Predicting Loneliness

IIan Shryock

Contents

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.7      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.0.0 --

## v broom        1.0.1     v rsample      1.1.0
## v dials        1.1.0     v tune         1.0.1
## v infer        1.0.3     v workflows    1.1.2
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.3     v yardstick    1.1.0
## v recipes      1.0.2

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(rio)
library(here)
```

```
## here() starts at /Users/ishryock/Documents/GitHub/behavior-prediction/03-scripts/03-nomothetic
```

```
library(rsample)
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
#library(tidyroll)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
#library(nestedcv)
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(finalfit)
library(cutpointr)
```

```
##
## Attaching package: 'cutpointr'

## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity

## The following objects are masked from 'package:yardstick':
##
##     accuracy, npv, ppv, precision, recall, sensitivity, specificity
```

```
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
library(rattle)
```

```
## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(viridis)
```

```
## Loading required package: viridisLite

##
## Attaching package: 'viridis'

## The following object is masked from 'package:scales':
##
##     viridis_pal
```

1 Read Data

```
i_am(path = "03-scripts/03-nomothetic/nomothetic_models.Rmd")
```

```
## here() starts at /Users/ishryock/Documents/GitHub/behavior-prediction
```

```
train_files <- list.files(here("04-data/03-train-data"),
                    pattern = "*lonely_full_all_time.RData",
                    full.names = TRUE)

train_files <- setNames(train_files, train_files)

col_names1 <- names(import(train_files[1]))

data <- map_dfr(train_files,
                ~import(.),
                .id = "file") %>%
  mutate(file2 = str_remove(file, "/Users/ishryock/Documents/GitHub/behavior-prediction/04-data/03-trai
           id = as.factor(paste0("p_", str_remove(file2, "_lonely_full_all_time.RData")))) %>%
  clean_names() %>%
  group_by(id) %>%
  arrange(full_date) %>%
  mutate(surv_num = seq_along(full_date),
         max_surv = max(surv_num)) %>%
select(-night)# zero variance
```

2 Preprocess

```
data2 <- data %>%
  select(o_value, everything(), -surv_num, -max_surv, -full_date, -file, -file2) %>%
  mutate(o_value = factor(as.numeric(o_value), levels = c(1, 2), labels = c("notlonely", "lonely"))) %>%
  data.frame()


missingness <- ff_glimpse(data2) %>%
  bind_rows() %>%
  select(label, missing_percent) %>%
  filter(missing_percent < 75)
```

2.1 Blueprint

```
dummy_vars <- c("o_value", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"
                , "morning", "midday", "evening", "argument"
                , "interacted", "lost_Smthng", "late", "frgt_Smthng", "brd_S_Wk"
                , "exc_S_Wk", "Anx_S_Wk", "tired", "sick", "sleeping", "class"
                , "music", "internet", "TV", "study", "id") %>%
  str_to_lower()

blueprint_lonely <- recipe(x = data2,
                           vars = colnames(data2),
                           roles = c("outcome", rep("predictor", 67))) %>%
  step_dummy(one_of(dummy_vars), -all_outcomes()) %>%
      step_zv(all_numeric()) %>%
    step_normalize(all_numeric())
```

3 Partition Data

```
set.seed(123)

trainIndex <- createDataPartition(data2$id, list = FALSE,
                                  p = .08, times = 1)



train_data2 <- data2[-trainIndex, ]
test_data2 <- data2[trainIndex, ]
```

4 Logistic

4.1 Tune

```
set.seed(123)

log_grid = data.frame(alpha = 0,
                  lambda = 0)


fitControl <-  trainControl(method = "cv",
                            number = 10,
                            classProbs = TRUE,
                            summaryFunction = mnLogLoss)
```

4.2 Train

```
log_mod <- caret::train(blueprint_lonely,
                        data = train_data2,
                        method = "glmnet",
                        family = "binomial",
                        metric = "logLoss",
                        trControl = fitControl,
                        tuneGrid = log_grid)
```

```
## Loading required namespace: glmnet

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:bitops':
##
##     %&%
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-4
```

```
save(log_mod, file = here("05-results/edld", "log_mod.rda"))
```

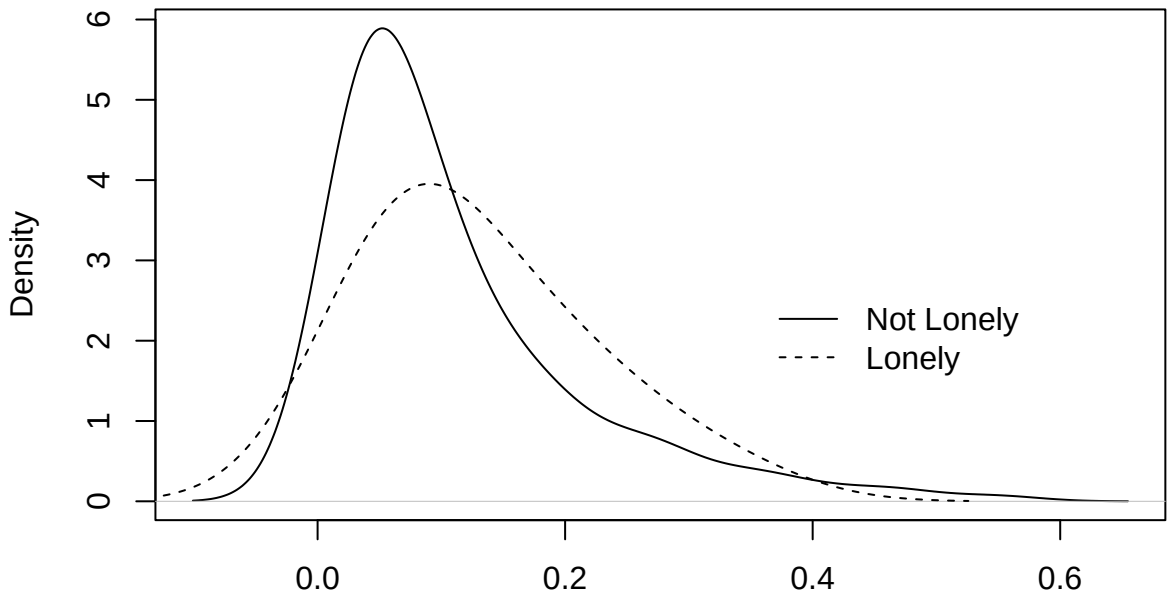
4.3 Get Predictions

