OkayCupid Capstone

January 20, 2019

1 OkayCupid - Capstone Project

Author: Justin Bard

In this capstone I will create a presentation about my findings on sample data provided by OkCupid. The purpose of this capstone is to practice formulating questions and implementing Machine Learning techniques to answer the following questions:

1.1 Project Questions

- Can gender be determined based on words used in essay answers
- Can income be determined based off words used in essay answers
- Can education be determined based off words used in essay answers

```
In [1]: import re, time, os
        import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_selection import chi2
        from sklearn.model_selection import train_test_split
```

```
In [2]: %matplotlib inline
```

2 Cleaning Up the Data

This function is used to clean up the essay data. Essentially it's just removing punctuation and HTML tags from the essay data.

```
In [3]: def data_therapy(dataFrame):
            temp_holder = []
            dataFrame = dataFrame.replace(['<br />', '<b>', '</b>','<a.*>', '\n', '-', '[,.!"/()
            for i in range(len(dataFrame)):
                temp_holder.append(re.sub(' +', ' ', dataFrame[i].strip().lower()))

            return temp_holder
```

```
In [5]: data = pd.read_csv('profiles.csv') ## Create DataFrame
```

2.1 Adding Columns

To make the data easier to work with later, I created a key map for both the education and the gender columns. To do this I broke down each unique entry for both categories and then assigned a number to them.

Afterwards, all unneeded columns were removed for efficiency.

```
In [6]: educations = data.education.replace(np.nan, 'not_set', regex=True)
        educations = educations.sort_values().unique()

        counter = 0
        education_map = {}

        for education in educations:
            education_map[education] = int(counter)
            counter += 1

        data["education_code"] = data.education.map(education_map)
        data = data[pd.notnull(data['education_code'])]
        data["education_code"] = data.education_code.astype('int64')

        education_code = data["education_code"]
        educations_code = education_code.replace(np.nan, 'not_set', regex=True)

        genders = data.sex.replace(np.nan, '', regex=True)
        genders = genders.sort_values().unique()

        counter = 0
        gender_map = {}

        for gender in genders:
            gender_map[gender] = counter
            counter += 1

        data["gender_code"] = data.sex.map(gender_map)
```

```
In [7]: col = ['gender_code', 'sex', 'education_code', 'education', 'essay0']
        essay_data = data[col]
        essay_data = essay_data[pd.notnull(data['essay0'])]
        essay_data = essay_data[pd.notnull(data['gender_code'])]
        essay_data = essay_data[pd.notnull(data['education_code'])]
```

```
essay_data.columns = col
essay_data = essay_data.reset_index(drop=True)

essay_data['essay0'] = data_therapy(essay_data['essay0'])
```

```
/home/jbard/anaconda3/envs/capstone/lib/python3.5/site-packages/ipykernel_launcher.py:4: UserWar
  after removing the cwd from sys.path.
/home/jbard/anaconda3/envs/capstone/lib/python3.5/site-packages/ipykernel_launcher.py:5: UserWar
  """
```

2.1.1 The new dataframe:

In [9]:
```
essay_data.head(n=10)
```

Out[9]:
```
   gender_code sex  education_code                         education  \
0            1   m              25      working on college/university
1            1   m              31              working on space camp
2            1   m              12     graduated from masters program
3            1   m              25      working on college/university
4            1   m               9  graduated from college/university
5            1   m               9  graduated from college/university
6            0   f               9  graduated from college/university
7            1   m              32         working on two-year college
8            1   m               9  graduated from college/university
9            1   m              28          working on masters program

                                              essay0
0  about me i would love to think that i was some...
1  i am a chef this is what that means 1 i am a w...
2  i'm not ashamed of much but writing public tex...
3               i work in a library and go to school
4  hey how's it going currently vague on the prof...
5  i'm an australian living in san francisco but ...
6  life is about the little things i love to laug...
7  my names jake i'm a creative guy and i look fo...
8  i was born in wisconsin grew up in iowa and mo...
9  i just moved to the bay area from austin tx or...
```

