: The two interprets describe different patterns in the relationship between predictor variables and the predicted outcome. The PDP 2-var interpret suggests that there is an interaction effect, where the effect of one predictor variable depends on the value of the other predictor variable. This means that the relationship between the predictor variables and the predicted outcome is not independent but rather depends on the joint values of the predictor variables. Additionally, the fact that the z-axis is highest when the y-axis is high and the x-axis is low, and lowest when the y-axis is low and the x-axis is high, suggests that the two predictor variables are working in opposite directions. On the other hand, the ALE plot interpret suggests that the relationship between the predictor variable and the predicted outcome is not linear. There may be discrete intervals

or steps where the effect of the predictor variable on the predicted outcome changes. This means that the relationship between the predictor variable and the predicted outcome may be due to different factors or conditions that affect the outcome, such as different subgroups or categories of observations that have different relationships between the predictor variable and the predicted outcome. Therefore, the PDP 2-var interpret suggests an interaction effect between the predictor variables, while the ALE plot interpret suggests non-linearity in the relationship between the predictor variable and the predicted outcome.

- **model3 (XGBoost)**: The PDP 2-var and ALE plot interprets are different in their focus and conclusions. The PDP 2-var interpretation describes a linear relationship between two predictor variables and the predicted outcome. It suggests that as one predictor variable increases, the effect of the other predictor variable on the predicted outcome is becoming weaker. It also suggests that the two predictor variables are working in opposite directions, with high values of one variable and low values of the other leading to high predicted outcomes. This interpretation does not suggest any non-monotonic behavior or complex interactions between the predictor variables and other variables in the model. On the other hand, the ALE plot interpretation suggests a non-linear and potentially complex relationship between the predictor variable and the predicted outcome. It describes ups and downs in the ALE plot, indicating that the relationship between the predictor variable and the predicted outcome may change direction or magnitude at certain values of the predictor variable. This behavior could arise due to complex interactions or nonlinear relationships with other variables in the model. This interpretation suggests that further investigation is needed to understand the factors contributing to the non-monotonic behavior.

# model1: GLM

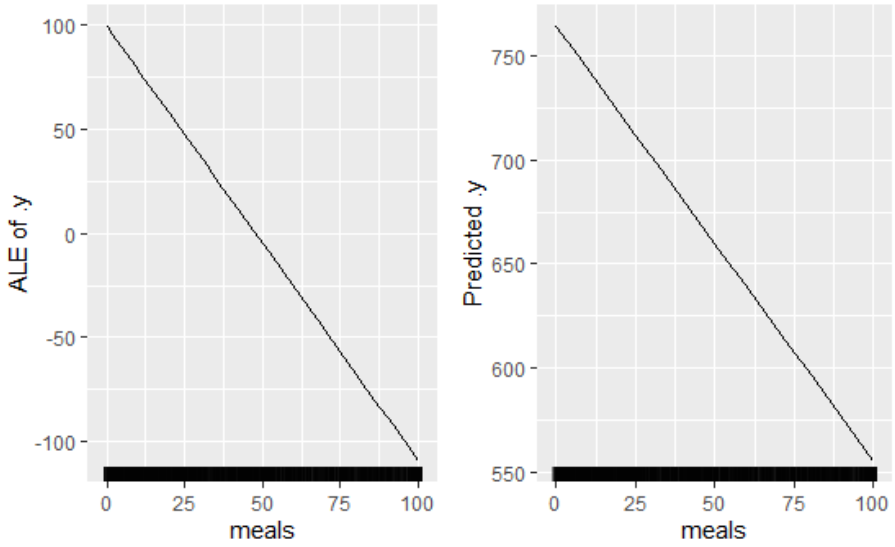
GLM: Generalized Linear Model

**Interpretation**:

It suggests a negative relationship between the predictor variable (x-axis) and the predicted outcome (y-axis). This means that as the value of the predictor variable increases, the predicted outcome decreases.

