# The imposter syndrome is strong with this one

```
library(tidyverse)
library(tidymodels)
library(stacks)
dir <- '20210302'
path_coefs_glm <- here::here(dir, 'coefs_glm.rds')
path_res_knn <- here::here(dir, 'res_knn.rds')
path_res_rf <- here::here(dir, 'res_rf.rds')
path_fit_ens <- here::here(dir, 'fit_ens.rds')
path_preds_holdout <- here::here(dir, 'preds_holdout.csv')
path_probs_holdout <- here::here(dir, 'probs_holdout.csv')
df <-
  here::here(dir, 'sliced-s00e01-data.csv') %>%
  read_csv(guess_max = 20000) %>%
  select(-X1)

df_holdout <-
  here::here(dir, 'sliced-s00e01-holdout.csv') %>%
  read_csv()
```

Wow there are a lot of columns. Some of these numeric ones are really categorical features in disguise.

```
df %>%
  skimr::skim()
```

Data summary

| | |
|---|---|
| Name | Piped data |
| Number of rows | 5934 |
| Number of columns | 119 |
| _______________________ | |
| Column type frequency: | |
| character | 4 |
| numeric | 115 |
| ________________________ | |
| Group variables | None |

**Variable type: character**

| skim_variable | n_missing | complete_rate | min | max | empty | n_unique | whitespace |
|---|---|---|---|---|---|---|---|
| field | 58 | 0.99 | 3 | 51 | 0 | 194 | 0 |
| undergra | 1902 | 0.68 | 2 | 49 | 0 | 213 | 0 |
| from | 74 | 0.99 | 2 | 58 | 0 | 207 | 0 |
| career | 84 | 0.99 | 1 | 77 | 0 | 294 | 0 |

**Variable type: numeric**

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| iid | 0 | 1.00 | 310.30 | 179.37 | 1.00 | 102.00 | 362.0 | 467.00 | 552.00 | ▇▁▃▇▇ |

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| id | 1 | 1.00 | 8.65 | 5.37 | 1.00 | 4.00 | 8.0 | 13.00 | 22.00 | |
| gender | 0 | 1.00 | 0.50 | 0.50 | 0.00 | 0.00 | 1.0 | 1.00 | 1.00 | |
| idg | 0 | 1.00 | 16.68 | 10.65 | 1.00 | 8.00 | 15.0 | 25.00 | 44.00 | |
| condtn | 0 | 1.00 | 1.80 | 0.40 | 1.00 | 2.00 | 2.0 | 2.00 | 2.00 | |
| wave | 0 | 1.00 | 12.25 | 6.85 | 1.00 | 4.00 | 14.0 | 19.00 | 21.00 | |
| round | 0 | 1.00 | 16.24 | 4.44 | 6.00 | 14.00 | 18.0 | 19.00 | 22.00 | |
| position | 0 | 1.00 | 8.77 | 5.40 | 1.00 | 4.00 | 8.0 | 13.00 | 22.00 | |
| positin1 | 1846 | 0.69 | 9.05 | 5.59 | 1.00 | 4.00 | 8.0 | 13.00 | 22.00 | |
| order | 0 | 1.00 | 8.61 | 5.34 | 1.00 | 4.00 | 8.0 | 13.00 | 22.00 | |
| partner | 0 | 1.00 | 8.65 | 5.37 | 1.00 | 4.00 | 8.0 | 13.00 | 22.00 | |
| pid | 10 | 1.00 | 310.61 | 179.36 | 1.00 | 102.00 | 362.0 | 467.00 | 552.00 | |
| match | 0 | 1.00 | 0.17 | 0.37 | 0.00 | 0.00 | 0.0 | 0.00 | 1.00 | |
| int_corr | 148 | 0.98 | 0.19 | 0.30 | -0.73 | -0.03 | 0.2 | 0.42 | 0.91 | |
| samerace | 0 | 1.00 | 0.40 | 0.49 | 0.00 | 0.00 | 0.0 | 1.00 | 1.00 | |
| age_o | 99 | 0.98 | 26.31 | 3.67 | 18.00 | 23.00 | 26.0 | 29.00 | 55.00 | |
| race_o | 68 | 0.99 | 2.80 | 1.25 | 1.00 | 2.00 | 2.0 | 4.00 | 6.00 | |
| pf_o_att | 84 | 0.99 | 24.24 | 13.84 | 2.00 | 15.00 | 20.0 | 30.00 | 100.00 | |
| pf_o_sin | 84 | 0.99 | 17.00 | 7.65 | 0.00 | 10.00 | 20.0 | 20.00 | 60.00 | |
| pf_o_int | 84 | 0.99 | 20.48 | 7.32 | 0.00 | 17.00 | 20.0 | 25.00 | 50.00 | |
| pf_o_fun | 93 | 0.98 | 17.45 | 6.80 | 0.00 | 13.00 | 20.0 | 20.00 | 50.00 | |
| pf_o_amb | 102 | 0.98 | 9.77 | 6.22 | 0.00 | 5.00 | 10.0 | 15.00 | 53.00 | |
| pf_o_sha | 124 | 0.98 | 11.28 | 6.77 | 0.00 | 5.00 | 10.0 | 15.00 | 30.00 | |
| dec_o | 0 | 1.00 | 0.42 | 0.49 | 0.00 | 0.00 | 0.0 | 1.00 | 1.00 | |
| attr_o | 201 | 0.97 | 6.19 | 1.98 | 0.00 | 5.00 | 6.0 | 8.00 | 10.50 | |
| sinc_o | 242 | 0.96 | 7.20 | 1.77 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| intel_o | 254 | 0.96 | 7.36 | 1.58 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| fun_o | 289 | 0.95 | 6.44 | 1.98 | 0.00 | 5.00 | 7.0 | 8.00 | 11.00 | |
| amb_o | 531 | 0.91 | 6.78 | 1.83 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| shar_o | 769 | 0.87 | 5.45 | 2.20 | 0.00 | 4.00 | 6.0 | 7.00 | 10.00 | |
| like_o | 218 | 0.96 | 6.15 | 1.85 | 0.00 | 5.00 | 6.0 | 7.00 | 10.00 | |
| prob_o | 276 | 0.95 | 5.20 | 2.17 | 0.00 | 4.00 | 5.0 | 7.00 | 10.00 | |
| met_o | 333 | 0.94 | 1.97 | 0.21 | 1.00 | 2.00 | 2.0 | 2.00 | 7.00 | |
| age | 90 | 0.98 | 26.31 | 3.68 | 18.00 | 23.00 | 26.0 | 29.00 | 55.00 | |
| field_cd | 77 | 0.99 | 7.41 | 3.89 | 1.00 | 5.00 | 8.0 | 10.00 | 18.00 | |
| mn_sat | 3431 | 0.42 | 1293.85 | 118.83 | 914.00 | 1210.00 | 1310.0 | 1400.00 | 1470.00 | |
| tuition | 2981 | 0.50 | 21023.54 | 6754.80 | 2406.00 | 15004.00 | 25020.0 | 26562.00 | 34300.00 | |

