EDLD 640 Capstone

Diana DeWald[1] & Dare Baldwin[1]

[1] University of Oregon

Abstract

How is young children's exploration impacted by adult pedagogy? Can we create Machine Learning models to predict how a child will explore the causal features of an object based upon the pedagogy they are exposed to? Our goal is to establish predictive models of preschoolers' causal learning outcomes within educational settings based upon teachers' pedagogical styles. Using pre-existing samples of pedagogy and child outcomes, we created three machine learning models to capture 1) the extent to which adults produce pedagogy that is likely to have the intended effect on preschoolers' play behavior when prompted accordingly, and 2) the extent to which pedagogy that has different intent based upon the prompt differs in sentiment. We found that......

Keywords: causal learning models, pedagogy, text analysis

Word count: 1933

EDLD 640 Capstone

Introduction

Developmentalists and educators have long debated the benefits of child-directed (Montessori style) exploration contra adult-directed (pedagogical) instruction on learning outcomes. While young children (age 3-6) often re-structure their hypotheses about the world based upon self-directed exploration á la Montessori, there are many subjects children cannot master without adult-directed pedagogical guidance (e.g., novel object labels, the alphabet, color and shape labels, historical events, the existence of entities such as germs, etc.). Such subjects are often culturally and linguistically bound, but even causal learning related to the physical properties of objects and entities can benefit from adult-directed pedagogy. Clarifying the extent to which pedagogy supports—and in some cases diminishes—effective causal learning is essential for a) informing teaching strategies in a time when many preparatory schools in the U.S. suffer from a lack of funding and teacher support (**SRCD?**; **NIERR?**), and b) elevating early education outcomes following relative dips in school preparedness over the past three years (Jalongo, 2021; (**gonzalez2022school?**)). Those who have sought to address the impact of adult-directed pedagogy on causal learning describe a pedagogical trade-off model. This model proposes that adult instruction increases the proportion of time children spend exploring an object's pedagogy-relevant properties but limits their investigation of other properties. Conversely, child-directed exploration is understood to produce broader discovery of the complete set of an object's causal properties but diminish the time spent investigating any particular property.While such behavioral outcomes are established, little is known about the differential impact of diverse pedagogy types (such method tend to rely on one pedagogical condition statement, usually "This is how my toy works"). In this project, we were interested in the relation between adults' use of language while teaching and children's learning outcomes. Specifically, we wanted to investigate how diverse and naturalistic pedagogy types that were produced to encourage children's play based upon findings from the pedagogy trade-off model would align with the pedagogy utilized within typical empirical methods for this model. To examine this, we took pre-existing text data from a survey where we asked UO students to watch a video depicting a toy and generate a text response detailing how they would go about teaching a preschooler about the toy. In one between subjects condition, they were instructed to 'enhance' exploration (i.e., by providing pedagogy that would encourage a child to explore beyond one object property). In a second condition, they were instructed to 'constrain exploration (i.e., by providing pedagogy that would encourage a child to only explore the one property demonstrated in the video). After presenting descriptives on the survey text dataset, we go on to describe three machine learning models to investigate different aims. The first model was created to predict the likelihood of a positive or negative sentiment occurring based upon the condition (enhance or constrain), source (qualtrics survey or pre-existing study pedagogy) and function (we included both a 'squeaker' and a 'light' function in the object introduction video). The second model was created to predict the distance among expected outcomes paired via a manual coding system to the pedagogy samples from the survey (K-nearest neighbors). The third model again used logistic regression to create a best-performing model predicting child play outcomes (we collected in Fall of 2022 at the Oregon Museum of Science and Industry) from 4 pedagogy types used in that method.

Methods

```
In Fall of 2022, we conducted a Qualtrics survey of undergraduate students
```

(N = 168) at the University of Oregon, asking participants to report how they would teach a child about the causal properties of a novel object. Participants were randomly sorted into 2 conditions. In the 'enhance' condition, participants were instructed to generate pedagogy for two object properties with the intention to produce broad exploratory behaviors from a child (prompt: “what would you say) to intoduce this toy in a way that encourages wide-ranging exploration?”). In the ‘constrain’ condition, participants were asked to generate pedagogy intended to produce limited exploratory behaviors from a child (prompt: “What would you say) to introduce this toy in a way that discourages wide-ranging exploration?”). We subsequently assessed overall word frequencies, and word frequencies by condition (see ‘Results: Descriptive Plots’). Next, we investigated the extent to which the ‘constrain’ pedagogy differed in sentiment from the ‘enhance’ pedagogy (see

‘Results: Model 1’) In Model 2, we utilized a coding classifier system pairing participant-generated pedagogy with 7 pedagogy-type categories from previous research. These 7 pedagogy categories were linked to specific child outcomes from prior studies conducted by us as well as other labs. See Results: Model 2 for the model performance. Finally, using child data outcome data from our lab, we created a model to predict child outcomes based upon four pedagogy types from that study (Model 3). Having previously paired these four pedagogy types with items on the 7-point scale, this gives us the ability to indirectly assess the likelihood that the survey-generated pedagogy text data will produce the desired child outcomes.

