Solution

Ahmed Alawami

July 18, 2017

Set up

First we set up the problem by:

- loading packages and data.
- cleaning up the data and imputing missing values.

```
### load packages ------------------------------------------------------------------------
suppressMessages(purrr::walk(c("doMC", "gbm", "randomForest", "xgboost", "stringr",
                               "caret", "magrittr", "lubridate", "forcats", "tidyverse"),
                             library, character.only = TRUE))

# Register mutliple cores for parallelization on caret
# registerDoMC(cores = 4)

### read in data -------------------------------------------------------------------------
suppressMessages(survey <- read_csv("../data/survey.csv"))

# Data Checks
glimpse(survey)

survey[, !(names(survey) %in% c('Timestamp', 'comments'))]  %>% lapply(unique)
survey[, !(names(survey) %in% c('Timestamp', 'comments'))]  %>% lapply(table)

# Data Cleaning
survey %<>% mutate(
  Timestamp = ymd_hms(Timestamp),
  Country = factor(Country),
  # Most are not self-employed given the small size of missing values
  # we may impute them as "No"
  self_employed = factor(self_employed, levels = c("No", "Yes")),
  self_employedImputed = ifelse(is.na(self_employed), "No", "Yes"),
  self_employedImputed = factor(self_employedImputed, levels = c("No", "Yes")),
  family_history = factor(family_history, levels = c("No", "Yes")),
  treatment = factor(treatment, levels = c("No", "Yes")),
  work_interfere = ifelse(is.na(work_interfere), "Not Applicable", work_interfere),
  work_interfere = factor(work_interfere, levels = c("Not Applicable", "Never", "Rarely",
                                                     "Sometimes", "Often")),
  no_employees = factor(no_employees, levels = c("1-5", "6-25", "26-100", "100-500",
                                                 "500-1000", "More than 1000")),
  remote_work = factor(remote_work, levels = c("No", "Yes")),
  tech_company = factor(tech_company, levels = c("No", "Yes")),
  benefits = factor(benefits, levels = c("No", "Don't know", "Yes")),
  care_options = factor(care_options, levels = c("No", "Not sure", "Yes")),
  wellness_program = factor(wellness_program, levels = c("No", "Don't know", "Yes")),
  seek_help = factor(seek_help, levels = c("No", "Don't know", "Yes")),
  anonymity = factor(anonymity, levels = c("No", "Don't know", "Yes")),
  leave = factor(leave, levels = c("Very easy", "Somewhat easy", "Don't know",
```

```
                                    "Somewhat difficult", "Very difficult")),
  mental_health_consequence = factor(mental_health_consequence,
                                     levels = c("No", "Maybe", "Yes")),
  phys_health_consequence = factor(phys_health_consequence,
                                   levels = c("No", "Maybe", "Yes")),
  coworkers = factor(coworkers, levels = c("No", "Some of them", "Yes")),
  supervisor = factor(supervisor, levels = c("No", "Some of them", "Yes")),
  mental_health_interview = factor(mental_health_interview,
                                   levels = c("No", "Maybe", "Yes")),
  phys_health_interview = factor(phys_health_interview, levels = c("No", "Maybe", "Yes")),
  mental_vs_physical = factor(mental_vs_physical, levels = c("No", "Don't know", "Yes")),
  obs_consequence = factor(obs_consequence, levels = c("No", "Yes"))
)

# Fix unreasonable age enteries
survey %<>% mutate(
  Age = ifelse(Age < 18 | Age > 100, NA, Age),
  ageWithheld = factor(ifelse(is.na(Age), "Unreported", "Reported"))
)
summary(survey$Age)
# survey$Age %>% hist()

survey %<>% mutate(
  AgeImputed = ifelse(is.na(Age), median(Age, na.rm = TRUE), Age)
)

# Standardize gender entries
# Note: spelling errors are likely to be correlated with country
survey %<>% mutate(binnedGender = Gender)
survey$binnedGender[survey$Gender %in%
                      c("M", "m", "male", "Mal", "Make", "maile", "Malr", "Mail", "msle",
                        "Man", "Cis Male", "Male (CIS)", "cis male", "Cis Man")] <- "Male"
survey$binnedGender[survey$Gender %in%