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| race | 58 | 0.99 | 2.80 | 1.25 | 1.00 | 2.00 | 2.0 | 4.00 | 6.00 | |
| imprace | 74 | 0.99 | 3.70 | 2.81 | 0.00 | 1.00 | 3.0 | 6.00 | 10.00 | |
| imprelig | 74 | 0.99 | 3.63 | 2.81 | 1.00 | 1.00 | 3.0 | 6.00 | 10.00 | |
| zipcode | 661 | 0.89 | 85708.89 | 615631.05 | 0.00 | 10021.00 | 19422.0 | 76513.00 | 9971200.00 | |
| income | 0 | 1.00 | 22008.95 | 25253.35 | -1.00 | -1.00 | -1.0 | 42096.00 | 109031.00 | |
| goal | 74 | 0.99 | 2.15 | 1.43 | 1.00 | 1.00 | 2.0 | 2.00 | 6.00 | |
| date | 92 | 0.98 | 4.98 | 1.48 | 1.00 | 4.00 | 5.0 | 6.00 | 7.00 | |
| go_out | 74 | 0.99 | 2.13 | 1.13 | 1.00 | 1.00 | 2.0 | 3.00 | 7.00 | |
| career_c | 133 | 0.98 | 5.45 | 3.42 | 1.00 | 2.00 | 6.0 | 7.00 | 17.00 | |
| sports | 74 | 0.99 | 6.36 | 2.67 | 1.00 | 4.00 | 7.0 | 9.00 | 10.00 | |
| tvsports | 74 | 0.99 | 4.59 | 2.85 | 1.00 | 2.00 | 4.0 | 7.00 | 10.00 | |
| exercise | 74 | 0.99 | 6.16 | 2.46 | 1.00 | 4.00 | 6.0 | 8.00 | 10.00 | |
| dining | 74 | 0.99 | 7.90 | 1.73 | 1.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| museums | 74 | 0.99 | 6.97 | 2.10 | 0.00 | 6.00 | 7.0 | 9.00 | 10.00 | |
| art | 74 | 0.99 | 6.75 | 2.30 | 0.00 | 5.00 | 7.0 | 8.00 | 10.00 | |
| hiking | 74 | 0.99 | 5.59 | 2.63 | 0.00 | 3.00 | 6.0 | 8.00 | 10.00 | |
| gaming | 74 | 0.99 | 3.82 | 2.61 | 0.00 | 1.00 | 3.0 | 6.00 | 14.00 | |
| clubbing | 74 | 0.99 | 5.82 | 2.49 | 0.00 | 4.00 | 6.0 | 8.00 | 10.00 | |
| reading | 74 | 0.99 | 7.53 | 2.01 | 1.00 | 6.00 | 8.0 | 9.00 | 13.00 | |
| tv | 74 | 0.99 | 5.41 | 2.47 | 1.00 | 4.00 | 6.0 | 7.00 | 10.00 | |
| theater | 74 | 0.99 | 6.87 | 2.30 | 0.00 | 5.00 | 7.0 | 9.00 | 10.00 | |
| movies | 74 | 0.99 | 7.96 | 1.76 | 0.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| concerts | 74 | 0.99 | 6.94 | 2.12 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| music | 74 | 0.99 | 7.85 | 1.82 | 1.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| shopping | 74 | 0.99 | 5.76 | 2.52 | 1.00 | 4.00 | 6.0 | 8.00 | 10.00 | |
| yoga | 74 | 0.99 | 4.37 | 2.70 | 0.00 | 2.00 | 4.0 | 7.00 | 10.00 | |
| exphappy | 96 | 0.98 | 5.61 | 1.72 | 1.00 | 5.00 | 6.0 | 7.00 | 10.00 | |
| expnum | 4134 | 0.30 | 5.57 | 4.76 | 0.00 | 2.00 | 4.0 | 8.00 | 20.00 | |
| attr1_1 | 74 | 0.99 | 24.26 | 13.86 | 2.00 | 15.00 | 20.0 | 30.00 | 100.00 | |
| sinc1_1 | 74 | 0.99 | 17.00 | 7.65 | 0.00 | 10.00 | 20.0 | 20.00 | 60.00 | |
| intel1_1 | 74 | 0.99 | 20.47 | 7.32 | 0.00 | 17.00 | 20.0 | 25.00 | 50.00 | |
| fun1_1 | 84 | 0.99 | 17.44 | 6.80 | 0.00 | 13.00 | 20.0 | 20.00 | 50.00 | |
| amb1_1 | 94 | 0.98 | 9.76 | 6.22 | 0.00 | 5.00 | 10.0 | 15.00 | 53.00 | |
| shar1_1 | 116 | 0.98 | 11.28 | 6.77 | 0.00 | 5.00 | 10.0 | 15.00 | 30.00 | |
| attr4_1 | 1884 | 0.68 | 32.44 | 15.47 | 5.00 | 20.00 | 30.0 | 40.00 | 95.00 | |
| sinc4_1 | 1884 | 0.68 | 12.08 | 7.27 | 0.00 | 10.00 | 10.0 | 17.00 | 35.00 | |

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| intel4_1 | 1884 | 0.68 | 14.35 | 6.70 | 0.00 | 10.00 | 15.0 | 20.00 | 35.00 | |
| fun4_1 | 1884 | 0.68 | 17.88 | 6.94 | 0.00 | 15.00 | 20.0 | 20.00 | 45.00 | |
| amb4_1 | 1884 | 0.68 | 10.86 | 7.86 | 0.00 | 5.00 | 10.0 | 15.00 | 50.00 | |
| shar4_1 | 1906 | 0.68 | 12.23 | 6.39 | 0.00 | 10.00 | 10.0 | 15.00 | 40.00 | |
| attr2_1 | 74 | 0.99 | 33.56 | 17.29 | 0.00 | 20.00 | 30.0 | 40.00 | 100.00 | |
| sinc2_1 | 74 | 0.99 | 12.41 | 7.54 | 0.00 | 5.00 | 10.0 | 20.00 | 50.00 | |
| intel2_1 | 74 | 0.99 | 13.82 | 6.67 | 0.00 | 10.00 | 15.0 | 20.00 | 40.00 | |
| fun2_1 | 74 | 0.99 | 18.43 | 7.20 | 0.00 | 15.00 | 20.0 | 20.00 | 50.00 | |
| amb2_1 | 84 | 0.99 | 10.71 | 6.96 | 0.00 | 5.00 | 10.0 | 15.00 | 35.00 | |
| shar2_1 | 84 | 0.99 | 11.12 | 6.51 | 0.00 | 5.00 | 10.0 | 15.00 | 30.00 | |
| attr3_1 | 100 | 0.98 | 7.02 | 1.39 | 2.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| sinc3_1 | 100 | 0.98 | 8.26 | 1.41 | 2.00 | 8.00 | 8.0 | 9.00 | 10.00 | |
| fun3_1 | 100 | 0.98 | 7.72 | 1.58 | 2.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| intel3_1 | 100 | 0.98 | 8.31 | 1.08 | 3.00 | 8.00 | 8.0 | 9.00 | 10.00 | |
| amb3_1 | 100 | 0.98 | 7.50 | 1.77 | 2.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| attr5_1 | 1910 | 0.68 | 7.00 | 1.48 | 2.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| sinc5_1 | 1910 | 0.68 | 7.97 | 1.63 | 1.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| intel5_1 | 1910 | 0.68 | 8.24 | 1.33 | 3.00 | 8.00 | 8.0 | 9.00 | 10.00 | |
| fun5_1 | 1910 | 0.68 | 7.56 | 1.78 | 2.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| amb5_1 | 1910 | 0.68 | 7.58 | 1.80 | 1.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| dec | 0 | 1.00 | 0.42 | 0.49 | 0.00 | 0.00 | 0.0 | 1.00 | 1.00 | |
| attr | 191 | 0.97 | 6.19 | 1.97 | 0.00 | 5.00 | 6.0 | 8.00 | 10.00 | |
| sinc | 232 | 0.96 | 7.20 | 1.77 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| intel | 244 | 0.96 | 7.35 | 1.58 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| fun | 279 | 0.95 | 6.44 | 1.98 | 0.00 | 5.00 | 7.0 | 8.00 | 10.00 | |
| amb | 521 | 0.91 | 6.78 | 1.83 | 0.00 | 6.00 | 7.0 | 8.00 | 10.00 | |
| shar | 760 | 0.87 | 5.45 | 2.20 | 0.00 | 4.00 | 6.0 | 7.00 | 10.00 | |
| like | 208 | 0.96 | 6.15 | 1.85 | 0.00 | 5.00 | 6.0 | 7.00 | 10.00 | |
| prob | 267 | 0.96 | 5.19 | 2.17 | 0.00 | 4.00 | 5.0 | 7.00 | 10.00 | |
| met | 323 | 0.95 | 0.99 | 0.99 | 0.00 | 0.00 | 1.0 | 2.00 | 7.00 | |
| match_es | 538 | 0.91 | 3.17 | 2.33 | 0.00 | 2.00 | 3.0 | 4.00 | 12.00 | |
| attr1_s | 4145 | 0.30 | 21.98 | 17.69 | 3.00 | 8.00 | 17.0 | 30.00 | 95.00 | |
| sinc1_s | 4145 | 0.30 | 13.54 | 8.14 | 0.00 | 8.00 | 10.0 | 20.00 | 50.00 | |
| intel1_s | 4145 | 0.30 | 14.74 | 7.52 | 0.00 | 9.00 | 15.0 | 20.00 | 40.00 | |
| fun1_s | 4145 | 0.30 | 13.30 | 6.42 | 1.00 | 9.00 | 10.0 | 20.00 | 40.00 | |
| amb1_s | 4145 | 0.30 | 8.52 | 4.94 | 0.00 | 5.00 | 9.0 | 10.00 | 20.00 | |

