STAT 222: Project 4
Countable Care

Yuan He, Lindsey Lee, Jin Rou New, Tianyi Zhu
University of California, Berkeley

April 9, 2015

Abstract

A Bayesian analysis of a randomized response survey was carried out to study the use of cognition-enhancing drugs in the UCB student population and gender-specific differences in this use. About three quarters of the student population do not use cognition-enhancing drugs, while about a tenth use them with a prescription and 14% use them without a prescription. We concluded that there is no statistically significant difference in drug use between male and female students.

Keywords: countable care, driven data competition, machine learning

1 Introduction

The competition we have chosen to work on is DrivenData.orgs Countable Care: Modeling Women's Health Care Decisions. The link to the competition can be found at: http://www.drivendata.org/competitions/6/. DrivenData is the equivalent of Kaggle for social causes.

2 Data

2.1 Data description

The data consists of responses by women in United States to the National Survey of Family Growth carried out by the United States Center for Disease Control and Prevention. Questions in the survey span topics such as demographics, marriage and reproductive health. There are a total of 14,644 observations, with each observation representing an individuals responses for one release of the survey (an individual may have participated in multiple rounds of the survey), and a total of 1378 features, including the survey round and 116 numeric (standardized), 1050 categorical and 211 ordinal features that are responses to survey questions. The data has been obfuscated, so it is not given what each feature corresponds to in real terms.

An overview of the survey data is given in Table 1.

	Male	Female	Missing	Total
All observations	62	46	9	117
Excluding fraternity observations	41	45	8	94

Table 1: **Summary of the survey observations**

2.2 Data problems

The data is very sparse, with 83% of the data missing in the train set. There are many features with a high proportion of missing values as shown in Fig 1. This is because are many missing values as some survey questions depend on the response to previous survey questions and may be skipped.

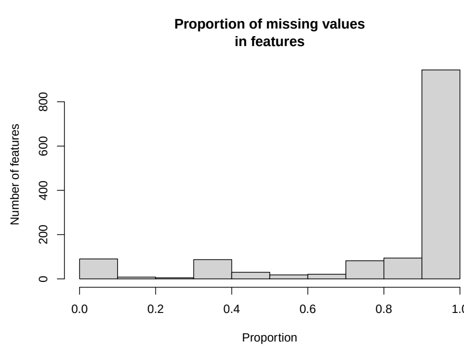

Figure 1: **Proportion of missing values in features.**

Secondly, columns of predictors/whether a women goes to a healthcare provider for different services may be dependent on each other. Fitting separate independent models for each health care service may result in a loss of information.

2.3 Data processing

Features that contain only missing values or one unique value (i.e. constant value for all women) were dropped from the data. There were 14 and 20 of such features respectively.

3 Methods

3.1 Feature engineering

We engineered a number of features, including:

- Number of numeric features with missing values for each woman
- Number of ordinal features with missing values for each woman
- Number of categorical features with missing values for each woman

3.2 Models

We fitted a variety of models, including:

- Gradient Boosting Machine (GBM)

3.3 Computation

All calculations were carried out in the `R` programming language, with additional functions from the `caret` package. Annotated code is given in the Appendix in Section 6.

4 Results

4.1 Prediction objective and evaluation metric

Binary outcomes denoting if a woman went to a healthcare provider for each of 14 services in the 12 months preceding the survey are given in the training set. The prediction objective is to predict the probability for each of the 3,661 women in the test set of going to a healthcare provider for each of the 14 services. In other words, 3,661 x 14 predicted probabilities are required. The 14 services are not mutually exclusive; a woman can go to more than one service in the 12 months.

The evaluation metric here is the logarithmic loss, defined by $-\frac{1}{n}$. (FILL IN!)

4.2 Prediction results

A summary of the performance of our models on the test set is given in Table 2.