                      c("female", "F", "f", "Woman", "woman", "femail", "Cis Female",
                        "cis-female/femme", "Female (cis)", "Femake")] <- "Female"
survey$binnedGender[survey$Gender %in%
                    c("Trans-female", "Genderqueer", "Trans woman", "Female (trans)")] <-
                      "Trans/Intersex/Questioning"
survey$binnedGender[survey$Gender %in%
  c("Nah", "All", "fluid", "Guy (-ish) ^_^", "ostensibly male, unsure what that really means",
    "p", "A little about you", "Neuter", "Agender", "Enby", "non-binary", "Genderqueer",
    "male leaning androgynous", "queer/she/they", "Male-ish", "Androgyne", "something kinda male?",
    "queer")] <- "Trans/Intersex/Questioning"

# Create new features for length of comment
survey %<>% mutate(
  binnedGender = factor(binnedGender,
                        levels = c("Male", "Female", "Trans/Intersex/Questioning")),
  commented = ifelse(is.na(comments) | comments == "-", "Didn't comment", "commented") %>%
    factor,
  # Eliminate leading and trailing spaces and count the length
  commentLength = ifelse(is.na(comments),
                         0,
```

```
                          nchar(gsub("^\\s+|\\s+$|^>\\s*", "", comments)))
  )
# survey$comments %>% nchar %>% table

summary(survey)

survey %<>% mutate(
  state = ifelse(Country != "United States", "nonUS", state),
  state = ifelse(is.na(state), "Unspecified", state) %>% factor
)

# Create a variable for continent. Since there are a lot of countries most have very few
# observations.
library(gapminder)

survey %<>% left_join(gapminder %>% distinct(country, continent),
                      by = c('Country' = 'country'))
survey %<>% mutate(
  continent = as.character(continent)
)
survey %<>% mutate(
  continent = replace(continent, Country == 'Russia', 'Asia'),
  continent = replace(continent, Country == 'Bahamas, The', 'Oceania'),
  continent = replace(continent, Country == 'Moldova', 'Europe'),
  continent = replace(continent, Country == 'Georgia', 'Asia'),
  continent = replace(continent, Country == 'Latvia', 'Europe'),
  continent = ifelse(Country %in% c('Canada', 'United States', 'Mexico'),
                     'North America', continent),
  continent = factor(continent),
  Country = replace(Country, Country == 'Bahamas, The', 'The Bahamas'),
  Country = factor(Country)
)
levels(survey$continent)[2] <- "South America"

# Since the US account for a large part of the data it might be helpful to
# add information about the state regionally

library(datasets)

survey %<>% mutate(
  us = ifelse(Country == 'United States', 'US', 'nonUS') %>% factor
)
survey %<>% left_join(tibble(state = c(state.abb, 'DC', 'nonUS', 'Unspecified'),
                             region = c(as.character(state.division), 'South',
                                        'nonUS', 'UnspecifiedUS'))) %>%
  mutate(region = factor(region))
```

```
## Joining, by = "state"
```

```
summary(survey)

# Construct Mental Health Score (mhs)
survey %<>% mutate(
  mhs = (as.numeric(benefits) - 2) + (as.numeric(wellness_program) - 2) +
```

```
    (3 - as.numeric(leave))/2 + (2 - as.numeric(mental_health_consequence)) +
    (2 - as.numeric(coworkers)) + (2 - as.numeric(supervisor)) +
    (as.numeric(mental_health_interview) - 2) + (as.numeric(mental_vs_physical) - 2) +
    (as.numeric(obs_consequence) - 2)
)

set.seed(42)
trainIndex <- sample(nrow(survey), round(0.7 * nrow(survey)))
train <- survey[trainIndex, ]
test <- survey[-trainIndex, ]

# Data percularities:
# - A few observations had the same comments, which is unexpected to be possible.
```

Analysis

There are certainly a few things that are not certain in the variables definition. It would be of interest to know if someone is suffering or undergoing a mental health issue, but alas there is no variable that captures that. The next best thing is the variable `treatment` which observes with a treatment for mental health condition is sought. I build a predictive model to predict `treatment`.