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| shar1_s | 4145 | 0.30 | 10.18 | 6.31 | 0.00 | 5.00 | 10.0 | 15.00 | 30.00 | |
| attr3_s | 4160 | 0.30 | 7.18 | 1.30 | 3.00 | 7.00 | 7.0 | 8.00 | 10.00 | |
| sinc3_s | 4160 | 0.30 | 8.19 | 1.38 | 1.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| intel3_s | 4160 | 0.30 | 8.11 | 1.18 | 4.00 | 7.25 | 8.0 | 9.00 | 10.00 | |
| fun3_s | 4160 | 0.30 | 7.83 | 1.55 | 3.00 | 7.00 | 8.0 | 9.00 | 10.00 | |
| amb3_s | 4160 | 0.30 | 7.57 | 1.75 | 2.00 | 7.00 | 8.0 | 9.00 | 10.00 | |

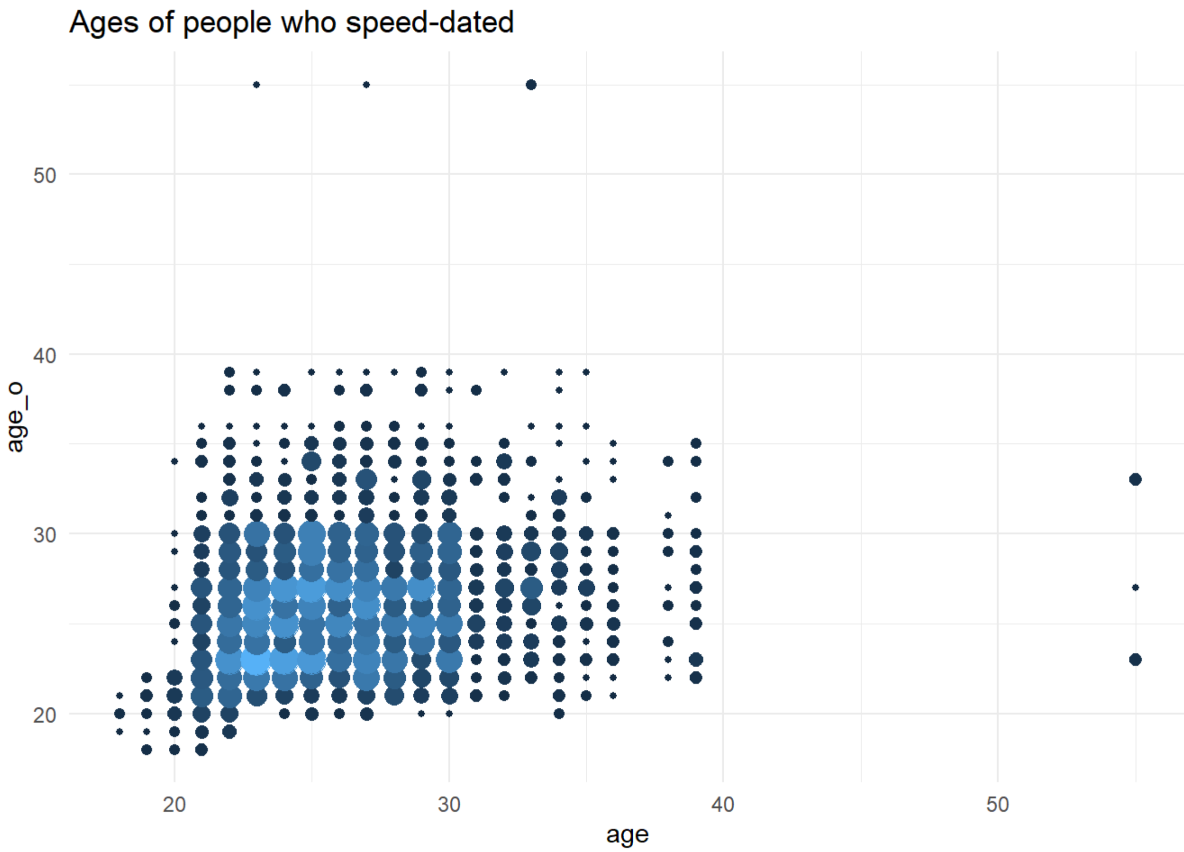
Correlation eda spared for later…

More futile eda trying to gain some insight

```
df %>% count(dec_o)
```

```
## # A tibble: 2 x 2
##   dec_o     n
## * <dbl> <int>
## 1     0  3450
## 2     1  2484
```

```
gg_age <-
  df %>%
  group_by(age, age_o) %>%
  summarize(across(dec, sum)) %>%
  ungroup() %>%
  ggplot() +
  aes(x = age, y = age_o) +
  geom_point(aes(size = dec, color = dec)) +
  theme_minimal() +
  labs(title = 'Ages of people who speed-dated') +
  guides(color = FALSE, size = FALSE)
gg_age
```




Logistic regression with every variable, what could go wrong?

```
f_glm <- function(col_x) {
  fit <- glm(
    formula(glue::glue('match ~ {col_x}')),
    data = df,
    family = 'binomial'
  )
  tidy(fit)
}
col_y <- 'match'
nms <- df %>% names()
cols_x <- nms %>% setdiff(col_y)
cols_x %>% length()
```

```
## [1] 118
```

```
cols_x_filt <- cols_x %>% str_subset('id$', negate = TRUE)
```