Data set	Model	Log loss
Missing value cut-off of 50%	Gradient boosting method	0.2796
Missing value cut-off of 80%	Gradient boosting method	0.2812

Table 2: **Summary of model performance on test set.**

5 Discussion

6 References

Appendix

```r
rm(list = ls())
gc()
run_on_server <- TRUE
if (!run_on_server)
  setwd("~/Copy/Berkeley/stat222-spring-2015/stat222sp15/projects/countable-care")
data_dir <- "data"
fig_dir <- "fig"
results_dir <- "results"
dir.create(fig_dir, showWarnings = FALSE)
dir.create(results_dir, showWarnings = FALSE)
dir.create("submit", showWarnings = FALSE)
get_notifications <- ifelse(run_on_server, TRUE, FALSE)
if (get_notifications) {
  library(RPushbullet)
  # options(error = function() { # Be notified when there is an error
  #   pbPost("note", "Error!", geterrmessage(), recipients = c(1, 2))
  # })
}

write_submission <- function(probs, model_name) {
  file_path <- file.path("submit", paste0(model_name, ".csv"))
  submit <- read.csv("data/SubmissionFormat.csv")
  submit[, 2:ncol(submit)] <- probs
  if (file.exists(file_path))
    stop(paste0(file_path, " already exists!"))
  write.csv(submit, file.path(file_path), row.names = FALSE)
  message(paste0("Results written to ", file_path))
}

seed <- 12345
set.seed(seed)
library(caret)
library(e1071)
# List all models in caret
# names(getModelInfo())

# Load data
prop_missing_cutoff <- 0.5
load(file = file.path(data_dir, paste0("data_cutoff", prop_missing_cutoff, ".rda")))
train <- data$train
test <- data$test
ytrain <- data$ytrain

# Create data partitions of 80% and 20%
ntrain <- nrow(train)
train_indices <- sample(1:ntrain)[1:floor(ntrain*0.8)]
train_val <- train[-train_indices, ]

# Set up caret models
train_control <- trainControl(method = "cv", number = 10, returnResamp = "none")
```

```r
mod_types <- c("gbm", "rf")
mod <- list()
probs <- matrix(NA, nrow(test), ncol(ytrain))
for (mod_type in mod_types) {
  for (svc_index in 1:ncol(ytrain)) {

    # Testing!!!
    # mod_type <- "knn"
    # train_indices <- 1:100
    # svc_index <- 1

    # Train all the models with train data
    mod[[svc_index]] <- train(train[train_indices, ], ytrain[train_indices, svc_index],
                              method = mod_type, trControl = train_control)

    # Predict on test data
    probs[, svc_index] <- predict(object = mod[[svc_index]], newdata = test,
                                  type = "prob")$yes

    # Get predictions for each model and add them back to themselves
    # train_val[[paste0(mod_type, "_PROB")]] <- predict(mod[[svc_index]], train_val, type = "prob")
    # test[[paste0(mod_type, "_PROB")]] <- predict(mod[[svc_index]], test, type = "prob")

    # Run an ensemble model to blend all the predicted probabilities
    # mod_ensemble[[svc_index]] <- train(train_val, ytrain[-train_indices, svc_index],
    #                                    method = "lasso", trControl = train_control)

    # Predict on test data
    # preds <- predict(mod_ensemble[[svc_index]], test, type = "prob")
  }
  write_submission(probs, paste0(mod_type, "_cutoff", prop_missing_cutoff))
  save(mod, file = file.path(results_dir,
                             paste0("mod_", mod_type, "_cutoff", prop_missing_cutoff, ".rda")))
  if (get_notifications)
    pbPost(type = "note",
           title = "stat222",
           body = paste0(mod_type, " done!"),
           recipients = c(1, 2))
}
```

```r
rm(list = ls())
gc()
setwd("~/Copy/Berkeley/stat222-spring-2015/stat222sp15/projects/countable-care")
data_dir <- "data"
fig_dir <- "fig"
results_dir <- "results"
dir.create(fig_dir, showWarnings = FALSE)
dir.create(results_dir, showWarnings = FALSE)
dir.create("submit", showWarnings = FALSE)
prop_missing_cutoff <- 0.8

# Read in data
train_readin <- read.csv(file.path(data_dir, "train_values.csv"),
```

```r
                              stringsAsFactors = FALSE, na.strings = "")