I have run various types of models and it seems that tree based methods do best (though I was able to build a logistic regression model with step-wise variable selection that is as good but it took longer to train) . Below are three models, random forest, gradient boosting and XGboost. Several metrics could have been used such as accuracy, log-loss, and AUC. I choose accuracy for simplicity of exploration and presentation. I used 5-fold cross-validation with a hold out test set. Another thing I could have done is parameter tuning using either random or grid search. I print out the variable importance for random forest. The highest accuracy achieved on the test set is 86.77%.

Random Forest

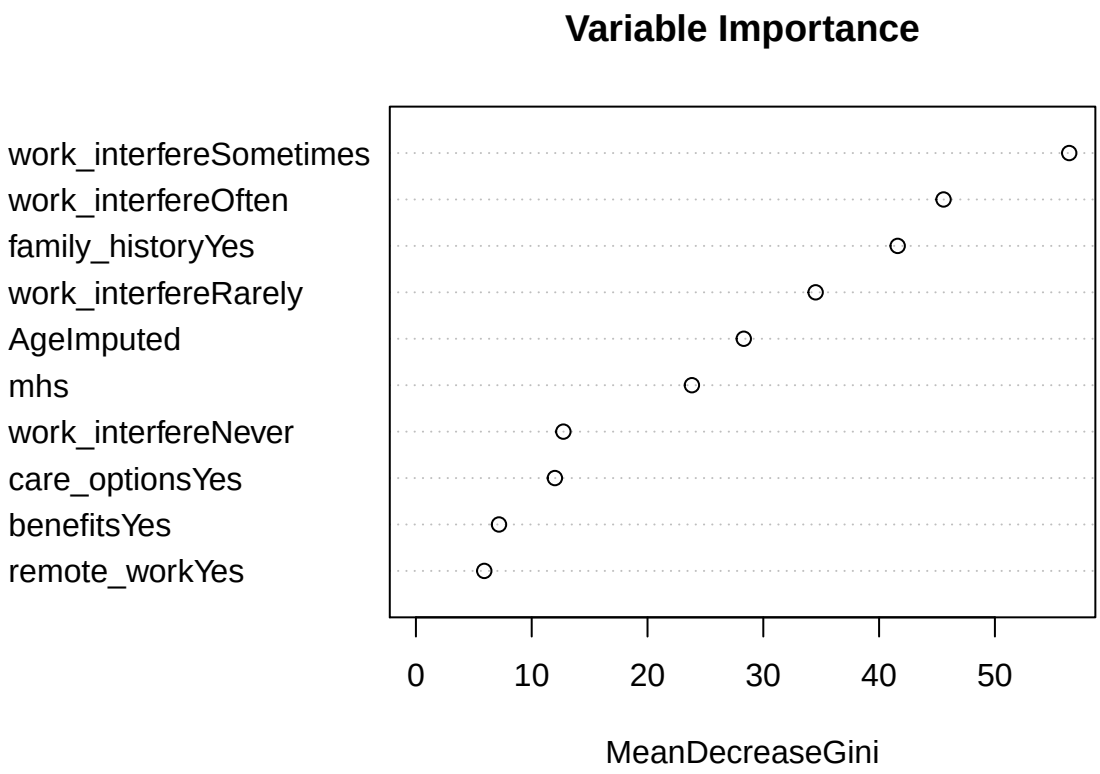
```
# Create trainControl object: myControl
myControl <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  verboseIter = FALSE
)

# Fit random forest model: model

set.seed(42)
rf_model <- train(treatment ~ .,
                  data = train %>% select(-Timestamp, -Age, -Gender, -Country, -state,
                                          -self_employed, -comments),
                  method = "rf",
                  trControl = myControl
)
# varImp(rf_model$finalModel) %>% rownames_to_column("variable") %>% arrange(desc(Overall))
# varImpPlot(rf_model$finalModel, scale = TRUE, main = "Variable Importance")
```

```r
pred <- predict(rf_model, test, type = 'raw')
```

```r
varImpPlot(rf_model$finalModel, scale = TRUE, main = "Variable Importance", n.var = 10)
```




```r
confusionMatrix(test$treatment, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  143  41
##        Yes  20 174
##
##                Accuracy : 0.8386
##                  95% CI : (0.7976, 0.8743)
##     No Information Rate : 0.5688
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6761
##  Mcnemar's Test P-Value : 0.01045
##
##             Sensitivity : 0.8773
##             Specificity : 0.8093
##          Pos Pred Value : 0.7772
##          Neg Pred Value : 0.8969
##              Prevalence : 0.4312
##          Detection Rate : 0.3783
##    Detection Prevalence : 0.4868
##       Balanced Accuracy : 0.8433
```

```
##
##        'Positive' Class : No
##
```

