Fragile Families Challenge

Group 5

Introduction

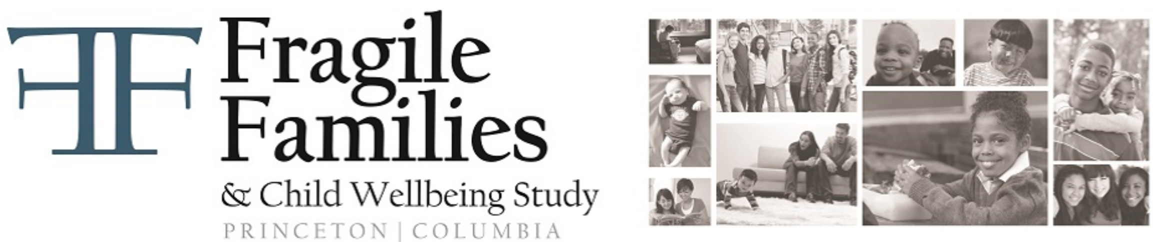


Figure 1:

- **BACKGROUND**

The Fragile Families Challenge is a mass collaboration that will combine predictive modeling, causal inference, and in-depth interviews to yield insights that can improve the lives of disadvantaged children in the United States. By working together we can discover things that none of us can discover individually.

The Fragile Families Challenge is based on the Fragile Families and Child Wellbeing Study, which has followed thousands of American families for more than 15 years. During this time, the Fragile Families study collected information about the children, their parents, their schools, and their larger environments.

Figure 2:

- **HELP THE WORLD:**

Designed to produce scientific knowledge that can be used to improve the lives of disadvantaged children in the United States. Even more than that, we hope the Fragile Families Challenge can serve as a model for how social scientists and data scientists can collaborate on problems of societal importance.

• **WIN PRIZES:**

We will award prizes to participants who make important contributions to the project. All prize winners will be given an all-expenses paid trip to Princeton University for the scientific workshop at the end of the project.

Research Data

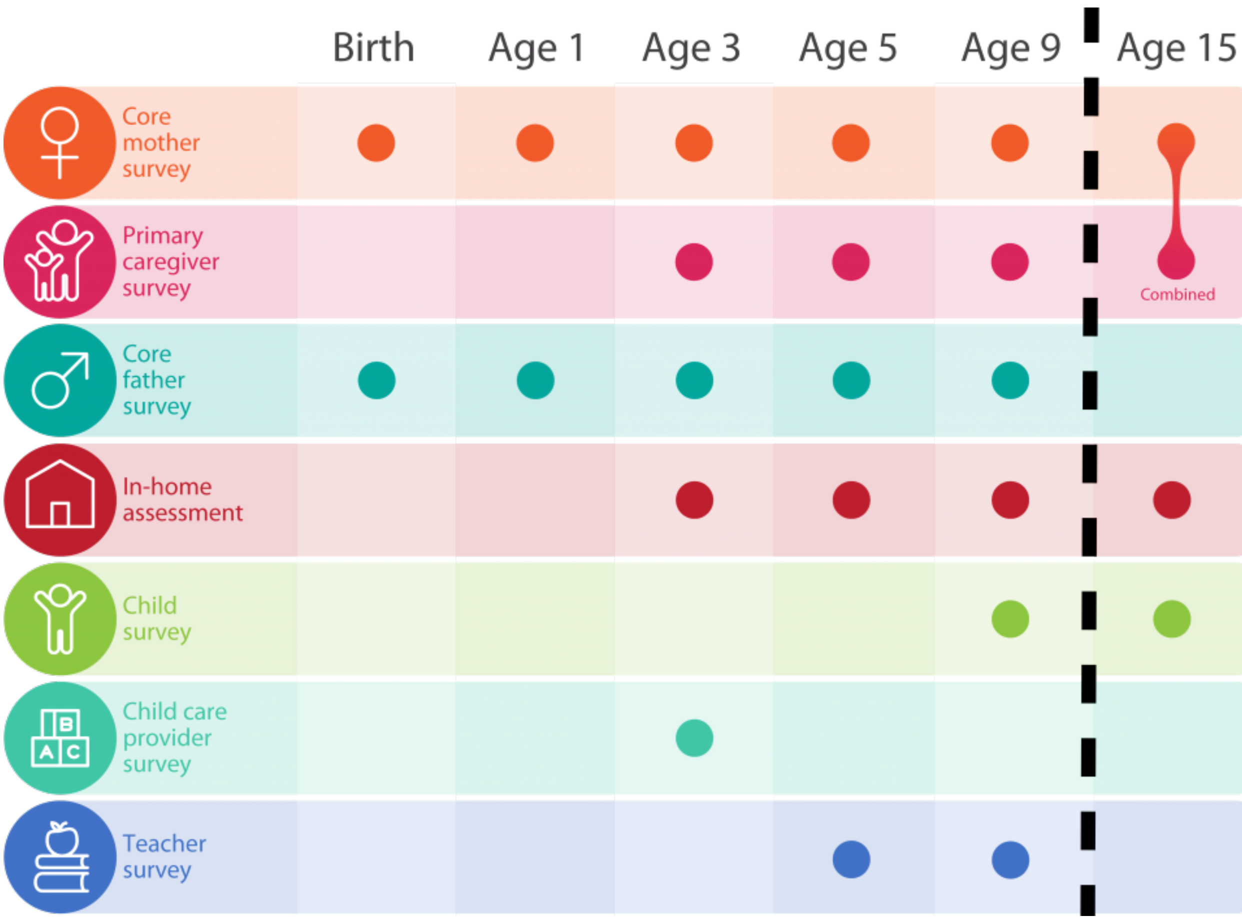


Figure 3:

