Parkinson Classification Based on Demographic Information and Voice Features

Bowen Xiao

May 22, 2018

Introduction

Using demographic information and GeMAPS extracted features of voice to classify a patient diagnosed with Parkinson's disease. I started with demogrphics data (~80% negative cases), testing different classification methods (achieved ~90% accuracy), and then went on combining with voice features (~60% negative cases). Finally, I used a hierarchical regularized logistic regression and achieved 90% accuracy and 86% recall.

Also, I packed the combined data to a simple neural network (single hidden layer with some dropout, mini batch and scale adjustment), I set the estimated parameter of logistic regression with demographic data as initial weights and it turned out to be a tiny improvement (achieved ~90% accuracy and ~80% callback).

Data Preparation

Demographic dataset has 6627 records and about 84% are negative cases. I split the data into train set (80%) and test set (20%). A quick look of the data is also shown as following.

```r
setwd("~/Parkinson_Classification_Based_on_Demographic_Information_and_Voice_Features")
parkinson <- read.csv("./data/parkinson.csv")
parkinson$brain<-as.factor(parkinson$brain)
parkinson$edu<-as.factor(parkinson$edu)
parkinson$emp<-as.factor(parkinson$emp)
parkinson$gender<-as.factor(parkinson$gender)
parkinson$mar<-as.factor(parkinson$mar)
parkinson$race<-as.factor(parkinson$race)
parkinson$smoke<-as.factor(parkinson$smoke)
parkinson$diag<-as.factor(parkinson$diag)

#visulazation
round(nrow(parkinson[parkinson$diag=='FALSE',])/nrow(parkinson),2)
```

[1] 0.84

```r
library(knitr)
kable(round(prop.table(table(parkinson$brain,parkinson$diag)),2))
```

	FALSE	TRUE
FALSE	0.67	0.15
TRUE	0.01	0.02
UNK	0.16	0.00

```r
kable(round(prop.table(table(parkinson$edu,parkinson$diag)),2))
```

	FALSE	TRUE
2-year college degree	0.05	0.01
4-year college degree	0.24	0.04
Doctoral Degree	0.05	0.02
High School Diploma/GED	0.08	0.01
Master's Degree	0.13	0.04
Some college	0.20	0.02
Some graduate school	0.05	0.01
Some high school	0.02	0.00
UNK	0.01	0.00

```
kable(round(prop.table(table(parkinson$emp,parkinson$diag)),2))
```

	FALSE	TRUE
A homemaker	0.01	0.00
A student	0.16	0.00
Employment for wages	0.52	0.05
Out of work	0.03	0.00
Retired	0.02	0.07
Self-employed	0.08	0.02
Unable to work	0.01	0.02
UNK	0.00	0.00

```
kable(round(prop.table(table(parkinson$gender,parkinson$diag)),2))
```

	FALSE	TRUE
Female	0.16	0.06
Male	0.68	0.11
Prefer not to answer	0.00	0.00
UNK	0.00	0.00

```
kable(round(prop.table(table(parkinson$mar,parkinson$diag)),2))
```

	FALSE	TRUE
Divorced	0.03	0.01
Married or domestic partnership	0.36	0.13
Other	0.01	0.00
Separated	0.00	0.00
Single, never married	0.43	0.01
UNK	0.00	0.00
Widowed	0.00	0.01

```
kable(round(prop.table(table(parkinson$race,parkinson$diag)),2))
```

	FALSE	TRUE
"Black or African"	0.02	0.00
"Caribbean"	0.00	0.00

	FALSE	TRUE
"East Asian"	0.04	0.00
"Latino/Hispanic"	0.08	0.00
"Middle Eastern"	0.02	0.00
"Mixed"	0.02	0.00
"Native American"	0.00	0.00
"Other"	0.01	0.00
"Pacific Islander"	0.00	0.00
"South Asian"	0.03	0.00
"White or Caucasian"	0.58	0.15
multi	0.04	0.00
UNK	0.00	0.00

```
kable(round(prop.table(table(parkinson$smoke,parkinson$diag)),2))
```

	FALSE	TRUE
FALSE	0.52	0.11
TRUE	0.28	0.06
UNK	0.04	0.00

```
# split it into train set and test set
set.seed(123)
index=sample(1:nrow(parkinson),0.8*nrow(parkinson))
parkinson_train<-parkinson[index,]
parkinson_test<-parkinson[-index,]
```