```
load(here("05-results/edld", "log_mod.rda"))

predicted_log <- predict(log_mod, test_data2, type='prob')
head(predicted_log)
```

```
##   notlonely     lonely
## 1 0.8380372 0.16196280
## 2 0.8058113 0.19418867
## 3 0.9576283 0.04237174
## 4 0.9336749 0.06632513
## 5 0.9852908 0.01470920
## 6 0.9091354 0.09086458
```

4.4 Check Separations

```
notlonely <- which(test_data2$o_value=="notlonely")
lonely <- which(test_data2$o_value=="lonely")


plot(density(predicted_log[notlonely,]$lonely,adjust=1.5),xlab='',main='')
points(density(predicted_log[lonely,]$lonely,adjust=1.5),lty=2,type='l')
legend(x=0.35,y=2.75,c('Not Lonely','Lonely'),lty=c(1,2),bty='n')
```

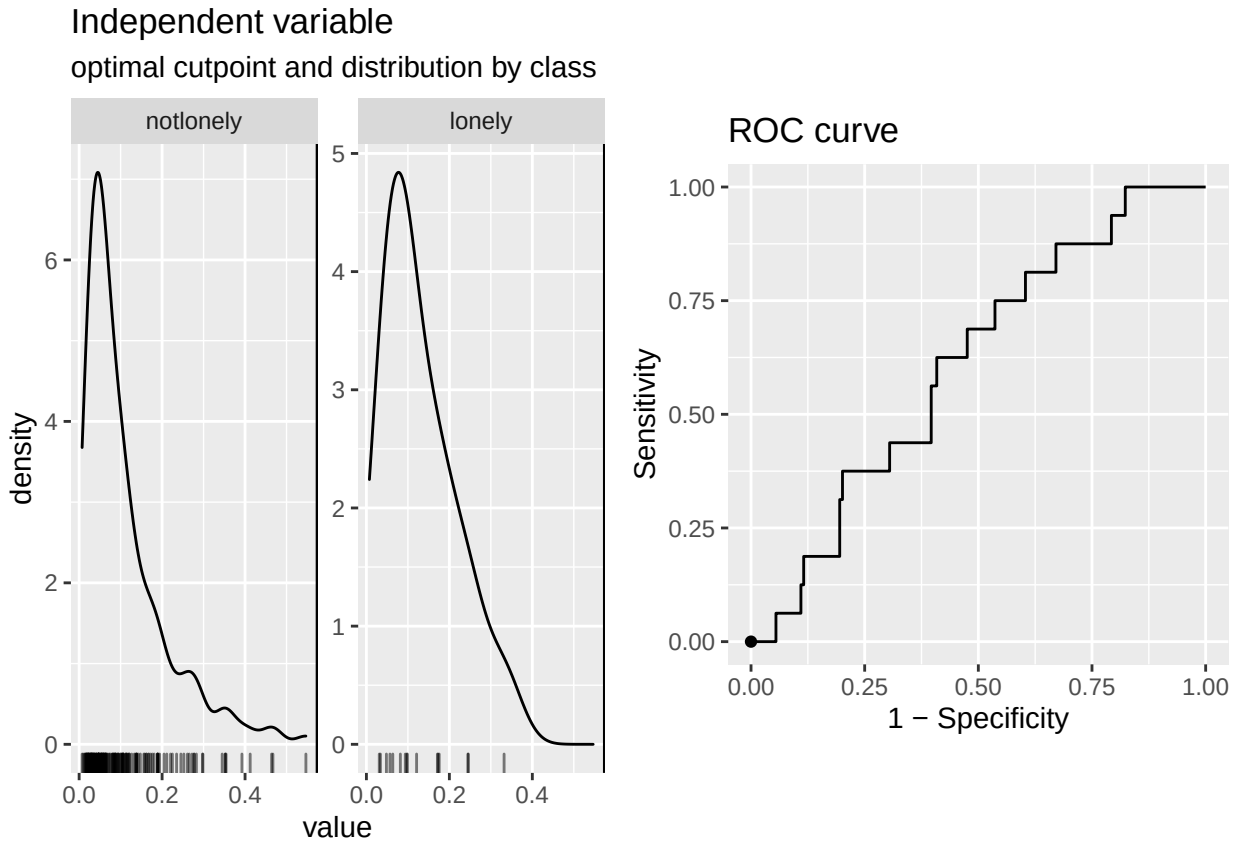


4.5 Get Cutpoint