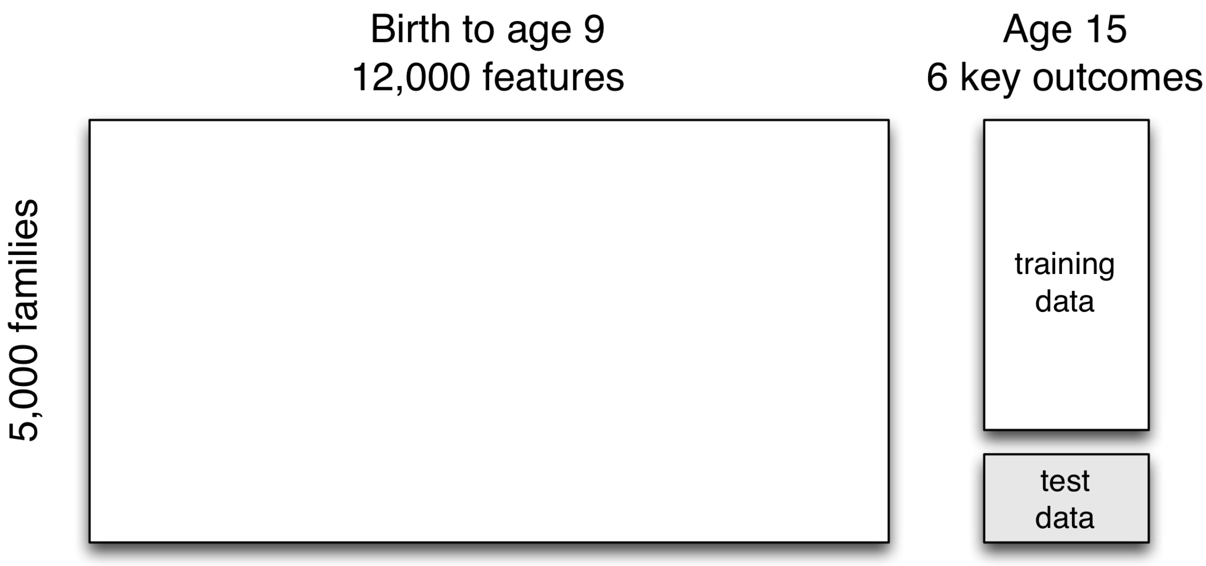


Figure 4:

Continuous outcomes:

- GPA
- Grit
- Material hardship

Binary outcomes:

- Housing eviction
- Layoff of a caregiver
- Job training for a caregiver

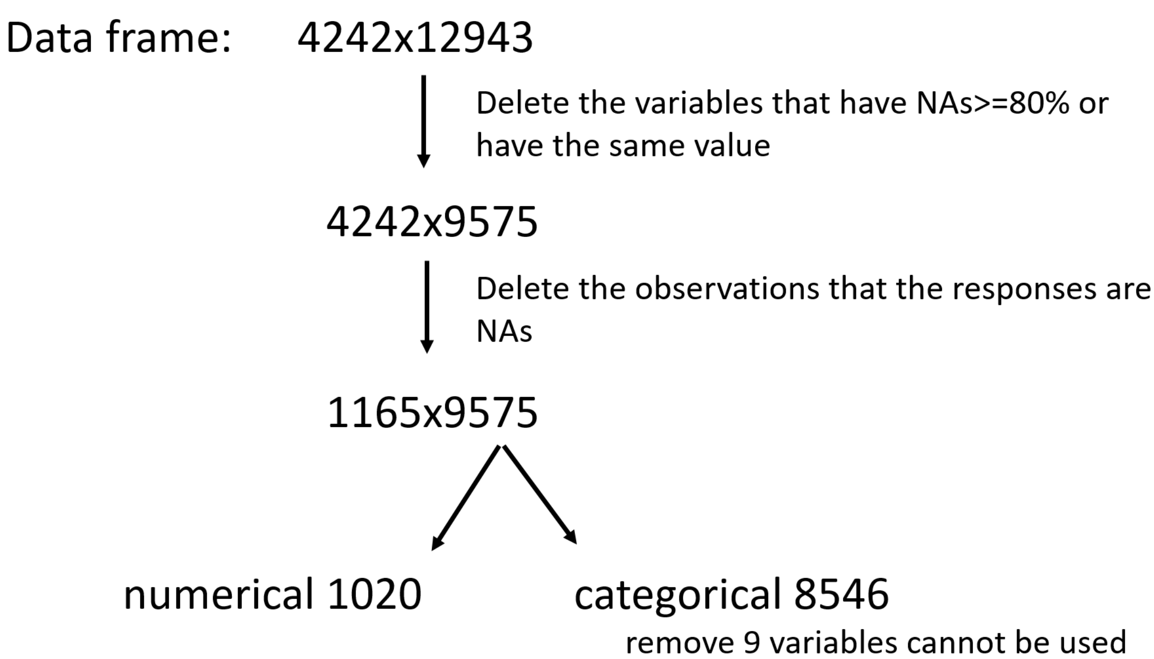
Figure 5:

