Speed Dating Data

Arturo Pérez

22/6/2020

1 Overview

In this project we are going to explore the Speed Dating Data compiled by Ray Fisman and Sheena Iyengar at Columbia Business School for their paper *Gender Differences in Mate Selection: Evidence From a Speed Dating Experiment*. The data in the set was gathered from participants in experimental speed dating events from 2002-2004. The participants would have a four minute "first date" with every other participant of the opposite sex. At the end of their four minutes, participants were asked if they would like to see their date again. If they both say yes we have a match!

The goal of this project is to find out to what extent a match can be predicted based solely on prior information about individuals.

After downloading the dataset we need to select relevant information for our purposes. There are lots of variables collected before, during and after the experiment and only some of them are aligned with our objective. Once we have a curated dataset, we will test two model to predict matches: in the first one we will try to predict the outcome as linear combination of potential features; in the second one we will use machine learning algorithms to take advantage of nonlinearities. We will finally present the results and make our conclusions.

2 Dataset preparation

The Speed Dating Dataset can be downloaded at the following url:

http://www.stat.columbia.edu/~gelman/arm/examples/speed.dating/Speed%20Dating%20Data.csv

There is also a Word document with the description of each variable at:

http://www.stat.columbia.edu/~gelman/arm/examples/speed.dating/Speed%20Dating%20Data%20Key.doc

```
# Required packages

if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")

# File download

if (!file.exists('sdd.csv')) {
  download.file(
```

```
    'http://www.stat.columbia.edu/~gelman/arm/examples/speed.dating/Speed%20Dating%20Data.csv',
    'sdd.csv'
  )
}

# Read data
rawdata <- fread('sdd.csv')
dim(rawdata)
```

```
## [1] 8378  195
```

As we can see, the dataset is composed of 8378 observations of 195 variables.

Each speed date generates two observations in the dataset, each focused on one of the partners. Every observation is composed of a mixture of data: personal information about the "main" participant (age, race, education...), information about the organization of the date and its result, information about the partner and, mainly, opinions of the participant about preferences, competitors and potential partners. We are going to discard lots of variables, because they were collected after the participants met each other and we want to use only prior information.

We are going to extract the following information for each participant:

Personal information[1]:

- **iid**: Identifier of the participant
- **gender**: 0=Female, 1=Male
- **age**: Age of the participant
- **race**: Code representing the race of the participant
- **imprace**: Importance from 1 to 10 of the partner being of the same etnical background
- **imrelig**: Importance from 1 to 10 of the partner being of the same religious background
- **career_c**: Code of the intended career

Interests:

Each of the following variables captures the interest (scored from 1 to 10) of the participant in the corresponding activity:

sports, **tvsports**, **exercise**, **dining**, **museums**, **art**, **hiking**, **gaming**, **clubbing**, **reading**, **tv**, **theater**, **movies**, **concerts**, **music**, **shopping**, **yoga**

Preferences and self perception:

Participants are given 100 points to distribute over 6 attributes (*attractive*, *sincere*, *intelligent*, *fun*, *ambitious*, *shared interests*) to reflect their preferences in potential partners. The scores are recorded in the folloging variables:

attr1_1, **sinc1_1**, **intel1_1**, **fun1_1**, **amb1_1**, **shar1_1**

Participants are also given 100 points to distribute over 5 attributes (*attractive*, *sincere*, *intelligent*, *fun*, *ambitious*) to reflect their self perception. The scores are recorded in the folloging variables:

attr3_1, **sinc3_1**, **fun3_1**, **intel3_1**, **amb3_1**

[1] Some of this variables are categorical and each category is denoted with an integer. You can find the meaning of each code in the appendices.

```r
# We create a data frame with the relevant data for each participant

participants <- rawdata %>% distinct(
  iid, gender, age, race, imprace, imprelig, career_c,
  sports, tvsports, exercise, dining, museums, art, hiking, gaming, clubbing,
  reading, tv, theater, movies, concerts, music, shopping, yoga,
  attr1_1, sinc1_1, intel1_1, fun1_1, amb1_1, shar1_1,
  attr3_1, sinc3_1, intel3_1, fun3_1, amb3_1
)
```

We need to remove records containing NAs.

```r
# Remove participants with missing information
participants <- participants[rowSums(is.na(participants)) == 0,]
```