Results: Descriptive plots (word counts)

```
# parsing words from the 'pedagogy' (text) column
tidy_words <- mydata %>%
  unnest_tokens(word, pedagogy)

# removing numbers
tidy_words <- tidy_words[-grep("\\b\\d+\\b", tidy_words$word),]
```

```r
# removing common/under-informative words
exclu <- tibble(word = c("the", "this", "I"))


tidy_words <- tidy_words %>%
  anti_join(exclu, by = "word")



#plot
tidy_words %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>% # make y-axis ordered by n
  slice(1:15) %>% # select only the first 15 rows
  ggplot(aes(n, word)) +
  geom_col(fill = "royalblue", alpha = .7) +
  scale_x_continuous(expand = c(0,0)) +
  theme_minimal() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.major.x = element_line(color = "gray80")
  ) +
  labs(
    x = "Word Frequency",
    y = "Word",
    title = "Figure 1: Top 15 most frequently occurring words across all pedagogy types"
  )
```

Figure 1: Top 15 most frequently occurring words across all pedagogy ty

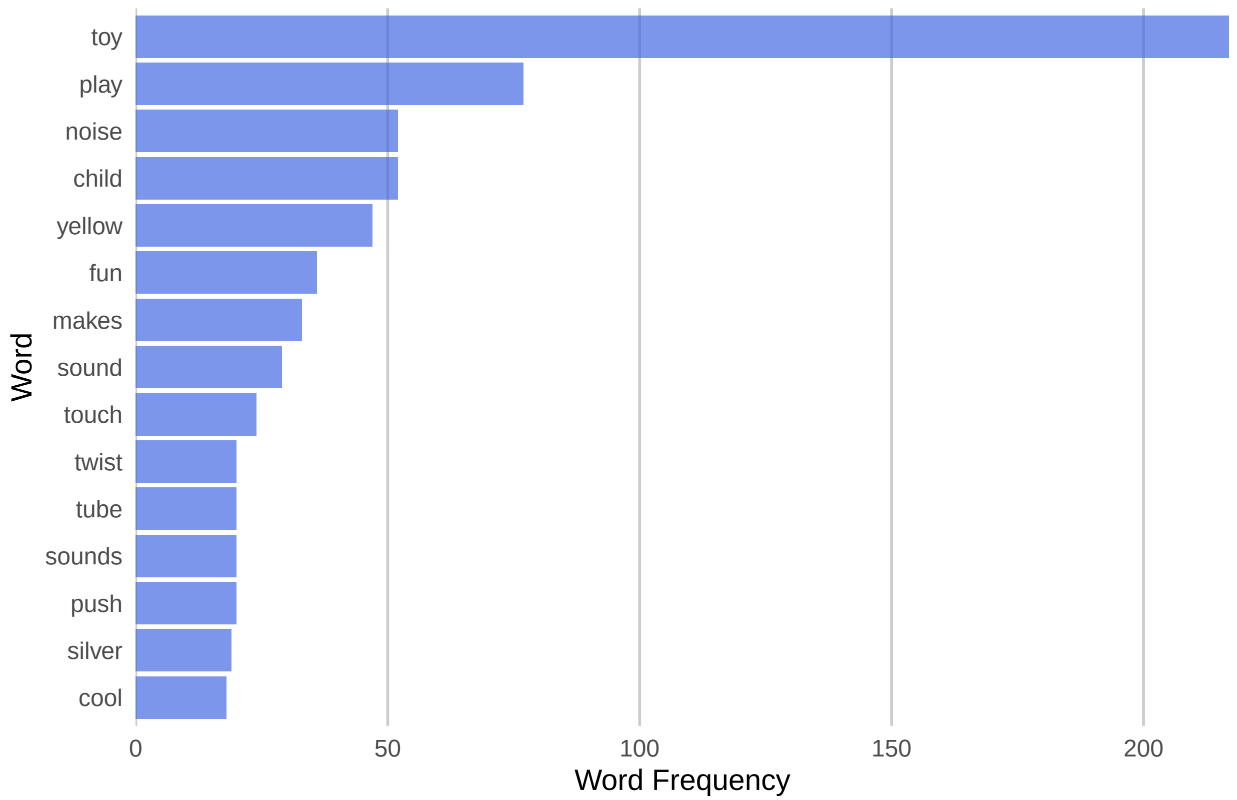


Figure 2: Word Frequency Cloud (Across Conditions)

```
# visualing: word cloud
library(wordcloud)


tokens = textdata %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)


# top words
word_count = tokens%>%
  group_by(word)%>%
```

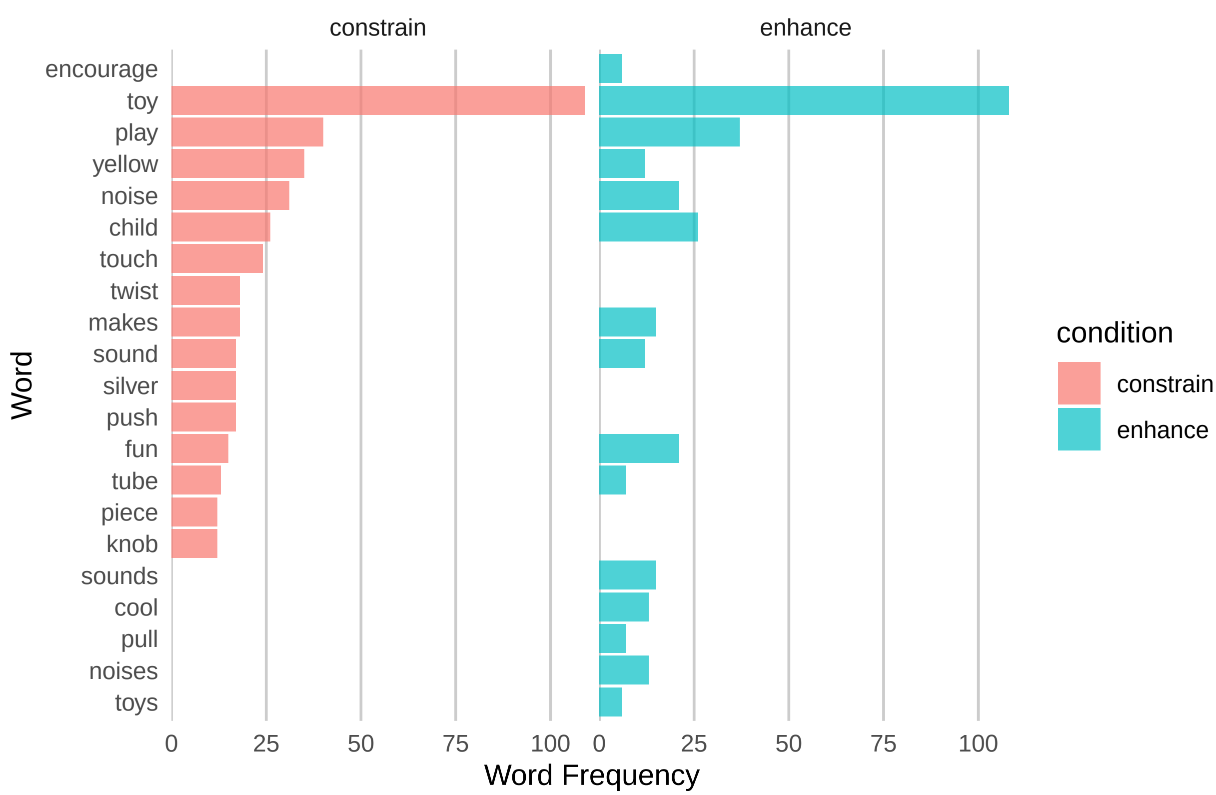
```
  summarise(count = n())%>%
  arrange(desc(count))%>%
  slice(1:10)



# word cloud--zoom in
cloud <- tokens %>%
  group_by(source, word) %>%
  summarise(count = n())%>%
  arrange(desc(count))%>%
  slice(1:10)



wordcloud(tokens$word, max.words = 75, colors=brewer.pal(6, "Dark2"))
```