Data Matrix

Outcomes

Step 0: Clean Data

- **-9 Not in wave** – Did not participate in survey/data collection component
- -8 Out of range – Response not possible; rarely used
- -7 Not applicable (also -10/-14) – Rarely used for survey questions
- **-6 Valid skip** – Intentionally not asked question; question does not apply to respondent or response known based on prior information.
- -5 Not asked "Invalid skip" – Respondent not asked question in the version of the survey they received.
- -3 Missing – Data is missing due to some other reason; rarely used
- **-2 Don't know** – Respondent asked question; Responded "Don't Know".
- **-1 Refuse** – Respondent asked question; Refused to answer question




```r
library(readr)
library(Matrix)
library(mlr)
library(xgboost)
library(plyr)

bg=read_csv('background.csv')
train=read_csv('train.csv')

# Delete the records that corresponding to NA in the train
```

```r
naid<- which(is.na(train$gpa))
bgtrain<- bg[setdiff(train$challengeID,train$challengeID[naid]),]

# Delete the variables that are 80% NAs
n <- ncol(bgtrain)
na_count<- rep(NA, n)
for(i in 1:n){
        na_count[i]<- sum(is.na(bgtrain[,i]))
}
na_index<- c(1:n)[na_count>=0.8*nrow(bgtrain)]
bgtrain<- bgtrain[,-na_index]

# Delete the variables that have the same value
namelist=names(bgtrain)
for (f in namelist) {
        if(nrow(unique(bgtrain[f]))==1) {
                bgtrain[f]=NULL
        }
}

# Delete the variables that can neither be used as charater
n<- ncol(bgtrain)
t<- rep(NA, n)
for(i in 1:n){
        t[i]<- typeof(bgtrain[[1,i]])
}
# unique(t)
char_index<- c(1:n)[t=="character"]
inter_index<- c(1:n)[t=="integer"]
dou_index<- c(1:n)[t=="double"]

# seperate the variable by different type, charater are those that cannot
# be convert to any type of data, doubles are continuous data. And intergers may
# contain part of continuous data. Thus, we should also handle these variables.
bgtrain_char<- bgtrain[, char_index]
bgtrain_inter<- bgtrain[, inter_index]
bgtrain_dou<- bgtrain[, dou_index]

# For interger-type variables, check the factor number it contains. And we consider those have
# more than 20 factors as continuous number.
fac_num<- as.numeric(apply(bgtrain_inter, 2, function(vec)
{return(length(table(vec)))} ))
fac_index<- which(fac_num<=20)
bgtrain_dou<- cbind(bgtrain_inter[,-fac_index], bgtrain_dou)
bgtrain_inter<- bgtrain_inter[,fac_index]

# Fianlly, we get two seperate dataset to deal with in next steps
bgtrain_dou<- data.frame(bgtrain_dou)
bgtrain_inter<- data.frame(bgtrain_inter)
```

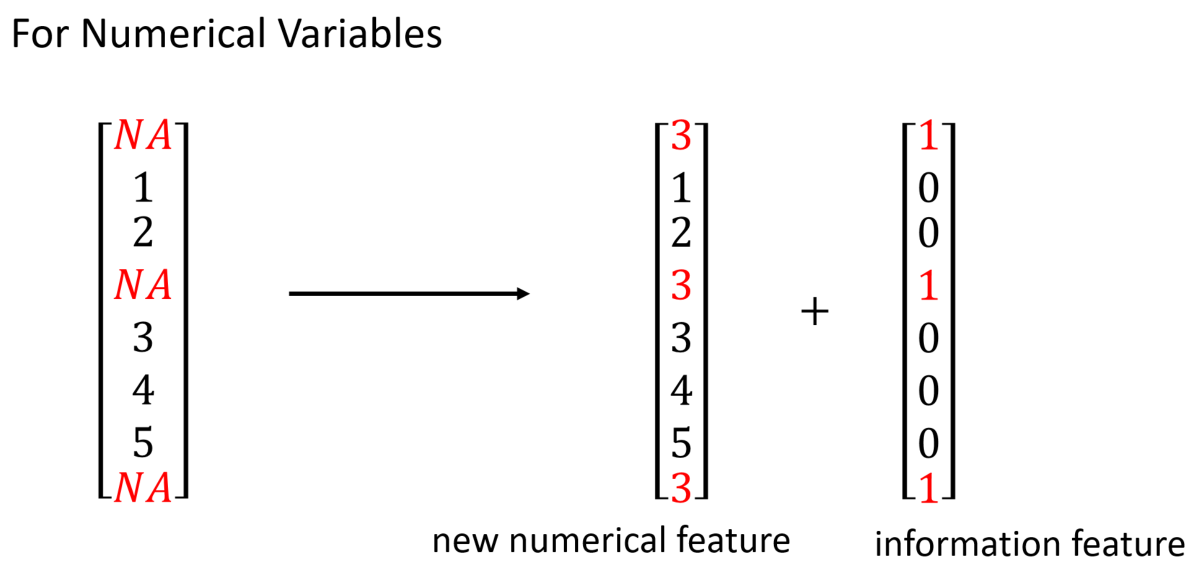


Figure 6: Numerical

Deal with Numerical Variables

```
### fill NA and create new data
fill_each_column <- function(each_col){
        na_label <- is.na(each_col)
        if(sum(na_label) > 0){
                cate_col <- ifelse(na_label == T, 1, 0) ## T = 1, is NA
                fill <- median(each_col, na.rm = T)
                each_col[na_label] <- fill
                return(list(NEW_COLUMN = each_col, NEW_CATE = cate_col))
        }else{
                ## no NA in a column
                return(list(NEW_COLUMN = rep(0,length(each_col)), NEW_CATE = rep(0,length(each_col))))
        }

}
```

Deal with Categorical Variables

Step 1: Regression and Prediction on GPA

Model 1: 9-year-old kid numerical variables + numerical information variables

```
my.text <- "^([a-z]{1,3}5)"
indices <- grepl(my.text,colnames(bgtrain_dou))
mat <- bgtrain_dou[,indices]

## INPUT data= continuous dataframe
gp5data_dou <- mat
```

