**PRESENTATION**

* Do it over-what would you do differently?
* Discuss restarting
* Misunderstood data – regression
* Data- thought it was clean and coded
  + WRONG
    - 8378 entries
    - 195 features
  + This took me the longest time – deciphering every feature b/c all were labeled the same just with a different reference # at the end to indicate when in the survey / experiment the question had been administered and/or who (subject or partner)
    - Had to sort though every feature to figure out which would be relevant to my question 🡪 issue there was I couldn’t figure out the approach I wanted to take on my question so instead of making up my mind, I decided to try and keep all the day (bad idea)
* Started by pulling everything I thought would be interesting; had a lot of questions about the data
* After some observations, decided I wanted to make a condensed dataset per iid
* Wrote functions to condense data
* NaN value issues
  + Had to refine my subsets; only use data to answer specific question

Further exploration – look at gender perceptions; how does that effect dating outcomes?

### Does one’s perception of their gender generalizations differ from their own evaluations of what’s important when it comes to selecting mates?

\*\*Question 2B\*\*: \*Does this differ from self-evaluations (Q3)? E.g.: do men rate ‘attractiveness’ as less important for their own dating choices but more important for other men?\*

\*\*Hypothesis\*\*: men will rate ‘attractiveness’ as less important for their own dating choices but more important for other men’s decisions when choosing a partner.

Other question:

Look at probability rating and score; e.g. if person gave high score but thought the prob of other person choosing them was low, how that affects decision (y/n); ultimately predicting match

NOTE:

for sums of # of matches, I put an idea in the answer above. but assigning the resulting value to a column such as df.num\_matches = \* should work. then you can do linear regression to predict how many matches they'd get, or if you normalize it 0-1 then you could also do logistic regression or even classification (where you'd be classifying if they got 'a lot' of matches vs not)

for viewing the data, you can always do .head(a) or .tail(b) where a and b are optional, but specify the # of rows returned - so the top a rows or the bottom b rows. there are also scatter plots of an independent variable vs the outcome variable, and histograms of any single variable. compared to describe(), I think histograms are probably what you're looking for. df.variable.plot(kind='hist')

exploratory analysis

data exploration and analysis (visuals)

scatter w/ 2 variables (color code)

MODELING

Random forests (determine which features are important)

* using those, use logistic regression

KNN

Accuracy tests

False positive vs. true negative

E.g. better to get false negative than a false positive: would rather think you aren't getting a match and then get one instead of thinking you are getting a match and then not get one

* match\_sum | match\_ave : 0.885822
* match\_es | match\_es\_ave : 0.875759
* like\_o\_ave |attr\_o\_ave : 0.856744
* dec\_sum | dec\_ave : 0.840868
* amb\_iRateMe\_exp |amb\_iMeasUp\_2 : 0.830971
* attr\_oPercveMe\_2 |attr\_iMeasUp\_2 : 0.815613
* dec\_o\_ave | attr\_o\_ave : 0.811872
* attr\_iMeasUp\_2 | attr\_iRateMe\_exp : 0.808513
* dec\_o\_sum | dec\_o\_ave : 0.798661
* fun\_iMeasUp\_2 | fun\_oPercveMe\_2 : 0.797956
* like\_o\_ave | dec\_o\_ave : 0.794109
* attr\_iMeasUp\_1 | attr\_oPercveMe\_1 : 0.791036
* amb\_iRateMe\_exp | amb\_iMeasUp\_1 : 0.783401
* like\_o\_ave | fun\_o\_ave : 0.779505
* fun\_iMeasUp\_2 | fun\_iRateMe\_exp : 0.778293
* fun\_iMeasUp\_1 | fun\_oPercveMe\_1 : 0.768977
* attr\_iMeasUp\_1 | attr\_iMeasUp\_2 : 0.754649
* attr\_oPercveMe\_1 |attr\_oPercveMe\_2 : 0.753079
* intel\_oPercveMe\_1 | intel\_iMeasUp\_1 : 0.746656
* fun\_iRateMe\_exp | fun\_oPercveMe\_2 : 0.732414
* amb\_iMeasUp\_1 | amb\_iMeasUp\_2 : 0.731679
* like\_ave | attr\_ave : 0.730203
* intel\_iMeasUp\_2 | intel\_iRateMe\_exp : 0.728748
* like\_ave | fun\_ave : 0.727753
* fun\_iMeasUp\_1 | fun\_iRateMe\_exp : 0.725670
* fun\_iMeasUp\_2 | fun\_oPercveMe\_1 : 0.721469
* fun\_iMeasUp\_1 | fun\_iMeasUp\_2 : 0.717412
* fun\_oPercveMe\_1 | fun\_oPercveMe\_2 : 0.701235

|  |
| --- |
| **match\_sum:** |
| **match\_ave** | | 0.885822 |
| **dec\_sum** | | 0.600592 |
| **dec\_match\_ave** | | 0.58332 |
| **dec\_o\_sum** | | 0.558067 |
| **dec\_ave** | | 0.486358 |
| **match\_es** | | 0.459059 |
| **dec\_o\_ave** | | 0.434573 |
| **like\_o\_ave** | | 0.389363 |
| **attr\_o\_ave** | | 0.379035 |
| **fun\_o\_ave** | | 0.350454 |
| **match\_es\_ave** | | 0.315399 |

|  |  |
| --- | --- |
| **dec\_match\_ave:** | |
| **dec\_o\_ave** | 0.684324 | |
| **attr\_o\_ave** | 0.620261 | |
| **match\_ave** | 0.616025 | |
| **dec\_o\_sum** | 0.586397 | |
| **match\_sum** | 0.58332 | |
| **like\_o\_ave** | 0.575325 | |
| **fun\_o\_ave** | 0.417569 | |
| **prob\_o\_ave** | 0.362303 | |
| **shar\_o\_ave** | 0.306149 | |