87

```
tidy_words %>%

  group_by(condition) %>%

  anti_join(stop_words) %>%

  count(word, sort = TRUE) %>%

  mutate(word = reorder(word, n)) %>% # make y-axis ordered by n

  slice(1:15) %>% # select only the first 15 rows

  ggplot(aes(n, word, fill = condition)) +

  geom_col(alpha = .7) +

  facet_wrap(~condition) +

  scale_x_continuous(expand = c(0,0)) +

  theme_minimal() +

  theme(
```

```
    panel.grid.major.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.major.x = element_line(color = "gray80")
  ) +
  labs(
    x = "Word Frequency",
    y = "Word",
    title = "Figure 3: Top 15 most frequently occurring words by condition",
  )
```

Figure 3: Top 15 most frequently occurring words by condition



In Figure 3, we see the top 15 words used in the enhance and constrain conditions. In the 'enahnce' condition, there was greater use of words like "sounds," "cool," "pull," "noises," and "toys." In the 'constrain' condition, there was greater use of "touch," "silver," "push," and "knob."

Description of sentiment analysis. After trying for many hours to use the ‘transformers’ package from ‘huggingface’ via a virtual python environment without success, we decided to utilize sentiment analysis program from “bing” to perform analyses for Model 1. We also utilized three other programs (“afinn,” “loughran,” and “nrc”), and a summary of their positive and negative assessments by condition is included in Table 1 at the end of this document.

```
sentimentdata <- import(here("data", "sentiment_an.xlsx"))
sentiment <- import(here("data", "sentiment.xlsx"))

sentiment_by_condition <- sentimentdata %>%
  group_by(condition, analysis) %>%
  count(sentiment)

# table of sentiment for four groups ("bing", "nrc", "afinn", "loughran")
sentimenttable <- import(here("data", "sentiment_table.xlsx"))
table <- sentimenttable[1:2, 2:11]

options(kableExtra.auto_format = FALSE)
library(kableExtra)



table %>%
  kbl(caption = "Sentiment by analysis tool") %>%
  kable_classic(html_font = "Cambria") %>%
  column_spec(column = 1:10, width = "0.57in") %>%
  footnote(general_title = "Note.", footnote_as_chunk=TRUE, threeparttable=TRUE, general
```

Model 1: Predicting a Categorical Outcome (sentiment: positive or negative) using Regularized Logistic Regression

```
# Recipe for the sentiment dataset

outcome <- c('sentiment')

ID <- c('id')

categorical <- c('condition', 'source', 'function')

 blueprint <- recipe(x  = sentiment,
                    vars  = c(categorical, outcome, ID),
                    roles = c(rep('predictor',3), 'outcome', 'ID')) %>%
  step_zv(all_of(categorical))



# splitting data into training and testing
# Let the training data have the 80% of cases and the test data have the 20% of the ca

set.seed(11102021)  # for reproducibility

sen      <- sample(1:nrow(sentiment), round(nrow(sentiment) * 0.8))
sen_train  <- sentiment[sen, ]
sen_test  <- sentiment[-sen, ]

# 10-fold cross-validation without regularization
```

```
set.seed(11152021) # for reproducibility

sen_tr = sen_train[sample(nrow(sen_train)),]

  # Creating 10 folds with equal size

folds = cut(seq(1,nrow(sen_tr)),breaks=10,labels=FALSE)

  # Creating the list for each fold
sen.indices <- vector('list',10)

      for(i in 1:10){
        sen.indices[[i]] <- which(folds!=i)
      }

 cv <- trainControl(method    = "cv",
                   index          = sen.indices,
                   classProbs     = TRUE,
                   summaryFunction = mnLogLoss)

# Train the model
 caret_mod <- caret::train(blueprint,
                          data      = sen_tr,
                          method    = "glm",
```

```
                          family    = 'binomial',
                          metric    = 'logLoss',
                          trControl = cv)

# caret_mod

 # Evaluate the model on the test data

# Predict the probabilities for the observations in the test dataset

 predicted_test <- predict(caret_mod, sen_test, type='prob')

# head(predicted_test)


# Evaluate the model on the test dataset
predicted_te <- predict(caret_mod, sen_test)

predicted_te <- as.numeric(predicted_te)
sen_test_numeric <- sen_test$sentiment_numeric


predicted_eval <- data.frame(predicted_te, sen_test_numeric)
predicted_eval <- predicted_eval[complete.cases(predicted_eval), ]


rsq_te <- cor(predicted_eval$predicted_te,predicted_eval$sen_test_numeric)^2
```

```
# rsq_te

mae_te <- mean(abs(predicted_eval$sen_test_numeric - predicted_eval$predicted_te))
# mae_te

rmse_te <- sqrt(mean((predicted_eval$sen_test_numeric - predicted_eval$predicted_te)^2))
# rmse_te
```