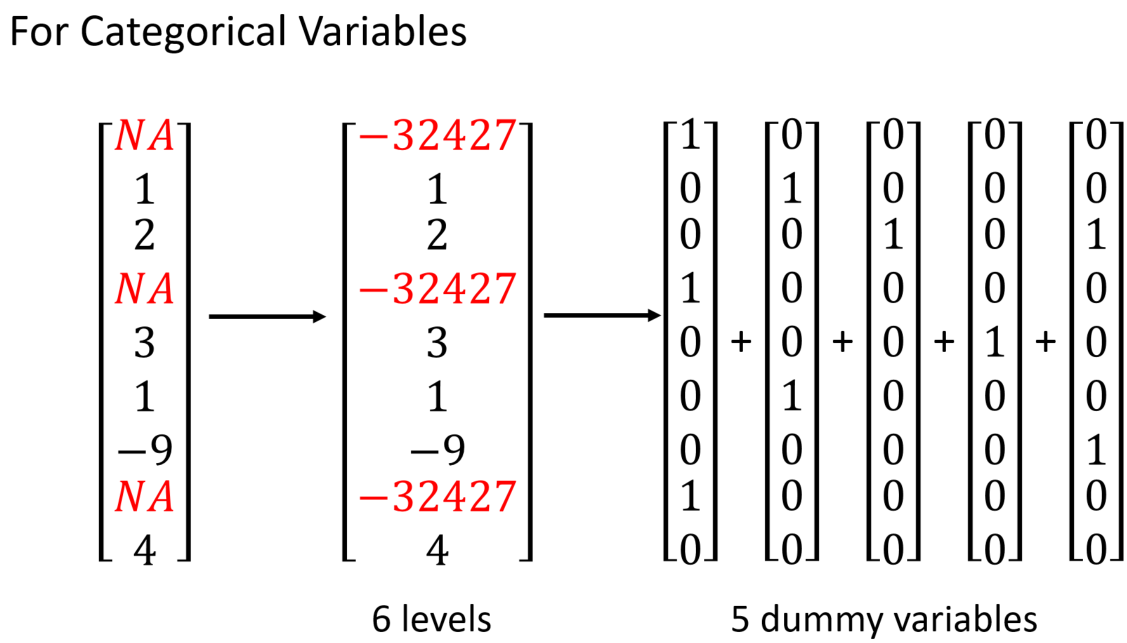
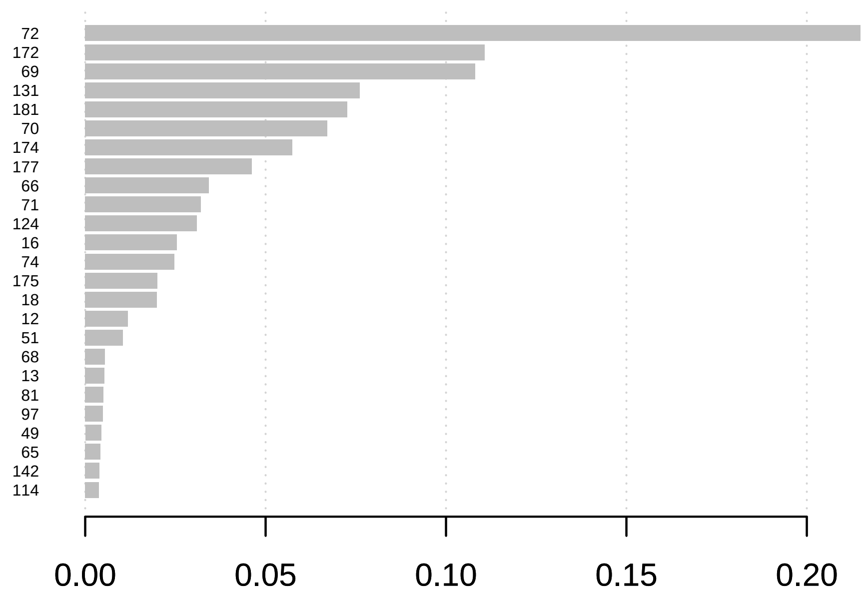


Figure 7: Categorical

```
gp5data_dou <- matrix(unlist(gp5data_dou), nrow(gp5data_dou))
## 1 for new column, 2 for new categorical features
gp5data_dou_RMNA <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_dou_na <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[2]]})

# remove NON-NA columns
non_na_dou <- colSums(cate_dou_na) == 0
cate_dou_na <- cate_dou_na[,!non_na_dou]
bg_dou <- bg[,colnames(mat)]

gp5pred_dou <- bg_dou
gp5pred_dou <- matrix(unlist(gp5pred_dou), nrow(gp5pred_dou))
## 1 for new column, 2 for new categorical features
gp5pred_dou_RMNA <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_pred_dou_na <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[2]]})

dtrain=xgb.DMatrix(cbind(data.matrix(mat),cate_dou_na),label=train$gpa[which(!is.na(train$gpa))])
dtest=xgb.DMatrix(cbind(data.matrix(bg_dou),cate_pred_dou_na))
#dtest=xgb.DMatrix(new_features_test)
params=list(
        objective='reg:linear',
        subsample=0.9,
        colsample_bytree=0.8,
        eta=0.05,
        max_depth=1
)

xgb.cv(nfold=10,data=dtrain,params = params,nround=300, print_every_n = 50)
```

```
## [1]  train-rmse:2.344140+0.005558    test-rmse:2.344033+0.052417
## [51] train-rmse:0.633966+0.003240    test-rmse:0.645640+0.029830
## [101]    train-rmse:0.594008+0.002820    test-rmse:0.617642+0.030568
## [151]    train-rmse:0.583770+0.002661    test-rmse:0.620125+0.030095
## [201]    train-rmse:0.575551+0.002612    test-rmse:0.622352+0.028966
## [251]    train-rmse:0.568277+0.002556    test-rmse:0.624437+0.027284
## [300]    train-rmse:0.561870+0.002574    test-rmse:0.627034+0.026839
```

```
model=xgb.train(data=dtrain,params=params,nrounds=100)
imp=xgb.importance(model=model)
xgb.plot.importance(importance_matrix = imp)
```




```
sub=predict(model,dtest)
```