Gradient Boosting

```
# Fit gradient boosting model

set.seed(42)
gbm_model <- train(treatment ~ .,
                   data = train %>% select(-Timestamp, -Age, -Gender, -Country, -state,
                                           -self_employed, -comments),
                   method = "gbm",
                   verbose = FALSE,
                   trControl = myControl
)
# varImp(gbm_model$finalModel) %>% rownames_to_column("variable") %>%
# arrange(desc(Overall)) %>% `[`(1:7,) %>%
#   ggplot(aes(Overall, reorder(variable, Overall))) +
#     geom_point() + ylab('Variable') + xlab('Importance')

pred <- predict(gbm_model, test, type = 'raw')
```

```
confusionMatrix(test$treatment, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  145  39
##        Yes  18 176
##
##                Accuracy : 0.8492
##                  95% CI : (0.8091, 0.8837)
##     No Information Rate : 0.5688
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6973
##  Mcnemar's Test P-Value : 0.008071
##
##             Sensitivity : 0.8896
##             Specificity : 0.8186
##          Pos Pred Value : 0.7880
##          Neg Pred Value : 0.9072
##              Prevalence : 0.4312
##          Detection Rate : 0.3836
##    Detection Prevalence : 0.4868
##       Balanced Accuracy : 0.8541
##
##        'Positive' Class : No
##
```

XGboost

```
# Fit XGboost model: model
set.seed(42)
xg_model <- train(treatment ~ .,
               data = train %>% select(-Timestamp, -Age, -Gender, -Country, -state,
                                       -self_employed, -comments),
               method = "xgbTree",
               trControl = myControl
)
# (xgb.importance(names(train %>% select(-Timestamp, -Age, -Gender, -Country, -state,
# -self_employed, -comments)), model = xg_model$finalModel)) %>%
#   ggplot(aes(Gain, reorder(Feature, Gain))) + geom_point() + ylab('Feature')

pred <- predict(xg_model, test, type = 'raw')
```

```
confusionMatrix(test$treatment, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  143  41
##        Yes   9 185
##
##                Accuracy : 0.8677
##                  95% CI : (0.8294, 0.9002)
##     No Information Rate : 0.5979
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7341
##  Mcnemar's Test P-Value : 1.165e-05
##
##             Sensitivity : 0.9408
##             Specificity : 0.8186
##          Pos Pred Value : 0.7772
##          Neg Pred Value : 0.9536
##              Prevalence : 0.4021
##          Detection Rate : 0.3783
##    Detection Prevalence : 0.4868
##       Balanced Accuracy : 0.8797
##
##        'Positive' Class : No
##
```