Analysis of Demographic Information

Logistic Regression

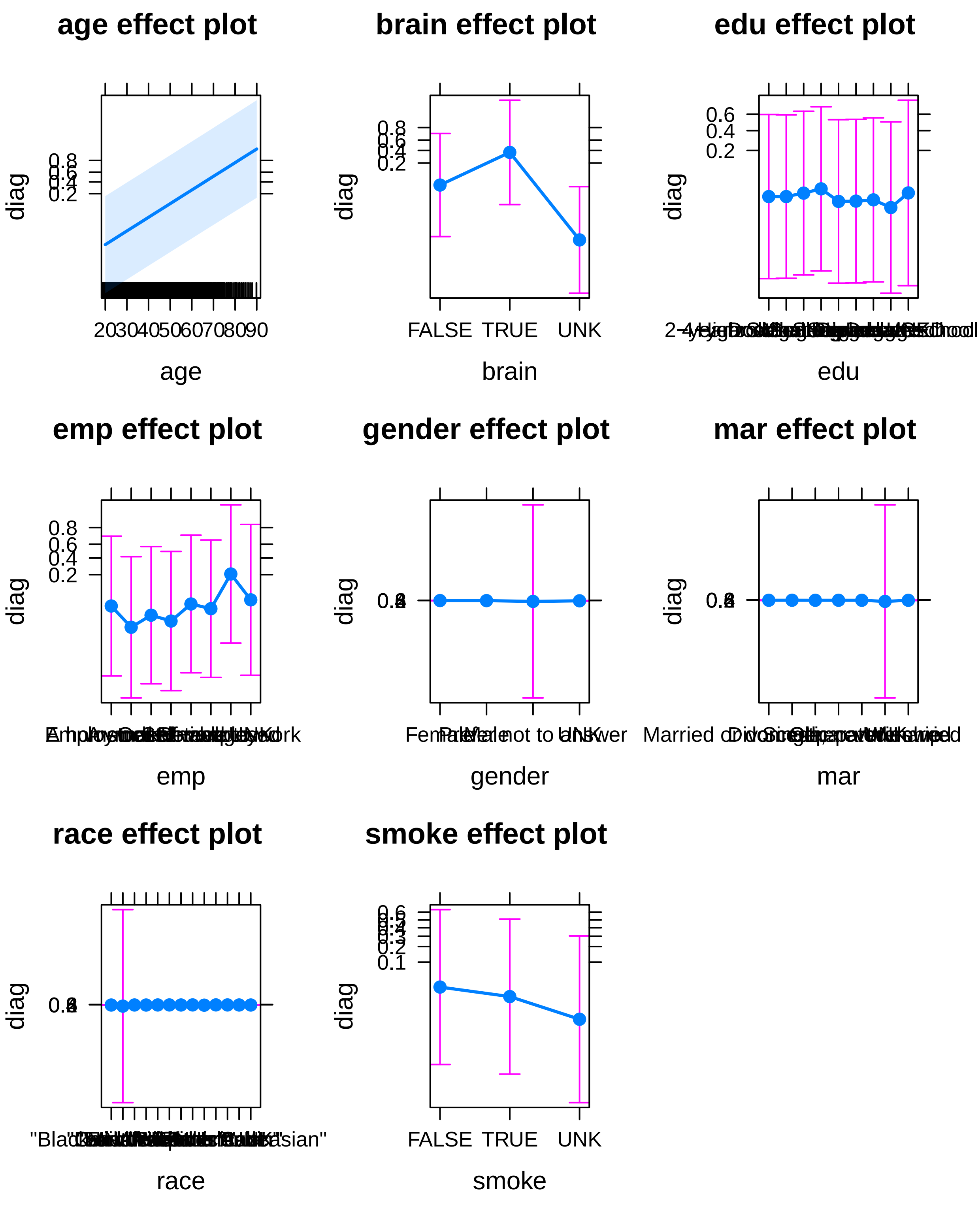
```r
model1<-glm(diag~.,data=parkinson_train,family=binomial(link='logit'))
y_1<-predict.glm(model1,newdata = parkinson_test,type='response')
y_1=ifelse(y_1>=0.5,'TRUE','FALSE')
A1<-mean(y_1==parkinson_test$diag)
R1<-mean(y_1[parkinson_test$diag=='TRUE']==parkinson_test[parkinson_test$diag=='TRUE',]$diag)
```

```r
library(effects)
```

```
## Loading required package: carData
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```r
plot(allEffects(model1))
```



As is shown, `gender` and `race` seem not to be significant predictor.