```
log.cut.obj <- cutpointr(x     = predicted_log$lonely,
                     class = test_data2$o_value,
                     method = maximize_metric,
                     metric = accuracy)
```

```
## Assuming the positive class is lonely

## Assuming the positive class has higher x values
```

```
plot(log.cut.obj)
```




```
log.cut.obj$optimal_cutpoint ## not good...
```

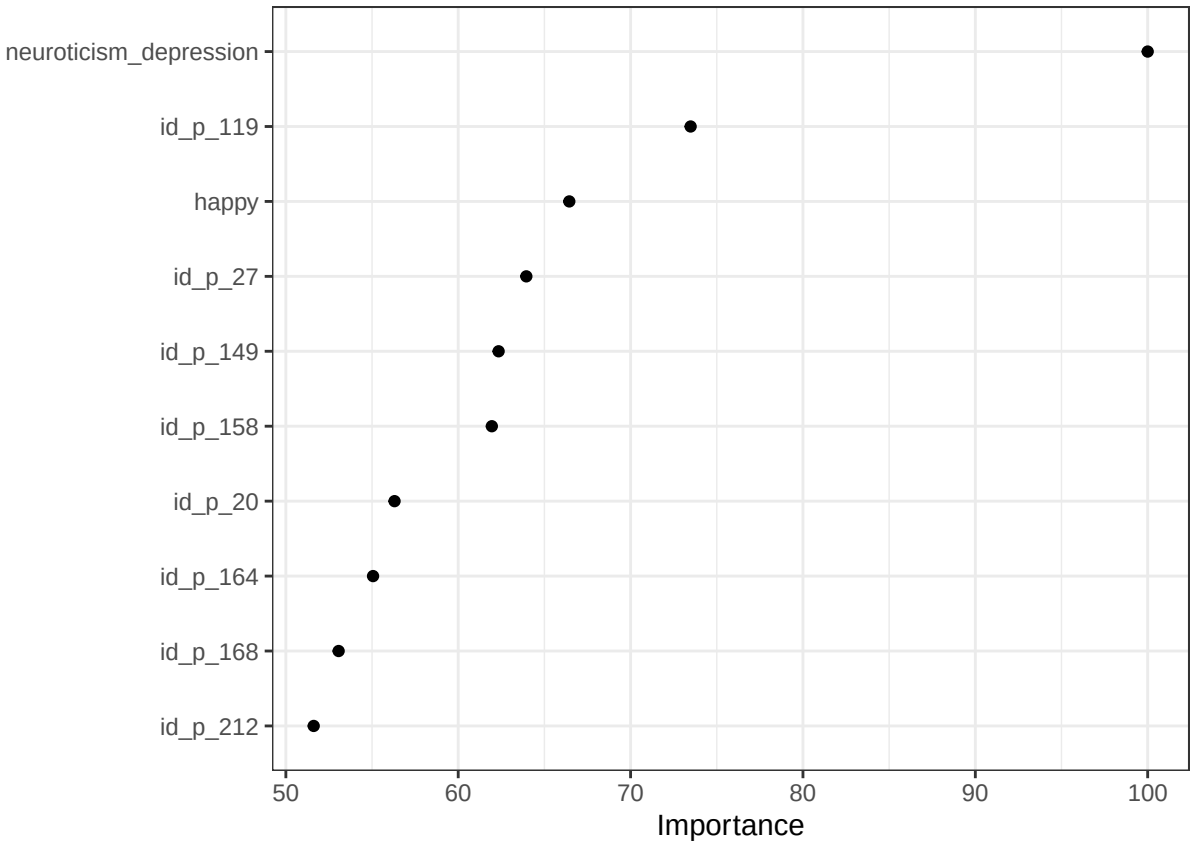
```
## [1] Inf
```

4.6 Get Confusion Matrix

```
pred_class_log <- ifelse(predicted_log$lonely>.2,1,0) ## chose an arbitrary value since cutpointr didn'

confusion_log <- table(test_data2$o_value,pred_class_log)
```

4.7 Variable Importance

```
vip(log_mod,num_features = 10, geom = "point") + theme_bw()
```



Get Coefs

```
log_coefs <- matrix(coef(log_mod$finalModel,log_mod$bestTune$lambda)) %>% data.frame()
log_coefs <- coef(log_mod$finalModel,log_mod$bestTune$lambda)

ind_log   <- order(abs(log_coefs),decreasing=T)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
top_log_coef <- head(as.matrix(log_coefs[ind_log[-1], ]), 20)
```

5 GLMNET

5.1 Tune