ytrain <- read.csv(file.path(data_dir, "train_labels.csv"))
test_readin <- read.csv(file.path(data_dir, "test_values.csv"),
                             stringsAsFactors = FALSE, na.strings = "")

# Data exploration
# Column types
colnames_all <- names(train_readin)
colnames_type <- sapply(colnames_all, function(x) strsplit(x, "_")[[1]][1])
table(colnames_type[-c(1, 2)])

# Check proportion of missing values in features
prop_missing <- sapply(train_readin, function(x) mean(is.na(x)))
par(mar = c(4.5, 4.5, 4.5, 2))
pdf(file.path(fig_dir, "prop-missing-in-columns.pdf"), width = 7, height = 5)
hist(prop_missing, main = "Proportion of missing values\nin features",
     xlab = "Proportion", ylab = "Number of features", col = "lightgrey")
dev.off()

# Missing pattern plot
# library(mi)
# missing.pattern.plot(train)

# Data processing
# Check original number of features, not including id
ncol(train_readin[, -1]) # 1378

# Check for features with only constant values
cols_constant <- sapply(train_readin, function(x) length(unique(x)) == 1)
sum(cols_constant) # 20
# names(train_readin)[cols_constant]

# Feature engineering
# Column types
colnames_all <- names(train_readin)
colnames_type <- sapply(colnames_all, function(x) strsplit(x, "_")[[1]][1])
table(colnames_type[-c(1, 2)])
cols_numeric <- colnames_type == "n"
cols_ordinal <- colnames_type == "o"
cols_categorical <- colnames_type == "c"

# Number of missing numeric/ordinal/categorical features
num_missing_numeric <- apply(train_readin[, cols_numeric], 1, function(x) sum(is.na(x)))
num_missing_ordinal <- apply(train_readin[, cols_ordinal], 1, function(x) sum(is.na(x)))
num_missing_categorical <- apply(train_readin[, cols_categorical], 1, function(x) sum(is.na(x)))
hist(num_missing_numeric, freq = FALSE,
     main = "Distribution of missing\nnumeric features across all women",
     xlab = "Number of missing numeric features")
hist(num_missing_ordinal, freq = FALSE,
     main = "Distribution of missing\nordinal features across all women",
     xlab = "Number of missing ordinal features")
hist(num_missing_categorical, freq = FALSE,
     main = "Distribution of missing\ncategorical features across all women",
```

```r
     xlab = "Number of missing categorical features")
num_missing_numeric_test <- apply(test_readin[, cols_numeric], 1, function(x) sum(is.na(x)))
num_missing_ordinal_test <- apply(test_readin[, cols_ordinal], 1, function(x) sum(is.na(x)))
num_missing_categorical_test <- apply(test_readin[, cols_categorical], 1, function(x) sum(is.na(x)))

# Check for features with proportion of missing values > prop_missing_cutoff
prop_missing <- sapply(train_readin, function(x) mean(is.na(x)))
cols_missing <- prop_missing > prop_missing_cutoff
sum(cols_missing) # 1159 for 0.5, 1038 for 0.8

# Remaining number of features, not including id
sum(!cols_constant & !cols_missing) - 1 # 213 for 0.5, 334 for 0.8

# Remove above features and id
train <- train_readin[, names(train_readin) != "id" & !cols_constant & !cols_missing]
test <- test_readin[, names(train_readin) != "id" & !cols_constant & !cols_missing]

# Column types
colnames_all <- names(train)
colnames_type <- sapply(colnames_all, function(x) strsplit(x, "_")[[1]][1])
table(colnames_type[-c(1, 2)])
cols_numeric <- colnames_type == "n"
cols_ordinal <- colnames_type == "o"
cols_categorical <- colnames_type == "c"

# Missing value imputation for remaining features
# Numeric features: Set as 0
for (i in 1:sum(cols_numeric)) {
  train[, cols_numeric][, i] <- ifelse(is.na(train[, cols_numeric][, i]),
                                       0, train[, cols_numeric][, i])
  test[, cols_numeric][, i] <- ifelse(is.na(test[, cols_numeric][, i]),
                                      0, test[, cols_numeric][, i])
}
# Ordinal features: Set as -1
# table(sapply(train[, cols_ordinal], min, na.rm = TRUE)) # min is 0 or 1
for (i in 1:sum(cols_ordinal)) {