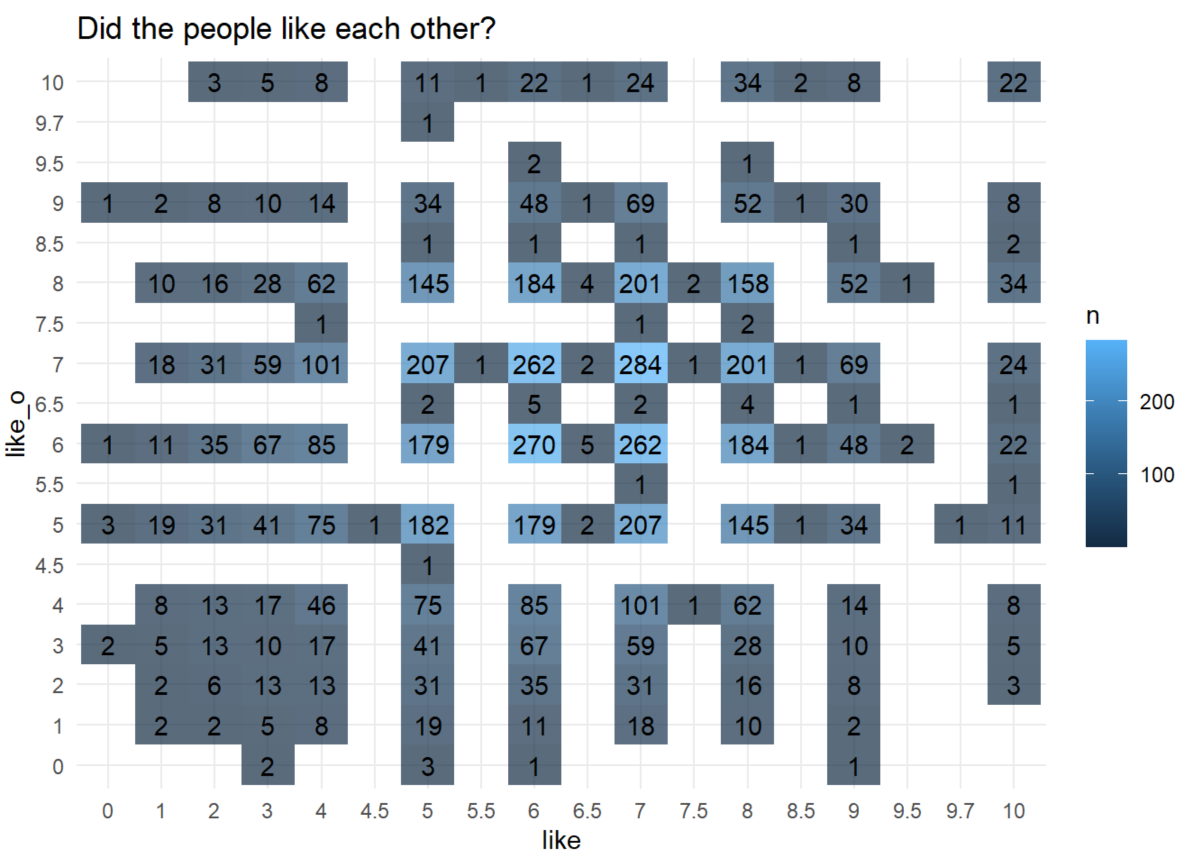
```
coefs_glm <-
  cols_x_filt %>% map_dfr(f_glm)
```

```
coefs_glm %>%
  filter(term != '(Intercept)') %>%
  arrange(p.value)
```

```
## # A tibble: 1,015 x 6
##    term   estimate std.error statistic   p.value   idx
##    <chr>     <dbl>     <dbl>     <dbl>     <dbl> <int>
##  1 like      0.560    0.0255      22.0 3.13e-107  1102
##  2 like_o    0.560    0.0255      22.0 3.55e-107    54
##  3 fun       0.460    0.0227      20.2 4.79e- 91  1096
##  4 fun_o     0.460    0.0227      20.2 5.27e- 91    48
##  5 attr      0.417    0.0216      19.3 6.21e- 83  1090
##  6 attr_o    0.417    0.0216      19.3 6.43e- 83    42
##  7 shar      0.382    0.0201      19.0 1.22e- 80  1100
##  8 shar_o    0.382    0.0201      19.0 1.25e- 80    52
##  9 prob      0.358    0.0188      19.0 1.31e- 80  1104
## 10 prob_o    0.358    0.0188      19.0 1.58e- 80    56
## # ... with 1,005 more rows
```

On average, people rated each other around a 5-7.

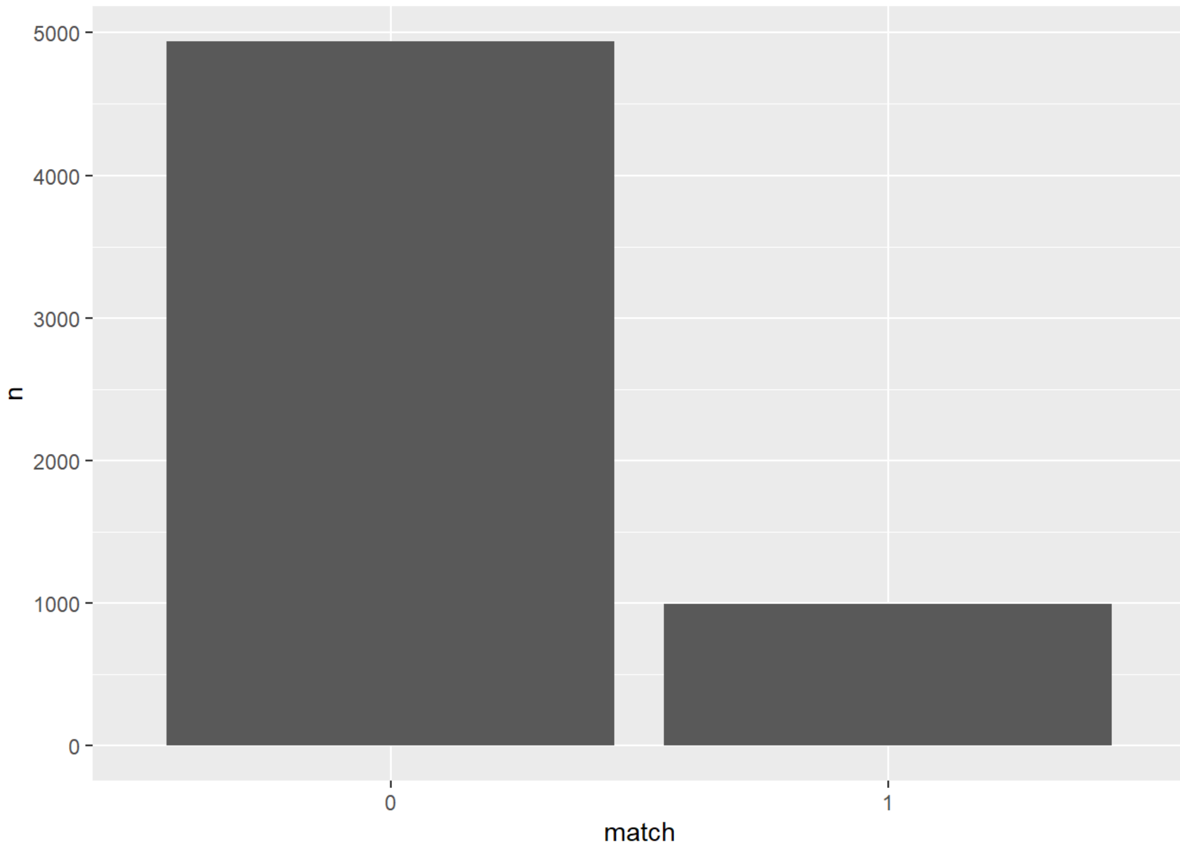
```
df %>%
  count(like, like_o) %>%
  mutate(across(c(like, like_o), factor)) %>%
  drop_na() %>%
  ggplot() +
  aes(x = like, y = like_o) +
  geom_tile(aes(fill = n), alpha = 0.7) +
  geom_text(aes(label = n)) +
  theme_minimal() +
  labs(
    title = 'Did the people like each other?'
  )
```




We've got a ton of the these {attribute} and {attribute}_o (other person) feature pairs. Surely we can do something with them

