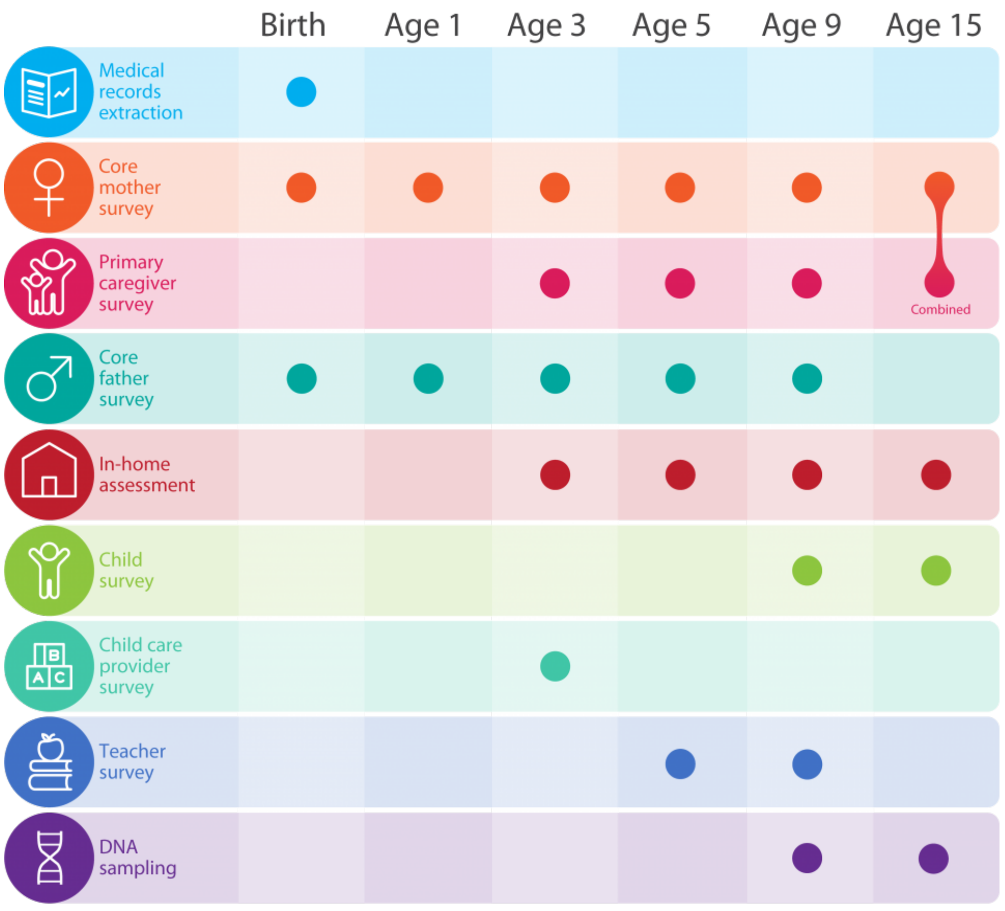
Fragile Families Challenge

Yue Gao

2017/04/27

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.

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.



These data have been used in hundreds of scientific papers and dozens of dissertations, and insight from these studies are routinely shared with policy makers around the world through the Future of Children, which is jointly published by Princeton University and Brookings Institution.