  train[, cols_ordinal][, i] <- as.integer(ifelse(is.na(train[, cols_ordinal][, i]),
                                              -1, train[, cols_ordinal][, i]))
  test[, cols_ordinal][, i] <- as.integer(ifelse(is.na(test[, cols_ordinal][, i]),
                                             -1, test[, cols_ordinal][, i]))
}

# Categorical features
# Check for: i) features with categories in test but not train set.
# ii) features with missing values in test but not train set
cols_unknownlevels <- NULL
cols_nomissingintrain <- NULL
for (i in 1:sum(cols_categorical)) {
  if (any(is.na(test[, cols_categorical][, i])) & all(!is.na(train[, cols_categorical][, i]))) {
    print(i)
    cols_nomissingintrain <- c(cols_nomissingintrain, i)
  }
  if (any(!is.na(test[, cols_categorical][, i]) &
```

```
            !(test[, cols_categorical][, i] %in% unique(train[, cols_categorical][, i])))) {
    cols_unknownlevels <- c(cols_unknownlevels, i)
    levels_train <- unique(train[, cols_categorical][, i])
    levels_train <- ifelse(is.na(levels_train), "missing", as.character(levels_train))
    levels_test <- unique(test[, cols_categorical][, i])
    levels_test <- ifelse(is.na(levels_test), "missing", as.character(levels_test))
    if (!("missing" %in% levels_train)) {
      print(i)
      print(levels_test[!(levels_test %in% levels_train)])
      print("---")
      cols_nomissingintrain <- c(cols_nomissingintrain,
                                 rep(i, length(levels_test[!(levels_test %in% levels_train)])))
    }
  }
}

# Categorical features: Set as new category missing
for (i in 1:sum(cols_categorical)) {
  train[, cols_categorical][, i] <- as.factor(ifelse(is.na(train[, cols_categorical][, i]),
                                                 "missing", train[, cols_categorical][, i]))
  test[, cols_categorical][, i] <- as.factor(ifelse(is.na(test[, cols_categorical][, i]),
                                                  "missing",
                                                  ifelse(!(test[, cols_categorical][, i] %in%
                                                    unique(train[, cols_categorical][, i])),
                                                    "missing", test[, cols_categorical][, i])))
}

# Set values in test set to most frequently-occurring category in train set for:
# i) features with categories in test but not train set.
# ii) features with missing values in test but not train set
for (c in seq_along(cols_nomissingintrain)) {
  i <- cols_nomissingintrain[c]
  value_new <- names(which.max(table(test[, cols_categorical][, i])))
  test[, cols_categorical][, i] <- as.factor(ifelse(as.character(test[, cols_categorical][, i]) ==
                                                    "missing",
                                                  value_new,
                                                  as.character(test[, cols_categorical][, i])))
}

# Add engineered features to data
train$num_missing_numeric <- num_missing_numeric
train$num_missing_ordinal <- num_missing_ordinal
train$num_missing_categorical <- num_missing_categorical
test$num_missing_numeric <- num_missing_numeric_test
test$num_missing_ordinal <- num_missing_ordinal_test
test$num_missing_categorical <- num_missing_categorical_test

# Convert release variable to factor, else it throws error
train$release <- as.factor(train$release)
test$release <- as.factor(test$release)

# Convert prediction label to alphabetical factor,
# else it throws error in caret::predict
```

```r
for (i in 1:ncol(ytrain))
  ytrain[, i] <- factor(ifelse(ytrain[, i] == 1, "yes", "no"))

# Save processed data to file
data <- list(train = train,
             ytrain = ytrain[, -1], # Drop id column
             test = test,
             prop_missing_cutoff = prop_missing_cutoff)
save(data, file = file.path(data_dir, paste0("data_cutoff", prop_missing_cutoff, ".rda")))
```