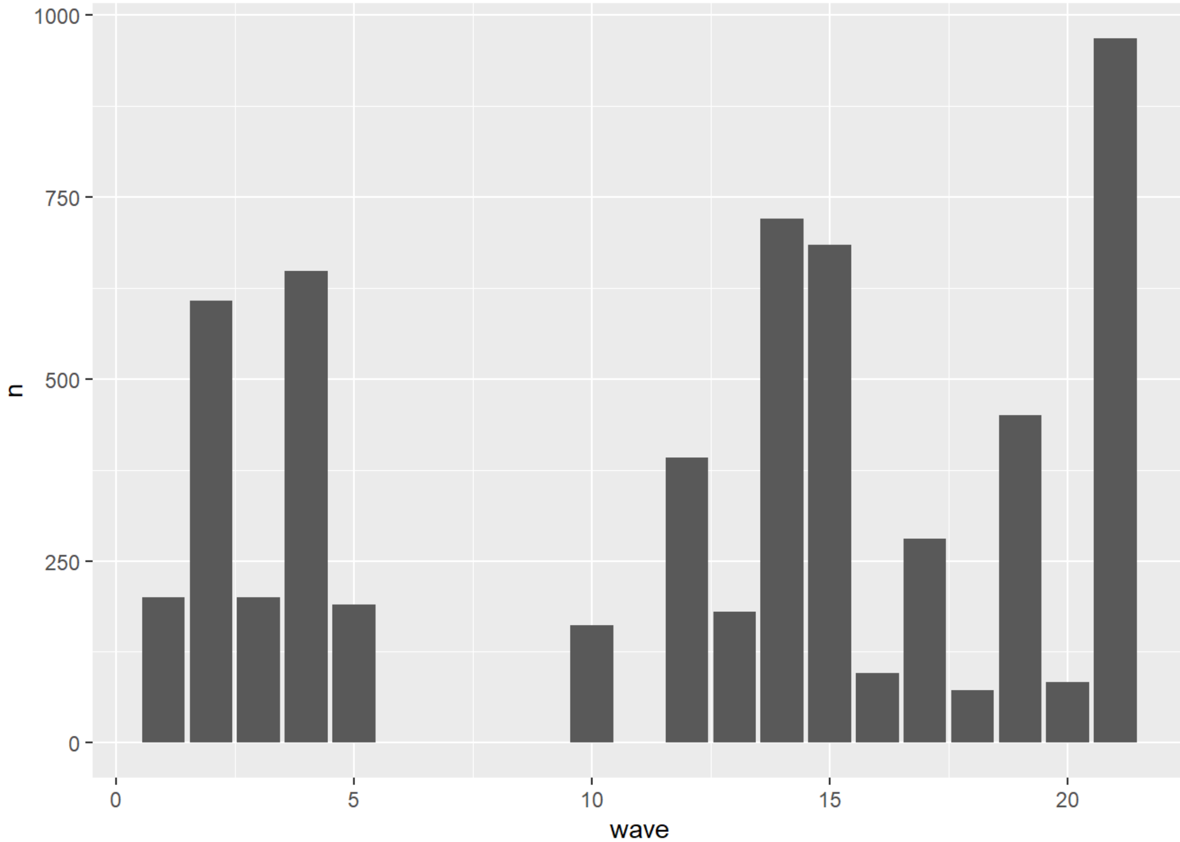
Note that there is a big class imbalance.

```
df %>%
  count(match) %>%
  mutate(across(match, factor)) %>%
  ggplot() +
  aes(x = match, y= n) +
  geom_col()
```




There’s variation in the number of people in each wave of speed-dating. Probably not useful for a quick model.

```
df %>%
  count(wave) %>%
  ggplot() +
  aes(x = wave, y = n) +
  geom_col()
```




I noticed that the p.values were smaller for paired features, and it might just make sense to focus on them.

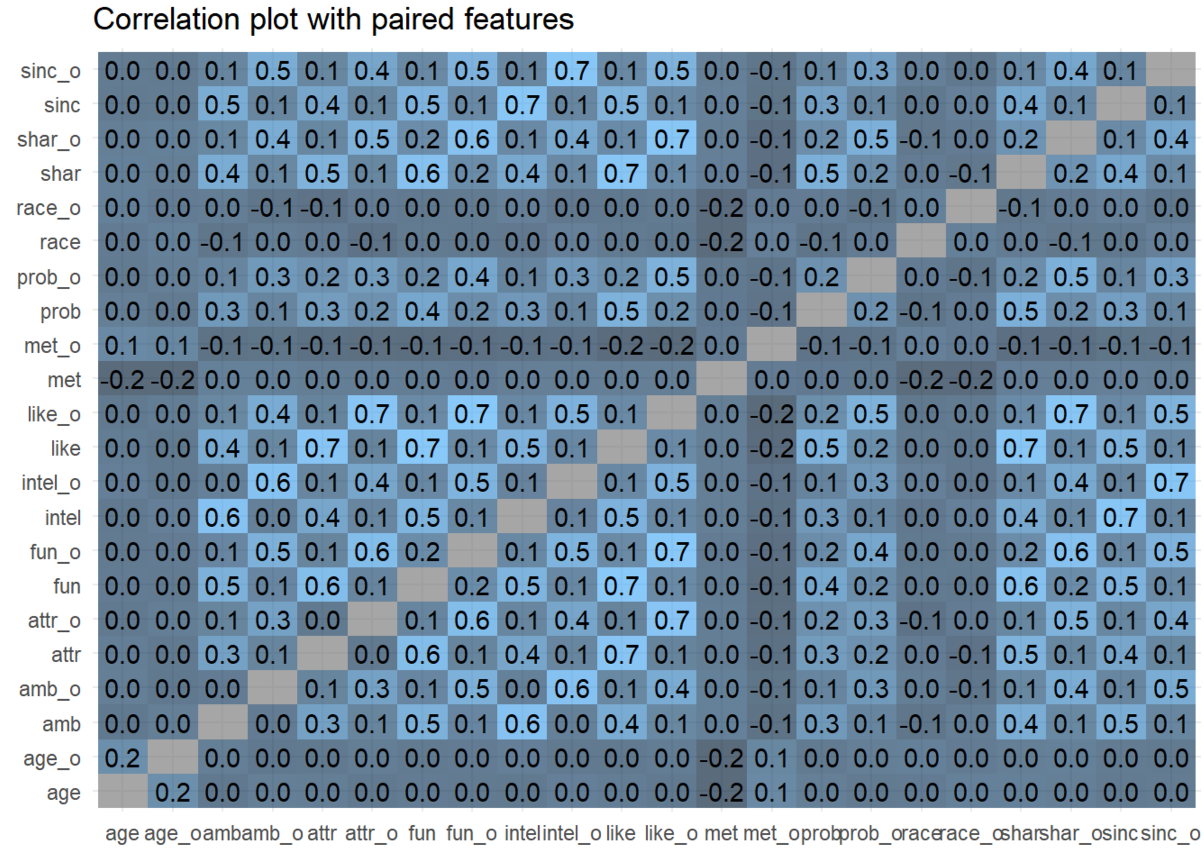
```
cols_x_paired <-
  cols_x %>%
  str_remove('_o$') %>%
  tibble(col = .) %>%
  count(col) %>%
  filter(n > 1L) %>%
  filter(col != 'dec')
cols_x_paired
```

```
## # A tibble: 11 x 2
##    col       n
##    <chr> <int>
##  1 age       2
##  2 amb       2
##  3 attr      2
##  4 fun       2
##  5 intel     2
##  6 like      2
##  7 met       2
##  8 prob      2
##  9 race      2
## 10 shar      2
## 11 sinc      2
```

```r
f_select <- function(data) {
  res <-
    data %>%
    select(
    any_of(col_y),
    one_of(cols_x_paired %>% pull(col)),
    one_of(cols_x_paired %>% pull(col) %>% paste0('_o'))
  )

  if(any('match' %in% colnames(res))) {
    res <-
      res %>%
      mutate(across(match, factor))
  }
  res
}

df_slim <- df %>% f_select()
df_holdout_slim <- df_holdout %>% f_select()
df_slim_nona <- df_slim %>% drop_na()
```

