Preprocesamiento de datos

El siguiente archivo presenta el procedimiento de recodificación para los archivos `train_cupid.csv` y `test_cupid.csv`.

Descripción de variables

- `body_type`: rather not say, thin, overweight, skinny, average, fit, athletic, jacked, a little extra, curvy, full figured, used up.
- `diet`: mostly/strictly; anything, vegetarian, vegan, kosher, halal, other.
- `drinks`: very often, often, socially, rarely, desperately, not at all.
- `drugs`: never, sometimes, often.
- `education`: graduated from, working on, dropped out of; high school, two-year college, university, masters program, law school, med school, Ph.D program, space camp.
- `ethnicity`: Asian, middle eastern, black, native American, indian, pacific islander, Hispanic/latin, white, other.
- `height`: en pulgadas.
- **income**: (US $, -1 significa que prefiere no reportarlo) -1, 20000, 30000, 40000, 50000, 60000 70000, 80000, 100000, 150000, 250000, 500000, 1000000.
- `job`: student, art/music/writing, banking/finance, administration, technology, construction, education, entertainment/media, management, hospitality, law, medicine, military, politics/government, sales/marketing, science/engineering, transportation, unemployed, other, rather not say, retire.
- `offspring`: has a kid, has kids, doesnt have a kid, doesn't want kids; ,and/,but might want them, wants them, doesnt want any, doesnt want more.

- `orientation`: straight, gay, bisexual.
- `pets`: has dogs, likes dogs, dislikes dogs; and has cats, likes cats, dislikes cats.
- `religion`: agnosticism, atheism, Christianity, Judaism, Catholicism, Islam, Hinduism, Buddhism, Other; and very serious about it, and somewhat serious about it, but not too serious about it, and laughing about it.
- `sex`: m, f
- `sign`: aquarius, pices, aries, Taurus, Gemini, cancer, leo, virgo, libra, scorpio, saggitarius, Capricorn; but it doesn't matter, and it matters a lot, and it's fun to think about.
- `smokes`: yes, sometimes, when drinking, trying to quit, no.
- `speaks`: English (fluently, okay, poorly). Afrikaans, Albanian, Arabic, Armenian, Basque, Belarusan, Bengali, Breton, Bulgarian, Catalan, Cebuano, Chechen, Chinese, C++, Croatian, Czech, Danish, Dutch, Esperanto, Estonian, Farsi, Finnish, French, Frisian, Georgian, German, Greek, Gujarati, Ancient Greek, Hawaiian, Hebrew, Hindi, Hungarian, Icelandic, Ilongo, Indonesian, Irish, Italian, Japanese, Khmer, Korean, Latin, Latvian, LISP, Lithuanian, Malay, Maori, Mongolian, Norwegian, Occitan, Other, Persian, Polish, Portuguese, Romanian, Rotuman, Russian, Sanskrit, Sardinian, Serbian, Sign Language, Slovak, Slovenian, Spanish, Swahili, Swedish, Tagalog, Tamil, Thai, Tibetan, Turkish, Ukranian, Urdu, Vietnamese, Welsh, Yiddish (fluently, okay, poorly).
- `essay0`: My self summary.
- `essay1`: What I'm doing with my life.
- `essay2`: I'm really good at.
- `essay3`: The first thing people usually notice about me.

- `essay4`: Favorite books, movies, show, music, and food.
- `essay5`:The six things I could never do without.
- `essay6`: I spend a lot of time thinking about.
- `essay7`: On a typical Friday night I am.
- `essay8`: The most private thing I am willing to admit.
- `essay9`: You should message me if...
- `status`: single, seeing someone, married, in an open relationship. De esta variable surgen sus vectores objetivos.

Elimine los siguientes atributos

Se eliminan los ensayos y aquellas columnas que presenten un **.35** o más de valores nulos, así como `last_online`.

Recodificación Status

Se dejan como categoría de referencia a `married` y `unknown`.

Cada una de las categorías `single`, `seeing someone` y `available` son consideradas como vectores objetivos.