---

3 Data Analysis

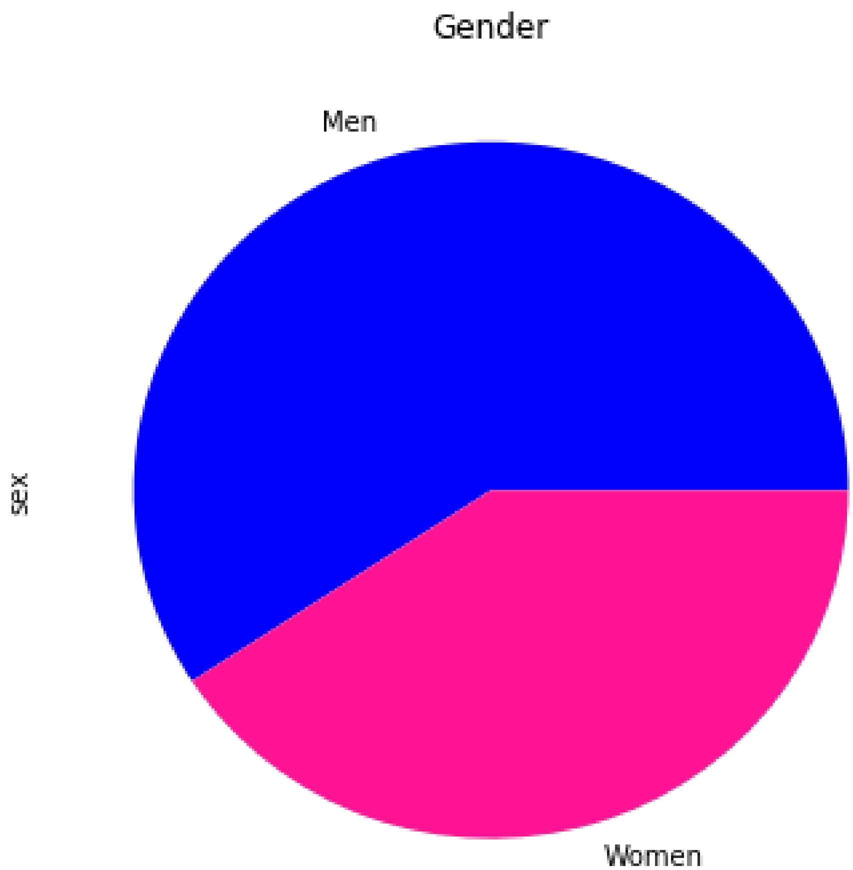
Here is an exploration of the data that will be anaylized to see how feesable the project questions are.

3.1 Gender Analysis

```
In [9]: male_count, female_count = data['sex'].value_counts()
        gender_data = data['sex'].value_counts()
        male_by_age = data.loc[data['sex'] == 'm'].groupby('age').count()['sex']
        female_by_age = data.loc[data['sex'] == 'f'].groupby('age').count()['sex']
```

```
In [10]: plt.title("Gender")
         gender_data.plot.pie(labels=['Men', 'Women'], colors=['blue', '#FF1493'], figsize=(6,6)
```

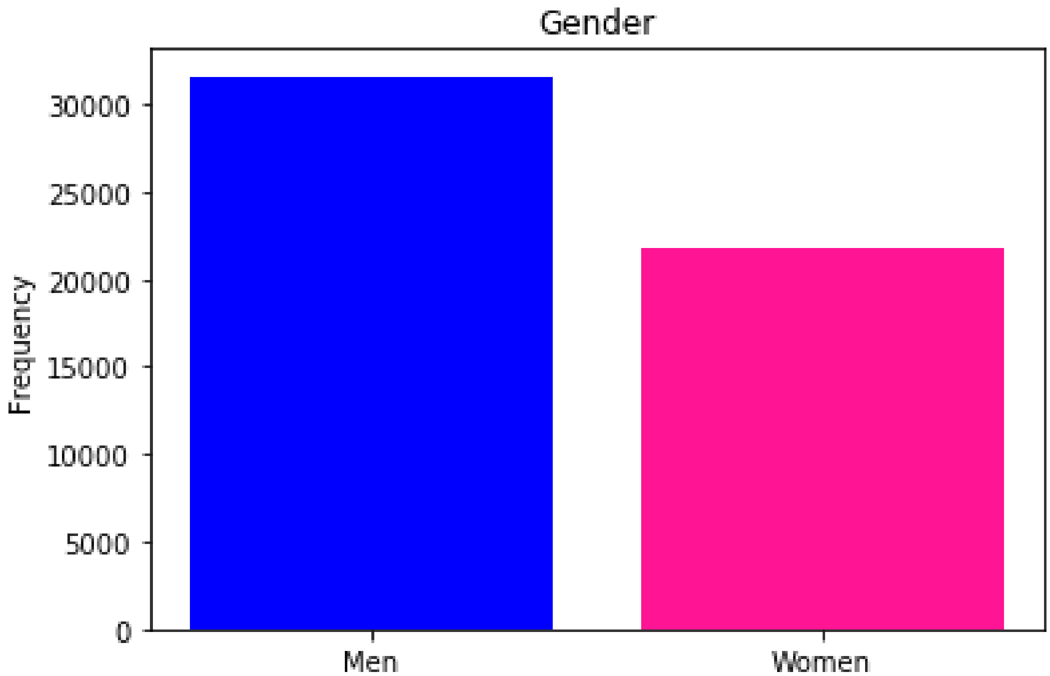
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0a4d04cc50>
```