Model 2: 9-year-old kid categorical and KMeans

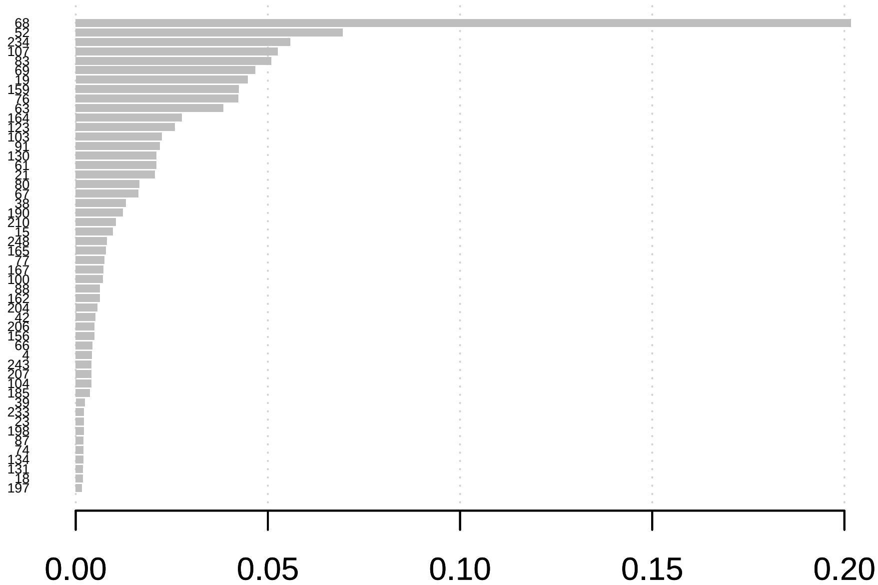
```
######## K-MEANS func
new_features_kmeans <- function(data, K){
  ## Return cluster.id
  kmeans_results <- kmeans(data, centers = K, iter.max = 500)
  return(kmeans_results$cluster)
}
####################
generate_new_f_kmeans <- function(data, cluster.id){
  new_data_lm <- data.frame(CLUS = cluster.id,
```

```r
                            Records = data)
  new_data_lm_done <- ddply((new_data_lm), .(CLUS), colMeans)
  return(DATA = t(new_data_lm_done[,-1]))
}

bg_dou <- bg[,colnames(bgtrain_dou)]

gp5pred_dou <- bg_dou
gp5pred_dou <- matrix(unlist(gp5pred_dou), nrow(gp5pred_dou))
## 1 for new column, 2 for new categorical features
gp5pred_dou_RMNA <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_pred_dou_na <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[2]]})

my.text <- "^([a-z]{1,3}5)"
indices <- grepl(my.text, colnames(bgtrain_inter))
mat <- bgtrain_inter[,indices]

#grp5_cate_422 <- bgtrain_inter
grp5_cate_422 <- mat
grp5_cate_422[is.na(grp5_cate_422)] <- -32767
for (iter in 1:ncol(grp5_cate_422 )){
  grp5_cate_422[,iter] <- as.factor(grp5_cate_422[,iter])
}
aaa <- apply(grp5_cate_422, 2, createDummyFeatures)

aaa1 <- matrix(ncol = 0, nrow = nrow(grp5_cate_422))
for (iter in 1:length(aaa)){
  aaa1 <- cbind(aaa1, aaa[[iter]][,2:ncol(aaa[[iter]])])
}

cs.id <- new_features_kmeans(t(aaa1), K = 250)
new_features_train <- generate_new_f_kmeans(t(aaa1), cs.id)

## bg_dou_cate begin
bg_inter <- bg[, colnames(mat)]
grp5_pre_cate_422 <- bg_inter
grp5_pre_cate_422[is.na(grp5_pre_cate_422)] <- -32767
grp5_pre_cate_422 <- data.frame(grp5_pre_cate_422)

for (iter in 1:ncol(grp5_pre_cate_422 )){
  set1 <- colnames(aaa[[iter]])
  grp5_pre_cate_422[!(grp5_pre_cate_422[,iter] %in% as.numeric(set1)),iter] <- -32767
  grp5_pre_cate_422[,iter] <- as.factor(grp5_pre_cate_422[,iter])
}

aaa_pre <- apply(grp5_pre_cate_422, 2, createDummyFeatures)
## This is a list
aaa_pre1 <- matrix(ncol = 0, nrow = nrow(grp5_pre_cate_422))
for (iter in 1:length(aaa_pre)){
  aaa_temp <- aaa_pre[[iter]][,colnames(aaa[[iter]])]
  aaa_temp <- aaa_temp[,2:ncol(aaa[[iter]])]
  aaa_pre1 <- cbind(aaa_pre1, aaa_temp)
}
```

```
## k-means features
new_features_test <- generate_new_f_kmeans(t(aaa_pre1), cs.id)

## XGBOOST FEATURES
dtrain=xgb.DMatrix(data.matrix(new_features_train),label=train$gpa[which(!is.na(train$gpa))])
dtest=xgb.DMatrix(data.matrix(new_features_test))
#dim(dtrain)
#dim(dtest)

params=list(
  objective='reg:linear',
  subsample=0.9,
  colsample_bytree=1,  ## This can be modified
  eta=0.05,
  max_depth=1
)

xgb.cv(nfold=10,data = dtrain, params = params, nround = 300, print_every_n = 50)
```

```
## [1]  train-rmse:2.344083+0.006263    test-rmse:2.343037+0.059368
## [51] train-rmse:0.637015+0.003758    test-rmse:0.650474+0.032155
## [101]    train-rmse:0.594667+0.004500    test-rmse:0.619267+0.037439
## [151]    train-rmse:0.583304+0.004524    test-rmse:0.618088+0.038732
## [201]    train-rmse:0.574586+0.004639    test-rmse:0.618981+0.039849
## [251]    train-rmse:0.567053+0.004635    test-rmse:0.619898+0.040609
## [300]    train-rmse:0.560503+0.004644    test-rmse:0.621161+0.041251
```

```
model8=xgb.train(data=dtrain,params=params,nrounds=200)
imp=xgb.importance(model=model8)
xgb.plot.importance(importance_matrix = imp)
```




```r
cate_result <- predict(model8,dtest)
```

Bagging the two results, that is, given equal weights to the two results and get our final prediction

```r
cate_result <- predict(model8,dtest)
nie <- read_csv('prediction1.csv')
yyy <- nie
yyy$gpa <- 0.5 * nie$gpa + 0.5 * cate_result
hist(yyy$gpa)
write_csv(yyy,'prediction.csv')
```

Step 3: Regression and Prediction on Grit & Material Hardship

Grit: Categorical variables related to 9-year-old kids

```r
naid<- which(is.na(train$grit))
bgtrain<- bg[setdiff(train$challengeID,train$challengeID[naid]),]
n <- ncol(bgtrain)
na_count<- rep(NA, n)
for(i in 1:n){
        na_count[i]<- sum(is.na(bgtrain[,i]))
```