SVM

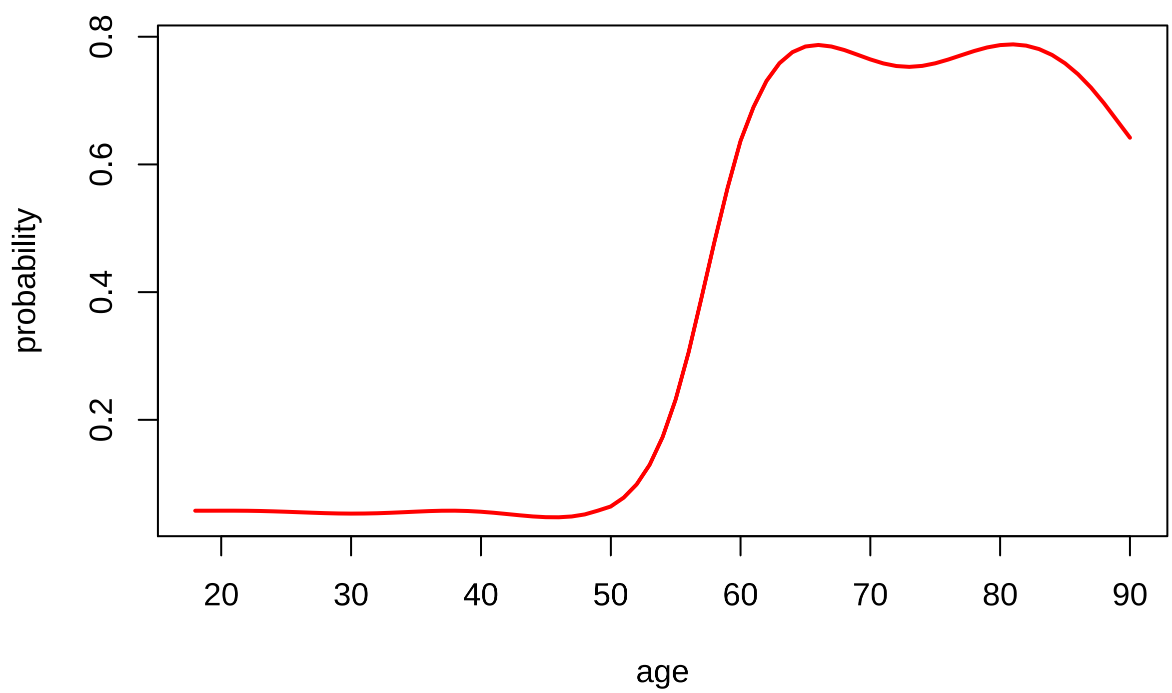
```
library(e1071)
svmfit<-svm(diag~.,data=parkinson_train,kernel="radial")
svmpred<-predict(svmfit,newdata = parkinson_test)
A2<-mean(svmpred==parkinson_test$diag)
R2<-mean(svmpred[parkinson_test$diag=='TRUE']==parkinson_test[parkinson_test$diag=='TRUE',]$diag)
```

A simple model with age can achieve 89% accuracy.

```
svmage<-svm(diag~age,data=parkinson_train,kernel="radial",probability=TRUE)
ppred<-predict(svmage,newdata = parkinson_test)
round(mean(ppred==parkinson_test$diag),2)
```

```
## [1] 0.89
```

```
pred <- predict(svmage, parkinson_train, decision.values = TRUE, probability = TRUE)
plot(parkinson_train$age[order(parkinson_train$age)],
     attr(pred, "probabilities")[,2][order(parkinson_train$age)],
     xlab='age',type='l',ylab='probability',col=2,lwd=2)
```



Naive Bayes Network

```
library(mlbench)
naive <- naiveBayes(diag ~ ., data = parkinson_train)
y_naive<-predict(naive,newdata = parkinson_test)
A3<-mean(y_naive==parkinson_test$diag)
R3<-mean(y_naive[parkinson_test$diag=='TRUE']==parkinson_test[parkinson_test$diag=='TRUE',]$diag)
```

For example, the marginal distribution of `gender` is shown as following.

```
kable(round(naive$tables$gender[,c(1,2)],2))
```

