OkCupid Analysis

Eric Kim, B.A.

Silicon Valley, CA

August 25, 2017

eric@kimanalytics.com
https://www.kimanalytics.com
https://github.com/kimanalytics/okcupidanalysis

Table of Contents

Abstract

We have a data set of San Francisco OkCupid user profiles from June 2012. (n = 59,946) The data contains variables both selected and user-inputted such as age, body type, income, and along with 10 essay questions. We do a brief analysis of averages and distribution of the data. We use the programming language Python to: clean up the data, visualize the tidy data, and use machine learning to create models and prediction of the data. The data science covered includes data wrangling, data visualization, exploratory data analysis, multivariate relationships, text analysis, and machine learning for prediction.

Keywords: Analysis, data science, matplotlib, naive bayes, numpy, okcupid, online dating, pandas, python, scipy, seaborn.

Introduction

The overall intention of this study is to demonstrate how user data can help both user and developer of online dating platforms. User data becomes valuable over time due to its ease of accessibility and affordability to store. Not only we can take the data at its face value, but we can do further analysis by finding relationships and outliers.

A benefit of finding relationships in data could help users find matching relationships with other users. A benefit of finding outliers could help developer filter out spam or low-quality accounts by possible addition of recommendation systems.

Finding inconsistency or missing values in the data could also assist both user and developer in finding out what is important to report or not. Something as simple as omitting data due to privacy concerns such as income or education could be addressed after finding its impact described in this study. Something already addressed by OkCupid lies in its “Education” user-selected response. Rather than claiming a classic education level such as “Graduated High School” or “College Dropout”, users can select a tongue-in-cheek response such as “Working on Space Camp”. This benefits the overall system by letting users have a full profile and developer with less missing values to work with.

After doing analysis of the data, we use a brief machine learning algorithm to make a model that predicts the gender of the user. While predicting sex may be useful for some users and not others, it could be easily reproduced with other combinations of variables to make other models. Other models could very well include: Education, body type, and age to predict income level. Also, drinking and drug habits to predict job type. The models we create for prediction could first help the developer match users with recommendations. It could also help fill in missing user profiles which help users achieve a full profile.

Data

The data consists of the public profiles of 59,946 OkCupid users who were living within 25 miles of San Francisco, had active profiles on June 26, 2012, were online in the previous year, and had at least one picture in their profile. Using a Python script, data was scraped from users' public profiles on June 30, 2012; any non-publicly facing information such as messaging was not accessible. Variables include typical user information (such as sex, sexual orientation, age, and ethnicity) and lifestyle variables (such as diet, drinking habits, smoking habits). Furthermore, text responses to the 10 essay questions posed to all OkCupid users are included as well, such as "My Self Summary," "The first thing people usually notice about me," and "On a typical Friday night I am..." For a complete list of variables and more details, see the accompanying codebook in my GitHub.

Before we continue we note that even though this data consists of publicly facing material, one should proceed with caution before scraping and using data in a fashion similar to ours, as the Computer Fraud and Abuse Act (CFAA)[1] makes it a federal crime to access a computer without authorization from the owner. In our case, permission to use and disseminate the data was given by its owners as long as the data remains public.

Part 1 - Understanding the Variables

Our goal for part 1 is to understand the variables in the OkCupid dataset. Just by listing all of the variables, we can already assume which are more interesting than others. However, let's looking into each just to understand the formats such as: selected word responses, numerical responses, and written responses.

1.1 Getting the name of all of the variables

```
list(df)
```

```
['age', 'body_type', 'diet', 'drinks', 'drugs', 'education', 'essay0', 'essay1', 'essay2',
'essay3', 'essay4', 'essay5', 'essay6', 'essay7', 'essay8', 'essay9', 'ethnicity', 'height',
'income', 'job', 'last_online', 'location', 'offspring', 'orientation', 'pets', 'religion',
'sex', 'sign', 'smokes', 'speaks', 'status']
```

1.1.1 Pedagogical Discussion

Looking at the variables listed from the data frame, we can count 31 variables. Using an example, suppose a person would want an opposite partner to have 5 matching criteria to be

[1] "18 U.S.C. 1030(a) - Legal Information Institute." https://www.law.cornell.edu/uscode/text/18/1030. Accessed 4 Aug. 2017.

considered “dateable”. Such criteria could include: Around 30 years old, No drugs, Graduated College, Male, and living in New York. Since we have 31 variables instead of a general 5, it gives the user a lot of power to filter out who matches to them. Also, it gives the opportunity for a user to still match if they have missing sections of their profile. What is beneficial to many variables is the ability to combine certain variables to predict another. A quick example is using education and job to predict income. Of course, a person could use a rough guess. However, it is possible to use machine learning to teach an algorithm how to make an accurate prediction based on user data.

1.2 Age, Body Type, and Diet

Here, we take a look at simple variables such as age, body type, and diet.

```
age = df['sex']]
age.describe()

age
count   59946.000000
mean    32.340290
std     9.452779
min     18.000000
25%     26.000000
50%     30.000000
75%     37.000000
max     110.000000
```

```
body_type = df[['body_type', 'sex']]
body_type_grouped = body_type.groupby('body_type')
body_type_grouped = body_type_grouped.count().reset_index()
body_type_grouped.rename(index=str, columns={"sex":"count"})

    body_type       count
0   a little extra  2629
1   athletic        11819
2   average         14652
3   curvy           3924
4   fit             12711
5   full figured    1009
6   jacked          421
7   overweight      444
8   rather not say  198
9   skinny          1777
10  thin            4711
11  used up         355
```

```
diet = df[['diet', 'sex']]
diet_grouped = diet.groupby('diet')
diet_grouped = diet_grouped.count()
diet_grouped = diet_grouped.reset_index()
diet_grouped.rename(index=str, columns={"sex":"count"})
```

	Diet	count
0	anything	6183
1	halal	11
2	kosher	11
3	mostly anything	16585
4	mostly halal	48
5	mostly kosher	86
6	mostly other	1007
7	mostly vegan	338
8	mostly vegetarian	3444
9	other	331
10	strictly anything	5113
11	strictly halal	18
12	strictly kosher	18
13	strictly other	452
14	strictly vegan	228
15	strictly vegetarian	875
16	vegan	136
17	vegetarian	667