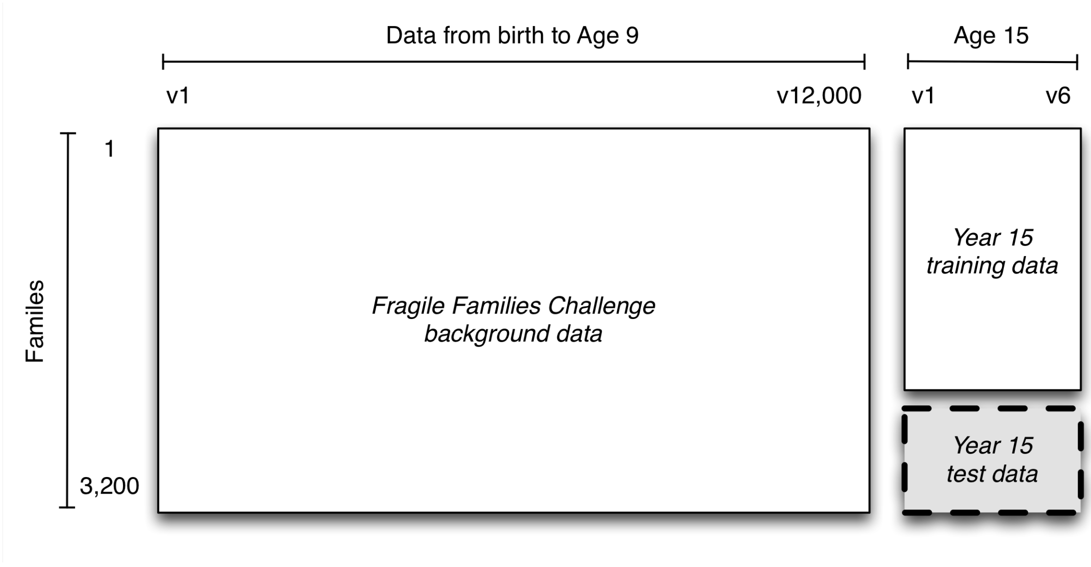
The Fragile Families Challenge is our attempt to create a new way of doing social research, one that is much more open to the talents and efforts of everyone. We expect that by combining ideas from social science and data science, we can—together—help address important scientific and social problems. And, we expect that through a mass collaboration we will accomplish things that none of us could accomplish individually.

The Challenge Problem:

Given all the background data from birth to year 9 and some training data from year 15, how well can you infer six key outcomes in the year 15 test data?

Continuous outcomes: GPA, Grit, Material hardship

Binary outcomes: Housing eviction, Layoff of a caregiver, Job training for a caregiver



Step 1: Data Cleansing

Our Assumption: Age 9 data has already included all the information from age 0-5

1.1 Extract the age 9 data from codebook and background file

```r
library(data.table)
library(stringr)
library(dmm)
library(Hmisc)

setwd("~/GitHub/Spr2017-proj5-grp3/")
load("../data/background.RData")
source("../lib/helper_data.R")
#background=read.csv("~/Documents/FFChallenge/background.csv",header=TRUE)

features<-colnames(background)
```

```
#create codebook
codebooks<-c("child","mom","dad","teacher")
data.info<-vector()

for (i in codebooks){
feat.table<-read.csv(paste0(".../data/codebook/ff_",i,"_cb9.csv"),header=FALS
E)
feat.table<-feat.table[,-1]
feat.table<-feat.table[-1,]
feat.table<-cbind(rep(i,nrow(feat.table)),feat.table)
data.info<-rbind(data.info,feat.table)
}

colnames(data.info)<-c("class","code","description")
data.info=as.data.frame(data.info)
featnum<-nrow(data.info)

extract.feature<-features %in% data.info$code
sum(extract.feature)

extract.data<-background[,extract.feature]
extract.data<-cbind(challengeID=background[,1],extract.data)

write.csv(extract.data,file="../data/extract_data.csv")
save(data.info, file="../data/data_info.RData")
```

1.2 Deal with missing data

Several missing value codes:

-9 Not in wave

-6 Valid skip

-2 Dont know

-1 Refuse

NA also used occasionally

Categorical feature:

Make NA a special level, NA=”-3”

Continuous feature:

Create a dummy variable indicating the missing situation of the feature

Impute the missing value with median

```r
#remove columns with missing values more than 80%
extract.data<-read.csv("../data/extract_data.csv")

ED<-divide.data(extract.data)
ED.categorical=ED[[1]]
ED.continuous=ED[[2]]

categorical=colnames(ED.categorical)

ED.factor=clean.factor(ED.continuous)

ED.continuous=ED.continuous[,!colnames(ED.continuous) %in% colnames(ED.factor
)]
ED.continuous=clean.continuous(ED.continuous)

categorical=c(categorical,colnames(ED.factor[,grep("*isna",colnames(ED.factor
))]),colnames(ED.continuous[,grep("*isna",colnames(ED.continuous))]))
#combine the data

ED.final<-cbind(ED.continuous,ED.categorical,ED.factor)

ED.final=as.data.frame(ED.final)

final.mis<-ED.final[,which(unlist(lapply(ED.final, function(x) anyNA(x))))]
missing=colnames(final.mis)

for(i in missing)
{
  ED.final[,i] = impute(ED.final[,i], fun = median)
}

write.csv(ED.final,file="../data/NAreplaced.csv")
save(categorical, file="../data/categorical.RData")
```