```
elnet_grid <- expand.grid(alpha = seq(0,1,.01), lambda = seq(0.001,0.5,.005))
```

5.2 Train

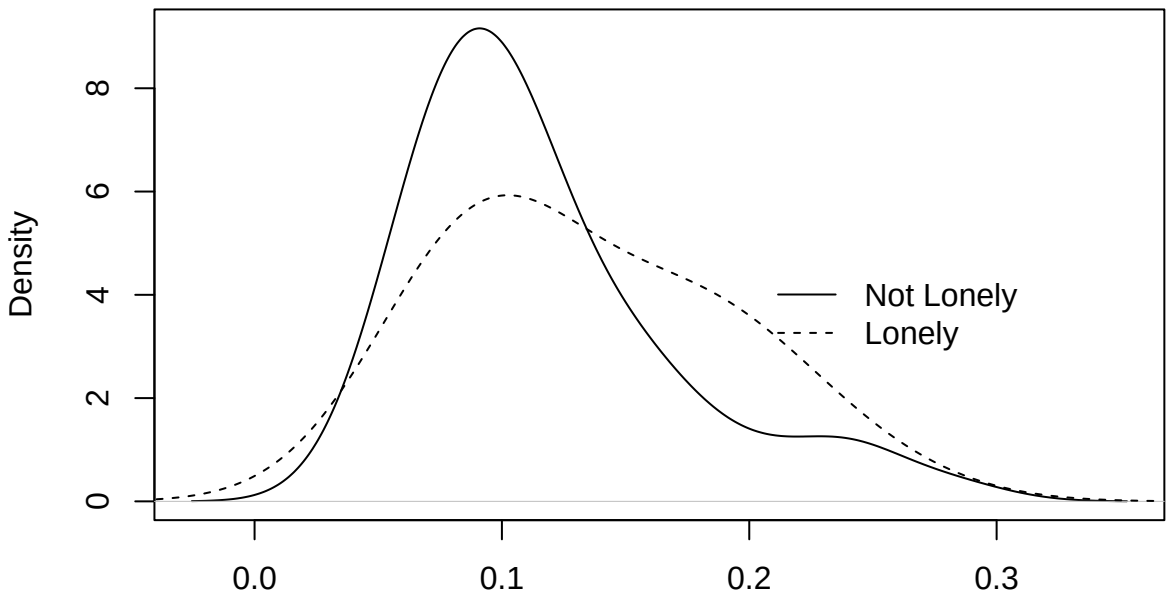
```
Sys.time()

elastic <- caret::train(blueprint_lonely,
                        data      = train_data2,
                        method    = "glmnet",
                        family = "binomial",
                        metric = "logLoss",
                        trControl = fitControl,
                        tuneGrid  = elnet_grid)

Sys.time()


save(elastic, file = here("05-results/edld", "elastic.rda"))
rm(elastic)
```

##Get Predictions

```
load(here("05-results/edld", "elastic.rda"))

predicted_elnet <- predict(elastic, test_data2, type='prob')
#head(predicted_log)
```

5.3 Check Separations

```
plot(density(predicted_elnet[notlonely,]$lonely,adjust=1.5),xlab='',main='')
points(density(predicted_elnet[lonely,]$lonely,adjust=1.5),lty=2,type='l')
legend(x=0.2,y=4.75,c('Not Lonely','Lonely'),lty=c(1,2),bty='n')
```

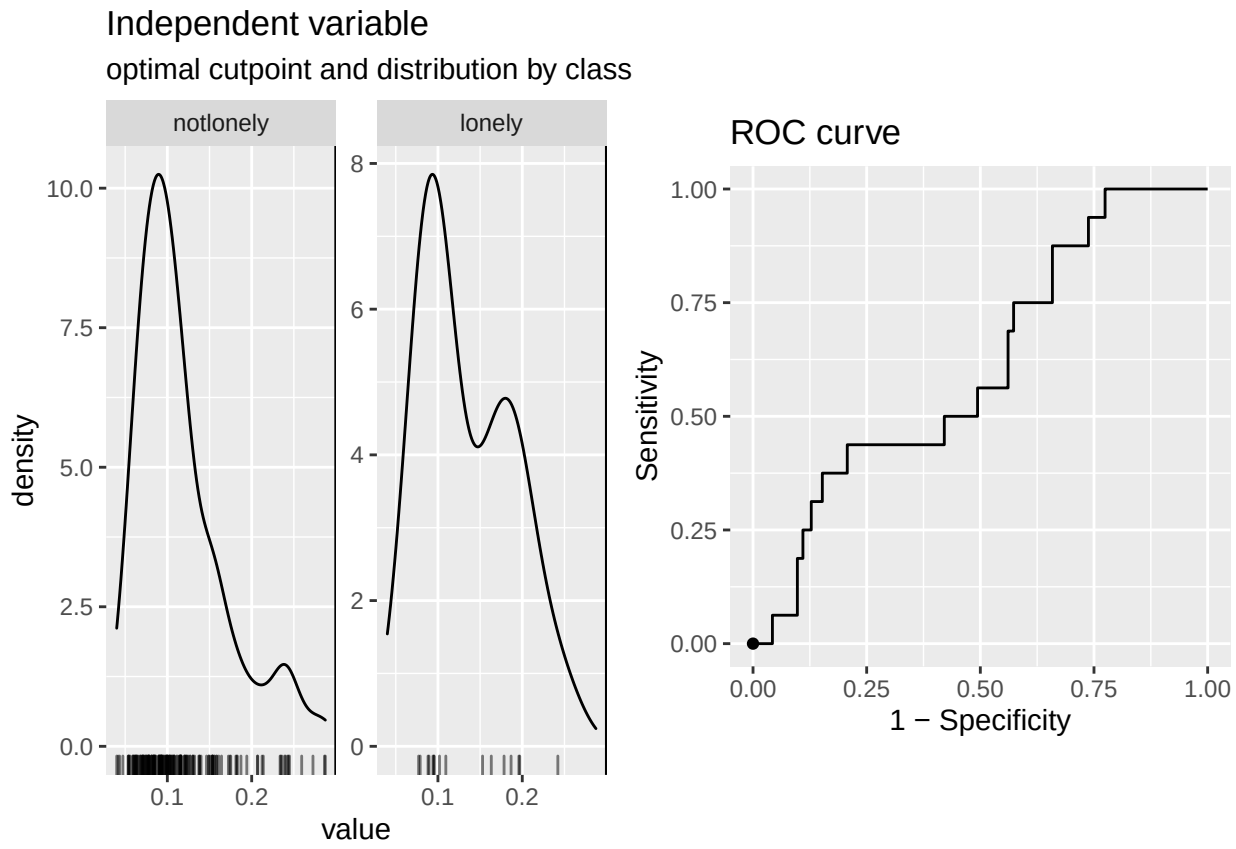


5.4 Get Cutpoint