Results Model 1

The first model is a regularized logistic regression predicting sentiment type (positive or negative) from a participant's condition (enhance or constrain), source (qualtrics survey or pre-existing study pedagogy) and function ('squeaker' or 'light). Results indicate that pedagogy did not differ significantly in sentiment based upon the condition participants were in... Model performance....

logLoss: 0.626 rsq_te: 0.026 mae_te: 0.614 rmse_te: 0.784

```
# import data
# textdummy <- import(here("data", "text_data_dummy.xlsx"))
 textdummy <-import(here("data", "text_dummy.xlsx"))

# train and test split

set.seed(10152022) # for reproducibility

loc <- sample(1:nrow(textdummy), round(nrow(textdummy) * 0.9))
rank_tr <- textdummy[loc, ]
rank_te <- textdummy[-loc, ]



# create row indices for 10-folds

  #randomly shuffle training data
rank_tr = rank_tr[sample(nrow(rank_tr)),]

    # Create 10 folds with equal size

      folds = cut(seq(1,nrow(rank_tr)),breaks=10,labels=FALSE)

    # Create the list for each fold

      my.indices <- vector('list',10)
      for(i in 1:10){
```

```
        my.indices[[i]] <- which(folds!=i)
      }


# Cross-validation settings


  cv <- trainControl(method = "cv",
                     index  = my.indices)


require(caret)
require(kknn)


getModelInfo()$kknn$parameters
```

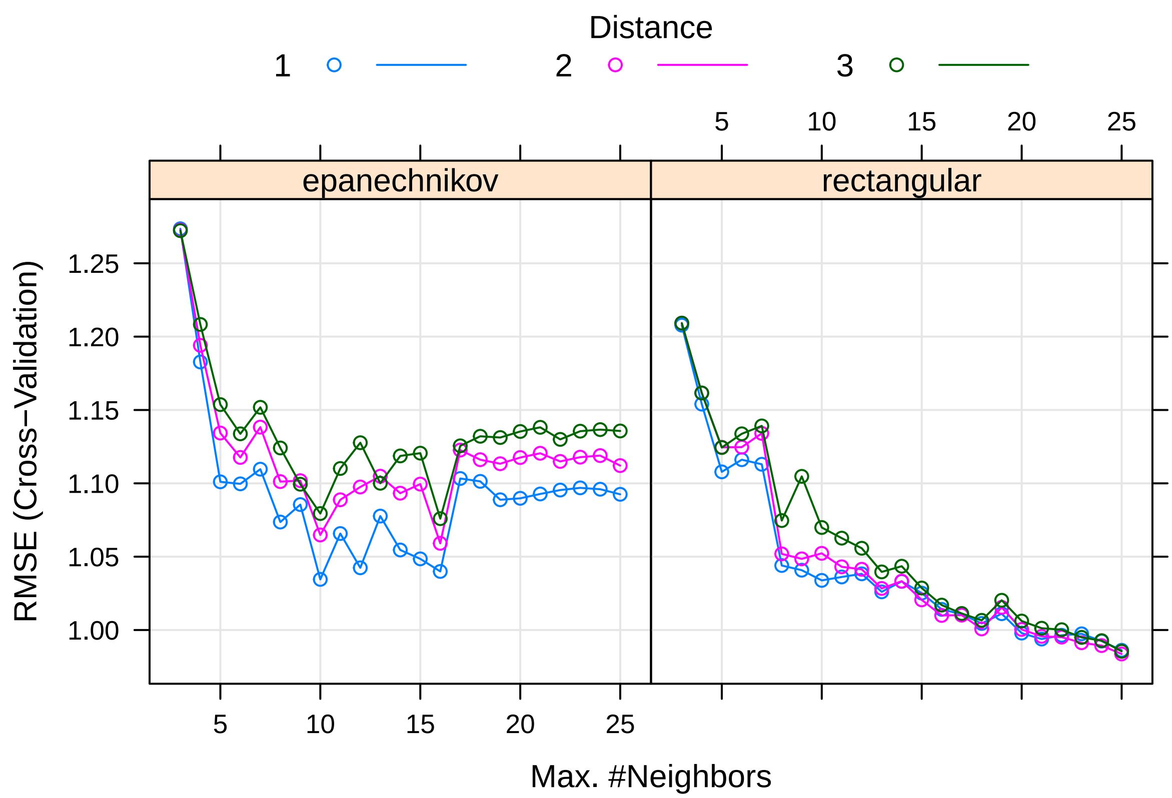
Model 2: K-nearest neighbors to predict rank ordered exploration-promotion from pedagogy text sample.

```
##   parameter     class           label
## 1      kmax   numeric Max. #Neighbors
## 2  distance   numeric        Distance
## 3    kernel character          Kernel
```