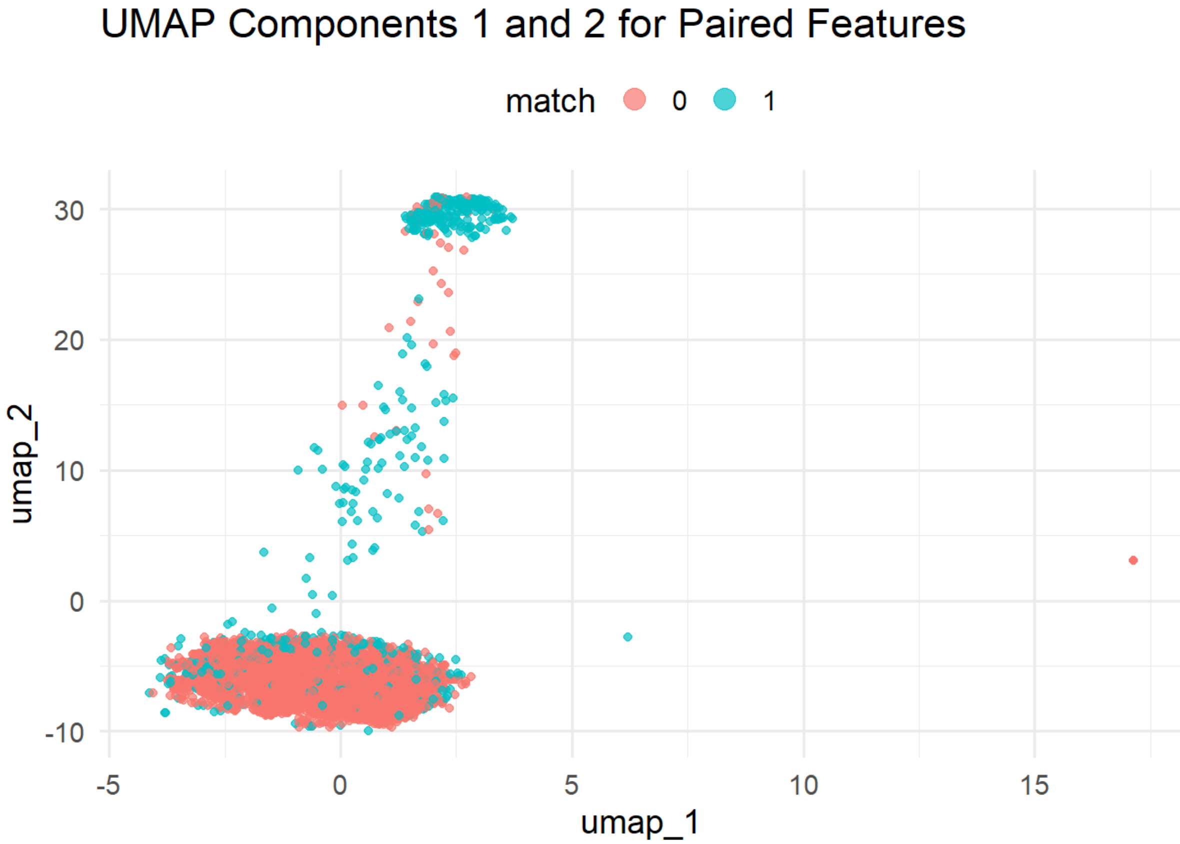
Canonical correlation plot

```r
cors <-
  df_slim_nona %>%
  select(where(is.numeric)) %>%
  corrr::correlate() %>%
  rename(col1 = rowname) %>%
  pivot_longer(
    -col1,
    names_to = 'col2',
    values_to = 'cor'
  )

gg_cors <-
  cors %>%
  ggplot() +
  aes(x = col1, y = col2) +
  geom_tile(aes(fill = cor), alpha = 0.7) +
  geom_text(aes(label = scales::number(cor, 0.1))) +
  theme_minimal() +
  guides(fill = FALSE) +
  labs(
    title = 'Correlation plot with paired features',
    x = NULL, y = NULL
  )
gg_cors
```




Can we differentiate solely based on these paired features? Also, I don’t think I need to split into train-test if we have a true holdout? Probably a bad call, but that’s why I’m an engineer and not a true DS.

```
# Use the nona data set for this cuz we don't like nas.
rec_umap <-
  recipe(match ~ ., data = df_slim_nona) %>%
  embed::step_umap(all_predictors(), outcome = vars(match), num_comp = 2)

juiced_umap <-
  rec_umap %>%
  prep() %>%
  juice()

gg_umap <-
  juiced_umap %>%
  ggplot() +
  aes(x = umap_1, umap_2) +
  geom_point(aes(color = match), alpha = 0.7) +
  theme_minimal(base_size = 14) +
  guides(
    color = guide_legend(override.aes = list(size = 4))
  ) +
  theme(legend.position = 'top') +
  labs(
    title = 'UMAP Components 1 and 2 for Paired Features'
  )
gg_umap
```




lol at the outlier in the umap plot above.

```
rec <-
  recipe(match ~ ., data = df_slim) %>%
  # One of these steps (probably near zero variance?) is causing a column to be dropped in rand forest, which sometimes raises a warning
  # step_nzv(all_numeric_predictors()) %>%
  step_zv(all_numeric_predictors()) %>%
  # step_lincomb(all_numeric_predictors()) %>%
  # Impute at the end?
  step_impute_knn(all_numeric_predictors()) %>%
  step_downsample(all_outcomes())

# Just checking that it works.
rec %>%
  prep() %>%
  juice()
```

```
## # A tibble: 1,992 x 23
##      age   amb  attr   fun intel  like   met  prob  race  shar  sinc age_o amb_o
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    30   5       6     6     5   6     2     4       4   4     6      23    10
##  2    29   5.4     6     3     5   5     2     3       2   5     5.6    28     6
##  3    36   5       7     5     7   5     0     2       2   4.8   7      33     8
##  4    23  10       7     8    10   7     2     5       2   7     8      27     6
##  5    24   6       7     5     5   3     0     2       4   1     8      25     3
##  6    27   8       6     8     8   8     1.2   8       4   8     7      27     6
##  7    23   5       8     9     7   8     2     1       2   4     7      27     7
##  8    23   9       3     5     9   6     2     7       4   2    10      21     7
##  9    26   6       2     6     7   6.4   0     6.4     2   6     8      29    10
## 10    23   6       6     6     6   4     0     6       4   6     6      26     7
## # ... with 1,982 more rows, and 10 more variables: attr_o <dbl>, fun_o <dbl>,
## #   intel_o <dbl>, like_o <dbl>, met_o <dbl>, prob_o <dbl>, race_o <dbl>,
## #   shar_o <dbl>, sinc_o <dbl>, match <fct>
```

I'm intentionally picking methods that are relatively different and don't have a lot of stuff to tune.

Parallel processing would be awesome here but it usually breaks my laptop.

```
spec_knn <-
  nearest_neighbor(neighbors = tune(), weight_func = tune()) %>%
  set_mode('classification') %>%
  set_engine('kknn')

wf_knn <-
  workflow() %>%
  add_recipe(rec) %>%
  add_model(spec_knn)

spec_rf <-
  rand_forest(mtry = tune(), trees = tune()) %>%
  set_mode('classification') %>%
  set_engine('ranger')

wf_rf <-
  workflow() %>%
  add_recipe(rec) %>%
  add_model(spec_rf)

ctrl_grid <- control_grid(save_pred = TRUE, save_workflow = TRUE, verbose = TRUE)
met_set <- metric_set(mn_log_loss, accuracy, roc_auc)
```

```
set.seed(6669)
folds <- df_slim %>% vfold_cv(strata = match, v = 10)
grid_knn <-
  grid_max_entropy(
    wf_knn %>% parameters(),
    size = 10
  )
grid_knn

# I feel like this is overfitting
grid_rf <-
  grid_max_entropy(
    trees(),
    # finalize(mtry(), df_slim),
    mtry(1, round(7/8*ncol(df_slim)-1))
    size = 10
  )
grid_rf

f_tune <-
  partial(
    tune_grid,
    resamples = folds,
    metrics = met_set,
    control = ctrl_grid,
    ... =
  )

res_knn <- wf_knn %>% f_tune(grid = grid_knn)
res_knn
res_rf <- wf_rf %>% f_tune(grid = grid_rf)
res_rf
```

```
params_best_rf <- res_rf %>% select_best(metric = 'mn_log_loss')
wf_rf_final <- wf_rf %>% finalize_workflow(params_best_rf)
wf_rf_final
```