	Female	Male
FALSE	0.18	0.81
TRUE	0.35	0.65

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
fit_rf<-randomForest(diag~.,data = parkinson_train)
rfpred<-predict(fit_rf,newdata = parkinson_test)
A4<-mean(rfpred==parkinson_test$diag)
R4<-mean(rfpred[parkinson_test$diag=='TRUE']==parkinson_test[parkinson_test$diag=='TRUE',]$diag)
```

Even a small tree can have a high accuracy.

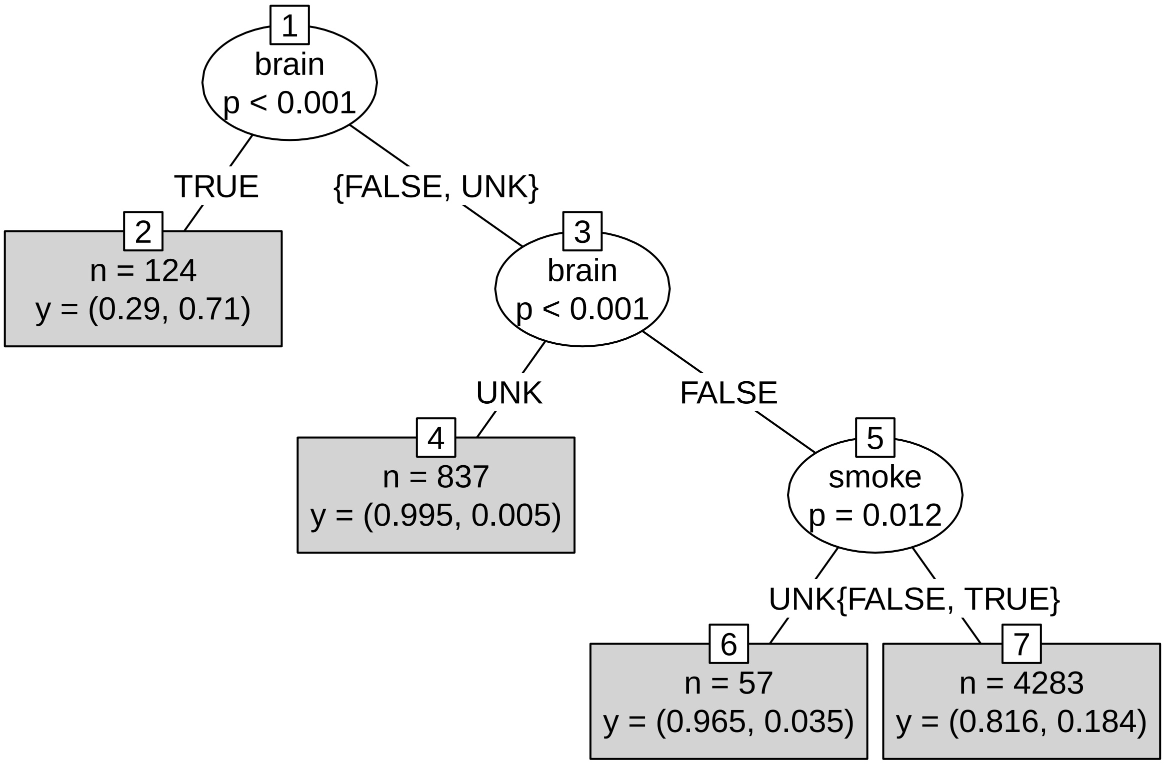
```
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
x <- ctree(diag~brain+smoke, data=parkinson_train)
xpred<-predict(x,newdata = parkinson_test)
round(mean(xpred==parkinson_test$diag),2)
```