Recodificación Body Type

Se generan dos variables binarias. La plantilla de recodificación es:

Categoría Original	Recodificación
'athletic'	Referencia
'fit'	Referencia
'jacked'	Referencia
'average'	'regular'
'skinny'	'regular'
'thin'	'regular'
'used up'	'regular'
'curvy'	'regular'
'full figured'	'overweight'
'overweight'	'overweight'

Recodificación Educación

Se generan dos variables binarias. La plantilla de recodificación es:

Categoría Original	Recodificación
'graduated from high school'	'high_school'
'dropped out of high school'	'high_school'
'high school'	'high_school'
'working on high school'	'high_school'
'graduated from college/university'	'undergrad_university'
'dropped out of college/university'	'undergrad_university'
'working on college/university'	'undergrad_university'
'graduated from two-year college'	'undergrad_university'
'working on two-year college'	'undergrad_university'
'dropped out of two-year college'	'undergrad_university'
'two-year college'	'undergrad_university'
'college/university'	'undergrad_university'
'working on masters program'	Referencia
'dropped out of masters program'	Referencia
'graduated from masters program'	Referencia
'masters program'	Referencia
'working on ph.d program'	Referencia
'dropped out of ph.d program'	Referencia
'graduated from ph.d program'	Referencia
'ph.d program'	Referencia
'working on law school'	Referencia
'dropped out of law school'	Referencia

'graduated from law school'	Referencia
'law school'	Referencia
'working on med school'	Referencia
'dropped out of med school'	Referencia
'graduated from med school'	Referencia
'med school'	Referencia
'working on space camp'	Referencia
'dropped out of space camp'	Referencia
'graduated from space camp'	Referencia
'space camp'	Referencia

Recodificación job

Se genera una variable binaria llamada `employed`. La plantilla de recodificación es:

Categoría Original	Recodificación
artistic / musical / writer	1
banking / financial / real estate	1
clerical / administrative	1
computer / hardware / software	1
construction / craftsmanship	1
education / academia	1
entertainment / media	1
executive / management	1
hospitality / travel	1
law / legal services	1
medicine / health	1
military	1
other	1
political / government	1
rather not say	1
retired	Referencia
sales / marketing / biz dev	1
science / tech / engineering	1
student	Referencia
transportation	1
unemployed	Referencia

Recodificación Drugs

Se generan dos variables binarias.

De las categorías existentes `'never', 'sometimes', 'often'` deben dejar a `'never'` como categorías de referencia.

Recodificación sex

Se genera una variable binaria.

De las categorías existentes `'m'` y `'f'`, deben dejar `'f'` como categoría de referencia.

Recodificación smokes

Se generan cuatro variables binarias.

De las categorías existentes `'yes', 'sometimes', 'when drinking', 'trying to quit', 'no'`, deben dejar `'no'` como categoría de referencia.

Recodificación orientation

Se generan dos variables binarias.

De las categorías existentes `gay', 'bisexual', 'straight',`, deben dejar `'bisexual'` como categoría de referencia.

Recodificación drinks

Se generan cinco variables binarias.

De las categorías existentes `'socially', 'rarely', 'often', 'not at all', 'very often', 'desperately'`, deben dejar `'desperately'` categoría de referencia.

Recodificación pets

Se generan dos variables binarias para Dog y Cat.

Categoría Original	Recodificación
'likes dogs and likes cats'	Referencia
'likes dogs'	Dog
'likes dogs and has cats'	Cat
'has dogs'	Dog
'has dogs and likes cats'	Dog
'likes dogs and dislikes cats'	Dog
'has dogs and has cats'	Referencia
'has cats'	Cat
'likes cats'	Cat
'has dogs and dislikes cats'	Dog
'dislikes dogs and likes cats'	Cat
'dislikes dogs and dislikes cats'	Referencia
'dislikes cats'	Referencia
'dislikes dogs and has cats'	Cat
'dislikes dogs'	Referencia

Recodificación Income

Se generan tres variables binarias. Income debe recodificarse en las siguientes categorías:

- non_reported ⇝ Referencia.
- income_between_25_50 ⇝ Binaria entre el 25% y el 49% de los percentiles. 0 de lo contrario.
- income_between_50_75 ⇝ Binaria entre el 50% y el 74% de los percentiles. 0 de lo contrario.
- income_over_75 ⇝ Binaria superior o igual al 75%. 0 de lo contrario.

Recodificación sign

Se generan once variables binarias. Recodifique en función de los siguientes grupos `'virgo', 'taurus', 'scorpio', 'pisces', 'libra', 'leo', 'gemini', 'capricorn', 'aries', 'aquarius','cancer','sagittarius'`, deben dejar `'capricorn'` como categoría de referencia.

Pueden buscar simplemente por la existencia del string en el registro con `np.where`

Recodificación speaks

Se generan cuatro variables binarias. Recodifique en función de los siguientes grupos `'spanish', 'chinese', 'french', 'german'`. Todas las demás opciones existentes deben ser consideradas como categorías de referencia.