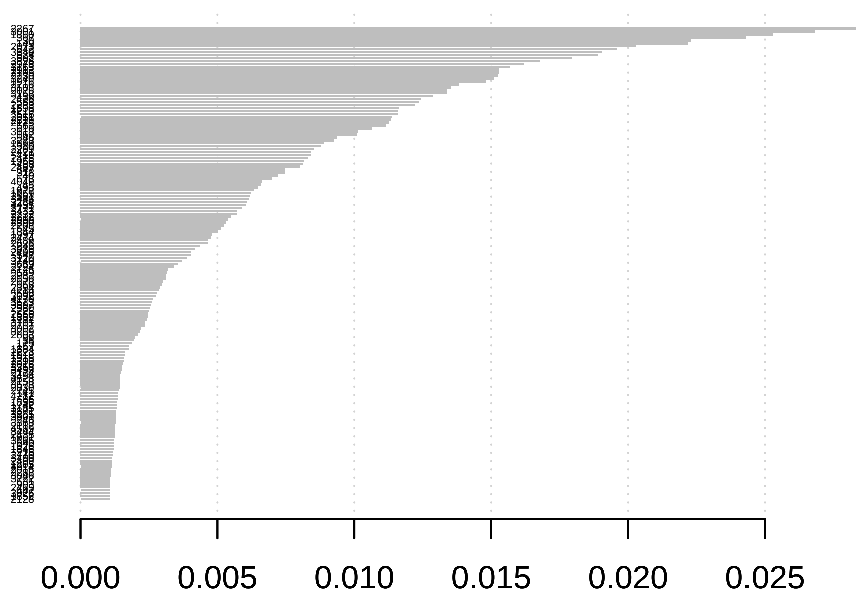
```
}
na_index<- c(1:n)[na_count>=0.8*nrow(bgtrain)]
bgtrain<- bgtrain[,-na_index]

namelist=names(bgtrain)
for (f in namelist) {
        if(nrow(unique(bgtrain[f]))==1) {
                bgtrain[f]=NULL
        }
}
n<- ncol(bgtrain)
t<- rep(NA, n)
for(i in 1:n){
        t[i]<- typeof(bgtrain[[1,i]])
}
char_index<- c(1:n)[t=="character"]
inter_index<- c(1:n)[t=="integer"]
dou_index<- c(1:n)[t=="double"]

bgtrain_char<- bgtrain[, char_index]
bgtrain_inter<- bgtrain[, inter_index]
bgtrain_dou<- bgtrain[, dou_index]

fac_num<- as.numeric(apply(bgtrain_inter, 2, function(vec)
{return(length(table(vec)))} ))
fac_index<- which(fac_num<=15)
bgtrain_dou<- cbind(bgtrain_inter[,-fac_index], bgtrain_dou)
bgtrain_inter<- bgtrain_inter[,fac_index]

bgtrain_dou<- data.frame(bgtrain_dou)
bgtrain_inter<- data.frame(bgtrain_inter)

### fill NA and create new data
fill_each_column <- function(each_col){
        na_label <- is.na(each_col)
        if(sum(na_label) > 0){
                cate_col <- ifelse(na_label == T, 1, 0) ## T = 1, is NA
                fill <- median(each_col, na.rm = T)
                each_col[na_label] <- fill
                return(list(NEW_COLUMN = each_col, NEW_CATE = cate_col))
        }else{
                ## no NA in a column
                return(list(NEW_COLUMN = rep(0,length(each_col)), NEW_CATE = rep(0,length(each_col))))
        }
}
## INPUT data= continuous dataframe
gp5data_dou <- bgtrain_dou
gp5data_dou <- matrix(unlist(gp5data_dou), nrow(gp5data_dou))
## 1 for new column, 2 for new categorical features
gp5data_dou_RMNA <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_dou_na <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[2]]})

## remove NON-NA columns
```

```
non_na_dou <- colSums(cate_dou_na) == 0
cate_dou_na <- cate_dou_na[,!non_na_dou]

bg_dou <- bg[,colnames(bgtrain_dou)]
gp5pred_dou <- bg_dou
gp5pred_dou <- matrix(unlist(gp5pred_dou), nrow(gp5pred_dou))
## 1 for new column, 2 for new categorical features
gp5pred_dou_RMNA <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_pred_dou_na <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[2]]})

my.text <- "^(k5|p5|o5|n5|hv5)"
indices <- grepl(my.text, colnames(bgtrain_inter))
mat <- bgtrain_inter[,indices]

grp5_cate_422 <- mat
grp5_cate_422[is.na(grp5_cate_422)] <- -32767
for (iter in 1:ncol(grp5_cate_422 )){
        grp5_cate_422[,iter] <- as.factor(grp5_cate_422[,iter])
}
aaa <- apply(grp5_cate_422, 2, createDummyFeatures)

aaa1 <- matrix(ncol = 0, nrow = nrow(grp5_cate_422))
for (iter in 1:length(aaa)){
        aaa1 <- cbind(aaa1, aaa[[iter]][,2:ncol(aaa[[iter]])])
}

bg_inter <- bg[, colnames(mat)]
grp5_pre_cate_422 <- bg_inter
grp5_pre_cate_422[is.na(grp5_pre_cate_422)] <- -32767
grp5_pre_cate_422 <- data.frame(grp5_pre_cate_422)

for (iter in 1:ncol(grp5_pre_cate_422 )){
        set1 <- colnames(aaa[[iter]])
        grp5_pre_cate_422[!(grp5_pre_cate_422[,iter] %in% as.numeric(set1)),iter] <- -32767
        grp5_pre_cate_422[,iter] <- as.factor(grp5_pre_cate_422[,iter])
}

aaa_pre <- apply(grp5_pre_cate_422, 2, createDummyFeatures) ## This is a list
aaa_pre1 <- matrix(ncol = 0, nrow = nrow(grp5_pre_cate_422))
for (iter in 1:length(aaa_pre)){
        aaa_temp <- aaa_pre[[iter]][,colnames(aaa[[iter]])]
        aaa_temp <- aaa_temp[,2:ncol(aaa[[iter]])]
        aaa_pre1 <- cbind(aaa_pre1, aaa_temp)
}
## bg_dou_cate end

dtrain=xgb.DMatrix(cbind(data.matrix(aaa1),cate_dou_na),label=train$grit[which(!is.na(train$grit))])
dtest=xgb.DMatrix(cbind(data.matrix(aaa_pre1),cate_pred_dou_na))
params=list(
        objective='reg:linear',
        subsample=0.9,
        colsample_bytree=0.6,
        eta=0.05,
```

```
        max_depth=1
)

xgb.cv(nfold=10,data=dtrain,params = params,nround=600,print_every_n = 100)
```

```
## [1]  train-rmse:2.823180+0.003997    test-rmse:2.822784+0.037435
## [101]    train-rmse:0.459673+0.002099    test-rmse:0.475749+0.021358
## [201]    train-rmse:0.444076+0.001846    test-rmse:0.471292+0.020297
## [301]    train-rmse:0.432404+0.001752    test-rmse:0.468903+0.019117
## [401]    train-rmse:0.422742+0.001636    test-rmse:0.468160+0.019136
## [501]    train-rmse:0.414342+0.001579    test-rmse:0.467944+0.018688
## [600]    train-rmse:0.406986+0.001585    test-rmse:0.468653+0.018982
```

```
model8=xgb.train(data=dtrain,params=params,nrounds=500)
imp=xgb.importance(model=model8)
xgb.plot.importance(importance_matrix = imp)
```




```
sub=predict(model8,dtest)
```