```
elnet.cut.obj <- cutpointr(x     = predicted_elnet$lonely,
                      class = test_data2$o_value,
                      method = maximize_metric,
                      metric = accuracy)
```

```
## Assuming the positive class is lonely

## Assuming the positive class has higher x values
```

```
plot(elnet.cut.obj)
```




```
elnet.cut.obj$optimal_cutpoint
```

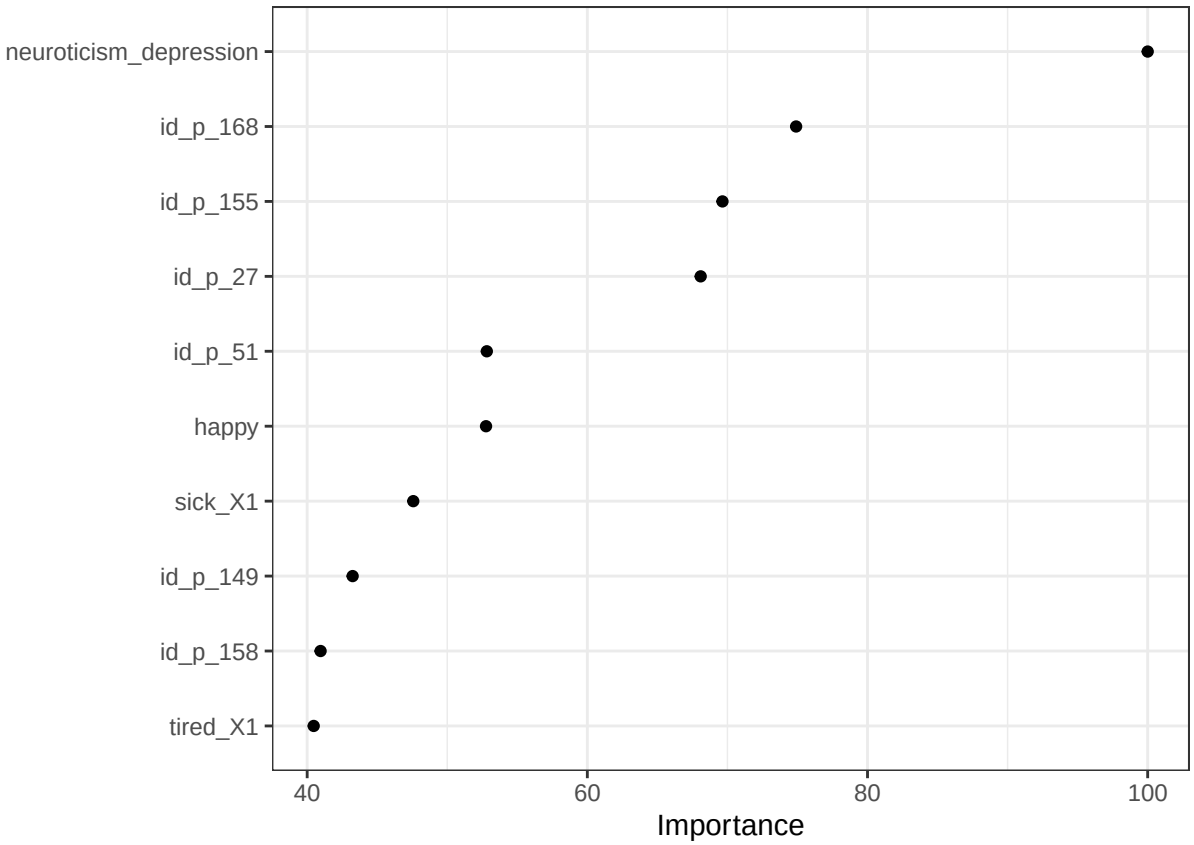
```
## [1] Inf
```

5.5 Get Confusion Matrix

```
pred_class_elnet <- ifelse(predicted_elnet$lonely>.2,1,0) ## chose an arbitrary value since cutpointr d

confusion_elnet <- table(test_data2$o_value,pred_class_elnet)
```

5.6 Variable Importance

```
vip(elastic,num_features = 10, geom = "point") + theme_bw()
```



5.7 Get Coefs

```
elnet_coefs <- matrix(coef(elastic$finalModel,elastic$bestTune$lambda)) %>% data.frame()
elnet_coefs <- coef(elastic$finalModel,elastic$bestTune$lambda)

elnet_ind   <- order(abs(elnet_coefs),decreasing=T)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
top_elnet_coef <- head(as.matrix(elnet_coefs[elnet_ind[-1], ]), 20)
```

6 Random Forest

6.1 Tune

