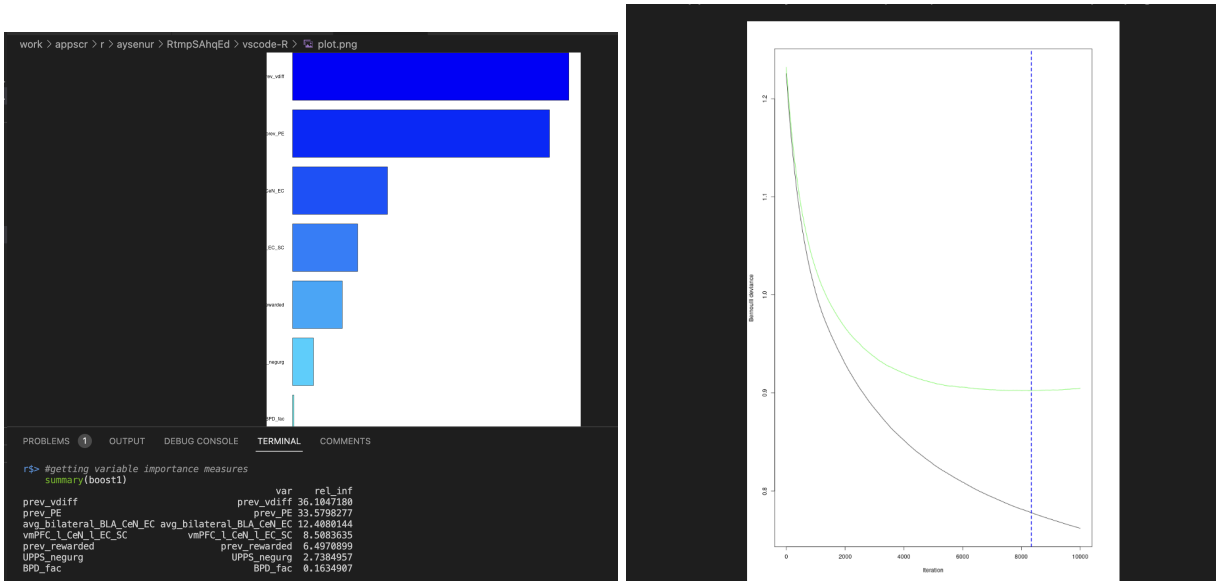
Looks good.
here, you might want to report sensitivity/specificity and such per cutpoint to see how well the model is predicting

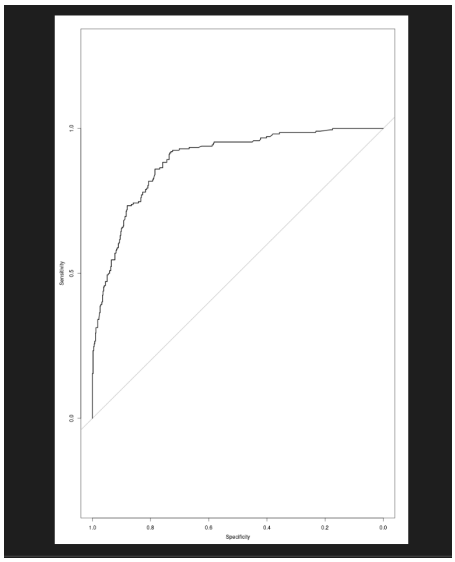
In this study, we investigate how Negative Urgency (a type of impulsivity) predicts exploration (i.e., switching choices between consecutive trials) in a reinforcement learning task that encourages participants to learn from the outcomes of their choices throughout the task. This task has a nested structure such that each participant completes 300 trials. Our outcome variable is a trial-level binary variable that shows whether participants switched their choice or stayed with the same one in consecutive trials (i.e., switch_choice). The predictors are split between trial- and participant- and group-levels. First, at the group level, participants are either in the Borderline Personality Disorder (i.e., BPD) group or in the Healthy Control (i.e., HC) group. At the participant-level, we have negative urgency scores measured by a self-report survey (i.e., negurg), functional connectivity as a neural predictor. Importantly, resting state fMRI data was analyzed using GIMME and identified different significant functional connectivity for BPD and HC groups: BLA->CeN connectivity in the BPD group and vmPFC-CeN connectivity in the HC group. Finally, at the trial level, we have the prediction error they got in the previous trial (i.e., prev_PE), whether they were rewarded in the previous trial (prev_rewarded) and the value difference between the chosen and the best option calculated by a computational algorithm (i.e., vdiff).

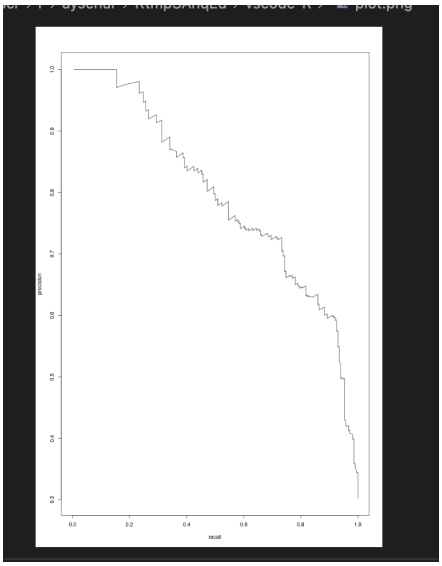
First, I try boosting with gbm() and get variable importance measures. I grow 1000 trees sequentially in random subsamples of 50% of the training dataset without replacement, with each tree having two splits/three leaves. I also specified a three-fold cross-validation to determine the optimal number of trees for prediction. The variable importance measures showed value difference (prev_vdiff_ and PE (prev_PE) in the previous trial, and BLA to CeN effective connectivity as the most important predictors, while study group, negative urgency and previous outcome were the least predictive variables. Testing the boosting performance showed that the

optimal number of trees was 8333. The training and testing datasets showed the same loss (1824611).