```
# Hyperparameter Tuning Grid


  grid <- expand.grid(kmax    = 3:25,
                     distance = c(1,2,3),
                     kernel   = c('epanechnikov','rectangular'))
 # grid
```

```
require(doParallel)

ncores <- 8

cl <- makePSOCKcluster(ncores)

registerDoParallel(cl)


# Train the model

textdummy$source <- as.vector(textdummy$source)
textdummy$rank <- as.vector(textdummy$rank)
textdummy$condition <- as.vector(textdummy$condition)
textdummy$funct <- as.vector(textdummy$funct)
textdummy$id <- as.vector(textdummy$id)

outcome <- c('rank')

ID <- c('id')

categorical <- c('condition', 'source', 'funct')

blueprint_textdummy <- recipe(x  = textdummy,
                    vars  = c(categorical, outcome, ID),
                    roles = c(rep('predictor',3), 'outcome', 'ID'))
```

```
caret_knn <- caret::train(blueprint_textdummy,
                                                data      = rank_tr,
                                                method    = "kknn",
                                                trControl = cv,
                                                tuneGrid  = grid)

plot(caret_knn)
```




```
caret_knn$bestTune
```

```
##     kmax distance      kernel
## 136   25        2 rectangular
```

```
# checking performance of knn algorithm on test dataset
predicted_te <- predict(caret_knn$finalModel, newdata = rank_te)


# r-square
cor(rank_te$rank, predicted_te)^2
```

```
## [1] 0.6127713
```

```
# rmse
sqrt(mean((rank_te$rank - predicted_te)*2))
```

```
## [1] 0.85
```

```
# mae
mean(abs(rank_te$rank -predicted_te))
```

```
## [1] 0.94125
```