```
rf_grid <- expand.grid(mtry = 20,splitrule='gini',min.node.size=2)
```

```
Sys.time()
```

```
   nbags <- c(5,seq(20,200,20))
   bags <- vector('list',length(nbags))

    for(i in 1:length(nbags)){
bags[[i]] <- caret::train(blueprint_lonely,
                          data      = train_data2,
                          method    = 'ranger',
                          metric = "logLoss",
                          trControl = fitControl,
                          tuneGrid  = rf_grid,
                          num.trees = nbags[[i]],
                          max.depth = 60)

    }


save(bags, file = here("05-results/edld", "bags.rda"))
```

6.2 Choose nBags

```
load(here("05-results/edld", "bags.rda"))

   nbags <- c(5,seq(20,200,20))

logLoss_ <- c()

for(i in 1:length(nbags)){

  logLoss_[i] = bags[[i]]$results$logLoss

}

ggplot()+
  geom_line(aes(x=nbags,y=logLoss_))+
  xlab('Number ofs')+
  ylab('Negative LogLoss')+
  ylim(c(0,1))+
  theme_bw()



nbags[which.min(logLoss_)] ## 140

rm(bags)
```

6.3 Run Final Model

```
rforest <- caret::train(blueprint_lonely,
                          data      = train_data2,
```

```
                        method    = 'ranger',
                        metric = "logLoss",
                        family = "binomial",
                        trControl = fitControl,
                        tuneGrid  = rf_grid,
                        num.trees = 140,
                        max.depth = 60,
                        importance = "impurity")
```

```
## Loading required namespace: e1071

## Loading required namespace: ranger

##
## Attaching package: 'e1071'

## The following object is masked from 'package:tune':
##
##     tune

## The following object is masked from 'package:rsample':
##
##     permutations

## The following object is masked from 'package:parsnip':
##
##     tune

##
## Attaching package: 'ranger'

## The following object is masked from 'package:rattle':
##
##     importance

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
```

```
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family

## Warning in ranger::ranger(dependent.variable.name = ".outcome", data = x, :
## Unused arguments: family
```

```
save(rforest, file = here("05-results/edld", "rforest.rda"))
rm(rforest)
```

6.4 Get Predictions

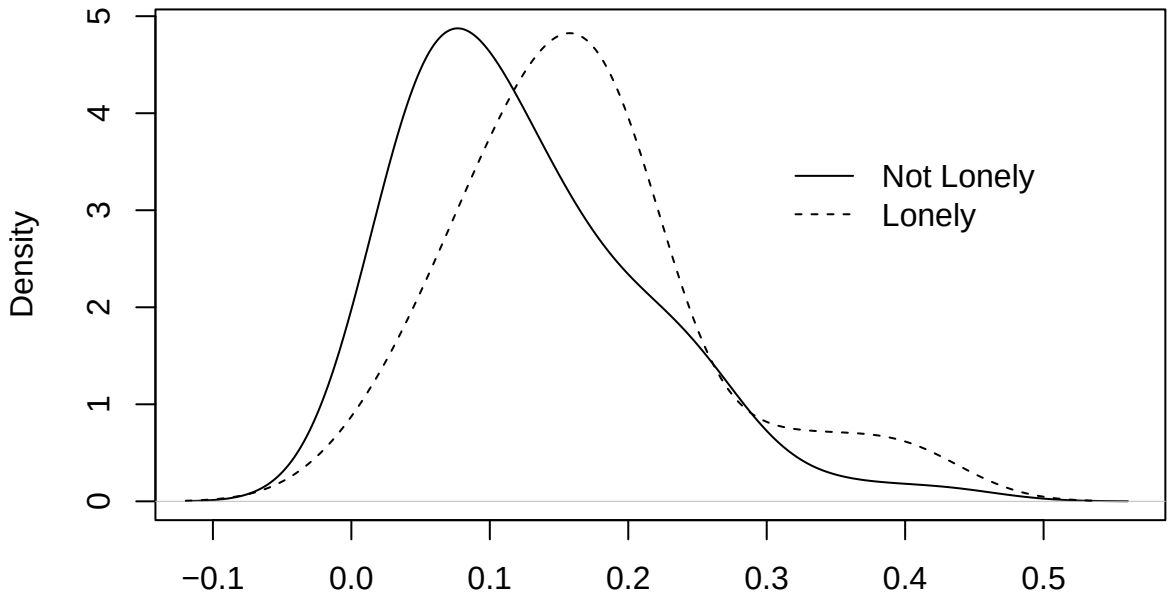
```
load(here("05-results/edld", "rforest.rda"))
rforest$results
```

```
##   mtry splitrule min.node.size   logLoss  logLossSD
## 1   20      gini             2 0.3592378 0.01458553
```

```
predicted_forest <- predict(rforest, test_data2, type = "prob")
```

6.5 Check Separations