```
In [11]: objects = ('Men', 'Women')
         y_pos = np.arange(len(objects))
```

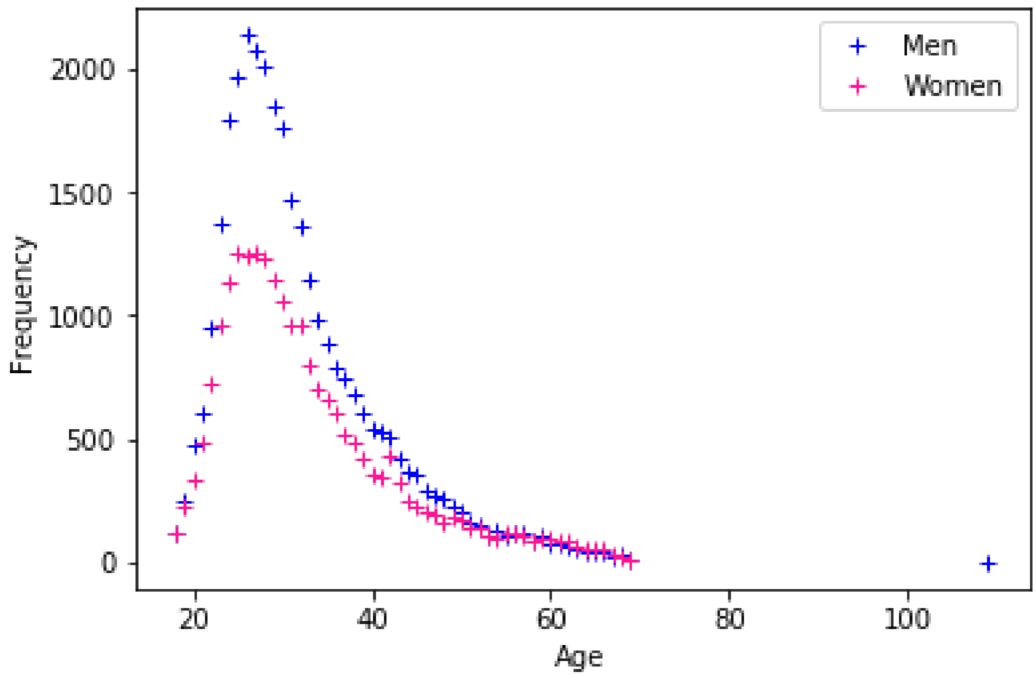
```
In [13]: plt.bar(y_pos, [male_count, female_count], color=['blue','#FF1493'], align='center')
         plt.xticks(y_pos, objects)
         plt.ylabel('Frequency')
         plt.title('Gender')
```

Out[13]: <matplotlib.text.Text at 0x7f0a4e5ef710>




```
In [14]: plt.plot(male_by_age, '+', color='blue')
         plt.plot(female_by_age, '+', color='#FF1493')
         plt.legend(['Men', 'Women'])
         plt.xlabel('Age')
         plt.ylabel('Frequency')
```

Out[14]: <matplotlib.text.Text at 0x7f0a4e679d30>



3.1.1 Results

The results clearly demonstrate that overall, more men than women are in the data. Additionally, the data is narrowing down the male to female ratio based off of age.

3.2 Income Analysis

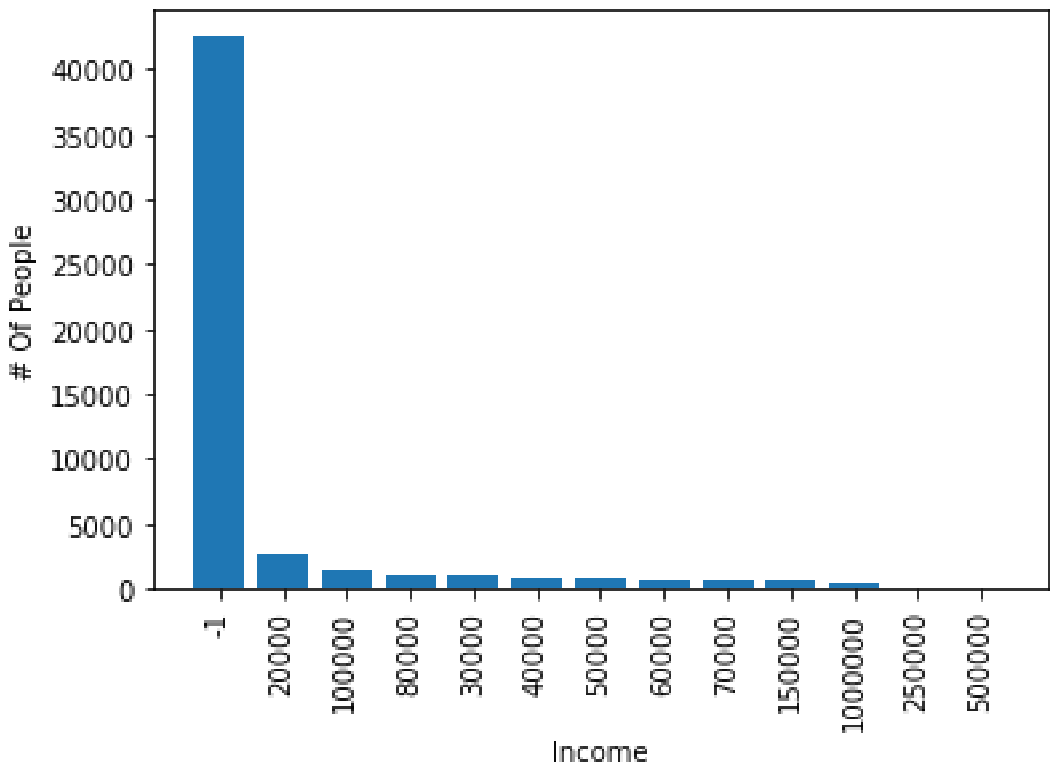
In [15]:
```
x_data = data['income'].sort_values().value_counts()
y_data = data['income'].sort_values().value_counts().index.tolist()