The decreasing slope of the ALE plot suggests that the relationship between the predictor variable and the predicted outcome is becoming weaker as the value of the predictor variable increases. This is consistent with the decreasing ALE of y and predicted y values with increasing x-axis values.

```
ALE(model1,
    "meals",
    apipop_train,
    apipop_train$api00
    )
```




# model2: CART

CART: Classification And Regression Tree

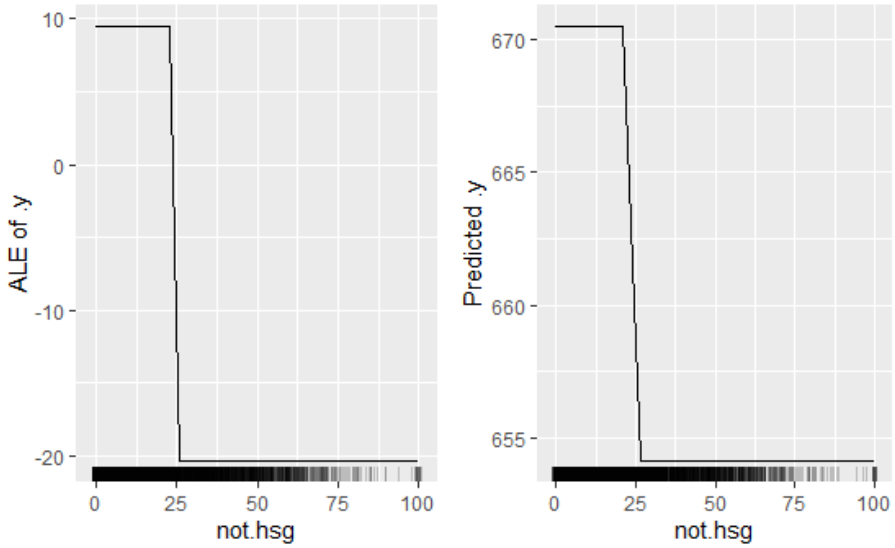
**Interpretation**:

It suggests that the relationship between the predictor variable and the predicted outcome is not linear. Instead, there may be discrete intervals or steps where the effect of the predictor variable on the predicted outcome changes.

The step-like pattern in the ALE plot suggests that the relationship between the predictor variable and the predicted outcome may be due to different factors or conditions that affect the outcome. For example, there may be different subgroups or categories of observations that have different relationships between the predictor variable and the predicted outcome.

```
ALE(model2,
    "not.hsg",
    apipop_train,
```

```
    apipop_train$api00
    )
```




# model3: XGBoost

XGBoost: eXtreme Gradient Boosting

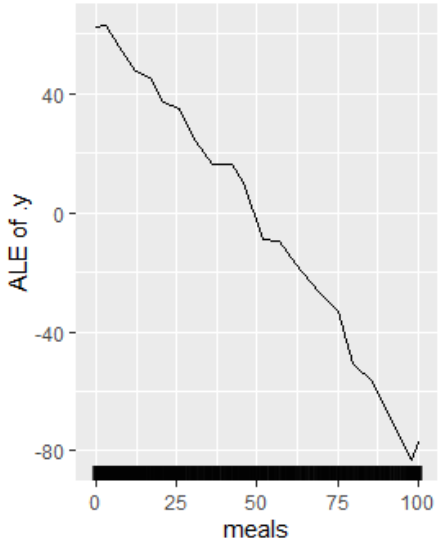
**Interpretation**:

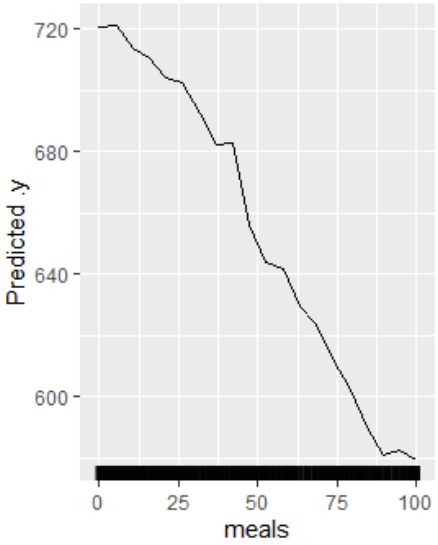
It suggests that the relationship between the predictor variable and the predicted outcome is not linear and may have some non-monotonic behavior.

The ups and downs in the ALE plot indicate that the relationship between the predictor variable and the predicted outcome may change direction or magnitude at certain values of the predictor variable. This non-monotonic behavior can arise due to complex interactions or nonlinear relationships between the predictor variable and other variables in the model.