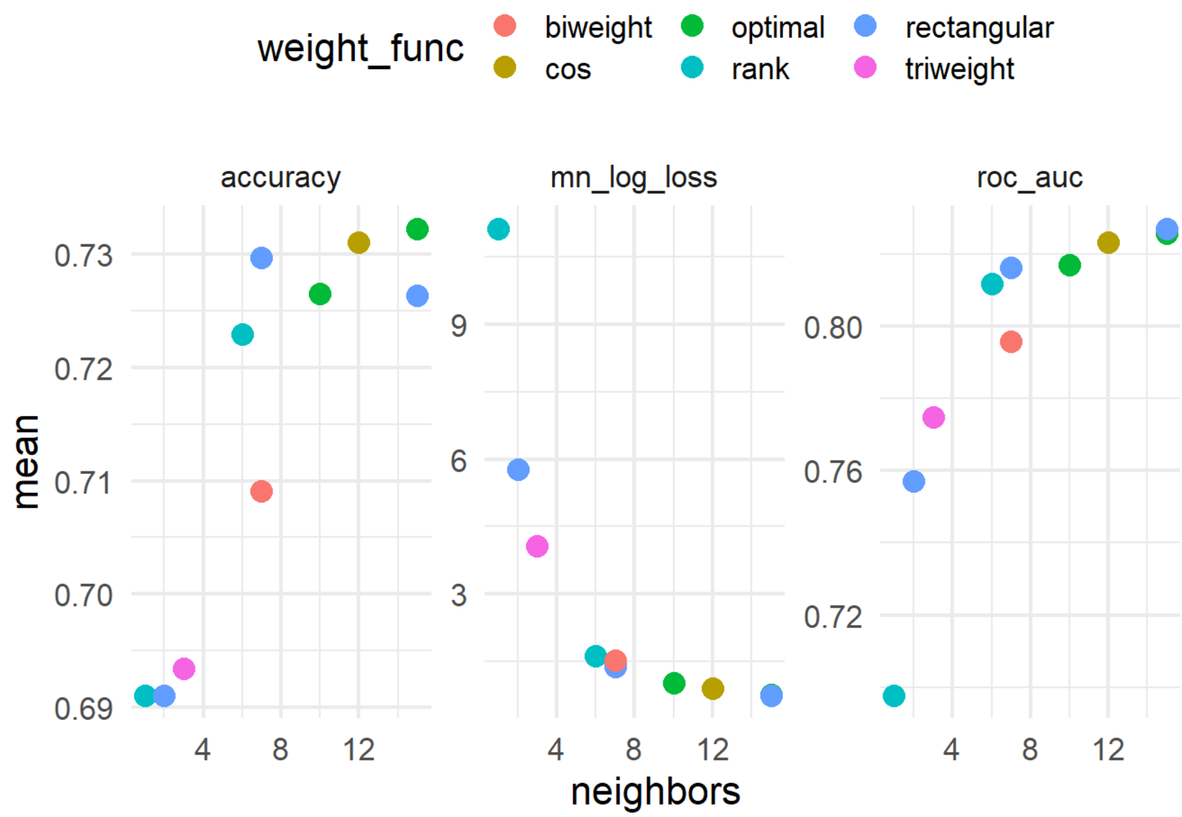
```
## == Workflow ================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_zv()
## * step_impute_knn()
## * step_downsample()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 12
##   trees = 1970
##
## Computational engine: ranger
```

Looking at how the tuning went.

```
mets_knn <-
  res_knn %>%
  collect_metrics()
params_best_knn <- res_knn %>% select_best('mn_log_loss')
wf_knn_best <-
  wf_knn %>%
  finalize_workflow(params_best_knn)
fit_knn_best <-
  wf_knn_best %>%
  fit(data = df_slim) %>%
  pull_workflow_fit()
fit_knn_best
```

```
## parsnip model object
##
## Fit time:  220ms
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(15L,     data, 5), kernel = ~"rectangular")
##
## Type of response variable: nominal
## Minimal misclassification: 0.2394578
## Best kernel: rectangular
## Best k: 15
```

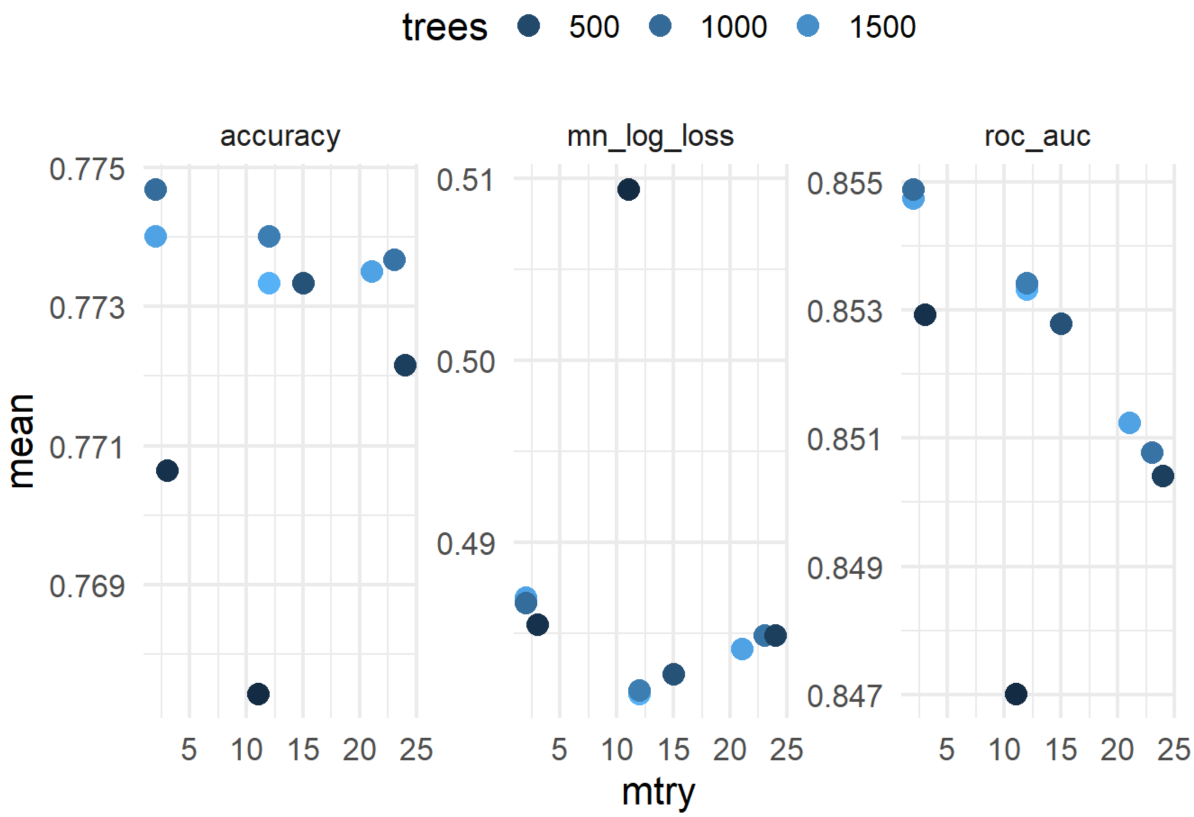
```
mets_knn %>%
  select(neighbors, weight_func, .metric, mean) %>%
  ggplot() +
  aes(x = neighbors, y = mean) +
  geom_point(aes(color = weight_func), size = 4) +
  facet_wrap(~.metric, scales = 'free') +
  guides(color = guide_legend(override.aes = list(size = 4))) +
  theme_minimal(base_size = 16) +
  theme(legend.position = 'top')
```




```
mets_rf <-
  res_rf %>%
  collect_metrics()
mets_rf
```

```
## # A tibble: 30 x 8
##     mtry trees .metric     .estimator  mean     n std_err .config
##    <int> <int> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
##  1     3   126 accuracy    binary     0.771    10 0.00427 Preprocessor1_Model01
##  2     3   126 mn_log_loss binary     0.486    10 0.00502 Preprocessor1_Model01
##  3     3   126 roc_auc     binary     0.853    10 0.00651 Preprocessor1_Model01
##  4    12  1970 accuracy    binary     0.773    10 0.00319 Preprocessor1_Model02
##  5    12  1970 mn_log_loss binary     0.482    10 0.00539 Preprocessor1_Model02
##  6    12  1970 roc_auc     binary     0.853    10 0.00625 Preprocessor1_Model02
##  7     2  1790 accuracy    binary     0.774    10 0.00524 Preprocessor1_Model03
##  8     2  1790 mn_log_loss binary     0.487    10 0.00410 Preprocessor1_Model03
##  9     2  1790 roc_auc     binary     0.855    10 0.00590 Preprocessor1_Model03
## 10    23  1128 accuracy    binary     0.774    10 0.00323 Preprocessor1_Model04
## # ... with 20 more rows
```

```
mets_rf %>%
  # filter(.metric == 'mn_log_loss') %>%
  #mutate(across(trees, factor)) %>%
  ggplot() +
  aes(x = mtry, y = mean) +
  geom_point(aes(color = trees, size = trees), size = 4) +
  facet_wrap(~.metric, scales = 'free') +
  guides(color = guide_legend(override.aes = list(size = 4))) +
  theme_minimal(base_size = 16) +
  theme(legend.position = 'top')
```