Material Hardship: Mother's categorical variables, and use Kmeans on categorical dummy variables

```
naid<- which(is.na(train$materialHardship))
bgtrain<- bg[setdiff(train$challengeID,train$challengeID[naid]),]
n <- ncol(bgtrain)
```

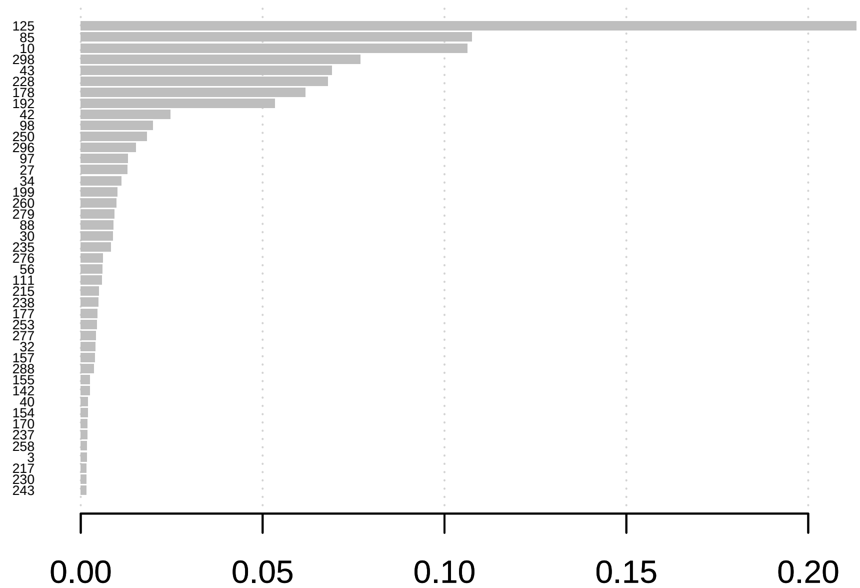
```
na_count<- rep(NA, n)
for(i in 1:n){
        na_count[i]<- sum(is.na(bgtrain[,i]))
}
na_index<- c(1:n)[na_count>=0.8*nrow(bgtrain)]
bgtrain<- bgtrain[,-na_index]

# Delete the variables that have the same value
namelist=names(bgtrain)
for (f in namelist) {
        if(nrow(unique(bgtrain[f]))==1) {
                bgtrain[f]=NULL
        }
}

n<- ncol(bgtrain)
t<- rep(NA, n)
for(i in 1:n){
        t[i]<- typeof(bgtrain[[1,i]])
}

char_index<- c(1:n)[t=="character"]
inter_index<- c(1:n)[t=="integer"]
dou_index<- c(1:n)[t=="double"]
bgtrain_char<- bgtrain[, char_index]
bgtrain_inter<- bgtrain[, inter_index]
bgtrain_dou<- bgtrain[, dou_index]
fac_index<- which(fac_num<=15)
bgtrain_dou<- cbind(bgtrain_inter[,-fac_index], bgtrain_dou)
bgtrain_inter<- bgtrain_inter[,fac_index]
bgtrain_dou<- data.frame(bgtrain_dou)
bgtrain_inter<- data.frame(bgtrain_inter)
### fill NA and create new data
fill_each_column <- function(each_col){
        na_label <- is.na(each_col)
        if(sum(na_label) > 0){
                cate_col <- ifelse(na_label == T, 1, 0) ## T = 1, is NA
                fill <- median(each_col, na.rm = T)
                each_col[na_label] <- fill
                return(list(NEW_COLUMN = each_col, NEW_CATE = cate_col))
        }else{
                ## no NA in a column
                return(list(NEW_COLUMN = rep(0,length(each_col)), NEW_CATE = rep(0,length(each_col))))
        }


}
my.text <- "^(m)"
indices <- grepl(my.text, colnames(bgtrain_dou))
mat <- bgtrain_dou[,indices]

## INPUT data= continuous dataframe
gp5data_dou <- mat
```

```r
gp5data_dou <- matrix(unlist(gp5data_dou), nrow(gp5data_dou))
## 1 for new column, 2 for new categorical features
gp5data_dou_RMNA <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_dou_na <- apply(gp5data_dou, 2, function(col){fill_each_column(col)[[2]]})
## remove NON-NA columns
non_na_dou <- colSums(cate_dou_na) == 0
cate_dou_na <- cate_dou_na[,!non_na_dou]

bg_dou <- bg[,colnames(mat)]
gp5pred_dou <- bg_dou
gp5pred_dou <- matrix(unlist(gp5pred_dou), nrow(gp5pred_dou))
## 1 for new column, 2 for new categorical features
gp5pred_dou_RMNA <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[1]]})
cate_pred_dou_na <- apply(gp5pred_dou, 2, function(col){fill_each_column(col)[[2]]})
dtrain=xgb.DMatrix(data.matrix(mat),label=train$materialHardship[which(!is.na(train$materialHardship))]
dtest=xgb.DMatrix(data.matrix(bg_dou))
params=list(
        objective='reg:linear',
        subsample=0.9,
        colsample_bytree=0.8,
        eta=0.05,
        max_depth=1
)

model=xgb.train(data=dtrain,params=params,nrounds=200)
imp=xgb.importance(model=model)
my.text <- "^(m)"
indices <- grepl(my.text, colnames(bgtrain_inter))
mat <- bgtrain_inter[,indices]

grp5_cate_422 <- mat
grp5_cate_422[is.na(grp5_cate_422)] <- -32767
for (iter in 1:ncol(grp5_cate_422 )){
        grp5_cate_422[,iter] <- as.factor(grp5_cate_422[,iter])
}
aaa <- apply(grp5_cate_422, 2, createDummyFeatures)

aaa1 <- matrix(ncol = 0, nrow = nrow(grp5_cate_422))
for (iter in 1:length(aaa)){
        aaa1 <- cbind(aaa1, aaa[[iter]][,2:ncol(aaa[[iter]])])
}
bg_inter <- bg[, colnames(mat)]
grp5_pre_cate_422 <- bg_inter
grp5_pre_cate_422[is.na(grp5_pre_cate_422)] <- -32767
grp5_pre_cate_422 <- data.frame(grp5_pre_cate_422)
for (iter in 1:ncol(grp5_pre_cate_422 )){
        set1 <- colnames(aaa[[iter]])
        grp5_pre_cate_422[!(grp5_pre_cate_422[,iter] %in% as.numeric(set1)),iter] <- -32767
        grp5_pre_cate_422[,iter] <- as.factor(grp5_pre_cate_422[,iter])
}
####

aaa_pre <- apply(grp5_pre_cate_422, 2, createDummyFeatures) ## This is a list
```

```r
aaa_pre1 <- matrix(ncol = 0, nrow = nrow(grp5_pre_cate_422))
for (iter in 1:length(aaa_pre)){
        aaa_temp <- aaa_pre[[iter]][,colnames(aaa[[iter]])]
        aaa_temp <- aaa_temp[,2:ncol(aaa[[iter]])]
        aaa_pre1 <- cbind(aaa_pre1, aaa_temp)
}
## bg_dou_cate end

new_features_kmeans <- function(data, K){
        ## Return cluster.id
        kmeans_results <- kmeans(t(data), centers = K, iter.max = 500)
        return(kmeans_results$cluster)
}

generate_new_f_kmeans <- function(data, cluster.id){
        new_data_lm <- data.frame(CLUS = cluster.id,
                                  Records = t(data))
        new_data_lm_done <- ddply((new_data_lm), .(CLUS), colMeans)
        return(DATA = t(new_data_lm_done[,-1]))
}

cs.id <- new_features_kmeans(data = aaa1, K =300)
new_features_train <- generate_new_f_kmeans(aaa1, cs.id)
new_features_test <- generate_new_f_kmeans(aaa_pre1, cs.id)

dtrain=xgb.DMatrix(cbind(data.matrix(new_features_train),cate_dou_na),label=train$materialHardship[whic
dtest=xgb.DMatrix(cbind(data.matrix(new_features_test),cate_pred_dou_na))
params=list(
        objective='reg:linear',
        subsample=0.9,
        colsample_bytree=0.8,
        eta=0.05,
        max_depth=1
)

xgb.cv(nfold=10,data=dtrain,params = params,nround=300, print_every_n = 50)
```

```
## [1]  train-rmse:0.407110+0.000783    test-rmse:0.407087+0.007435
## [51] train-rmse:0.144159+0.001198    test-rmse:0.146873+0.011020
## [101]    train-rmse:0.137017+0.001162    test-rmse:0.142438+0.011490
## [151]    train-rmse:0.134353+0.001118    test-rmse:0.141848+0.011476
## [201]    train-rmse:0.132350+0.001113    test-rmse:0.141836+0.011221
## [251]    train-rmse:0.130683+0.001125    test-rmse:0.141865+0.011235
## [300]    train-rmse:0.129242+0.001141    test-rmse:0.142015+0.011329
```

```r
model8=xgb.train(data=dtrain,params=params,nrounds=200)
imp=xgb.importance(model=model8)
xgb.plot.importance(importance_matrix = imp)
```