In this case, it may be useful to investigate the data further to identify the factors that contribute to the non-monotonic behavior. This could involve examining other variables that may interact with the predictor variable, or exploring subgroups or categories of observations that have different relationships with the predictor variable.

```
ALE(model3,
    "meals",
    apipop_train,
    apipop_train$api00
    )
```




# d. Interaction plots

- Compute the overall interaction H statistics
- as well as the feature specific statistics for the **ELL variable** for all of the models.
- Provide a table that includes as rows the predictors in the apipop_train data file and the H statistics for each model as columns.
- For the ELL specific H statistics, provide an interaction plot per model.

## model1: GLM

GLM: Generalized Linear Model

**Interpretation**:

- overall:
    - The overall interaction values for the Generalized Linear Model are all very small, with the largest value being 1.259390e-15 for the “cname” feature. This suggests that there is little to no interaction between the different features in the model, and that the features are largely independent in their influence on the target variable. In other words, the model does not detect any significant interactions between the features that would affect their contribution to the target variable. This result may indicate that the features have a simple linear relationship with the target variable or that the model is not complex enough to capture any non-linear relationships or interactions between the features.
- Feature specific:
    - The output of the specific interaction for the Generalized Linear Model indicates the strength of interaction between two variables, in this case, the interaction between “stype”

and “ell”. The values of the coefficients represent the strength of the interaction. A coefficient close to zero indicates little or no interaction, while a large coefficient indicates a strong interaction between the two variables. Looking at the specific interaction coefficients, we can see that the interaction between “stype” and “ell” is very small, with a coefficient of 1.243676e-15. This suggests that there is little interaction between these two variables in predicting the target variable. Similarly, the coefficients for “cname:ell”, “api00:ell”, and “api99:ell” are all zero, indicating no interaction between these variables. On the other hand, we can see a relatively strong interaction between “meals” and the target variable “ell” (coefficient of 3.420603e-16), as well as between “not.hsg” and “ell” (coefficient of 2.050319e-15) and between “some.col” and “ell” (coefficient of 4.600026e-15).

```
Interaction_overall(model1,
                    apipop_train,
                    apipop_train$api00
                    )
```

```
Attaching package: 'webshot'

The following object is masked from 'package:hardhat':

    shrink
```

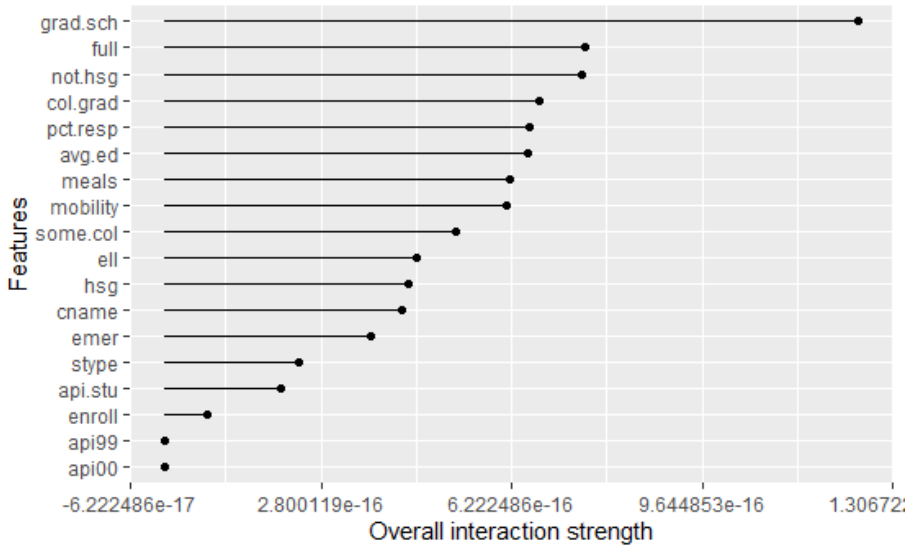
```
$results_head
```

| .feature <chr> | .interaction <dbl> |
|---|---|
| stype | 2.389279e-16 |
| cname | 4.260613e-16 |
| api00 | 0.000000e+00 |
| api99 | 0.000000e+00 |
| meals | 6.184668e-16 |
| ell | 4.503063e-16 |
| mobility | 6.133975e-16 |
| pct.resp | 6.544961e-16 |
| not.hsg | 7.475603e-16 |

| | |
|---|---|
| hsg | 4.381122e-16 |

1-10 of 18 rows Previous **1** 2 Next

```
$plot_output
```




```
Interaction_feature_specific(model1,
                             apipop_train,
                             apipop_train$api00,
                             "ell"
                             )
```