y_pos = np.arange(len(y_data))
x_data
```

Out[15]:
```
-1         42535
 20000      2795
 100000     1534
 80000      1031
 30000       965
 40000       929
 50000       902
 60000       689
 70000       665
 150000      605
```

```
 1000000      490
 250000       133
 500000        45
Name: income, dtype: int64
```

```
In [16]: plt.bar(y_pos, x_data)
         plt.xticks(y_pos, y_data, rotation=90)
         plt.ylabel('# Of People')
         plt.xlabel('Income')
```

```
Out[16]: <matplotlib.text.Text at 0x7f0a4e679a20>
```



3.2.1 Results

After analyzing the income data, it can be seen that the vast majority of the samples decided to not provide their income level. These findings suggest it will likely be very difficult to create accurate predictions of individual income based off of any other metric in the data set.

3.3 Education Analysis

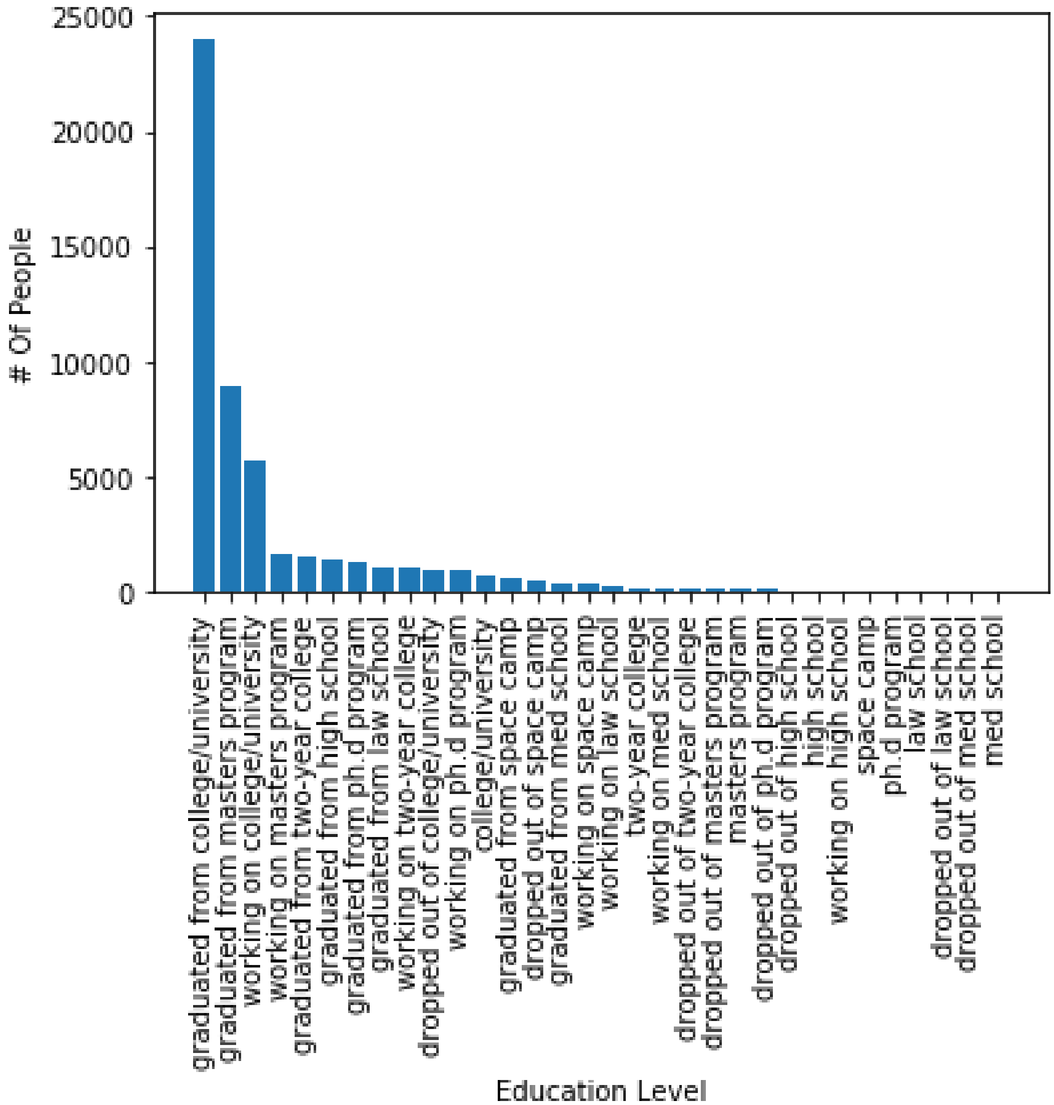
In [17]:
```
education_x = data['education'].value_counts()
education_y = data['education'].value_counts().index.tolist()