1.2.1 Pedagogical Discussion

1.2.1.1 Age

Looking at age, everything seems to be normal. The average user age is 32 ± 9 years old. This makes sense, after college or even during is when people will use dating sites to find partners. It is more convenient than going back to school or involving in office romances. The lowest age is 18 years old. This has to be the lowest since it is illegal for users to be under 18 years old and using a dating website in the US. The mid range is around 26-37 years old. This makes sense. However, the max age is 110 years old. This seems to be an outlier but it isn't affecting the mid-range averages so much with over 50,000 data points. Another observation that could be made is that users don't have much room to lie about their age if they choose to do so since they have a profile picture as well. If they do choose to lie, it shouldn't be too much off from the standard deviation, 9 years.

1.2.1.2 Body Type

For body types, there are 11 types. The highest count is the body type of "average" with 14,652 users which makes sense because it's average. The lowest count is "rather not say" with 198 users which could have some meaning with machine learning. Some interesting body types here are "jacked", "curvy", "fit", and "overweight". The first two could predict gender and the last

two could predict diet. With a combination of other variables, it could very well predict variables not listed in this study.

1.2.1.3 Diet

With diet, it is very similar to body type. Most popular diet is: “mostly anything” with 16,585 users and least diet is: “halal” or “kosher” with 11 users each. Diet could be used to predict variables such as religion. Although many users overlook diet when dating, I think it could be very useful in predicting other factors of a person.

1.3 Education

Education is a great variable for “smart” matches. It helps users in similar situations with similar goals.

```
education = df[['education', 'sex']]
education_grouped = education.groupby('education')
education_grouped = education_grouped.count().reset_index()
education_grouped.rename(index=str, columns={"sex":"count"})
```

	education	count
0	college/university	801
1	dropped out of college/university	995
2	dropped out of high school	102
3	dropped out of law school	18
4	dropped out of masters program	140
5	dropped out of med school	12
6	dropped out of ph.d program	127
7	dropped out of space camp	523
8	dropped out of two-year college	191
9	graduated from college/university	23959
10	graduated from high school	1428
11	graduated from law school	1122
12	graduated from masters program	8961
13	graduated from med school	446
14	graduated from ph.d program	1272
15	graduated from space camp	657
16	graduated from two-year college	1531
17	high school	96
18	law school	19
19	masters program	136
20	med school	11
21	ph.d program	26
22	space camp	58
23	two-year college	222
24	working on college/university	5712
25	working on high school	87
26	working on law school	269

```
27  working on masters program          1683
28  working on med school               212
29  working on ph.d program             983
30  working on space camp               445
31  working on two-year college         1074
```

1.3.1 Pedagogical Discussion

An interesting category in the education variable is 'space camp'. Through quick research, 'space camp' does exist.[2] However, I don't think all of the responses are honest. I'm going to keep this in mind when comparing it with other responses. Education could be a good predictor for matches with age and income. The largest group is college graduates with 23,959 users with the least being 11 users that are medical school graduates.

1.4 Essays

Essays 0-9 are fill in the blank prompts. The titles are:

- 0: Self Summary
- 1: What I'm doing with my life
- 2: I'm really good at
- 3: The first thing people usually notice about me
- 4: Favorite books, movies, show music, and food
- 5: The six things I could never do without
- 6: I spend a lot of time thinking about
- 7: On a typical Friday night I am
- 8: The most private thing I am willing to admit
- 9: You should message me if...

1.4.1 Pedagogical Discussion

Unlike many the other responses which are drag-drop pre filled responses, this gives the users an opportunity to express themselves in words. Perhaps they want to explain other variables such as income and religion with this. Or, they want to use this section to prepare for dates. Either way, this is an excellent way for users to get to know one another.

1.5 Essay Sample

Let's take a look at the first user of our dataset. Here, we have an Asian, 75 inches tall, 22 year-old-male that is working on college. His body is a little extra and he will eat strictly anything.

[2] "Space Camp." https://www.spacecamp.com/. Accessed 4 Aug. 2017.

He drinks but doesn’t do drugs. He works in transportation and doesn’t want to disclose his income. (Note: undisclosed income is noted as ‘-1’)

```
df.loc[0,['age', 'body_type', 'diet', 'drinks', 'drugs', 'education',
 'ethnicity', 'height', 'income', 'job', 'last_online', 'location',
 'offspring', 'orientation', 'pets', 'religion', 'sex', 'sign',
 'smokes', 'speaks', 'status']]
```

```
age                                                    22
body_type                                   a little extra
diet                                      strictly anything
drinks                                             socially
drugs                                                 never
education                       working on college/university
ethnicity                                    asian, white
height                                                 75
income                                                 -1
job                                           transportation
last_online                                2012-06-28-20-30
location                       south san francisco, california
offspring            doesn’t have kids, but might want them
orientation                                       straight
pets                                 likes dogs and likes cats
religion              agnosticism and very serious about it
sex                                                     m
sign                                               gemini
smokes                                          sometimes
speaks                                            english
status                                             single
```