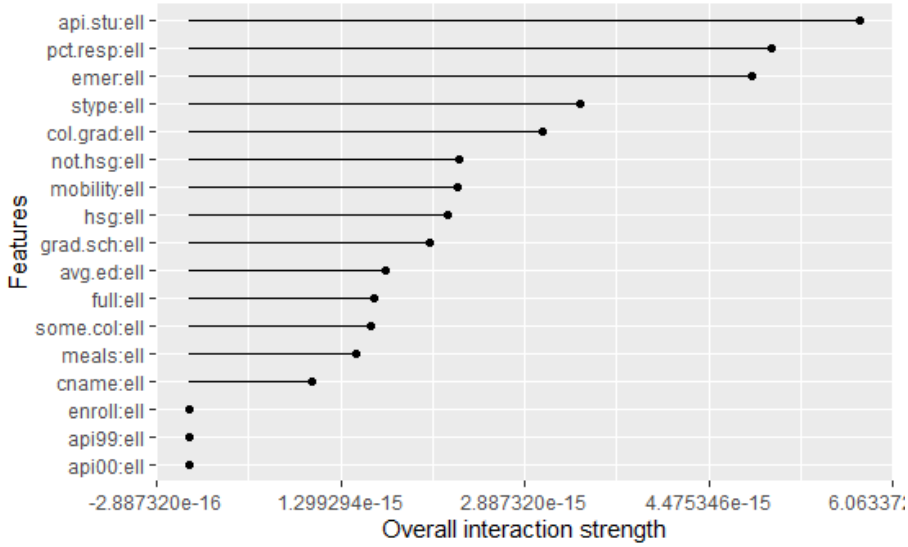
```
$results_head
```

| .feature<br><chr> | .interaction<br><dbl> |
|---|---|
| stype:ell | 3.372402e-15 |
| cname:ell | 1.050978e-15 |
| api00:ell | 0.000000e+00 |
| api99:ell | 0.000000e+00 |
| meals:ell | 1.427519e-15 |
| mobility:ell | 2.303482e-15 |
| pct.resp:ell | 5.015625e-15 |
| not.hsg:ell | 2.319581e-15 |
| hsg:ell | 2.227700e-15 |

| | |
|---|---|
| some.col:ell | 1.560570e-15 |

1-10 of 17 rows Previous **1** 2 Next

```
$plot_output
```




# model2: CART

CART: Classification and Regression Tree

**Interpretation**:

- overall:
  - The ".feature" column lists the names of the independent variables (predictors or features) in the dataset, which are represented as character strings. The ".interaction" column shows the importance of each feature in the CART model, as measured by the Gini index. The Gini index is a measure of the impurity of a set of samples, and in the context of a CART model, it represents the extent to which a feature can be used to split the data into subsets that are more homogeneous with respect to the outcome variable (i.e., the dependent variable). The values in the ".interaction" column indicate the relative importance of each feature in the model. Higher values indicate that the feature is more important for splitting the data and improving the accuracy of the model, while lower values indicate that the feature is less important or may not be useful for splitting the data. In this particular output, the features "stype", "meals", and "not.hsg" have the highest importance scores, with values of 0.2715888, 0.2641148, and 0.1518510, respectively. The feature "avg.ed" has an importance score of 0.1200937, while the remaining features have importance scores of 0 or close to 0, indicating that they are not useful for splitting the

data in this particular model.