Pueden buscar simplemente por la existencia del string en el registro con `np.where`

Recodificación religion

Se generan ocho variables binarias. Recodifique en función de los siguientes grupos: `'agnosticism','atheism','christianity','other','catholicism','buddhism', 'judaism','hinduism','islam'`, deben dejar `'christianity'` como categoría de referencia.

Se busca la existencia del string en el registro con `np.where`

Recodificación ethnicity

Se generan ocho variables binarias. Recodifique en función de los siguientes grupos `'white'`,`'asian'`,`'hispanic/latin'`,`'black'`,`'other'`,`'indian'`,`'pacific islander'`,`'native american'`,`'middle eastern'`, deben dejar `'white'` como categoría de referencia.

Se busca existencia del string en el registro con `np.where`

Recodificación location

Se generan 40 variables binarias. Recodifique en función de los siguientes grupos: `'california', ' colorado', ' new york', ' oregon', ' arizona', ' hawaii', ' montana', ' wisconsin', ' virginia', ' spain', ' nevada', ' illinois', ' vietnam', ' ireland', ' louisiana', ' michigan', ' texas', ' united kingdom', ' massachusetts', ' north carolina', ' idaho', ' mississippi', ' new jersey', ' florida', ' minnesota', ' georgia', ' utah', ' washington', ' west virginia', ' connecticut', ' tennessee', ' rhode island', ' district of columbia', ' canada', ' missouri', ' germany', ' pennsylvania', ' netherlands', ' switzerland', ' mexico', ' ohio'`. Deben dejar `'california'` como categoría de referencia.

Se busca la existencia del string en el registro con `np.where`.

Este procedimiento debe realizarse para train y test sets:

```
import pandas as pd
import numpy as np
df = pd.read_csv('profiles.csv')
pd.options.display.max_columns = None # Para que nos muestre las tablas completas, y no oculte columnas.
```

Revisamos cantidad de columnas con nulas superior al .35

```
remove_oversized_na = []
for colname, serie in df.iteritems():
   # Revisamos cuántas entradas de cada columna son nulas
   tmp_series = serie.isnull().value_counts('%')
   if tmp_series.get(True) and tmp_series.get(True) > .35:
       remove_oversized_na.append(colname)
```

Eliminamos atributos que no ocuparemos

```
remove_oversized_na
```

```
['diet', 'offspring']
```

```
df = df.drop(['essay0', 'essay1', 'essay2',
             'essay3', 'essay4', 'essay5',
             'essay6', 'essay7', 'essay8',
             'essay9'] + remove_oversized_na,axis=1) # Eliminamos los
ensayos y las columnas que tienen muchas entradas nulas
```

Evaluamos cuántas categorías únicas existen en cada atributo

Algunas de las categorías como `location`, `speaks`, `sign`, `religion` y `ethnicity` se recodificaron en menos categorías para capturar simplemente la mención del atributo específico. Más adelante se detalla el código de recodificación.

Dado la alta cantidad de categorías en `last_online`, ésta se eliminará.

```
# revisemos cuántas categorías únicas existen
fetch_uniqueness = {i: len(df[i].unique()) for i in df.columns}
```

```
fetch_uniqueness
```

```
{'age': 54,
'body_type': 13,
'drinks': 7,
'drugs': 4,
'education': 33,
'ethnicity': 218,
'height': 61,
'income': 13,
'job': 22,
'last_online': 30123,
'location': 199,
'orientation': 3,
'pets': 16,
'religion': 46,
'sex': 2,
'sign': 49,
'smokes': 6,
'speaks': 7648,
'status': 5}
```

Recod Body Type

```
# Reducimos la cantidad de tipos de cuerpo
body_map = {'athletic': 'athletic',
           'fit': 'athletic',
           'jacked': 'athletic',
           'average': 'regular',
           'skinny': 'regular',
           'thin': 'regular',
           'used up': 'regular',
           'curvy':'regular',
           'full figured': 'overweight',
           'overweight': 'overweight',
           'rathernot': 'rather not say'}

df['body_recod'] = df['body_type'].map(body_map)
```