```
## [1] 0.86
```

```r
plot(x, type="simple")
```



XGBoost

```r
library(xgboost)
data_train<-model.matrix(~.+0,data = parkinson_train[,1:8])
data_test<-model.matrix(~.+0,data = parkinson_test[,1:8])
dtrain <- xgb.DMatrix(data = data_train,label = ifelse(parkinson_train$diag=='TRUE',1,0))
dtest <- xgb.DMatrix(data = data_test,label = ifelse(parkinson_test$diag=='TRUE',1,0))
params <- list(booster = "gbtree", objective = "binary:logistic", eta=0.3, gamma=0,
               max_depth=6, min_child_weight=1, subsample=1, colsample_bytree=1)
xgbcv <- xgb.cv( params = params, data = dtrain, nrounds = 100, nfold = 5, showsd = T,
                 stratified = T, print.every.n = 10, early.stop.round = 20, maximize = F)
```

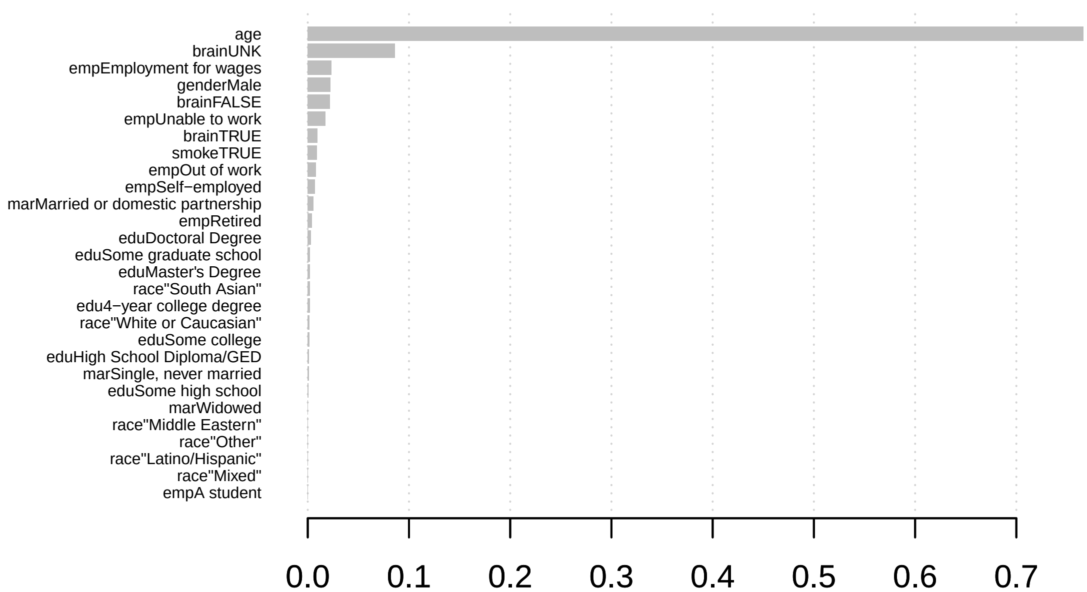
```
## Warning: 'print.every.n' is deprecated.
## Use 'print_every_n' instead.
## See help("Deprecated") and help("xgboost-deprecated").

## Warning: 'early.stop.round' is deprecated.
## Use 'early_stopping_rounds' instead.
## See help("Deprecated") and help("xgboost-deprecated").

## [1]  train-error:0.078900+0.001684   test-error:0.090361+0.009674
## Multiple eval metrics are present. Will use test_error for early stopping.
```

```
## Will train until test_error hasn't improved in 20 rounds.
##
## [11] train-error:0.073760+0.001726   test-error:0.082062+0.011808
## [21] train-error:0.070081+0.002628   test-error:0.081684+0.010094
## [31] train-error:0.067158+0.002895   test-error:0.083383+0.010256
## Stopping. Best iteration:
## [19] train-error:0.071496+0.002523   test-error:0.081307+0.010657
```

```r
fit_xgb<-xgb.train(data = dtrain, max_depth = 6, eta = 0.3, nthread = 2, nrounds = 11,
                   objective = "binary:logistic")
xgpred<-predict(fit_xgb,newdata = dtest)
A5<-mean(ifelse(xgpred<=0.5,0,1)==ifelse(parkinson_test$diag=='TRUE',1,0))
R5<-mean(ifelse(xgpred<=0.5,0,1)[parkinson_test$diag=='TRUE']==
           ifelse(parkinson_test$diag=='TRUE',1,0)[parkinson_test$diag=='TRUE'])
```

Importance of each predictor is shown as following.

```r
mat <- xgb.importance(feature_names = colnames(data_train),model=fit_xgb)
xgb.plot.importance (importance_matrix = mat)
```



Summary

Comparison of the 5 methods is shown as following.

```r
kable(data.frame(method=c('logistic','SVM','Naive Bayes','Random Forest','XGBoost'),
                 Accuracy=round(c(A1,A2,A3,A4,A5),2),
                 Recall=round(c(R1,R2,R3,R4,R5),2)))
```

method	Accuracy	Recall
logistic	0.91	0.68
SVM	0.91	0.64
Naive Bayes	0.91	0.81
Random Forest	0.92	0.74
XGBoost	0.91	0.75

Combining with Voice Features

```
setwd("~/Parkinson_Classification_Based_on_Demographic_Information_and_Voice_Features")
train <- read.csv("./src/R/train.csv", header=FALSE)
test <- read.csv("./src/R/test.csv", header=FALSE)
mean(train[,1]==1)
```

```
## [1] 0.6270381
```

Baseline - Logistic Regression Based on Demographic Information Or Voice Features

Combined dataset is decoded with dummy variables. There are 50107 records and 102 features (62 voice features).