Final Results

Results

#	User	GPA ▲	Grit ▲	Material hardship ▲	Eviction ▲	Layoff ▲	Job training ▲
1	kai_niubi	0.36440 (1)	0.21997 (33)	0.02880 (44)	0.05341 (27)	0.17435 (24)	0.20224 (22)
1	ADSgrp5	0.36440 (1)	0.21292 (8)	0.02453 (2)	0.05341 (27)	0.17435 (24)	0.20224 (22)
2	zn	0.36456 (2)	0.21997 (33)	0.02880 (44)	0.05341 (27)	0.17435 (24)	0.20224 (22)
3	ovarol	0.36571 (3)	0.21812 (21)	0.02481 (6)	0.05660 (35)	0.17422 (22)	0.20225 (23)
4	OldDriver.ffc	0.36630 (4)	0.21251 (7)	0.02431 (1)	0.05273 (19)	0.17187 (8)	0.20060 (11)
5	kai_666	0.36733 (5)	0.21997 (33)	0.02880 (44)	0.05341 (27)	0.17435 (24)	0.20224 (22)
6	wjlei1990	0.36742 (6)	0.21176 (4)	0.02431 (1)	0.05233 (16)	0.17149 (7)	0.20251 (26)
7	kgenova	0.36837 (7)	0.21502 (16)	0.02461 (3)	0.05112 (7)	0.17336 (14)	0.19382 (1)
8	malte	0.36912 (8)	0.21569 (17)	0.02484 (7)	0.05208 (11)	0.17461 (25)	0.20079 (13)
9	yjpeng	0.36974 (9)	0.21450 (13)	0.02491 (9)	0.05218 (12)	0.17120 (5)	0.20499 (29)

Figure 8: Leader Board