Recod Education

```
# Reducimos la cantidad de niveles de educación
educ_map = {'graduated from high school': 'high_school',
          'dropped out of high school': 'high_school',
          'high school': 'high_school',
          'working on high school': 'high_school',
          'graduated from college/university': 'undergrad_university',
          'dropped out of college/university': 'undergrad_university',
          'working on college/university': 'undergrad_university',
          'graduated from two-year college': 'undergrad_university',
          'working on two-year college': 'undergrad_university',
          'dropped out of two-year college': 'undergrad_university',
          'two-year college': 'undergrad_university',
          'college/university': 'undergrad_university',
          'working on masters program': 'grad_school',
          'dropped out of masters program': 'grad_school',
          'graduated from masters program': 'grad_school',
          'masters program': 'grad_school',
          'working on ph.d program': 'grad_school',
          'dropped out of ph.d program': 'grad_school',
          'graduated from ph.d program': 'grad_school',
          'ph.d program': 'grad_school',
          'working on law school': 'grad_school',
          'dropped out of law school': 'grad_school',
          'graduated from law school': 'grad_school',
          'law school': 'grad_school',
           'working on med school': 'grad_school',
          'dropped out of med school': 'grad_school',
          'graduated from med school': 'grad_school',
          'med school': 'grad_school',
           'working on space camp': 'grad_school',
          'dropped out of space camp': 'grad_school',
          'graduated from space camp': 'grad_school',
          'space camp': 'grad_school',}

df['educ_recod'] = df['education'].map(educ_map)
```

Recod Sign

```
# Cambiamos la columna de signo zodiaco por varias columnas booleanas,
una para cada signo.
for i in ['virgo', 'taurus', 'scorpio', 'pisces', 'libra',
          'leo', 'gemini','aries', 'aquarius','cancer','sagittarius']:
   df[i] = np.where(df['sign'] == i, 1, 0)
```

```
def mutate_str_to_list(a):
   if type(a) != float and a is not None:
       return (np.array(a.split(',')))[0]

df['ethnicity'] = df['ethnicity'].apply(mutate_str_to_list)
# Cambiamos la columna de etnia por varias columnas booleanas, una para
cada etnia.
for i in ['asian','hispanic / latin','black','other','indian','pacific
islander','native american', 'middle eastern',]:
   df[i] = np.where(df['ethnicity'] == i, 1, 0)
```

Recod Locations

```
values_locations=[]
def mutate_str_to_list_last_str(a):
   if type(a) != float and a is not None:
       value = (np.array(a.split(',')))[-1]
       if value not in values_locations:
           values_locations.append(value)
       return value

df['location'] = df['location'].apply(mutate_str_to_list_last_str)
values_locations.remove(' california')

# Cambiamos la columna de localidad por varias columnas booleanas, una
para cada localidad.
for i in values_locations:
   df[i] = np.where(df['location'] == i, 1, 0)
```

Recod Religion

```
values_religions=[]
def mutate_str_to_list_religion(a):
   if type(a) != float and a is not None:
       value = (np.array(a.split(' ')))[0]
       if value not in values_religions:
           values_religions.append(value)
       return value

df['religion'] = df['religion'].apply(mutate_str_to_list_religion)
values_religions.remove('christianity')

# Cambiamos la columna de religion por varias columnas booleanas, una
para cada religion.
for i in values_religions:
   df[i] = np.where(df['religion'] == i, 1, 0)
```

Recod Pets

```
doggos_maps = {'likes dogs and likes cats': 0,
            'likes dogs': 1,
            'likes dogs and has cats': 0,
            'has dogs': 1,
            'has dogs and likes cats': 1,
            'likes dogs and dislikes cats': 1,
            'has dogs and has cats': 0,
            'has cats': 0,
            'likes cats': 0,
            'has dogs and dislikes cats': 1,
            'dislikes dogs and likes cats': 0,
            'dislikes dogs and dislikes cats': 0,
            'dislikes cats': 0,
            'dislikes dogs and has cats': 0,
            'dislikes dogs': 0}

cattos_maps = {'likes dogs and likes cats': 0,
            'likes dogs': 0,
            'likes dogs and has cats': 1,
            'has dogs': 0,
            'has dogs and likes cats': 0,
            'likes dogs and dislikes cats': 0,
            'has dogs and has cats': 0,
            'has cats': 1,
            'likes cats': 1,
            'has dogs and dislikes cats': 0,
            'dislikes dogs and likes cats': 1,
            'dislikes dogs and dislikes cats': 0,
            'dislikes cats': 0,
            'dislikes dogs and has cats': 1,
            'dislikes dogs': 0}

df['pro_dogs'] = df['pets'].map(doggos_maps)
df['pro_cats'] = df['pets'].map(cattos_maps)
```