Next, although they lead to the exact loss, I still try to determine a different probability threshold in the training dataset using ROC curves to practice this. A threshold of 0.219 was optimal and seems to increase true positives and decrease true negatives. I also estimated the PR curves, and got the partial dependence plots for the variables with the highest/lowest variable importance.





```
---
title: "Assignment 3 - Boosting"
author: "Aysenur Okan"
date: "2024-02-08"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

library(rpart)
library(rpart.plot)
#install.packages("partykit")
library(partykit)
#install.packages("caret")
library(caret)
library(e1071)
library(party)
#install.packages("randomForest")
library(randomForest)
library(readr)
library(tidyverse)
#install.packages("gbm")
#install.packages("pROC")
#install.packages("pdp")
#install.packages("rpart.plot")$
#library(gbm)
library(pROC)
library(pdp)
#install.packages("devtools")
library("devtools")
#install_github("gbm-developers/gbm3")
library(gbm3)


#setwd("/Users/aysenur/Desktop/PSYC834 - Machine Learning")
setwd("/proj/mnhallqlab/users/aysenur/Machine Learning")

df <- read_csv("df_all_AO.csv")

df <- df %>% dplyr::select(id, BPD_fac, UPPS_negurg, avg_bilateral_BLA_CeN_EC,
vmPFC_l_CeN_l_EC_SC, prev_PE, prev_rewarded, prev_vdiff, switch_choice)
```
```

```
df$switch_choice <- factor(df$switch_choice)
df$BPD_fac <- factor(df$BPD_fac)

#Split data into training and testing
set.seed(150) # Set a seed for reproducibility
index <- createDataPartition(df$switch_choice, p = 0.5, list = FALSE) # Create an
index for splitting the data; use 50% for training and 50% for testing

# Create training and testing datasets
df <- df[index, ] #training data
tsdf <- df[-index, ] #testing data

#missing values are not allowed
df <- na.omit(df)
tsdf <- na.omit(tsdf)
df <- as.data.frame(df)
df <- as.data.frame(tsdf)
```

 # Boosting with gbm()

```{r}
#boosting model
#We specify that the distribution is "bernoulli" because our outcome only takes two
values: 0 or 1. There were 10000 trees sequentially grown on the negative gradients of
pf, and the trees have an interaction depth of 2, so each tree has two splits/three
leaves.
boost1=gbm(switch_choice~.-id, data=df, n.trees=10000,interaction.depth = 2,shrinkage
= .001, bag.fraction = .5,distribution='bernoulli',cv.folds = 3)

#getting variable importance measures
summary(boost1)
```

the performance output for the boosting procedure

```{r}
gbm.perf(boost1,method="cv")
```

```
ntree=gbm.perf(boost1,method='cv') #8333
```

obtain the predicted values in the training dataset, cross-tabulation of the classes, and the variable importance measures.

```{r}
boostpred1=predict(boost1,df,type='response',n.trees=ntree) #obtain predicted values from training d
head(boostpred1) #0.06364223 0.06381605 0.06381605 0.12076142 0.06381605 0.06495425

boostclass1=ifelse(boostpred1<.5,0,1)
boost.table=table(boostclass1,df$switch_choice) #cross-tabulation (rows are predicted, columns are observed) boost.table
1-(mean(boostclass1==df$switch_choice)) #0.1824611
```

obtain predicted values from testing dataset

```{r}
boostpred2=predict(boost1,tsdf,type='response',n.trees=ntree) #obtain predicted values from training d
head(boostpred2)
boostclass2=ifelse(boostpred2<.5,0,1)
boost.table2=table(boostclass2,tsdf$switch_choice) #cross-tabulation (rows are predicted, columns are observed)
boost.table2
1-(mean(boostclass2==tsdf$switch_choice)) #0.1824611
```

Although they lead to the exact loss, I still try to determine a different probability threshold in the training dataset using ROC curves to practice this setup

```{r}
l1=roc(predictor=boostpred1,response=df$switch_choice)
plot(l1)

thr=coords(l1,'best')[1]
thr #0.2188427
```

```
boostpredth=ifelse(boostpred2<c(thr),0,1)
table(boostpredth,tsdf$switch_choice)
```

estimate the PR curve.

```{r}
plot(precision ~ recall,
coords(l1, "all", ret = c("recall", "precision")),type='l')

```

the partial dependence plots for the variables with the highest/lowest variable importance.

```{r}
#most important
par1 <- partial(boost1, pred.var = c("prev_vdiff"), chull = TRUE,n.trees=ntree)
autoplot(par1,type='l',rug=T,train=df)

#least important
par2 <- partial(boost1, pred.var = c("BPD_fac"), chull = TRUE,n.trees=ntree)
autoplot(par2,type='l',rug=T,train=df)
```