Now that we have a glimpse of his variables, let’s dive into his essays.

```
## 0: Self Summary
df[['essay0']].iloc[0,0]
"about me:<br />\n<br />\ni would love to think that i was some some kind of
intellectual:\neither the dumbest smart guy, or the smartest dumb guy. can't say i\ncan tell
the difference. i love to talk about ideas and concepts. i\nforge odd metaphors instead of
reciting cliches. like the\nsimularities between a friend of mine's house and an
underwater\nsalt mine. my favorite word is salt by the way (weird choice i\nknow). to me most
things in life are better as metaphors. i seek to\nmake myself a little better everyday, in
some productively lazy\nway. got tired of tying my shoes. considered hiring a five year\nold,
but would probably have to tie both of our shoes... decided to\nonly wear leather shoes dress
shoes.<br />\n<br />\nabout you:<br />\n<br />\nyou love to have really serious, really deep
conversations about\nreally silly stuff. you have to be willing to snap me out of a\nlight
hearted rant with a kiss. you don't have to be funny, but you\nhave to be able to make me
laugh. you should be able to bend spoons\nwith your mind, and telepathically make me smile
while i am still\nat work. you should love life, and be cool with just letting the\nwind blow.
extra points for reading all this and guessing my\nfavorite video game (no hints given yet).
and lastly you have a\ngood attention span."
```

```
## 1: What I'm doing with my life
df[['essay1']].iloc[0,0]
'currently working as an international agent for a freight\nforwarding company. import,
export, domestic you know the\nworks.<br />\nonline classes and trying to better myself in my
free time. perhaps\na hours worth of a good book or a video game on a lazy sunday.'

## 2: I’m really good at
df[['essay2']].iloc[0,0]
'making people laugh.<br />\nranting about a good salting.<br />\nfinding simplicity in
complexity, and complexity in simplicity.'

## 3: The first thing people usually notice about me
df[['essay3']].iloc[0,0]
'the way i look. i am a six foot half asian, half caucasian mutt. it\nmakes it tough not to
notice me, and for me to blend in.'

## 4: Favorite books, movies, show music, and food
df[['essay4']].iloc[0,0]
"books:<br />\nabsurdistan, the republic, of mice and men (only book that made me\nwant to
cry), catcher in the rye, the prince.<br />\n<br />\nmovies:<br />\ngladiator, operation
valkyrie, the producers, down periscope.<br />\n<br />\nshows:<br />\nthe borgia, arrested
development, game of thrones, monty\npython<br />\n<br />\nmusic:<br />\naesop rock, hail mary
mallon, george thorogood and the delaware\ndestroyers, felt<br />\n<br />\nfood:<br />\ni'm
down for anything."

## 5: The six things I could never do without
df[['essay5']].iloc[0,0]
'food.<br />\nwater.<br />\ncell phone.<br />\nshelter.'

## 6: I spend a lot of time thinking about
df[['essay6']].iloc[0,0]
'duality and humorous things'

## 7: On a typical Friday night I am
df[['essay7']].iloc[0,0]
'trying to find someone to hang out with. i am down for anything\nexcept a club.'

## 8: The most private thing I am willing to admit
df[['essay8']].iloc[0,0]
'i am new to california and looking for someone to wisper my secrets\nto.'

## 9: You should message me if...
df[['essay9']].iloc[0,0]
'you want to be swept off your feet!<br />\nyou are tired of the norm.<br />\nyou want to
catch a coffee or a bite.<br />\nor if you want to talk philosophy.'
```

1.5.1 Pedagogical Discussion

The first thing that we must address is the formatting. The code is riddled with HTML formatting code. We will clean this up before text analysis. Let's focus on some interesting essay topics. Essay 3 gives an opportunity for the user to describe how it would be like to meet them in person at first sight. This is data that can't be selected by a drop-down menu. This could help in good matches if users understand to respect the value of it. Essay 5 could very well measure a person's moral and materialistic values. It gives users a deeper connection than the objective variables. It could also be a very good predictor of other variables such as income or gender. The last note that might be relevant is a sense of character. Many users are deterred from using dating sites due to fear of connecting with "creepy", "needy", or even "stalker" types. I believe the 10 essays give a great way for users seeking for additional information.

1.6 Income

Perhaps the most materialistic variable to filter by, there is the income variable. (Note: For income, a response of '-1' means: prefer not to answer)

```
income = df[['income', 'sex']]
income_grouped = income.groupby('income')
income_grouped = income_grouped.count().reset_index()
income_grouped.rename(index=str, columns={"sex":"count"})
```

	income	count
0	-1	48442
1	20000	2952
2	30000	1048
3	40000	1005
4	50000	975
5	60000	736
6	70000	707
7	80000	1111
8	100000	1621
9	150000	631
10	250000	149
11	500000	48
12	1000000	521

1.6.1 Pedagogical Discussion

Not surprising, 48,442 prefer not to list their income. This is over half of our users! Although it would be really nice to have a predictor of income, it becomes very difficult for machine learning when over half of the data is missing for users. It's also important to note honesty in this section. Income is easy to mask due to loans and just simply not spending money. However, it may just be the most redeeming quality to the llack of others. (Think: age, education, and body type)

Thus, could this be the most dishonest section, omitting the '-1' values? There are over 500 users that make a million a year. We will see in part 2, cleaning the data.

Part 2 - Cleaning the Data

In part 2, we clean up the data before we process and analyze it. We also focus on extreme responses that deviate away from the average response. We clean up the numbers so we can find averages and make it ready for machine learning. We clean up the essays for the same reason.

2.1 Height

We know that men are generally taller than women. Let's see if the data represents the same theory.

```
height = df[['height', 'sex']]
height_grouped = height.groupby('sex')
height_grouped.mean().reset_index()

    sex   height
0   f     65.103873
1   m     70.443492
```

This seems fair so far. The average male is 70 inches tall while the average women are 65 inches tall. Let's now check the extreme values to assess the integrity of the data.

Here, we have a female standing at 4 inches tall. Also, we have a male at 1 inch tall.

```
height = df[['height', 'sex']]
height_grouped = height.groupby('sex')
height_grouped.min().reset_index()

    sex  height
0   f    4.0
1   m    1.0
```

Here, we have a female standing at 95 inches tall. Also, a male at 95 inches tall. Just for reference, the average NBA[3] basketball player stands at 79 inches tall.

```
height = df[['height', 'sex']]
height_grouped = height.groupby('sex')
```

[3] "NBA.com." http://www.nba.com/. Accessed 3 Aug. 2017.

```
height_grouped.max().reset_index()

    sex  height.max
0   f    95.0
1   m    95.0
```

```
## Getting standard deviation of height.
height_grouped.std()

sex    height.std
f      2.926502
m      3.076521
```

2.1.1 Pedagogical Discussion

It seems that these two extremes show that some of the responses for height may be wrong. According to US data from cdc.gov[4], the average height for female is 63.8 inches and for male is 69.3 inches If the data is more than 5 standard deviations away from the mean, I will replace it with a NaN value. Coincidently, the average standard deviation for men and women is 3.00 inches. So I am going replace if the user's height is over or under the US average by 15.00 inches with NaN values.