Now that we have compiled the data about the participants, we are going to build a data frame joining the data of participants who have had a date together, and the result of the date (match or not). We are going to keep only one record for each date. As all the dates were arranged for couples with different gender we can simply use this variable to filter the dataset. Once the joining is made, we can drop the identifiers of the participants:

```r
# Create a data frame with the output (match),
# the data gathered for the first participant,
# and the data gathered for the second participant.

dates <- rawdata %>%
  filter(gender == 0) %>%
  select(iid, pid, match) %>%
  inner_join(participants, by='iid') %>%
  inner_join(participants, by=c('pid' = 'iid')) %>%
  select(-c(iid, pid))

rm(rawdata, participants)
```

Our dataset has a high prevalence of non-match dates. We are going to discard random non-matches to equalize the ratio:

```r
#### THE CODE WAS TESTED USING R VERSION 3.2.3, PAY ATTENTION TO THE SEED ####
set.seed(1) # if using R higher than 3.5 , use `set.seed(1, sample.kind="Rounding")`

# we separate matches and non-marches
matches = dates %>% filter(match == 1)
nonmatches = dates %>% filter(match == 0)

# and join all the matches and an equal number of (random) non-matches
dates <- rbind(matches, nonmatches[sample(nrow(nonmatches), nrow(matches)), ])

rm(matches, nonmatches)
```

Now we are going to split the data into a training set and a validation set.

```r
test_index <- createDataPartition(y = dates$match, times = 1, p = 0.1, list = FALSE)
training <- dates[-test_index,]
validation <- dates[test_index,]
rm(dates, test_index)
```

3 Analysis

As a first approach we are going to explore if we can predict the outcome using a linear combination of the following features:

- **age difference**:

$$agediff = age.x - age.y$$

- **race similarity ponderated by importance**:

$$srace.x = ifelse(race.x == race.y, 1, -1) * imprace.x$$

$$srace.y = ifelse(race.x == race.y, 1, -1) * imprace.y$$

- **importance of religion**: we are going to use the original values $imprelig.x$ and $imprelig.y$
- **intended career**: we will use the original values $career\_c.x$ and $career\_c.y$
- **attractiveness ponderated by preferences of the partner**:

$$attr.x = attr3\_1.x * attr1\_1.y$$

$$attr.y = attr3\_1.y * attr1\_1.x$$

- **sincerity ponderated by preferences of the partner**:

$$sinc.x = sinc3\_1.x * sinc1\_1.y$$

$$sinc.y = sinc3\_1.y * sinc1\_1.x$$

- **fun ponderated by preferences of the partner**:

$$fun.x = fun3\_1.x * fun1\_1.y$$

$$fun.y = fun3\_1.y * fun1\_1.x$$

- **intelligence ponderated by preferences of the partner**:

$$intel.x = intel3\_1.x * intel1\_1.y$$

$$intel.y = intel3\_1.y * intel1\_1.x$$

- **ambitiousness ponderated by preferences of the partner**:

$$amb.x = amb3\_1.x * amb1\_1.y$$

$$amb.y = amb3\_1.y * amb1\_1.x$$

- **shared interests ponderated by preferences of the partner**:

$$shared.x = shared\_interests * shar1\_1.y$$

$$shared.y = shared\_interests * shar1\_1.x$$

We will compute the *shared_interests* as the euclidean distance between the vectors of scores given to each activity by each partner.

The following code defines a function that computes these features given a dataframe with the original variables. We are also going to apply the function to our training and validation sets to be able to use the features for prediction.

```r
extract_features <- function(df) {

  # lets compute the shared interests as a separate data frame to use it later
  shared_interests <- sqrt(
    (df$sports.x    - df$sports.y)^2 +
    (df$tvsports.x  - df$tvsports.y)^2 +
    (df$exercise.x  - df$exercise.y)^2 +
    (df$dining.x    - df$dining.y)^2 +
    (df$museums.x   - df$museums.y)^2 +
    (df$art.x       - df$art.y)^2 +
    (df$hiking.x    - df$hiking.y)^2 +
    (df$gaming.x    - df$gaming.y)^2 +
    (df$clubbing.x  - df$clubbing.y)^2 +
    (df$reading.x   - df$reading.y)^2 +
    (df$tv.x        - df$tv.y)^2 +
    (df$theater.x   - df$theater.y)^2 +
    (df$movies.x    - df$movies.y)^2 +
    (df$concerts.x  - df$concerts.y)^2 +
    (df$music.x     - df$music.y)^2 +
    (df$shopping.x  - df$shopping.y)^2 +
    (df$yoga.x      - df$yoga.y)^2
  )

  return (
    # compute the features
    df %>% mutate(
      agediff = age.x - age.y,
      srace.x = ifelse(race.x == race.y, 1, -1) * imprace.x,
      srace.y = ifelse(race.x == race.y, 1, -1) * imprace.y,
      attr.x = attr3_1.x * attr1_1.y,
      attr.y = attr3_1.y * attr1_1.x,
      sinc.x = sinc3_1.x * sinc1_1.y,
      sinc.y = sinc3_1.y * sinc1_1.x,
      fun.x = fun3_1.x * fun1_1.y,
      fun.y = fun3_1.y * fun1_1.x,
      intel.x = intel3_1.x * intel1_1.y,
      intel.y = intel3_1.y * intel1_1.x,
      amb.x = amb3_1.x * amb1_1.y,
      amb.y = amb3_1.y * amb1_1.x,
      shared.x = shared_interests * shar1_1.y,
      shared.y = shared_interests * shar1_1.x
    ) %>% select(
      # and return only the result and the features
      match, agediff, srace.x, srace.y, imprelig.x, imprelig.y,
      career_c.x, career_c.y, attr.x, attr.y, sinc.x, sinc.y,
      fun.x, fun.y, intel.x, intel.y, amb.x, amb.y, shared.x, shared.y
    )
  )

}

# estract the features from training and validation set
feat_training <- extract_features(training)
```

```
feat_validation <- extract_features(validation)
```