- Feature specific:
  - The ".feature" column lists the names of the independent variables (predictors or features) in the dataset, which are represented as character strings, and the ".interaction" column shows the interaction (or importance) of each feature with a specific feature, "ell". In this specific interaction, the feature "ell" is used as the basis for splitting the data, and the values in the ".interaction" column represent the importance of each feature in splitting the data based on "ell". A value of 0 indicates that the feature is not useful for splitting the data based on "ell", while NaN (not a number) indicates that the feature was not used in the model for this specific interaction. Therefore, in this specific interaction, the features "stype", "meals", and "not.hsg" are useful for splitting the data based on "ell", with importance scores of 0. The features "avg.ed", "cname", "api00", "api99", "mobility", "pct.resp", "hsg", "some.col", "col.grad", "grad.sch", "full", "emer", "enroll", and "api.stu" are not useful for splitting the data based on "ell", as indicated by NaN importance scores. It's important to note that this is only a specific interaction of the CART model, and the importance of the features may differ when considering different interactions or the overall interaction of the model. Therefore, it's necessary to interpret the model's results and interactions in the context of the specific research question and dataset.

```
Interaction_overall(model2,
                    apipop_train,
                    apipop_train$api00
                    )
```

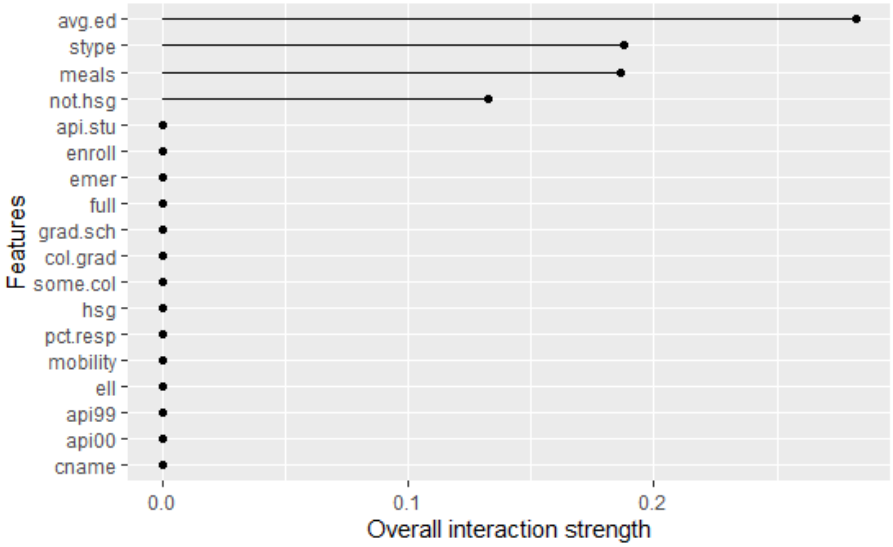
```
$results_head
```

| .feature<br><chr> | .interaction<br><dbl> |
|---|---|
| stype | 0.1878634 |
| cname | 0.0000000 |
| api00 | 0.0000000 |
| api99 | 0.0000000 |
| meals | 0.1866590 |
| ell | 0.0000000 |

| | |
|---|---|
| mobility | 0.0000000 |
| pct.resp | 0.0000000 |
| not.hsg | 0.1325161 |
| hsg | 0.0000000 |

1-10 of 18 rows Previous **1** 2 Next

```
$plot_output
```




```
Interaction_feature_specific(model2,
                             apipop_train,
                             apipop_train$api00,
                             "ell"
                             )
```

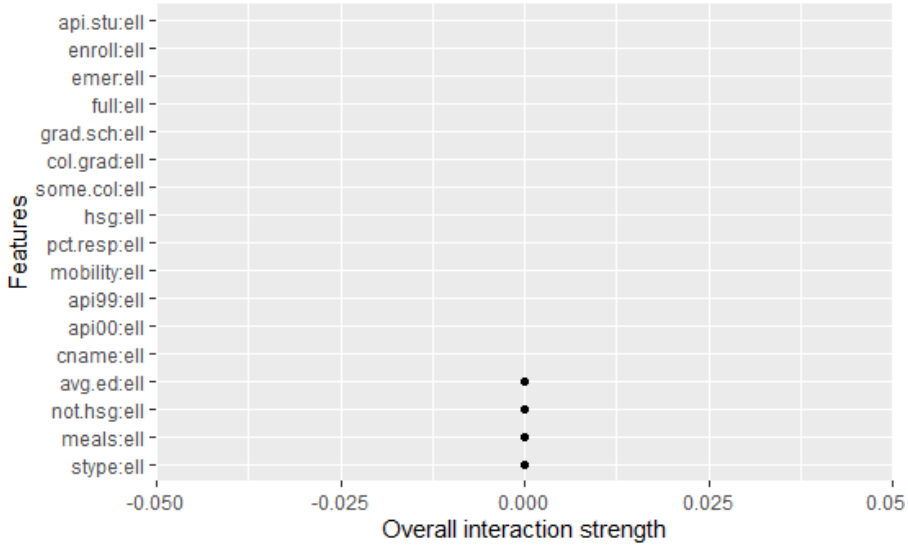
```
$results_head
```

| .feature | .interaction |
|---|---|
| <chr> | <dbl> |
| stype:ell | 0 |
| cname:ell | NaN |
| api00:ell | NaN |
| api99:ell | NaN |
| meals:ell | 0 |
| mobility:ell | NaN |
| pct.resp:ell | NaN |