Step 2: Feature Selection

We used Boruta Package to conduct feature selection.It works as a wrapper algorithm around Random Forest.

Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).

Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.

At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z score than the maximum Z score of its shadow features) and constantly removes features which are deemed highly unimportant.

Finally, the algorithm stops either when all features gets confirmed or rejected or it reaches a specified limit of random forest runs.

```r
library(Boruta)

load("../doc/data_info.RData")
ED.final<-read.csv("../data/NAreplaced.csv")
#names(ED.final[,1])="challengeID"
label<-read.csv("../data/train.csv")

# label<-label[,1:2] #gpa
# label<-label[,c(1,3)] #grit
# label<-label[,c(1,4)] #materialHardship
label<-label[,c(1,7)] #jobTraining
label=na.omit(label)
Index=ED.final$challengeID %in% label$challengeID

data.train<-ED.final[Index,]
data.train<-as.data.frame(data.train)

model.features = Boruta(data.train[,-1], label$jobTraining, maxRuns = 500)

results = as.data.frame(model.features$finalDecision)
which(results!="Rejected")
Bcodes = rownames(results)[which(results!="Rejected")]
```

What are the descriptions of the features (Boruta)?

```r
des = data.frame()
#score = data.frame()
for (i in 1:length(Bcodes)){
  if(Bcodes[i] %in% data.info$code){
    des = rbind(des, as.data.frame(data.info$description[data.info$code==Bcod
es[i]]))
   #score = rbind(score, as.data.frame(imp$meanImp[which(rownames(imp)==Bcode
s[i])]))
  }
}

decision = as.data.frame(model.features$finalDecision[model.features$finalDec
ision!="Rejected"])

Bdf = cbind(Bcodes, des, decision)
colnames(Bdf) = c("Codes", "Description", "Decision")
pred="jobTraining"
write.csv(Bdf, paste0("../data/",pred,"features.csv"), row.names = F)
```

Let's have a look at some of the selected features.

```
select <- read.csv('../data/Updated_Features/gpa_features.csv',stringsAsFacto
rs = FALSE)
head(cbind(select$Codes,select$Description))

##      [,1]
## [1,] "m5j1"
## [2,] "f5a4"
## [3,] "f5g35"
## [4,] "f5i13"
## [5,] "f5j1"
## [6,] "f5j6c"
##      [,2]

## [1,] "J1. Total household income before taxes/deductions in past 12 months
"
## [2,] "A4. Relationship with child's biological mother now"

## [3,] "G35. Age when you had sexual intercourse for the first time"

## [4,] "i13. how much you earn in that job, before taxes"

## [5,] "J1. Total household income before taxes/deductions in past 12 months
"
## [6,] "J6C. Value of vehicle if sold"
```

Step 3: Model Selection

We decided to try a series of models imcluding: Linear Regression, Full tree, Pruned tree, Random Forest, Conditional inference trees, Gradient Boosting, Support Vector Machine, LM+RF, SVM+RF, LDA, KNN etc.