Let's now fit a linear model using the transformed training set and compute the in-sample performance. We will predict a match (1) if the output of our model is greater or equal to 0.5 and a non-match (0) otherwise.

```
# Fit the linear model using the training features and outcomes
model1 <- lm(match ~ agediff + srace.x + srace.y + imprelig.x + imprelig.y +
            career_c.x + career_c.y + attr.x + attr.y + sinc.x + sinc.y +
             fun.x + fun.y + intel.x + intel.y + amb.x + amb.y + shared.x + shared.y,
             data = feat_training)

# Predict the outcome using only the training features
match_hat <- feat_training %>%
  select(-match) %>%
  mutate(match_hat = predict(model1, newdata = .)) %>%
  pull(match_hat)
match_hat <- ifelse(match_hat >= 0.5, 1, 0)

# Compute the accuracy
mean(match_hat == feat_training$match)
```

```
## [1] 0.5849057
```

Let's see the performance in our validation set:

```
# Predict the outcome using only the validation features
match_hat <- feat_validation %>%
  select(-match) %>%
  mutate(match_hat = predict(model1, newdata = .)) %>%
  pull(match_hat)
match_hat <- ifelse(match_hat >= 0.5, 1, 0)

# Compute the accuracy
mean(match_hat == feat_validation$match)
```

```
## [1] 0.6076923
```

A linear model may be not suitable for a complex task as predicting matches between partners. Maybe a machine learning algorithm can exploit patterns not captured in our first model. We are going to try the k-nearest neighbour algorithm:

```
# We need to convert the output to factors
feat_validation$match <- factor(feat_validation$match)
feat_training$match <- factor(feat_training$match, levels = levels(feat_validation$match))

# Let's train the model on the training set
train_knn <- train(
  x = feat_training %>% select(-match),
  feat_training$match,
  method='knn',
  tuneGrid = data.frame(k = seq(5, 21, 2))
)

# Then compute the output for the training set
match_hat_knn <- predict(train_knn, feat_training %>% select(-match), type = "raw")

# And display the accuracy
```

```r
mean(match_hat_knn == feat_training$match)
```

```
## [1] 0.6543739
```

This model has better performance in-sample than our inital model. Let's check the out-sample accuracy:

```r
# Then compute the output for the validation set
match_hat_knn <- predict(train_knn, feat_validation %>% select(-match), type = "raw")

# And display the accuracy
mean(match_hat_knn == feat_validation$match)
```

```
## [1] 0.5384615
```

There was a significant drop in the performance. This may be due to overfitting. Let's try another algorithm: random forests.

```r
# Let's train the model on the training set
train_rf  <- train(
  x = feat_training %>% select(-match),
  feat_training$match,
  method='rf',
  trControl = trainControl(number = 10)
)

# Then compute the output for the training set
match_hat_rf  <- predict(train_rf,  select(feat_training, -match), type = "raw")

# And display the accuracy
mean(match_hat_rf  == feat_training$match)
```

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

The accuracy in the training set is 1. That means that the random forest can make enouh partitions to fit all the data. It is not generalizing so we expect a big drop in out-sample performance:

```r
# Then compute the output for the validation set
match_hat_rf  <- predict(train_rf,  feat_validation %>% select(-match), type = "raw")

# And display the accuracy
mean(match_hat_rf  == feat_validation$match)
```

```
## [1] 0.6153846
```