```
model_b1<-glm(V1~.,data=train[,1:41],family=binomial(link='logit'))
y_b1<-predict.glm(model_b1,newdata = test,type='response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
y_b1=ifelse(y_b1>=0.5,1,0)
(A_b1<-mean(y_b1==test$V1))
```

```
## [1] 0.8909555
```

```
(R_b1<-mean(y_b1[test$V1==0]==test[test$V1==0,]$V1))
```

```
## [1] 0.7861322
```

```
model_b2<-glm(V1~.,data=train[,c(1,42:103)],family=binomial(link='logit'))
y_b2<-predict.glm(model_b2,newdata = test,type='response')
y_b2=ifelse(y_b2>=0.5,1,0)
(A_b2<-mean(y_b2==test$V1))
```

```
## [1] 0.7190069
```

```
(R_b2<-mean(y_b2[test$V1==0]==test[test$V1==0,]$V1))
```

```
## [1] 0.476273
```

Logistic Regression with Regularization (Lasso)

Penalty parameter λ is chosed based on cross validation.

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-16
```

```
lasso_cv<-cv.glmnet(x=as.matrix(train[,-1]),y=train[,1],alpha = 1,family="binomial")
model2<-glmnet(x=as.matrix(train[,-1]),y=train[,1],alpha = 1,family="binomial",
                     lambda = lasso_cv$lambda.min)
(A<-mean(predict(model2,newx=as.matrix(test[,-1]),type="class")==test[,1]))
```

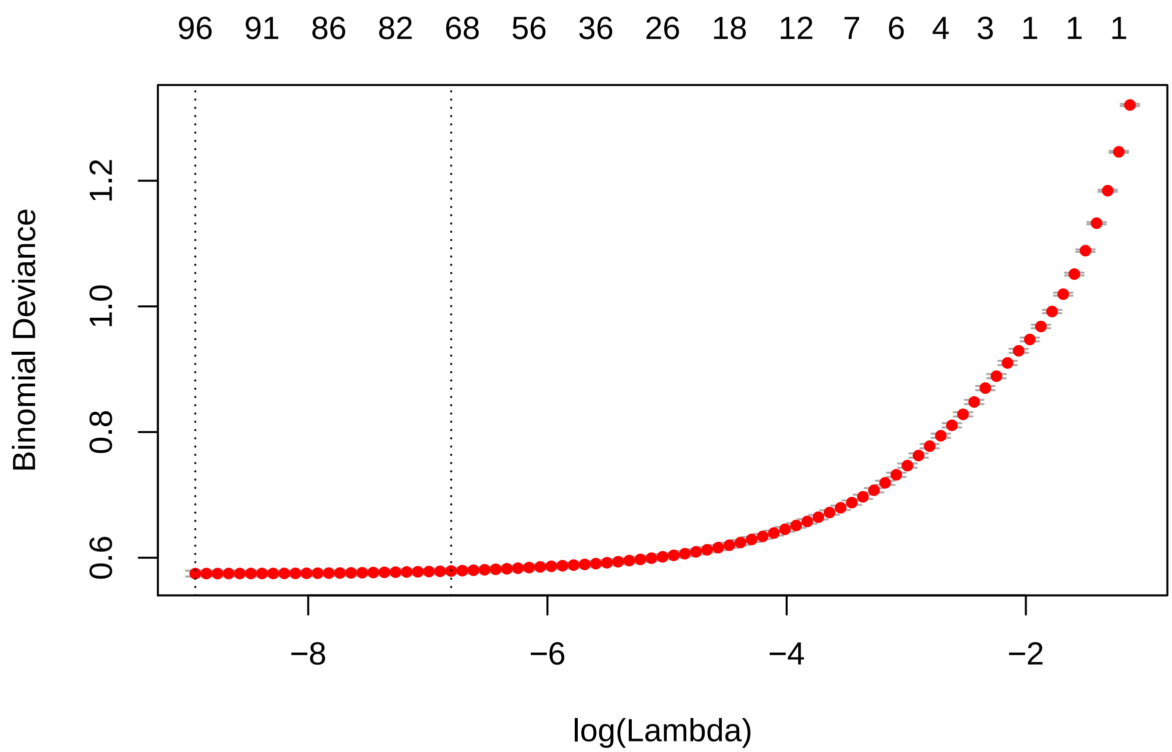
```
## [1] 0.8926319
```

```r
(R<-mean(predict(model2,newx=as.matrix(test[,-1]),type="class")[test[,1]==0]==test[test[,1]==0,1]))
```

```
## [1] 0.7874323
```