GPA

```
source("../lib/modelFunc.R")
load("../data/categorical.RData")
data.filtered <- read.csv('../data/NAreplaced.csv') #4242 1388
select <- read.csv('../data/Updated_Features/gpa_features.csv')
select.idx<-colnames(data.filtered) %in% as.character(select$Codes)
data.filtered <- data.filtered[,select.idx]

label <- read.csv('../data/train.csv')
label<-label[!is.na(label$gpa),]
Index<-as.numeric(rownames(data.filtered))%in% label$challengeID

data.train<-data.filtered[Index,]
data.train<-as.data.frame(data.train)
data.train<-cbind(label$gpa, data.train)
```

```
colnames(data.train)[1]<-"gpa"
cat.idx<-colnames(data.train) %in% categorical
# create training and test data set
set.seed(123)
train.index <- sample(1:nrow(data.train),800,replace = F)
train <- data.train[train.index,] #800*64
test <- data.train[-train.index,] #214*64
for(i in which(cat.idx)){
  data.train[,i]<-sapply(data.train[,i], factor)
  id <- which(!(test[,i] %in% unique(data.train[,i])))
  test[,i][id]<-sample(unique(data.train[,i]),length(id), replace = TRUE)
}

y<-train[,1]
model_selection_con(train[,-1], test, y)
```

results:

```
##                     Method    Test Error
## 1        Linear Regression  0.4082
## 2                Full tree  0.4303
## 3              Pruned tree  0.4617
## 4            Random Forest  0.3799
## 5    Conditional infe trees  0.4476
## 6        Gradient Boosting  0.3858
## 7    Support Vector Machine  0.3918
## 8                    LM+RF  0.3881
## 9                   SVM+RF  0.3824
```

Grit

results:

```
##                     Method    Test Error
## 1        Linear Regression  0.2443
## 2                Full tree  0.2609
## 3              Pruned tree  0.2587
## 4            Random Forest  0.241
## 5    Conditional infe trees  0.2577
## 6        Gradient Boosting  0.2396
## 7    Support Vector Machine  0.2539
## 8                    LM+RF  0.2401
## 9                   SVM+RF  0.2449
```

materialHardship

```
##                     Method    Test Error
## 1        Linear Regression  0.0181
## 2                Full tree  0.0199
## 3              Pruned tree  0.02
## 4            Random Forest  0.0178
## 5    Conditional infe trees  0.0201
## 6          Gradient Boosting  0.0174
## 7    Support Vector Machine  0.0195
## 8                      LM+RF  0.0195
## 9                     SVM+RF  0.0174
```

eviction

results:

```
##                      Method    Test Error
## 1                       glm  0.0653
## 2                 Full tree  0.0698
## 3               Pruned tree  0.0592
## 4             Random Forest    0.0577
## 5    Conditional infe trees  0.0592
## 6          Gradient Boosting  0.0592
## 7    Support Vector Machine  0.0592
## 8                      C5.0  0.0592
## 9                       LDA  0.0683
##10                       KNN  0.0592
```

layoff

results:

```
##                      Method    Test Error
## 1                       glm  0.2327
## 2                 Full tree  0.239
## 3               Pruned tree  0.2306
## 4             Random Forest  0.2306
## 5    Conditional infe trees  0.2306
## 6          Gradient Boosting  0.2411
## 7    Support Vector Machine  0.2306
## 8                      C5.0  0.2285
## 9                       LDA  0.2285
##10                       KNN  0.2516
```

jobTraining

results:

```
##                      Method    Test Error
## 1                       glm  0.2269
## 2                 Full tree  0.2572
## 3               Pruned tree  0.2269
## 4             Random Forest  0.2239
## 5    Conditional infe trees  0.2269
## 6          Gradient Boosting  0.2284
## 7    Support Vector Machine  0.2269
## 8                          C5.0  0.2405
## 9                           LDA  0.2315
##10                           KNN  0.2738
```

Step 4: Prediction

We decided to choose the following models:

Random Forest for GPA, eviction,job training

GBM for grit, material hardship

LDA for layoff

The test error evaluated on the predictions we submitted to the platform is:

GPA: 0.37260 (15)

Grit: 0.21490 (16)

Material Hardship: 0.02527 (15)

Eviction: 0.05660 (35)

Layoff: 0.24340 (44)

Job Training: 0.27736 (44)

This predictive modeling is not the end of the project, however. It is just the beginning. The researchers will use the models submitted to the Fragile Families Challenge to advance the scientific goals of the project, and they will publish the results in scientific journals, both individually and collectively.