2.2 Space Camp vs. Master's Degree

If a person were to lie about their education, what else would they lie about? Here, we compare the average income of users associated with space camp and users possessing master's degrees.

```
## Looking at users associated with space
## camp, their reported average income
## comes out to be $262607.76
space_camp_income = space_camp['income']
space_camp_income = space_camp[space_camp_income != -1]
print("Average income for users associated with space camp:",
      ('%.2f' % space_camp_income['income'].mean()))

Average income for users associated with space camp: 262607.76

## Let's check the income for college
## graduates to compare with the
## users associated with space camp
## which is $106695.94
masters_income = df[df['education'].str.contains('graduated from college/university')==True]
masters_income = masters_income[masters_income != -1]
print("Average income for users with masters:",
      ('%.2f' % masters_income['income'].mean()))
```

[4] "Centers for Disease Control and Prevention." https://www.cdc.gov/. Accessed 3 Aug. 2017.

```
Average income for users with masters: 106695.94
```

2.2.1 Pedagogical Discussion

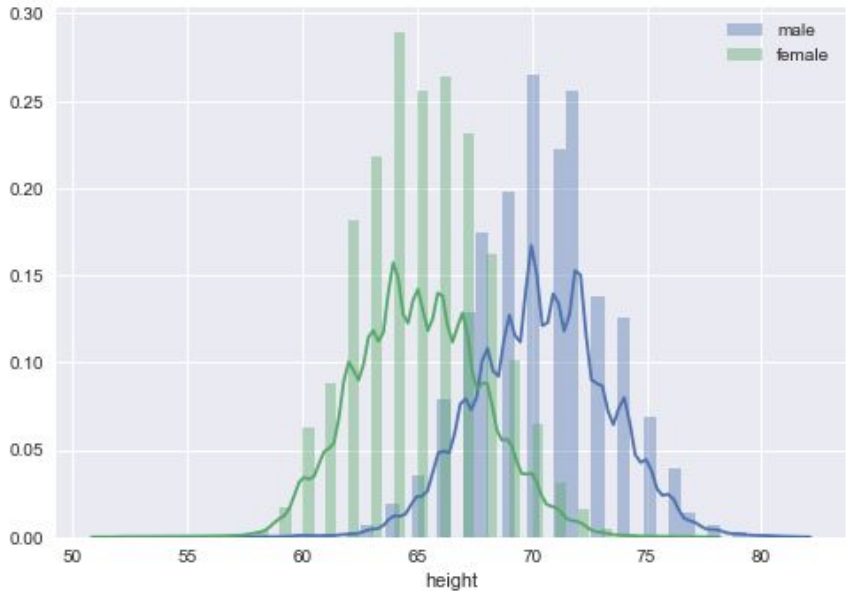
So according to the overall average, people who are associated with Space Camp seem to make more than double salary than the mean of all users. This is a significant difference, so we are going to have to omit it for users involved with space camp.

Part 3 - Analyzing the Data

Here, we use the clean dataset produced from part 2. We plot histograms, pie, and bar charts to see interesting relationships. If we see anything worth taking time with, we use it again for machine learning algorithms and predictions.

3.1 Male vs Female Heights

```
## Making a t-test
height_sex_df = df[['height', 'sex']]
male_height = height_sex_df[height_sex_df.sex=='m']
female_height = height_sex_df[height_sex_df.sex=='f']
stats.ttest_ind(male_height['height'].dropna(), female_height['height'].dropna())

Ttest_indResult(statistic=226.89713194246244, pvalue=0.0)

## Visualizing with seaborn
sns.distplot(male_height['height'].dropna(), label='male')
sns.distplot(female_height['height'].dropna(), label='female')
plt.legend()
```



3.1.1 Pedagogical Discussion

With a p-value of less than 0.01, we can confidently say that men are taller than women on average. We can also look at the distribution with seaborn after cleaning the data.

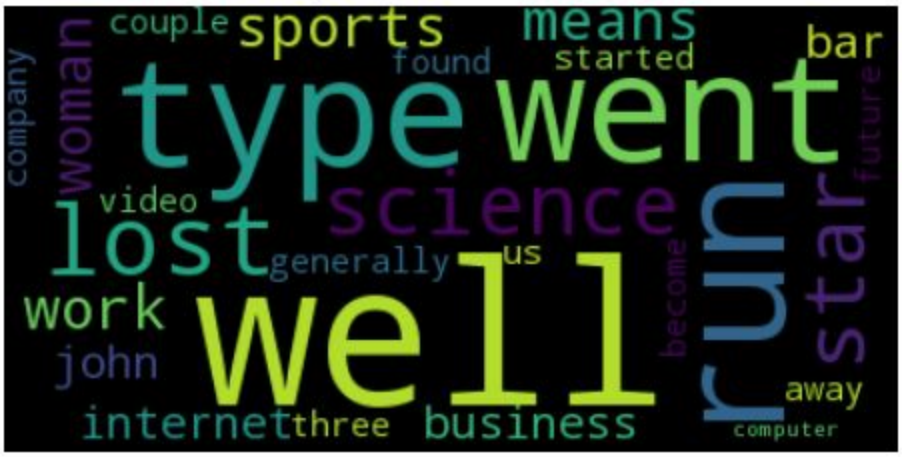
3.2 Unique words for Males vs Females

Here are the unique words from men and women.