As we can see, the random forest has a similar performance than our initial model using the same features. Let's see if the random forest is able to generalize when fed with the inicial training set before feature extraction.

```r
# We need to convert the output to factors
validation$match <- factor(validation$match)
training$match <- factor(training$match, levels = levels(validation$match))

# Let's train the model on the training set
train_rf_raw  <- train(
  x = training %>% select(-match),
  training$match,
  method='rf',
  trControl = trainControl(number = 10)
)
```

```r
# Then compute the output for the validation set
match_hat_rf_raw  <- predict(train_rf_raw,  training %>% select(-match), type = "raw")

# And display the accuracy
mean(match_hat_rf_raw  == training$match)
```

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

The random forest still can learn the entire training dataset. The out-sample performance is:

```r
# Then compute the output for the validation set
match_hat_rf_raw  <- predict(train_rf_raw,  validation %>% select(-match), type = "raw")

# And display the accuracy
mean(match_hat_rf_raw  == validation$match)
```

```
## [1] 0.6384615
```

4 Results

Our first model was a linear combination of several features extracted from the initial dataset. The features were carefully chosen due to their apparent relevance when estimating the result of the date between the partners, having nito account race, career, religion, shared interests and proximity between qualities of one partner and the preferences of the other. This simple model achieved an accuracy of about 60%. It may seem a poor performance, but the question we're trying to solve is inherently complex.

Precisely due to this complexity, it can be thought that a linear model is not suitable, that is why we have tried other models based on machine learning. Although we have been able to improve accuracy up to 64% using random forests and using raw data instead of the features used for our linear model, this result seems to be due to chance, because the model is not generalizing at all. The reason may be the nature of the data we are using (we are using lots of features based on discrete scores between 1 and 10) and the relative few observations.

The linear model, although less precise, has the added value of interpretability. If we explore the coefficients obtained, we can get an idea of the most relevant factors in the couple's decision and how they influence positively or negatively the outcome.

```r
model1
```

```
##
## Call:
## lm(formula = match ~ agediff + srace.x + srace.y + imprelig.x +
##     imprelig.y + career_c.x + career_c.y + attr.x + attr.y +
##     sinc.x + sinc.y + fun.x + fun.y + intel.x + intel.y + amb.x +
##     amb.y + shared.x + shared.y, data = feat_training)
##
## Coefficients:
## (Intercept)      agediff      srace.x      srace.y   imprelig.x   imprelig.y
##   2.838e-01    8.542e-04   -1.637e-03    4.927e-03   -6.980e-03   -7.502e-03
##  career_c.x   career_c.y       attr.x       attr.y       sinc.x       sinc.y
##  -4.671e-03   -2.020e-03    2.928e-04   -5.347e-05   -2.053e-04    2.658e-04
##       fun.x        fun.y      intel.x      intel.y        amb.x        amb.y
##   2.381e-04    9.755e-04    8.391e-04    2.275e-04   -4.254e-04    4.982e-04
##    shared.x     shared.y
##  -2.535e-04   -4.894e-04
```

5 Conclusion

In this project we wanted to explore the Speed Dating Data compiled by Ray Fisman and Sheena Iyengar at Columbia Business School. Our goal was to build a prediction model for the outcome of the dates (match or non-match) using only information gathered from the participants before they met. We combined the available data into meaninful features and developed a linear model based on them. Then we built a model based on random forests to see if that kind of algorithms could be able to better exploit the information but we found that this kind of models are not very suitable to our data. The accuracy obtained was not very good for prediction purposes, but the linear model can be used to gain valuable insights about the factors that leads to the decision. The construction of the model was not very rigorous, since its main objective was exploration. As future work, the definitions of the features could be reformulated (for example, transforming all to a similar scale) to facilitate the interpretation of the results. It would also be necessary to analyze the significance of the coefficients obtained in the regression to justify the inclusion of the features in the model. The Speed Dating Dataset has lots of information that can also be used to answer other interesting questions such as if people are consistent between their expectations and their matches, or if there is a halo effect (tendency for an impression created in one attribute to influence opinion in another attribute) in the ratings of the partners.

A Personal Information Codes

A.1 race

1. Black/African American
2. European/Caucasian-American
3. Latino/Hispanic American
4. Asian/Pacific Islander/Asian-American
5. Native American
6. Other

A.2 career_c

1. Lawyer
2. Academic/Research
3. Psychologist
4. Doctor/Medicine
5. Engineer
6. Creative Arts/Entertainment
7. Banking/Consulting/Finance/Marketing/Business/CEO/Entrepreneur/Admin
8. Real Estate
9. International/Humanitarian Affairs
10. Undecided
11. Social Work
12. Speech Pathology
13. Politics
14. Pro sports/Athletics
15. Other
16. Journalism
17. Architecture