```
plot(density(predicted_forest[notlonely,]$lonely,adjust=1.5),xlab='',main='')
points(density(predicted_forest[lonely,]$lonely,adjust=1.5),lty=2,type='l')
legend(x=0.3,y=3.75,c('Not Lonely','Lonely'),lty=c(1,2),bty='n')
```

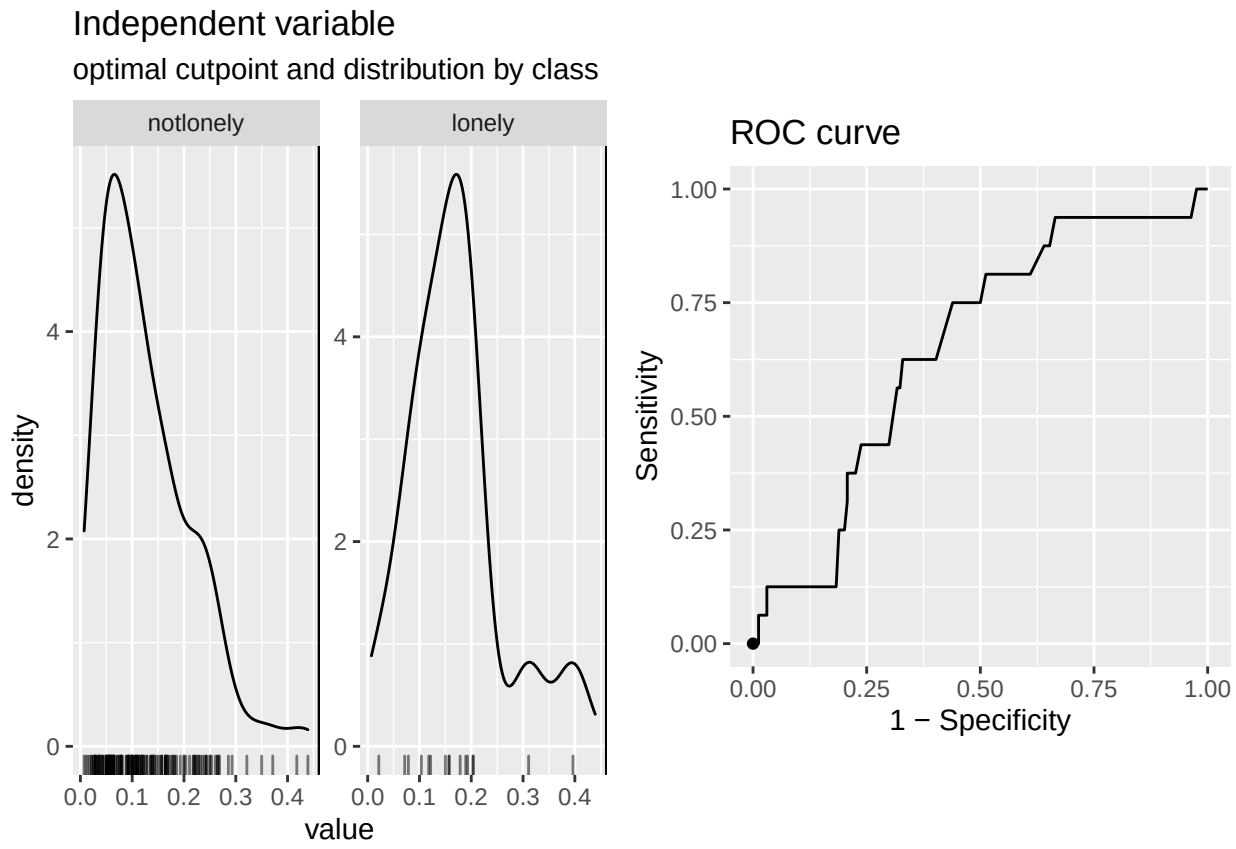


6.6 Get Cutpoint

```
forest.cut.obj <- cutpointr(x     = predicted_forest$lonely,
                     class = test_data2$o_value,
                     method = maximize_metric,
                     metric = accuracy)
```

```
## Assuming the positive class is lonely

## Assuming the positive class has higher x values
```

```
plot(forest.cut.obj)
```




```
forest.cut.obj$optimal_cutpoint
```

```
## [1] Inf
```

6.7 Get Confusion Matrix

```
pred_class_forest <- ifelse(predicted_forest$lonely>forest.cut.obj$optimal_cutpoint ,1,0) ## chose an a

confusion_forest <- table(test_data2$o_value,pred_class_forest)
```

6.8 Get Coefs/Importance

```
forest_imp <- importance(rforest$finalModel) %>%
  data.frame() %>%
  rownames_to_column()
names(forest_imp) <- c("rowname", "val")

top_forest_imp <- forest_imp %>%
 arrange(desc(abs(val))) %>%
    head(20) %>%
  mutate(model = "Forest")
```

7 Comparing Models

```
confusions = bind_rows(unlist(as.data.frame.matrix(confusion_log)), unlist(as.data.frame.matrix(confusi
                     unlist(as.data.frame.matrix(confusion_forest)))

colnames(confusions) <- c("TN", "FN", "FP", "TP")