education_y_pos = np.arange(len(education_y))
```

In [18]:
```
education_x
```

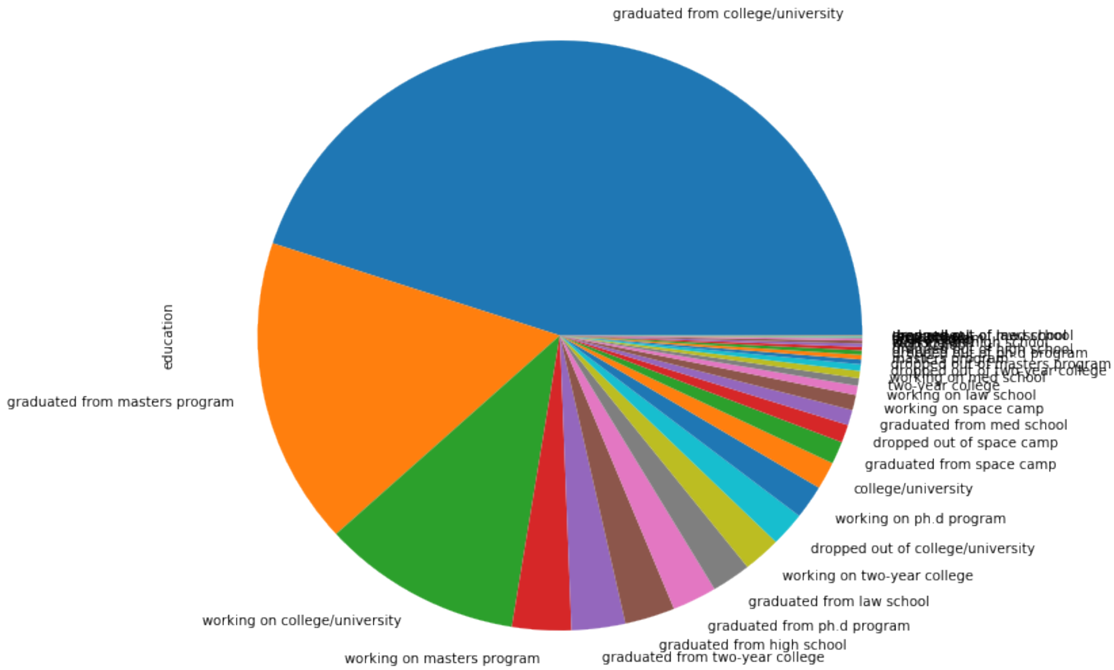
Out[18]:
```
graduated from college/university    23959
graduated from masters program        8961
working on college/university         5712
working on masters program            1683
graduated from two-year college       1531
graduated from high school            1428
graduated from ph.d program           1272
graduated from law school             1122
working on two-year college           1074
dropped out of college/university      995
working on ph.d program                983
college/university                     801
graduated from space camp              657
dropped out of space camp              523
graduated from med school              446
working on space camp                  445
working on law school                  269
two-year college                       222
working on med school                  212
dropped out of two-year college        191
dropped out of masters program         140
masters program                        136
dropped out of ph.d program            127
dropped out of high school             102
high school                             96
working on high school                  87
space camp                              58
ph.d program                            26
law school                              19
dropped out of law school               18
dropped out of med school               12
med school                              11
Name: education, dtype: int64
```

In [19]:
```
plt.bar(education_y_pos, education_x)
plt.xticks(education_y_pos, education_y, rotation=90)
plt.ylabel('# Of People')
plt.xlabel('Education Level')
```

Out[19]:
```
<matplotlib.text.Text at 0x7f9748196390>
```




```
In [20]: education_x.plot.pie(labels=education_y, figsize=(10,10))

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9743586ac8>
```



3.3.1 Results

It appears that "graduated from college/university" and "graduated from masters program" have the most data counts but there seems to be enough diversity that we can try to predict education level. I suspect that prediction accuracy may be most accurate for the larger datasets and less so with the education categories that contain fewer samples.

---

4 Training

One common approach for extracting features from text is to use the bag of words model: a model where for each document, a 'describe yourself' essay in our case, the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored.

Specifically, for each term in our dataset, we will calculate a measure called Term Frequency, Inverse Document Frequency, abbreviated to tf-idf. I use sklearn.feature_extraction.text.TfidfVectorizer to calculate a tf-idf vector for each of essay0 response provided.

4.1 Transformation!