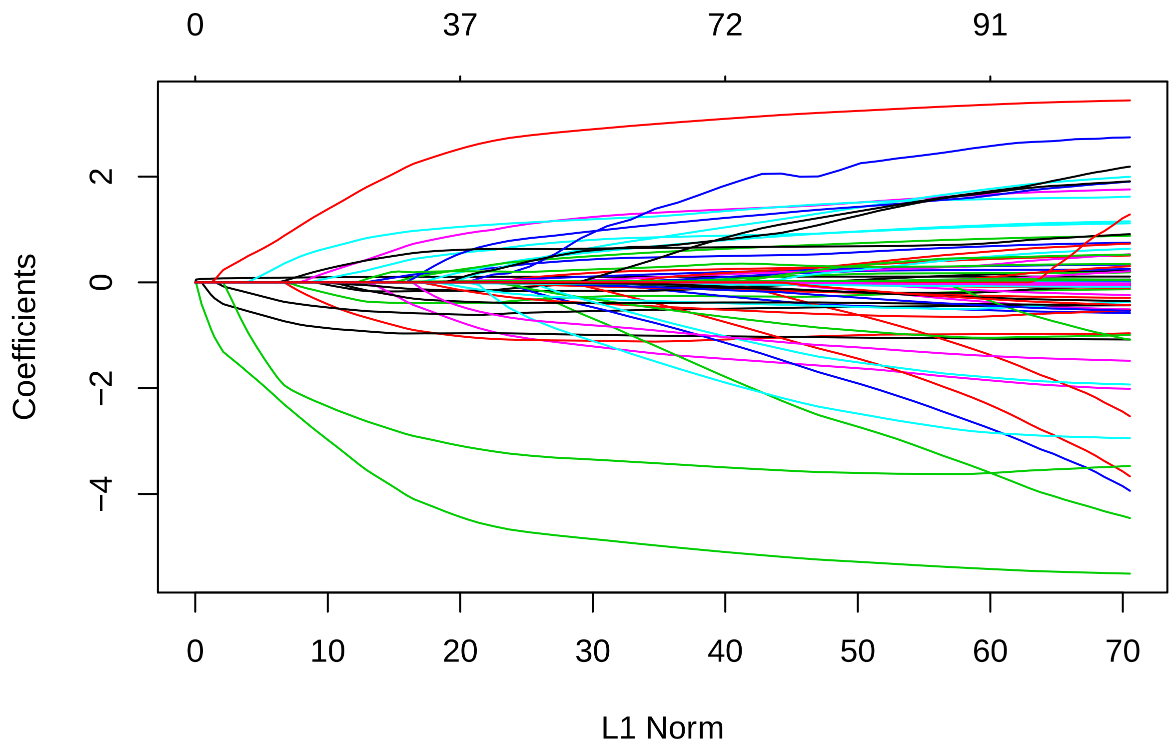
Variable selection can be shown as following.

```r
plot(lasso_cv)
```



Regularization paths can be shown as following.

```r
p_lasso<-glmnet(x=as.matrix(train[,-1]),y=train[,1],alpha = 1,family="binomial")
plot(p_lasso)
```



Final Model - Hierarchical Regularized Logistic Regression

Regularized logistic regression fails to make a big improvement. One issue is that messing up demographic information and voice features may not be wise.