Results Model 2

r-square: 0.613 rmse: 0.85 mae: 0.941

```
# omsi <- import(here("data", "omsidata.xlsx"))
omsi <- import(here("data", "omsi.xlsx"))

outcome <- c('squeaker_discovered')

ID <- c('participant')

categorical <- c('condition', 'gender', 'total_time', 'unique_actions', 'squeak_time', '

# old blueprint for data w/ character values:
# blueprint <- recipe(x  = omsi,
#                     vars  = c(categorical, outcome, ID),
#                     roles = c(rep('predictor'),'outcome','ID')) %>%
#  step_indicate_na(all_of(categorical)) %>%
#  step_zv(all_of(categorical)) %>%
#    step_num2factor(outcome,
#                   transform = function(x) x + 1,
#                   levels=c('Y', 'N')) %>%
#  step_num2factor(condition,
#                   transform = function(x) x + 1,
#                   levels=c('ped1', 'baseline', 'interrupted', 'naive'))


# omsi blueprint
 blueprint <- recipe(x  = omsi,
                    vars  = colnames(omsi),
```

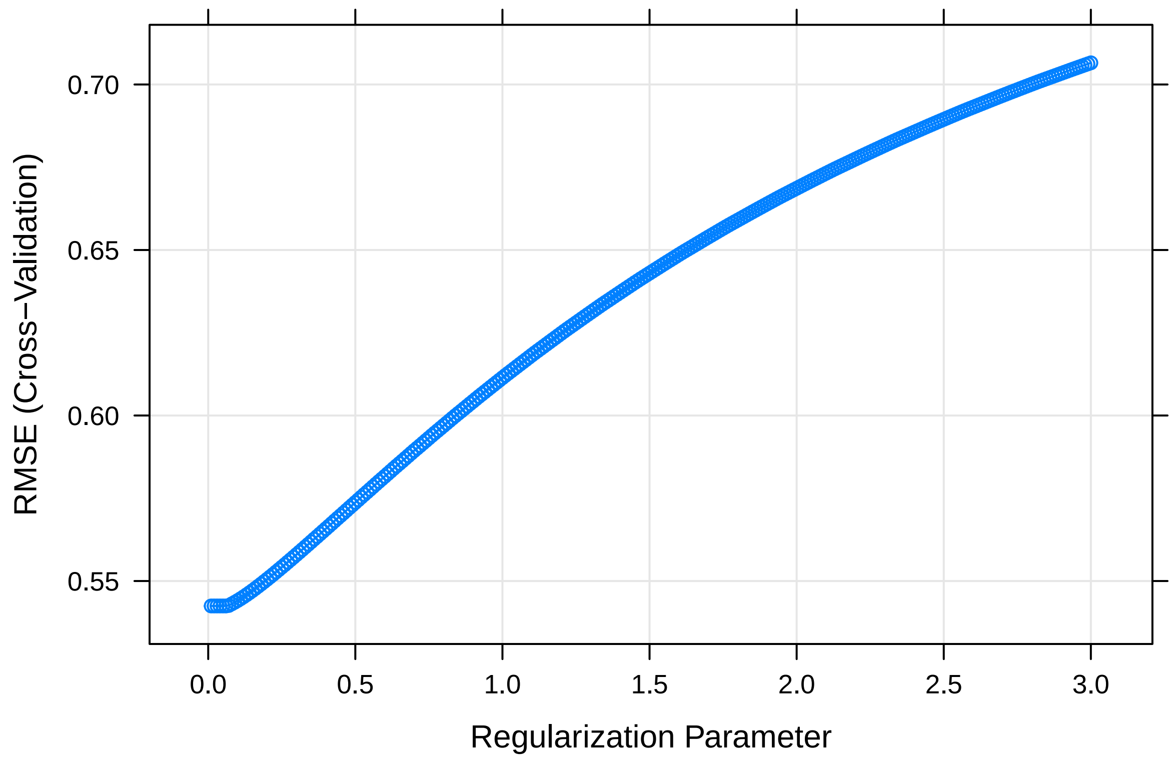
```
                    roles = c(rep('predictor',8), 'outcome', 'ID')) %>%
   step_zv(all_numeric()) %>%
    step_nzv(all_numeric()) %>%
    step_impute_mean(all_numeric()) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_corr(all_numeric(),threshold=0.9)



# splitting data into training and testing
# Let the training data have the 80% of cases and the test data have the 20% of the ca

set.seed(11102021)  # for reproducibility

om     <- sample(1:nrow(omsi), round(nrow(omsi) * 0.8))
omsi_tr  <- omsi[om, ]
omsi_te<- omsi[-om, ]

# 10-fold cross-validation without regularization

omsi_tr = omsi_tr[sample(nrow(omsi_tr)),]

  # Creating 10 folds with equal size

folds = cut(seq(1,nrow(omsi_tr)),breaks=10,labels=FALSE)

  # Creating the list for each fold
sen.indices <- vector('list',10)
```

```r
      for(i in 1:10){
        sen.indices[[i]] <- which(folds!=i)
      }



cv <- trainControl(method    = "cv",
                   index           = sen.indices)

# Train the model
grid <- data.frame (alpha = 0, lambda= seq(0.01, 3, .01))

ridge <- caret::train(blueprint,
                         data      = omsi_tr,
                         method    = "glmnet",
                         trControl = cv,
                         tuneGrid = grid)

# ridge$results
ridge$bestTune
```

Model 3: Logistic regression predicting whether a particular object function was discovered.

```
##   alpha lambda
## 6     0   0.06
```