```
In [21]: tfidf = TfidfVectorizer(sublinear_tf=True, min_df=2, norm='l2', encoding='latin-1', ngr
         features = tfidf.fit_transform(essay_data['essay0']).toarray()
```

- sublinear_df is set to True to use a logarithmic form for frequency.
- min_df is the minimum numbers of documents a word must be present in to be kept. - NOTE: This is very expensive
- norm is set to l2, to ensure all our feature vectors have a euclidian norm of 1.
- ngram_range is set to (1, 2) to indicate that we want to consider both unigrams and bigrams.
- stop_words is set to "english" to remove all common pronouns ("a", "the", ...) to reduce the number of noisy features.

```
In [22]: gender_labels = essay_data['sex'].factorize()[0]
         gender_id_df = essay_data[['sex', 'gender_code']].drop_duplicates().sort_values('gender
         gender_to_id = dict(gender_id_df.values)

         education_labels = essay_data['education'].factorize()[0]
         education_id_df = essay_data[['education', 'education_code']].drop_duplicates().sort_va
         education_to_id = dict(education_id_df.values)

         features.shape

Out[22]: (48885, 300896)
```

---

5 Gender - Unigrams and Bigrams:

```
In [22]: N = 5

         for sex, gender_code in sorted(gender_to_id.items()):
             features_chi2 = chi2(features, gender_labels == gender_code)
             indices = np.argsort(features_chi2[gender_code])
             feature_names = np.array(tfidf.get_feature_names())[indices]
             unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
             bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
             print("# '{}':".format(sex))
             print("     . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-N:])))
             print("     . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-N:])))
             print()

# 'f':
     . Most correlated unigrams:
. sassy
. gal
. love
. girl
. guy
```

```
     . Most correlated bigrams:
. nice guy
. girl loves
. going guy
. guy looking
. love laugh

# 'm':
     . Most correlated unigrams:
. syrupy
. danville
. attempts
. bilbao
. warrants
     . Most correlated bigrams:
. family drink
. illegal drugs
. band probably
. informed things
. directly proportional
```

Everything seems to make sense so far.

5.1 Gender Data - Training and Testing

```
In [23]: x_train, x_test, y_train, y_test = train_test_split(essay_data['essay0'], essay_data['s
         count_vect = CountVectorizer()
         x_train_counts = count_vect.fit_transform(x_train)
         tfidf_transformer = TfidfTransformer()
         x_train_tfidf = tfidf_transformer.fit_transform(x_train_counts)

         x_test_counts = count_vect.transform(x_test)
         x_test_tfidf = tfidf_transformer.transform(x_test_counts)

         classifier = MultinomialNB().fit(x_train_tfidf, y_train)
```

I picked a small random sample from the test set to see how good the predictions in action.

```
In [24]: random_list_for_gender_test = [23985, 32768, 3161, 20330, 25112]

         for i in random_list_for_gender_test:
```

```
            print ("Essay Response:")
            print (x_test[i])
            print ("------------------------------")
            print ("Prediction: "+str(classifier.predict(count_vect.transform([x_test[i]]))[0])
            print ("Actual     : "+y_test[i])
            print ()
```

```
Essay Response:
i'm an astute girl who enjoys warm and quick witted but not empty bantering individuals i happen
------------------------------
Prediction: f
Actual     : f

Essay Response:
after five years on the east coast i guess it's only appropriate for me to head out to the golde
------------------------------
Prediction: m
Actual     : m

Essay Response:
my friends call me joe i'm a creative person and i love to make new things out of whatever is ha
------------------------------
Prediction: m
Actual     : m

Essay Response:
i'm from the bay area i don't take life too seriously fun love and happiness are things that dri
------------------------------
Prediction: m
Actual     : m

Essay Response:
a modern romantic who's sometimes a realist but who's always sarcastic honest and opinionated th
------------------------------
Prediction: m
Actual     : f
```

```
In [25]: print ("Accuracy: "+str(classifier.score(x_test_tfidf, y_test)))
```

```
Accuracy: 0.621420389462
```

5.1.1 Results

Our accuracy is 62%! Not horriable but I was hoping for better.

6 Education - Unigrams and Bigrams:

```
In [ ]: N = 5

        for education, education_code in sorted(education_to_id.items()):
            features_chi2 = chi2(features, education_labels == education_code)
            indices = np.argsort(features_chi2[0])
            feature_names = np.array(tfidf.get_feature_names())[indices]
            unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
            bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
            print("# '{}':".format(education))
            print("     . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-N:])))
            print("     . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-N:])))
            print()
```