Recod speaks

```
df['spanish'] = df['speaks'].str.contains('spanish')
df['chinese'] = df['speaks'].str.contains('chinese')
df['french'] = df['speaks'].str.contains('french')
df['german'] = df['speaks'].str.contains('german')

df['spanish'] = pd.to_numeric(df['spanish'], errors='coerce')
df['chinese'] = pd.to_numeric(df['chinese'], errors='coerce')
df['french'] = pd.to_numeric(df['french'], errors='coerce')
df['german'] = pd.to_numeric(df['german'], errors='coerce')

df['spanish'] = df['spanish'].fillna(0)
df['chinese'] = df['chinese'].fillna(0)
df['french'] = df['french'].fillna(0)
df['german'] = df['german'].fillna(0)

df['spanish'] = df['spanish'].astype(int)
df['chinese'] = df['chinese'].astype(int)
df['french'] = df['french'].astype(int)
df['german'] = df['german'].astype(int)
```

Recod Status

```
df['single'] = np.where(df['status'] == 'single', 1, 0)
df['seeing_someone'] = np.where(df['status'] == 'seeing someone', 1, 0)
df['available'] = np.where(df['status'] == 'seeing someone', 1, 0)
```

```
# Cambiamos el resto de las columnas no-númericas por varias columnas
booleanas.
drugs_dummie = pd.get_dummies(df['drugs'], prefix="drugs",
drop_first=True)
drinks_dummie = pd.get_dummies(df['drinks'], prefix="drinks",
drop_first=True)
orientation_dummie = pd.get_dummies(df['orientation'],
prefix="orientation", drop_first=True)
sex_dummie = pd.get_dummies(df['sex'], prefix="sex", drop_first=True)
smokes_dummie = pd.get_dummies(df['smokes'], prefix="smokes",
drop_first=True)
body_type_dummie = pd.get_dummies(df['body_recod'], prefix="body_type",
drop_first=True)
education_dummie = pd.get_dummies(df['educ_recod'], prefix="education",
drop_first=True)
```

Recod Jobs

```
temp = np.where(np.logical_or(df['job'] == 'retired', df['job'] ==
'unemployed'), 1, 0)
df['employed'] = np.where(np.logical_or(df['job'] == 'student', temp),
0, 1)
```

```
df['income'].describe()
```

```
count     59946.000000
mean       20033.222534
std       97346.192104
min           -1.000000
25%           -1.000000
50%           -1.000000
75%           -1.000000
max     1000000.000000
Name: income, dtype: float64
```

```
# vamos a generar un binario que identifique a aquellos que no
reportaron ingresos
df['non_reported_income'] = np.where(df['income'] == -1, 1, 0)

df['income_between_25_50'] = np.where(np.logical_and(df['income'] >=
20000.0,
                                                     df['income'] <
50000.0), 1, 0)

df['income_between_50_75'] = np.where(np.logical_and(df['income'] >=
50000.0,
                                                     df['income'] <
100000.0), 1, 0)
df['income_over_75'] = np.where(df['income'] >= 100000.0, 1, 0)
```

```
df = df.drop(['body_type', 'drinks', 'drugs', 'education',
             'orientation', 'pets','sex','smokes','status',
             'body_recod', 'educ_recod', 'religion',
             'ethnicity','location', 'speaks', 'sign','job',
             'income', 'non_reported_income', 'other',
'last_online'],axis=1)
```

```
df_refac=pd.concat([df, drugs_dummie, drinks_dummie, orientation_dummie,
                  sex_dummie, smokes_dummie, body_type_dummie,
                  education_dummie], axis=1)
```

Just-In-Time csv function

```
from numba import jit

@jit
def to_csv(df, name):
   return df.to_csv(name)

to_csv(df_refac, "df.csv")
```

Random subsample

```
import numpy as np
import pandas as pd

#reassure sampling replicability
np.random.seed(11238)

new_df = pd.read_csv('df.csv')
new_df['split'] = np.random.randn(new_df.shape[0], 1)

#exclude remaining unused variables
new_df = new_df.drop([], axis=1)
# create random selector
msk = np.random.rand(len(new_df)) <= 0.5

#generate random selection
train = new_df[msk]
train = train.drop(['Unnamed: 0', 'split'], axis=1)
test = new_df[~msk]
test = test.drop(['Unnamed: 0', 'split'], axis=1)

# save dataframes
to_csv(train, "train_cupid.csv")
to_csv(test, "test_cupid.csv")
```

```
df = pd.read_csv('train_cupid.csv').drop(columns='Unnamed: 0')
```

```
df.to_csv('train_cupid.csv')
```

```
df
```