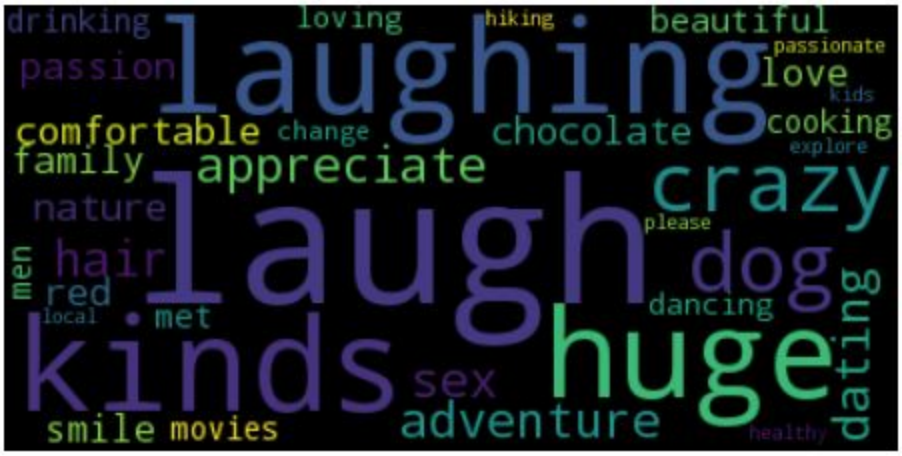
```
## Here we will take the top 500 words and find the unique
## words for each gender

men = pd.DataFrame(men_freq)[0]
women = pd.DataFrame(women_freq)[0]

unique_men = list(set(men) - set(women))
unique_women = list(set(women) - set(men))

unique_df = pd.concat([pd.DataFrame(unique_men), pd.DataFrame(unique_women)], axis=1)
unique_df.columns = ['unique_men', 'unique_women']
unique_df.head(15)
```

	unique_men	unique_women
0	type	laughing
1	well,	kinds
2	went	huge
3	that,	crazy
4	run	dog
5	lost	appreciate
6	science	sex
7	star	hair
8	sports	dating
9	here.	adventure
10	woman	laugh,
11	work.	comfortable
12	means	laugh.
13	2	chocolate
14	does	nature

Unique words for Men vs. Women



Unique words for Women vs. Men

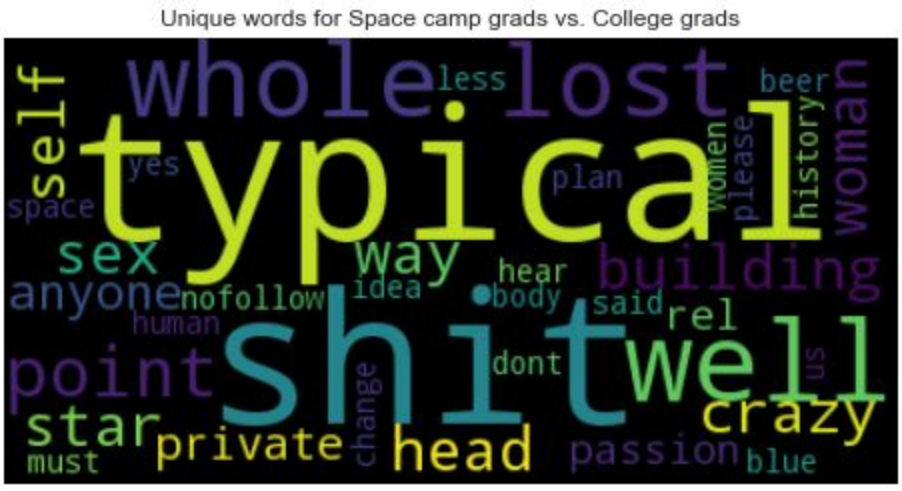


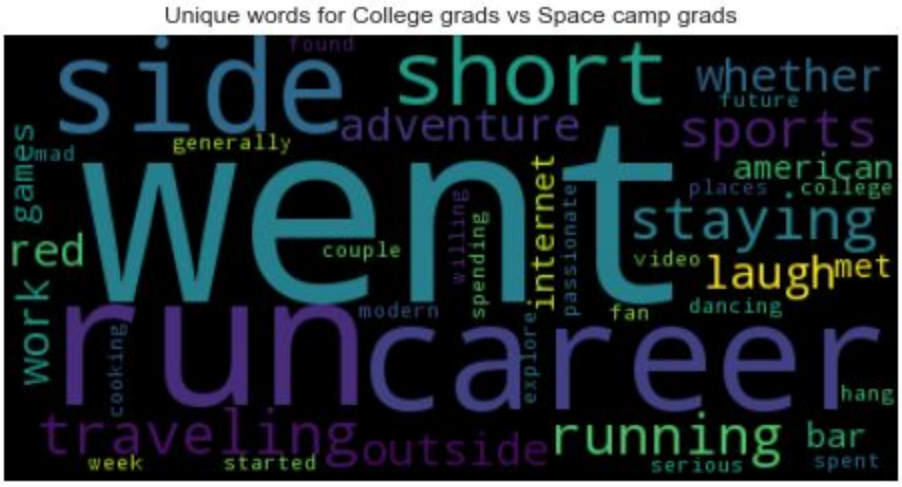
3.2.1 Pedagogical Discussion

Unique data here becomes especially useful since we can use it to determine other variables. If we can associate just a few words to classify men and women, we could also do it for variables such as education and others.

3.3 Unique words for Space camp grads vs College grads

Let’s compare space camp graduates versus college graduates. It would be nice to see if vocabulary level matches the degree.





3.3.1 Pedagogical Discussion

With the word cloud, we can already see what differs from space camp and college grads. This is useful in this case since we can use this data to possibly predict honesty in user profiles. If a college graduate's words deviated from the average excessively, a quick check on their other responses could be made to see if they are honest or not. Having accurate profiles should lead to happier users. If a user were to get a match based on false information, it could very well jeopardize their interest in using the website any further after a bad match based on false information.

Part 4 - Modeling the Data

Our final part of our analysis is focused on modeling the data with the help of Machine Learning. The overall intention here is to be able to predict data to help OkCupid users match more easily. We are going to be focusing on text data. Since we are using text, not numbers, we use classifiers. We have labeled data and are trying to predict categories so we use the Naive Bayes method.

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features.