```
# 'college/university':
     . Most correlated unigrams:
. sfsu
. im
. studying
. majoring
. student
     . Most correlated bigrams:
. academy art
. currently student
. going school
. time student
. college student

# 'dropped out of college/university':
     . Most correlated unigrams:
. wonky
. happenin
. xy
. hominid
. debauched
     . Most correlated bigrams:
. pretend wrote
. intrigued life
. better character
. new easy
. american spirit

# 'dropped out of high school':
     . Most correlated unigrams:
. india
. master
. masters
```

```
. mba
. im
     . Most correlated bigrams:
. social worker
. got master
. master degree
. business school
. grad school

# 'dropped out of law school':
     . Most correlated unigrams:
. school
. phd
. graduated
. im
. student
     . Most correlated bigrams:
. went school
. time student
. college student
. graduated college
. grad school

# 'dropped out of masters program':
     . Most correlated unigrams:
. trio
. applesauce
. spooks
. blahh
. im
     . Most correlated bigrams:
. best god
. hey want
. student college
. kick smoke
. likes lots

# 'dropped out of med school':
     . Most correlated unigrams:
. mba
. announced
. lak
. grad
. masters
     . Most correlated bigrams:
. mba student
. working mfa
. studying masters
```

```
. working masters
. grad student

# 'dropped out of ph.d program':
     . Most correlated unigrams:
. shhhh
. everbody
. ahhhh
. nifty
. torchwood
     . Most correlated bigrams:
. essay try
. best guy
. space reserved
. face winning
. looking distraction

# 'dropped out of space camp':
     . Most correlated unigrams:
. accumulating
. envolved
. watz
. tgirl
. semper
     . Most correlated bigrams:
. just holla
. meh don
. ergo sum
. french woman
. single happy

# 'dropped out of two-year college':
     . Most correlated unigrams:
. uninteresting
. statue
. dam
. slathered
. ragin
     . Most correlated bigrams:
. pinnacle good
. wear sunscreen
. ramble ramble
. geez really
. breaking waves

# 'graduated from college/university':
     . Most correlated unigrams:
. migrant
```

```
. ratty
. phd
. scientist
. postdoc
     . Most correlated bigrams:
. details later
. available person
. got phd
. postdoc cal
. finished phd

# 'graduated from high school':
     . Most correlated unigrams:
. bristol
. momentarily
. attorney
. law
. lawyer
     . Most correlated bigrams:
. bar exam
. went law
. lawyer like
. graduated law
. law school

# 'graduated from law school':
     . Most correlated unigrams:
. doctoral
. student
. grad
. ph
. phd
     . Most correlated bigrams:
. phd candidate
. working phd
. grad student
. phd student
. graduate student

# 'graduated from masters program':
     . Most correlated unigrams:
. esthetic
. julian
. bestfriend
. cobb
. gus
     . Most correlated bigrams:
. people vary
```

```
. complete soon
. tryin figure
. yep just
. words lol

# 'graduated from med school':
     . Most correlated unigrams:
. pouty
. excitment
. im
. summerize
. shuffling
     . Most correlated bigrams:
. straight foward
. understandable person
. therapist photographer
. asian spanish
. new construction

# 'graduated from ph.d program':
     . Most correlated unigrams:
. munchies
. navels
. medical
. kt
. med
     . Most correlated bigrams:
. amp blunt
. med student
. finishing med
. medical school
. med school

# 'graduated from space camp':
     . Most correlated unigrams:
. insofar
. typed
. curmudgeonly
. mog
. mornin
     . Most correlated bigrams:
. just embarrassing
. ll days
. movie question
. boston live
. small fit

# 'graduated from two-year college':
```

```
     . Most correlated unigrams:
. swingers
. gdata
. genghis
. khunt
. jaws
     . Most correlated bigrams:
. okcupid sort
. time moment
. simple complicated
. does want
. gregarious social

# 'high school':
     . Most correlated unigrams:
. veterinarian
. physician
. medical
. gam
. residency
     . Most correlated bigrams:
. went med
. graduating medical
. small animal
. residency bay
. family medicine

# 'law school':
     . Most correlated unigrams:
. opend
. swagg
. loveing
. becky
. im
     . Most correlated bigrams:
. working bear
. ask wanna
. im mexican
. just ask
. hi becky

# 'masters program':
     . Most correlated unigrams:
. aye
. moe
. catastrophe
. cougar
. whoop
```

```
    . Most correlated bigrams:
. confused fool
. aint nothin
. sweet geeky
. bitch ll
. words 500

# 'med school':
    . Most correlated unigrams:
. absentminded
. mobbing
. highschool
. 2456
. theodore
    . Most correlated bigrams:
. like ur
. wrestle like
. person stay
. im 18
. eric 18