Profiling individuals

```
# profile individuals
individual_characteristics = train.sample().T
individual_characteristics[individual_characteristics[individual_charact
eristics.columns[0]] != 0]
```

	7878
Unnamed: 0	7878.0
age	33.0
height	70.
spanish	1.0
single	1.0
employed	1.0
income_between_50_75	1.0
drinks_socially	1.0
orientation_straight	1.0

```
train.columns
```

```
Index(['Unnamed: 0', 'age', 'height', 'virgo', 'taurus', 'scorpio',
'pisces',
       'libra', 'leo', 'gemini', 'aries', 'aquarius', 'cancer',
'sagittarius',
       'asian', 'hispanic / latin', 'black', 'indian', 'pacific
islander',
       'native american', 'middle eastern', ' colorado', ' new york',
       ' oregon', ' arizona', ' hawaii', ' montana', ' wisconsin', '
virginia',
       ' spain', ' nevada', ' illinois', ' vietnam', ' ireland', '
louisiana',
       ' michigan', ' texas', ' united kingdom', ' massachusetts',
       ' north carolina', ' idaho', ' mississippi', ' new jersey', '
florida',
       ' minnesota', ' georgia', ' utah', ' washington', ' west
virginia',
       ' connecticut', ' tennessee', ' rhode island', ' district of
columbia',
       ' canada', ' missouri', ' germany', ' pennsylvania', '
netherlands',
       ' switzerland', ' mexico', ' ohio', 'agnosticism', 'atheism',
       'catholicism', 'buddhism', 'judaism', 'hinduism', 'islam',
'pro_dogs',
       'pro_cats', 'spanish', 'chinese', 'french', 'german', 'single',
       'seeing_someone', 'available', 'employed', 'income_between_25_50',
       'income_between_50_75', 'income_over_75', 'drugs_often',
       'drugs_sometimes', 'drinks_not at all', 'drinks_often',
'drinks_rarely',
       'drinks_socially', 'drinks_very often', 'orientation_gay',
       'orientation_straight', 'sex_m', 'smokes_sometimes',
       'smokes_trying to quit', 'smokes_when drinking', 'smokes_yes',
       'body_type_overweight', 'body_type_regular',
'education_high_school',
       'education_undergrad_university'],
     dtype='object')
```

Upload dataframes to psql

```
aaa=pd.read_csv('train_cupid.csv', ).drop(columns=['Unnamed: 0',
'Unnamed: 0.1'])
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call
last)

<ipython-input-42-f554eaf147cf> in <module>
----> 1 aaa=pd.read_csv('train_cupid.csv', ).drop(columns=['Unnamed: 0',
'Unnamed: 0.1'])
      2 aaa.isna()
```

```
~/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   3938                                                  index=index,
columns=columns,
   3939                                                  level=level,
inplace=inplace,
-> 3940                                                  errors=errors)
   3941
   3942     @rewrite_axis_style_signature('mapper', [('copy', True),
```

```
~/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   3778         for axis, labels in axes.items():
   3779             if labels is not None:
-> 3780                 obj = obj._drop_axis(labels, axis, level=level,
errors=errors)
   3781
   3782         if inplace:
```

```
~/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py in
_drop_axis(self, labels, axis, level, errors)
   3810                 new_axis = axis.drop(labels, level=level,
errors=errors)
   3811             else:
-> 3812                 new_axis = axis.drop(labels, errors=errors)
   3813             result = self.reindex(**{axis_name: new_axis})
   3814
```

```
~/anaconda3/lib/python3.6/site-packages/pandas/core/indexes/base.py in
drop(self, labels, errors)
   4962             if errors != 'ignore':
   4963                 raise KeyError(
-> 4964                     '{} not found in axis'.format(labels[mask]))
   4965             indexer = indexer[~mask]
   4966         return self.delete(indexer)
```

```
KeyError: "['Unnamed: 0' 'Unnamed: 0.1'] not found in axis"
```

```
aaa.shape
```

```
(29939, 98)
```

```
aaa.dropna().shape
```

```
(20081, 98)
```

```
aaa.dropna().to_csv('train_cupid.csv', index=False)
```

```
test.dropna().to_csv('test_cupid.csv', index=False)
```

```
a
```

```
seeing_someone = train.pop('status_seeing someone')
single
```