Finally, I am thinking in a hierarchical framework. I am going to group patients by their age, which is shown to be a strong demographic predictor. And I build 3 regularized logistic regression models with other covariates (both demographics and voice features) accordingly.

```
group.train<-c()
group.test<-c()

for (i in 1:nrow(train)){
  if (train[i,]$V2<=55) group.train<-c(group.train,1)
  if (train[i,]$V2>55&train[i,]$V2<=68) group.train<-c(group.train,2)
  if (train[i,]$V2>68) group.train<-c(group.train,3)
}

for (i in 1:nrow(test)){
  if (test[i,]$V2<=55) group.test<-c(group.test,1)
  if (test[i,]$V2>55&test[i,]$V2<=68) group.test<-c(group.test,2)
  if (test[i,]$V2>68) group.test<-c(group.test,3)
}

train.bayes<-data.frame(cbind(level=train$V1,group=group.train,apply(train[,3:103], MARGIN=2, FUN = func
test.bayes<-data.frame(cbind(level=test$V1,group=group.test,apply(test[,3:103], MARGIN=2, FUN = functio
```

```r
train.bayes1<-train.bayes[train.bayes$group==1,]
train.bayes2<-train.bayes[train.bayes$group==2,]
train.bayes3<-train.bayes[train.bayes$group==3,]
```

```r
#lasso_cv1<-cv.glmnet(x=as.matrix(train.bayes1[,-(1:2)]),y=train.bayes1[,1],alpha = 1,family="binomial"
modelFit1<-glmnet(x=as.matrix(train.bayes1[,-(1:2)]),y=train.bayes1[,1],alpha = 1,family="binomial",
                    lambda = 0.005)
#lasso_cv2<-cv.glmnet(x=as.matrix(train.bayes2[,-(1:2)]),y=train.bayes2[,1],alpha = 1,family="binomial"
modelFit2<-glmnet(x=as.matrix(train.bayes2[,-(1:2)]),y=train.bayes2[,1],alpha = 1,family="binomial",
                    lambda = 0.0002)
#lasso_cv3<-cv.glmnet(x=as.matrix(train.bayes3[,-(1:2)]),y=train.bayes3[,1],alpha = 1,family="binomial"
modelFit3<-glmnet(x=as.matrix(train.bayes3[,-(1:2)]),y=train.bayes3[,1],alpha = 1,family="binomial",
                    lambda = 0.0005)
```

```r
predFit<-c()

for (i in 1:nrow(test.bayes)){
  if (test.bayes[i,]$group==1){
    predFit<-c(predFit,predict(modelFit1,newx=as.matrix(test.bayes[i,-(1:2)]),type="class"))
  }
  if (test.bayes[i,]$group==2){
    predFit<-c(predFit,predict(modelFit2,newx=as.matrix(test.bayes[i,-(1:2)]),type="class"))
  }
  if (test.bayes[i,]$group==3){
    predFit<-c(predFit,predict(modelFit3,newx=as.matrix(test.bayes[i,-(1:2)]),type="class"))
  }
}

(AA<-mean(predFit==test.bayes$level))
```

```
## [1] 0.9006147
```

```r
(RR<-mean(predFit[test[,1]==0]==test[test[,1]==0,1]))
```

```
## [1] 0.8634886
```

Summary

```r
kable(data.frame(method=c('baseline1 - Demographics','baseline2 - Voice','Regularized Logistic','Hierar
                 Accuracy=round(c(A_b1,A_b2,A,AA),3),
                 Recall=round(c(R_b1,R_b2,R,RR),3)))
```

method	Accuracy	Recall
baseline1 - Demographics	0.891	0.786
baseline2 - Voice	0.719	0.476
Regularized Logistic	0.893	0.787
Hierarchical Regularized Logistic	0.901	0.863

Original Computational Environment

```r
sessionInfo()
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] glmnet_2.0-16       foreach_1.4.4       Matrix_1.2-14
##  [4] xgboost_0.6.4.6     party_1.3-0         strucchange_1.5-1
##  [7] sandwich_2.4-0      zoo_1.8-1           modeltools_0.2-21
## [10] mvtnorm_1.0-7       randomForest_4.6-14 mlbench_2.1-1
## [13] e1071_1.6-8         effects_4.0-1       carData_3.0-1
## [16] knitr_1.20
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16        compiler_3.4.4      nloptr_1.0.4
##  [4] highr_0.6           iterators_1.0.9     class_7.3-14
##  [7] tools_3.4.4         digest_0.6.15       lme4_1.1-16
## [10] evaluate_0.10.1     nlme_3.1-131.1      lattice_0.20-35
## [13] yaml_2.1.18         coin_1.2-2          stringr_1.3.0
## [16] rprojroot_1.3-2     nnet_7.3-12         data.table_1.10.4-3
## [19] survival_2.41-3     rmarkdown_1.9       multcomp_1.4-8
## [22] TH.data_1.0-8       minqa_1.2.4         magrittr_1.5
## [25] codetools_0.2-15    backports_1.1.2     htmltools_0.3.6
## [28] MASS_7.3-49         splines_3.4.4       colorspace_1.3-2
## [31] stringi_1.1.7       estimability_1.3    survey_3.33-2
```