# 'ph.d program':
    . Most correlated unigrams:
. bolivian
. irreverently
. irrepressibly
. intriguingly
. shields
    . Most correlated bigrams:
. smart impulsive
. secret menu
. talents interests
. words write
. american years

# 'space camp':
    . Most correlated unigrams:
. everthing
. koozie
. wham
. lovelies
. loveliest
    . Most correlated bigrams:
. im hella
. social girl
. francisco small
. 39 like
. rare bird
```

```
# 'two-year college':
     . Most correlated unigrams:
. miserably
. adorably
. evangelizing
. grrrl
. mend
     . Most correlated bigrams:
. cook sing
. music swimming
. sing study
. divorced looking
. just grown

# 'working on college/university':
     . Most correlated unigrams:
. modifications
. eveything
. praising
. kickit
. lmaoo
     . Most correlated bigrams:
. like pamper
. lookin friends
. laid bak
. mellow happy
. simple really

# 'working on high school':
     . Most correlated unigrams:
. ragged
. randall
. rents
. 2l
. law
     . Most correlated bigrams:
. website making
. ecstatic joy
. law school
. year law
. law student

# 'working on law school':
     . Most correlated unigrams:
. staten
. uncivilized
. factories
```

```
. appetites
. aquarian
     . Most correlated bigrams:
. know child
. kind mind
. new apt
. loving liberal
. astrology book

# 'working on masters program':
     . Most correlated unigrams:
. plagiarized
. showgirl
. platonian
. barbaric
. yawp
     . Most correlated bigrams:
. refine disarmingly
. man essence
. platonian ideal
. seeing ok
. winter quiet

# 'working on med school':
     . Most correlated unigrams:
. piquancy
. unadorned
. cashier
. enhancements
. avalanche
     . Most correlated bigrams:
. smart looking
. ways little
. fall super
. realize big
. scientist mind

# 'working on ph.d program':
     . Most correlated unigrams:
. cranberries
. florescent
. quarrel
. investigator
. pumpkin
     . Most correlated bigrams:
. like 11
. looks just
. special treat
```

```
. new tv
. private investigator
```

As expected - The categories with the larger datasets seem to make the most sense.

6.1 Education Training and Testing

```
In [26]: x_train, x_test, y_train, y_test = train_test_split(essay_data['essay0'], essay_data['e
         count_vect = CountVectorizer()
         x_train_counts = count_vect.fit_transform(x_train)
         tfidf_transformer = TfidfTransformer()
         x_train_tfidf = tfidf_transformer.fit_transform(x_train_counts)

         x_test_counts = count_vect.transform(x_test)
         x_test_tfidf = tfidf_transformer.transform(x_test_counts)

         classifier = MultinomialNB().fit(x_train_tfidf, y_train)

In [27]: random_list_for_gender_test = [23985, 32768, 3161, 20330, 25112]

         for i in random_list_for_gender_test:
             print ("Essay Response:")
             print (x_test[i])
             print ("-----------------------------")
             print ("Prediction: "+str(classifier.predict(count_vect.transform([x_test[i]]))[0])
             print ("Actual     : "+y_test[i])
             print ()
```

```
Essay Response:
i'm an astute girl who enjoys warm and quick witted but not empty bantering individuals i happen
-----------------------------
Prediction: graduated from college/university
Actual     : working on ph.d program

Essay Response:
after five years on the east coast i guess it's only appropriate for me to head out to the golde
-----------------------------
Prediction: graduated from college/university
Actual     : graduated from college/university

Essay Response:
my friends call me joe i'm a creative person and i love to make new things out of whatever is ha
-----------------------------
```

```
Prediction: graduated from college/university
Actual    : graduated from college/university

Essay Response:
i'm from the bay area i don't take life too seriously fun love and happiness are things that dri
-----------------------------
Prediction: graduated from college/university
Actual    : working on space camp

Essay Response:
a modern romantic who's sometimes a realist but who's always sarcastic honest and opinionated th
-----------------------------
Prediction: graduated from college/university
Actual    : working on college/university
```

```
In [28]: print ("Accuracy: "+str(classifier.score(x_test_tfidf, y_test)))
```

```
Accuracy: 0.453771886762
```

6.1.1 Results

An accuracy score of only 45%… Very disappointing.

---

7 Final Thoughts

While the accuracy wasn't as high as I wanted, I was able to prove that with some level of accuracy both gender and education level could be predicted based off of essay responses.

7.1 How we could improve our accuracy

I believe that accuracy could be improved if we included trigrams into our data transformation. Additionally we could include other essay responses to the data sample to provide as much data as possible to our training sets. To improve accuracy for education, we could roll up some of the categories with less datasets into a more broad category. I.E. All the dropped out responses for a specific category could be rolled up into a more generic "dropped out" category. We could expand on that to split the education levels into two categories, graduated/dropped out.