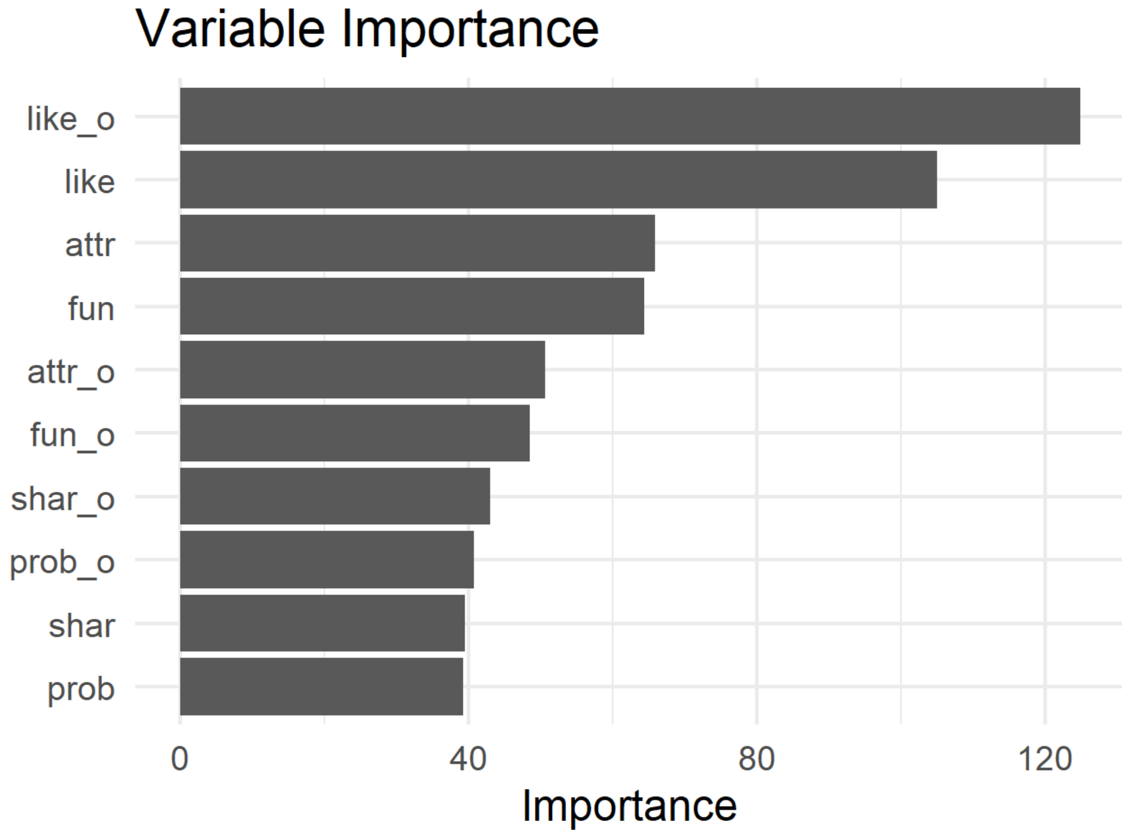



Variable importance time. (This is where i realized i left in `dec` and `dec_o` on accident.)

```
spec_rf_vi <-
  rand_forest(mtry = 12, trees = 1970) %>%
  set_mode('classification') %>%
  set_engine('ranger', importance = 'impurity')
wf_rf_vi <-
  workflow() %>%
  add_recipe(rec) %>%
  add_model(spec_rf_vi)
fit_rf_vi <- wf_rf_vi %>% fit(data = df_slim)
fit_rf_vi
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_zv()
## * step_impute_knn()
## * step_downsample()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~12,      x), num.trees = ~1970, importance = ~"impu
rity", num.threads = 1,      verbose = FALSE, seed = sample.int(10^5, 1), probability = TRUE)
##
## Type:                             Probability estimation
## Number of trees:                  1970
## Sample size:                      1992
## Number of independent variables:  22
## Mtry:                             12
## Target node size:                 10
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.1534695
```

```
fit_rf_final <-
  fit_rf_vi %>%
  pull_workflow_fit()
gg_vi <-
  fit_rf_vi %>%
  pull_workflow_fit() %>%
  vip::vip() +
  theme_minimal(base_size = 18) +
  labs(
    title = 'Variable Importance'
  )
gg_vi
```




Ensembling time with a package i just learned how to use yesterday! This is basically a regularized linear regression on our knn and rf models.

```
ens_st <-
  stacks() %>%
  add_candidates(res_knn) %>%
  add_candidates(res_rf)
ens_st
```

```
## # A data stack with 2 model definitions and 20 candidate members:
## #   res_knn: 10 model configurations
## #   res_rf: 10 model configurations
## # Outcome: match (factor)
```

```
fit_ens <-
  ens_st %>%
  blend_predictions() %>%
  fit_members()
fit_ens
```

```
## # A tibble: 6 x 3
##   member                type             weight
##   <chr>                 <chr>             <dbl>
## 1 .pred_1_res_rf_1_03   rand_forest      2.28
## 2 .pred_1_res_rf_1_05   rand_forest      1.20
## 3 .pred_1_res_rf_1_10   rand_forest      0.828
## 4 .pred_1_res_rf_1_07   rand_forest      0.735
## 5 .pred_1_res_knn_1_09 nearest_neighbor 0.590
## 6 .pred_1_res_rf_1_08   rand_forest      0.0443
```

A blend of 6 rf models were chosen by the ensembling

```
# preds_ens_holdout <-
#   fit_ens %>%
#   predict(df_holdout_slim, type = 'pred')
df_holdout_slim
```

```
## # A tibble: 882 x 22
##      age   amb  attr   fun intel  like   met  prob  race  shar  sinc age_o amb_o
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    28     5     6     6     7     6     0     8     3     2     6    29     6
##  2    28     6     4     6     6     6     0     8     3     6     5    24     8
##  3    28     9     7     4     9     2     0     4     3     3     6    21     6
##  4    28     5     2     5     5     3     0     3     3     1     4    26     9
##  5    28     9     2     1     1     1     0     2     3     5     1    25    NA
##  6    28     8     1     1     8     2     0     8     3     1     2    32     8
##  7    28     8     4     1     5     6     0     8     3     7     2    25     8
##  8    28     3     1     1     8     2     0     8     3     1     9    25     8
##  9    28     6     4     6     8     4     0     8     3    10     6    27     5
## 10    28     5     1     7     5     1     0     4     3     1     2    23     8
## # ... with 872 more rows, and 9 more variables: attr_o <dbl>, fun_o <dbl>,
## #   intel_o <dbl>, like_o <dbl>, met_o <dbl>, prob_o <dbl>, race_o <dbl>,
## #   shar_o <dbl>, sinc_o <dbl>
```

```
preds_holdout <-
  # fit_rf_final %>%
  fit_knn_best %>%
  predict(df_holdout_slim, type = 'class')
preds_holdout
```

```
## # A tibble: 664 x 1
##    .pred_class
##    <fct>
##  1 0
##  2 1
##  3 0
##  4 0
##  5 0
##  6 0
##  7 0
##  8 1
##  9 0
## 10 0
## # ... with 654 more rows
```

```
probs_holdout <-
  # fit_rf_final %>%
  fit_knn_best %>%
  predict(df_holdout_slim, type = 'prob') %>%
  rename(prob_0 = .pred_0, prob_1 = .pred_1)
probs_holdout
```

```
## # A tibble: 664 x 2
##    prob_0 prob_1
##     <dbl>  <dbl>
##  1  0.933 0.0667
##  2  0.267 0.733
##  3  0.733 0.267
##  4  0.933 0.0667
##  5  1     0
##  6  0.867 0.133
##  7  0.933 0.0667
##  8  0.4   0.6
##  9  1     0
## 10  0.933 0.0667
## # ... with 654 more rows
```

```
write_csv(preds_holdout, path_preds_holdout)
write_csv(probs_holdout, path_probs_holdout)
```