confusions$Model <- c("logistic", "elastic", "rforest")
confusions$LL <- c(log_mod$results$logLoss, min(elastic$results$logLoss), min(rforest$results$logLoss))
confusions$AUC = c(
        auc(cutpointr(x     = predicted_log$lonely,
                      class = test_data2$o_value)),
        auc(cutpointr(x     = predicted_elnet$lonely,
                      class = test_data2$o_value)),
        auc(cutpointr(x     = predicted_forest$lonely,
                      class = test_data2$o_value))
        )
```

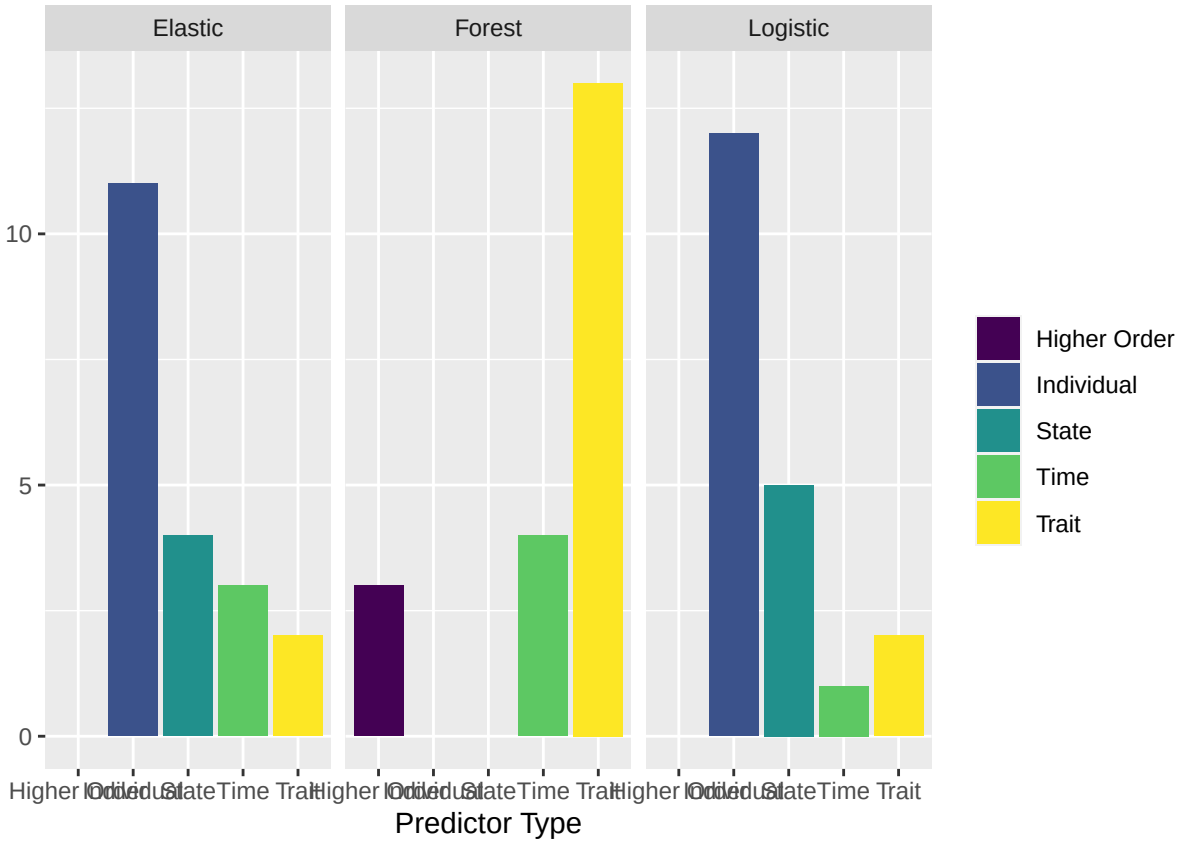
```
## Assuming the positive class is lonely

## Assuming the positive class has higher x values

## Assuming the positive class is lonely

## Assuming the positive class has higher x values

## Assuming the positive class is lonely

## Assuming the positive class has higher x values
```

```
confusions <- confusions %>%
  mutate(  ACC = (TP + TN)/(TP + TN + FP + FN),
           TPR = TP / (TP + FN),
           TNR = TN / (TN + FP),
           PRE = TP / (TP + FP)) %>%
  select(Model, everything())

confusions
```

```
## # A tibble: 3 x 11
##   Model       TN    FN    FP    TP    LL   AUC    ACC     TPR    TNR     PRE
##   <chr>    <int> <int> <int> <int> <dbl> <dbl>  <dbl>   <dbl>  <dbl>   <dbl>
## 1 logistic   140    13    24     3 0.386 0.607  0.794  0.188   0.854  0.111
## 2 elastic    148    15    16     1 0.354 0.608  0.828  0.0625  0.902  0.0588
## 3 rforest    164    16    NA    NA 0.359 0.650 NA     NA      NA     NA
```

8 Comparing Coefficients/Importance

```r
top_log_coef2 <- top_log_coef %>%
  data.frame() %>%
  rownames_to_column() %>%
  mutate(model = "Logistic")

top_elnet_coef2 <- top_elnet_coef %>%
  data.frame() %>%
  rownames_to_column() %>%
  mutate(model = "Elastic")

top_coefs <- bind_rows(top_elnet_coef2, top_log_coef2)



names(top_coefs) <- c("rowname", "val", "model")

top_coefs <- bind_rows(top_coefs, top_forest_imp) %>%
  mutate(predictor_type = case_when(
    str_detect(rowname, "agreeableness|neuroticism|extraversion|conscientiousness|openness")~"Trait",
    str_detect(rowname, "id_")~"Individual",
    str_detect(rowname, "cos1|sin1|sat_|morning|sin2|cos2")~"Time",
    str_detect(rowname, "cub|linear|quad")~"Higher Order",
    TRUE ~"State"
 ))

top_coefs %>%
  ggplot(aes(predictor_type))+
  geom_bar(aes(fill = predictor_type))+
  scale_fill_viridis_d()+
  facet_wrap(~model)+
xlab("Predictor Type")+
  ylab("")+
labs(fill = "")
```


Elastic
Forest
Logistic
10
5
0
Higher Order
Individual
State
Time
Trait
Predictor Type