4.1 Exploring Data

Here, we check on different categories for potential helpfulness in predicting OkCupid users. We have several variables at our disposal. One thing that we could do is combine categories for clustering. However, we have the most data in the number of essays. With over half a million essays, it could very well be enough of a sample size to predict another category. Something very simple that we can do is predict gender. However, we could predict education, income, or just about any other category with the same algorithms to be used in this notebook.

```
# Exploring possible variables for Machine Learning
n_men = len(df[df.sex == 'm'])
n_women = len(df[df.sex == 'f'])
n_essays = (df.loc[:,'essay0':'essay9']).size
n_body = df.body_type.unique().size
n_drink = df.drinks.unique().size
n_drug = df.drugs.unique().size
n_education = df.education.unique().size
n_ethnicity = df.ethnicity.unique().size
n_income = df.income.unique().size
n_orientation = df.orientation.unique().size

print("Number of male users: {:d}".format(n_men))
print("Number of essays: {:d}".format(n_essays))
print("Number of female users: {:d}".format(n_women))
print("Number of body types: {:d}".format(n_body))
print("Number of drinking types: {:d}".format(n_drink))
print("Number of drug types: {:d}".format(n_drug))
print("Number of ethnicity combinations: {:d}".format(n_ethnicity))
print("Number of income levels: {:d}".format(n_income))
print("Number of orientation types: {:d}".format(n_orientation))

Number of male users: 35829
Number of essays: 599460
Number of female users: 24117
Number of body types: 13
Number of drinking types: 7
```

```
Number of drug types: 4
Number of ethnicity combinations: 218
Number of income levels: 14
Number of orientation types: 3
```

4.1.1 Pedagogical Discussion

The reason why we use gender to predict first is that it has the least missing data. It is also very difficult to lie about your gender compared to all other variables. We could also use a combination of variables such as body type and diet if we don't have enough data for prediction of another category.

4.2 Data frame for Machine Learning

Here we make a smaller data frame to work with. We will eventually split the X and y values so that X can predict y. In this case, essays (X) will predict gender (y).

```
# Combining all 10 essays to a single column for each user
# Also fixing missing essays to blank values
df['all_essays'] = ''
essay_names = df.loc[:,'essay0':'essay9']
for essay_name in essay_names:
    df[essay_name] = df[essay_name].replace(np.nan, ' ')
    df['all_essays'] = df[essay_name] + ' ' + df['all_essays']

# Dataframe for essay to gender predictor
# Could reproduce to include other variables
essay_sex = df[['all_essays', 'sex']]
essay_sex.head(15)
```

	all_essays	sex
0	you want to be swept off your feet! you are t...	m
1	i am very open and will share just about any...	m
2	you are bright, open, intense, silly, ironic, ...	m
3	you feel so inclined. Cats and german phil...	m
4	music: bands, rappers, musicians at...	m
5	you're awesome. i cried on my first day at sch...	m
6	my typical friday night plotting to take ove...	f
7	out and about or relaxing at home with a g...	f
8	http://www.youtube.com/watch?v=4dxbwzuwsxk let...	f
9	you can rock the bells and say hi....	m
10	you are a complex woman with healthy self-este...	m
11	if you know who you are, who you want, where y...	m
12	bang my shit bang	m
13		f
14	...you genuinely think we'd be a good match. h...	f

4.3 Vector Space Model for Naive Bayes

First, we convert essays into a vector space model to act as a search engine for machine learning. Then, we MultinomialNB with the naive Bayes algorithm for multinomial distributed data. It is also one of the two classic naive Bayes variants used in text classification.

```
# Converting essays into a vector space model
from sklearn.feature_extraction.text import CountVectorizer

def make_xy(essay_sex, vectorizer=None):
    #Your code here
    if vectorizer is None:
        vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(essay_sex.all_essays)
    X = X.tocsc()  # some versions of sklearn return COO format
    y = (essay_sex.sex == 'm').values.astype(np.int)
    return X, y
X, y = make_xy(essay_sex)

from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, y)
clf = MultinomialNB().fit(xtrain, ytrain)
print("MN Accuracy: %0.2f%%" % (100 * clf.score(xtest, ytest)))
MN Accuracy: 72.42%

training_accuracy = clf.score(xtrain, ytrain)
test_accuracy = clf.score(xtest, ytest)

print("Accuracy on training data: %0.2f" % (training_accuracy))
print("Accuracy on test data:     %0.2f" % (test_accuracy))
Accuracy on training data: 0.79
Accuracy on test data:     0.72
```

4.4 Interpretation

We use a neat trick to identify strongly predictive features (i.e. words). first, we create a data set such that each row has exactly one feature. This is represented by the identity matrix. use the trained classifier to make predictions on this matrix sort the rows by predicted probabilities, and pick the top and bottom K rows.

```
# Getting list of words that predict gender
words = np.array(vectorizer.get_feature_names())

x = np.eye(xtest.shape[1])
probs = clf.predict_log_proba(x)[:, 0]
ind = np.argsort(probs)
```

```
men_words = words[ind[:10]]
women_words = words[ind[-10:]]

men_prob = probs[ind[:10]]
women_prob = probs[ind[-10:]]

print("Men words\t           P(Men word | word)")
for w, p in zip(men_words, men_prob):
    print("{:>20}".format(w), "{:.2f}".format(1 - np.exp(p)))

print("Women words\t        P(Men word | word)")
for w, p in zip(women_words, women_prob):
    print("{:>20}".format(w), "{:.2f}".format(1 - np.exp(p)))
```

```
  Men words            P(Men word | word)
         Soundcloud   0.90
           mechanic   0.90
          computers   0.90
            robocop   0.90
              beard   0.90
               nbsp   0.89
              djing   0.89
           hyperion   0.89
          guitarist   0.89
             escher   0.89

  Women words          P(Men word | word)
          oleander   0.17
            petite   0.17
             girly   0.17
             zumba   0.17
             femme   0.16
           mascara   0.15
            girlie   0.14
         lipstick   0.13
            tomboy   0.10
             gloss   0.10
```

4.4.1 Pedagogical Discussion

Here we get words that our Naive Bayes model thinks are strongly correlated with male versus females. Notice how the words are different than what we achieved in our wordcloud. This is due to attaching a probability to each term rather than just getting the top word counts.