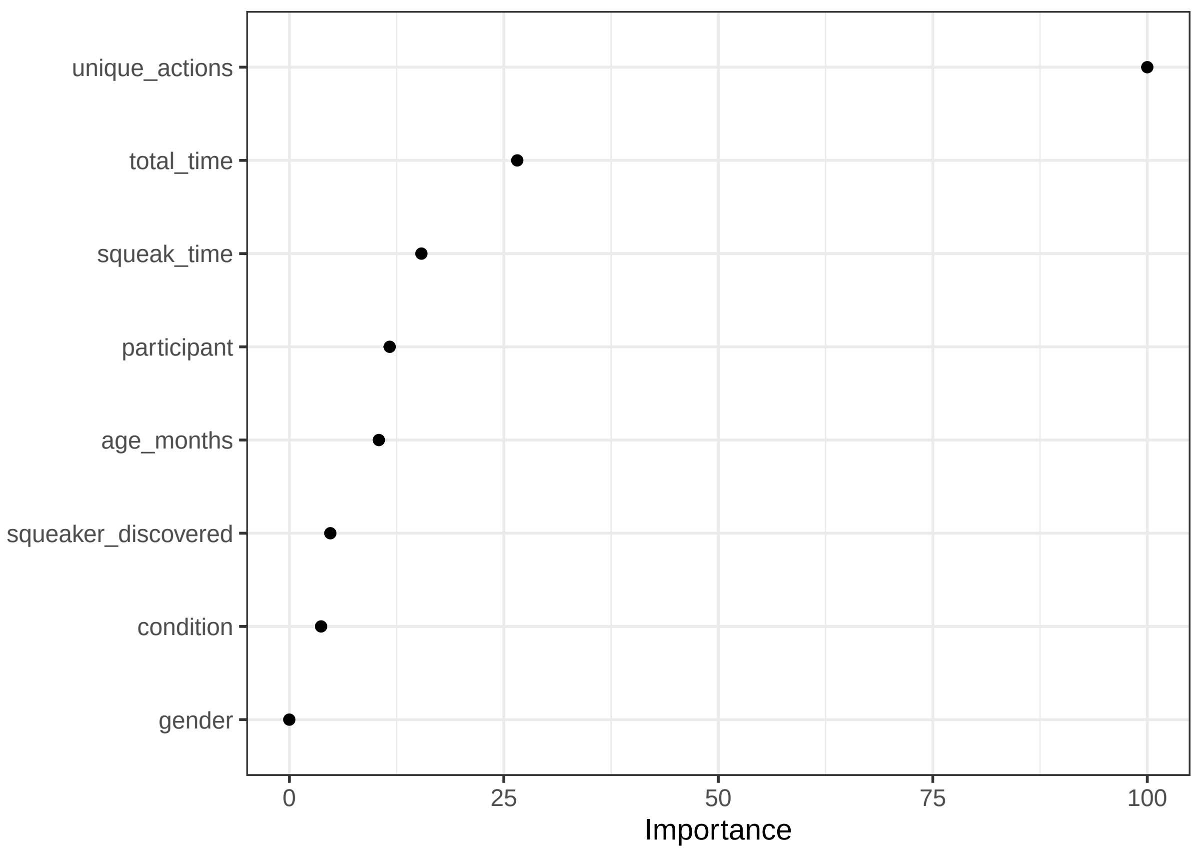
```
plot(ridge)
```



126

```
# Evaluate the model on the test data

# Predict the probabilities for the observations in the test dataset

predicted_te_ridge <- predict(ridge, omsi_te)

rsq_te <- cor(predicted_te_ridge, omsi_te$squeaker_discovered)^2
# rsq_te

mae_te <- mean(abs(omsi_te$squeaker_discovered - predicted_te_ridge))
# mae_te
```

```
rmse_te <- sqrt(mean((omsi_te$squeaker_discovered -predicted_te_ridge)^2))
# rmse_te




# variable importance
require(vip)

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



127

```r
coefs <- coef(ridge$finalModel,ridge$bestTune$lambda)

ind   <- order(abs(coefs),decreasing=T)

head(as.matrix(coefs[ind[-1],]),10)
```

```
##                            [,1]
## unique_actions       0.545425593
## total_time           0.147585830
## squeak_time         -0.087064667
## participant         -0.067041738
## age_months           0.060184145
## squeaker_discovered -0.029554122
## condition           -0.023706733
## gender              -0.003659261
```

Results Model 3

rsq: 0.00051 mae: 0.674 rmse: 0.833

The most important varialbes impacting whether or not the squeaker was discovered were (in order): number of unique actions, total play time, squeaker play time, participant, age in months, condition, and gender.

Discussion

References

We used packages from R [Version 4.1.1; R Core Team (2021)] and the R-package *papaja* [Version 0.1.0.9997; Aust and Barth (2020)] for all our analyses.

Aust, F., & Barth, M. (2020). *papaja: Create APA manuscripts with R Markdown.* Retrieved from https://github.com/crsh/papaja

R Core Team. (2021). *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Table 1

(#tab:sent anal)Sentiment by analysis tool

Pos bing	Neg bing	Pos nrc	Neg nrc	Pos loughran	Neg loughran	Pos afinn	Neg afinn	Total Positive	Total Negative
64	66	277	73	18	73	61	10	420	222
41	116	274	94	40	28	54	26	409	264

Note. Positive (Pos) and Negative (Neg) sentiment analysis by individual words using 4 analysis tools: bing, nrc, loughran, and afinn. Results demonstrate that Total Positive & Negative sentiment was roughly equal by condition (enhance or constrain). However, positive sentiment was slightly higher for the enhance condition and negative sentiment was slight higher for the constrain condition, which is the expected result.