| | |
|---|---|
| not.hsg:ell | 0 |
| hsg:ell | NaN |
| some.col:ell | NaN |

1-10 of 17 rows Previous **1** 2 Next

```
$plot_output
```




# model3: XGBoost

XGBoost: eXtreme Gradient Boosting

**Interpretation**:

- overall:
  - The ".feature" column lists the names of the independent variables (predictors or features) in the dataset, and the ".interaction" column shows the importance of each feature for the overall model. In this case, the values in the ".interaction" column indicate the relative contribution of each feature to the model's performance in predicting the outcome variable. According to the importance scores, "meals" is the most important feature for the model, with an importance score of 0.22003425. "pct.resp" and "avg.ed" also have relatively high importance scores of 0.11972590 and 0.10180762, respectively. Other features that are relatively important for the model include "cname", "ell", "not.hsg", and "api.stu", with importance scores ranging between 0.07118320 and 0.09305863. On the other hand, features such as "api00", "api99", "hsg", "some.col", "col.grad", "grad.sch", "emer", "enroll", and "mobility" have relatively low importance scores, suggesting that they have

less influence on the model's predictions. It's worth noting that the importance scores reflect the relative contribution of each feature to the overall model and that they should be interpreted in the context of the specific research question and dataset. The importance scores may also vary depending on the hyperparameters and settings used in the XGBoost model.

- Feature specific:
    - This specific interaction output is showing the interaction between each pair of feature and the target variable, where the target variable is 'ell' (percentage of English language learners in a school). The values indicate the importance of the interaction between each feature and the target variable in predicting the target variable. A higher value indicates that the interaction between the two features is more important for predicting the target variable. For example, the interaction between 'cname' (county name) and 'ell' has a value of 0.17532633, which indicates that the county name has a strong interaction with the percentage of English language learners in a school. Similarly, the interaction between 'not.hsg' (percentage of adults in the community without a high school diploma) and 'ell' has a value of 0.19020011, which indicates that the percentage of adults in the community without a high school diploma has a strong interaction with the percentage of English language learners in a school.

```
Interaction_overall(model3,
                    apipop_train,
                    apipop_train$api00
                    )
```

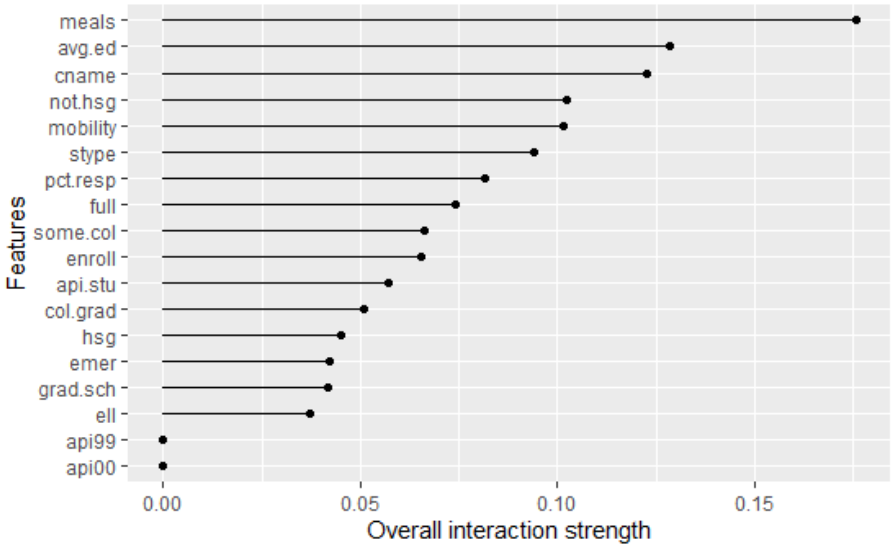
```
$results_head
```

| .feature <chr> | .interaction <dbl> |
|---|---|
| stype | 0.09411668 |
| cname | 0.12270191 |
| api00 | 0.00000000 |
| api99 | 0.00000000 |
| meals | 0.17572996 |
| ell | 0.03740807 |
| mobility | 0.10152227 |

| | |
|---|---|
| pct.resp | 0.08174127 |
| not.hsg | 0.10253179 |
| hsg | 0.04530117 |