4.5 Misinterpretation

Our model isn't perfect. It can't always predict genders accurately. Some simple variables that could influence essays are variables like orientation and education. Here we explore the mis-predicted quotes.

```
x, y = make_xy(essay_sex, vectorizer)

prob = clf.predict_proba(x)[:, 0]
predict = clf.predict(x)

bad_women = np.argsort(prob[y == 0])[:3]
bad_men = np.argsort(prob[y == 1])[-3:]

print("Mis-predicted Women quotes")
print('---------------------------')
for row in bad_women:
    print(essay_sex[y == 0].all_essays.iloc[row])
    print('')
```

```
Mis-predicted Women quotes
---------------------------
you want to chat or get to know me, have similar interests. i like guys who are responsible, fun/silly, ambitious, have a job (or are in school), and i prefer that they don't smoke. i like when they are caring, smart, have (some) manners (haha), is ambitious, kind, honest, near to my age, don't live too far, and it's nice if they like to work out a few times a week. i need more friends that live nearby! either home watching tv or a movie, doing homework,  out with a friend/friends, or  online... what i want to "do with my life", career/going back to school  worrying :)  homework/studying  forcing myself to go to the gym...haha  guys  music movies  tv  family  friends my family  my friends  food  music  movies/tv  my contacts/glasses books: <a class="ilink" href="/interests?i=harry+potter">harry potter</a> series, <a class="ilink" href= "/interests?i=the+time+traveller%27s+wife">the time traveller's...


```

```
print("Mis-predicted Men quotes")
print('-------------------------')
for row in bad_men:
    print(essay_sex[y == 1].all_essays.iloc[row])
    print('')
```

```
Mis-predicted Men quotes
-------------------------
you're kind and beautiful.  you practice mindfulness, or at least are open to meditation.  you have a big heart, love nature and knowledge.  you recycle and compost!    you're brave. as in, brave enough to jump on the back of a motorcycle and take the less traveled path. brave enough to be kind to strangers and strive for the life you envision. i've gotten in the habit of doing 24 hour fasts, usually from 6pm to 6pm, about every other week. i started this practice out of gratitude for the unbelievable abundance that we have in our lives. since then i've been researching fasting and the health benefits are awesome. i won't list them all here but
```

```
check it out for yourself! slow dancing with myself in the kitchen... or hanging out with
friends. i'm not into the club or late night scene but love to be social and be around good
people. dinner parties and dive bars are pretty typical.  i also love getting out of...
```

4.5.1 Pedagogical Discussion

It seems that the simple reason to a mis-prediction for a female is too much html code formatting. Things like adding hyperlinks and formatting could throw off the algorithm. Also, using crude language is a case as well. For men, it seems that sexual orientation strongly affects predictions. We could teach our machine learning algorithm to ignore excess html or sexual orientation but these data points may be important to assess user data integrity. Instead of ignoring, we should instead combine categories such as orientation or hobbies with essays to predict gender.

4.6 Testing Machine Learning Algorithm

It is important that we test our model on new data. Retrospective analysis is good, but predicting future outcomes is even better. Here we try a few sentences to predict gender.

First example is very simple. As expected, we get a good result at “96.81 percent likely from a man’s essay”.

```
text = ['I am a programmer with a beard']

vectorizer = CountVectorizer(min_df=0.001,
                             vocabulary=vectorizer.get_feature_names())
x = vectorizer.fit_transform(text)
prob = clf.predict_proba(vectorizer.transform(text))
man_prob = prob[0, 1]
woman_prob = prob[0, 0]
man_prob = ('%.2f' % (100*man_prob))
woman_prob = ('%.2f' % (100*woman_prob))

# Prediction
print("Text/Essay: ", text)
print('')
if clf.predict(x) == 1:
    print("This is", man_prob, "percent likely from a man's essay")
else:
    print("This is", woman_prob, "percent likely from a woman's essay")

Text/Essay:  ['I am a programmer with a beard']
This is 96.81 percent likely from a man's essay
```

Second example is similar to the first, we get a good result at “87.92% percent likely from a woman’s essay”.

```
text = ['I like lipgloss and lipstick']
```

```
vectorizer = CountVectorizer(min_df=0.001,
                             vocabulary=vectorizer.get_feature_names())
x = vectorizer.fit_transform(text)
prob = clf.predict_proba(vectorizer.transform(text))
man_prob = prob[0, 1]
woman_prob = prob[0, 0]
man_prob = ('%.2f' % (100*man_prob))
woman_prob = ('%.2f' % (100*woman_prob))

# Prediction
print("Text/Essay: ", text)
print('')
if clf.predict(x) == 1:
    print("This is", man_prob, "percent likely from a man's essay")
else:
    print("This is", woman_prob, "percent likely from a woman's essay")

Text/Essay:  ['I like lipgloss and lipstick']
This is 87.92 percent likely from a woman's essay
```

Let's try doing clever manipulations on a simple sentence which would reverse its meaning. We start with this below.

```
text = ['I like computers']
vectorizer = CountVectorizer(min_df=0.001,
                             vocabulary=vectorizer.get_feature_names())
x = vectorizer.fit_transform(text)
prob = clf.predict_proba(vectorizer.transform(text))
man_prob = prob[0, 1]
woman_prob = prob[0, 0]
man_prob = ('%.2f' % (100*man_prob))
woman_prob = ('%.2f' % (100*woman_prob))

# Prediction
print("Text/Essay: ", text)
print('')
if clf.predict(x) == 1:
    print("This is", man_prob, "percent likely from a man's essay")
else:
    print("This is", woman_prob, "percent likely from a woman's essay")

Text/Essay:  ['I like computers']
This is 90.21 percent likely from a man's essay
```

Seeing the positive result above, we try to see if we can get the opposite by reversing the word "like" to "hate". Our model does not do so well, it still classifies the user as a male but with less confidence.

```
text = ['I hate computers']
```

```
vectorizer = CountVectorizer(min_df=0.001,
                             vocabulary=vectorizer.get_feature_names())
x = vectorizer.fit_transform(text)
prob = clf.predict_proba(vectorizer.transform(text))
man_prob = prob[0, 1]
woman_prob = prob[0, 0]
man_prob = ('%.2f' % (100*man_prob))
woman_prob = ('%.2f' % (100*woman_prob))
# Prediction
print("Text/Essay: ", text)
print('')
if clf.predict(x) == 1:
    print("This is", man_prob, "percent likely from a man's essay")
else:
    print("This is", woman_prob, "percent likely from a woman's essay")

Text/Essay:  ['I hate computers']
This is 87.96 percent likely from a man's essay
```

4.7 Possible Improvements

Seeing how our model differentiates "like" and "hate" properly, we can look deeper into the text analysis that we used. Something that we could look into is n-grams.

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles.

An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of n in modern language, e.g., "four-gram", "five-gram", and so on.

A simple example could be from our example previously used. For a unigram: "I" "hate" "coumputers". This would be influenced more by the term: computers which is generally associated with males. For a bi-gram: "I hate" "hate computers". This would be influenced more by the term: "hate computers" which is generally not associated with males.

Testing with uni/bigrams

Compared to our previous model of 73/71 percent for traning/test data, there is only a 1 percent improvement. This could be utilized in a next study, but not enough to affect our current results.

```
vectorizer = CountVectorizer(ngram_range=(1,2),min_df=0.001)
X, y = make_xy(essay_sex, vectorizer)
xtrain=X[mask]
ytrain=y[mask]
xtest=X[~mask]
```

```
ytest=y[~mask]

clf = MultinomialNB(alpha=5).fit(xtrain, ytrain)

training_accuracy = clf.score(xtrain, ytrain)
test_accuracy = clf.score(xtest, ytest)

print("Accuracy on training data: {:2f}".format(training_accuracy))
print("Accuracy on test data:     {:2f}".format(test_accuracy))

Accuracy on training data: 0.747152
Accuracy on test data:     0.716415
```

Testing with only bigrams

The results with bigrams are better than our 73/71 model but overfitted. We should skip this.

```
vectorizer = CountVectorizer(ngram_range=(2,2),min_df=0.001)
X, y = make_xy(essay_sex, vectorizer)
xtrain=X[mask]
ytrain=y[mask]
xtest=X[~mask]
ytest=y[~mask]

clf = MultinomialNB(alpha=5).fit(xtrain, ytrain)

training_accuracy = clf.score(xtrain, ytrain)
test_accuracy = clf.score(xtest, ytest)

print("Accuracy on training data: {:2f}".format(training_accuracy))
print("Accuracy on test data:     {:2f}".format(test_accuracy))

Accuracy on training data: 0.757257
Accuracy on test data:     0.717860
```

Testing with random forests

Here, we get an amazing 98 percent accuracy on the training data but it is the most overfitted model we have seen. Skip this.

```
from sklearn.ensemble import RandomForestClassifier

vectorizer = CountVectorizer(min_df=0.001)
X, y = make_xy(essay_sex, vectorizer)
xtrain=X[mask]
ytrain=y[mask]
xtest=X[~mask]
ytest=y[~mask]

rforest = RandomForestClassifier()
```

```
clf = rforest.fit(xtrain,ytrain)

training_accuracy = clf.score(xtrain, ytrain)
test_accuracy = clf.score(xtest, ytest)

print("Accuracy on training data: {:2f}".format(training_accuracy))
print("Accuracy on test data:     {:2f}".format(test_accuracy))

Accuracy on training data: 0.980673
Accuracy on test data:     0.679493
```

4.5.1 Pedagogical Discussion

Although implementing n-grams may produce better results, we shouldn't dismiss other classifiers such as random forests. If we were to add more variables in with our essays such as orientation, diet, and height, perhaps other classifiers that did not perform so well would be more useful.

Conclusions and Recommendations for OkCupid

With this study, there are some conclusions that could help out OkCupid and its users. We are going to focus on getting more data, filtering out inconsistent users, and suggestions for users.

Getting more data

OkCupid does an amazing job at having many variables so that users can answer some versus very few or even none. However, data is important to OkCupid more than its users. It could be used to improve its platform to beat competitors.

For example, OkCupid could simply send out an anonymous survey to its users asking what 5 variables do they look for most in a partner. Or they could even do a blind study of seeing what users filter for the most. Suppose if users care about income and education the most, OkCupid should do everything to ensure responses are made. Declining to share income or even "space camp" for education might not be helpful for future studies to help users get what they want.

Filtering out inconsistent users

Nothing is more frustrating than finding a match online and finding out you been catfished.[5] "Catfished" is casually defined as "being deceived by false information online" by urban dictionary.[6] This could be tracked by inconsistent responses in a user's profile.

Some ways OkCupid could prevent this is by comparing similar response categories with the average mean. For example, if a 19-year-old college student happened to be a millionaire, they should be compared with what other 19-year-old college students make. Instead of banning their account, OkCupid could just give them fewer matches or matches with other inconsistent users. The reason why OkCupid shouldn't outright ban inconsistent users is that sometimes, the user is actually being honest. Also, banning users would bring in less income overall for OkCupid.

Suggestion for users

Suppose that it is difficult for a user to give responses or is unaware that a good full profile will result in more quality matches. A full profile could mean honesty and transparency for users. This is a lot better than a weak profile which could lead to a blind date feeling.

In order to help out users, OkCupid could use the Machine Learning predictions to help users auto-complete their profiles or even encourage users through positive reinforcement. Badges could be earned for complete profiles. This is similar in concept to LinkedIn's "all-star" profile where users have a complete profile.[7]

---

[5] "10 Ways To Catch Out A Catfish - eHarmony." http://www.eharmony.com.au/dating-advice/trust-and-safety/10-ways-to-catch-out-a-catfish. Accessed 5 Aug. 2017.

[6] "Urban Dictionary: catfished." http://www.urbandictionary.com/define.php?term=catfished. Accessed 5 Aug. 2017.

[7] "How to Achieve LinkedIn All-Star Profile Status | David Weaver, CCA ...." 28 Mar. 2016, https://www.linkedin.com/pulse/how-achieve-linkedin-all-star-profile-status-david-weaver. Accessed 5 Aug. 2017.