1-10 of 18 rows Previous 1 2 Next

```
$plot_output
```




```
Interaction_feature_specific(model3,
                             apipop_train,
                             apipop_train$api00,
                             "ell"
                             )
```

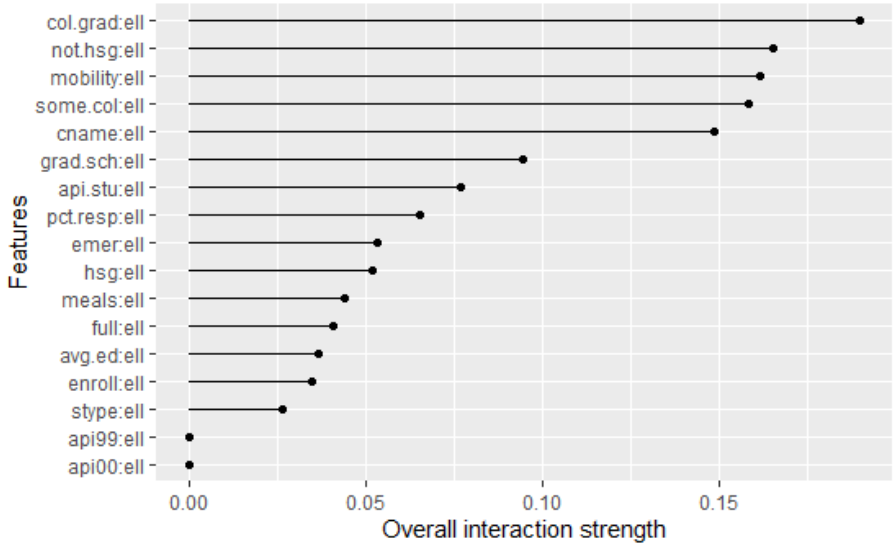
```
$results_head
```

| .feature<br><chr> | .interaction<br><dbl> |
|---|---|
| stype:ell | 0.02629532 |
| cname:ell | 0.14850579 |
| api00:ell | 0.00000000 |
| api99:ell | 0.00000000 |
| meals:ell | 0.04375712 |
| mobility:ell | 0.16160580 |
| pct.resp:ell | 0.06516777 |

| | |
|---|---|
| not.hsg:ell | 0.16555725 |
| hsg:ell | 0.05173210 |
| some.col:ell | 0.15852779 |

1-10 of 17 rows Previous 1 2 Next

```
$plot_output
```




# e. Prediction performance

Evaluate the prediction performance of each model using the test set that is included in the workspace (e.g. apipop_test).

## model1: GLM

GLM: Generalized Linear Model

```
# prediction on test data
yhat <- predict(model1, s = model1, apipop_test)
# RMSE for test data
error.test <- yhat - apipop_test$api00
rmse.test <- sqrt(mean(error.test^2))
rmse.test
```

```
[1] 48.56357
```

## model2: CART

CART: Classification And Regression Tree

Hide

```
# prediction on test data
yhat <- predict(model1, s = model2, apipop_test)
# RMSE for test data
error.test <- yhat - apipop_test$api00
rmse.test <- sqrt(mean(error.test^2))
rmse.test
```

```
[1] 48.56357
```

# model3: XGBoost

XGBoost: eXtreme Gradient Boosting

```
# prediction on test data
yhat <- predict(model3, s = model1, apipop_test)
# RMSE for test data
error.test <- yhat - apipop_test$api00
rmse.test <- sqrt(mean(error.test^2))
rmse.test
```

```
[1] 44.55533
```

# f.

Considering the results for the tasks above, which model object belongs to which method? Explain your choice!

We could not get what the f part is asking for. So, we put an interpretation for each part.