**DATA ANALYSIS FOR THE PROFILES DATASET**

Description of data:

The Data selected is “Profiles” data . There are 31 rows and 2 of them are interval data , where as the rest of them are categorical data. The data describes the profiles of people like age, gender, status of relationship, orientation of gender, various intrests, etc.

STEP 1: Loading libraries and loading the csv file into a variable dataset.

# reading the data into variable 'dataset'

dataset=pd.read\_csv('C:/Users/Indir/OneDrive/Desktop/profiles.csv')

STEP 2 : Checking for missing data

print(dataset.isnull().sum())

age 3

body\_type 5298

diet 24398

drinks 2988

drugs 14083

education 6631

essay0 5491

essay1 7576

essay2 9641

essay3 11480

essay4 10541

essay5 10856

essay6 13777

essay7 12456

essay8 19231

essay9 12609

ethnicity 5686

height 9

income 6

job 8204

last\_online 6

location 7

offspring 35566

orientation 7

pets 19927

religion 20233

sex 7

sign 11063

smokes 5519

speaks 57

status 7

OBSERVATION : It is seen that the data set has some missing values. So before proceeding with further steps, the missing data has to be dealt with.

STEP 2 : Calculating the percentage of missing values in each row.

miss\_val\*100/len(dataset)

age 0.005004

body\_type 8.837512

diet 40.697927

drinks 4.984237

drugs 23.491635

education 11.061069

essay0 9.159452

essay1 12.637408

essay2 16.082003

essay3 19.149611

essay4 17.583279

essay5 18.108726

essay6 22.981201

essay7 20.777661

essay8 32.078934

essay9 21.032878

ethnicity 9.484729

height 0.015013

income 0.010009

job 13.684966

last\_online 0.010009

location 0.011677

offspring 59.327095

orientation 0.011677

pets 33.239921

religion 33.750354

sex 0.011677

sign 18.454019

smokes 9.206159

speaks 0.095081

status 0.011677

OBSERVATION : There are rows which has high percentage of missing values like offspring (59%) and there are few clean rows like speaks, status,etc.

For the sake of simplification of the dataset, lets drop all those rows which have missing values more than 15%.

Usually in case of qualitative data the missing values are dealt with by imputing with mean/median/mode , etc. But in case of categorical variables, there are ways like either dropping rows with excess of missing values, using KNN means algorithm to fill the missing values or filling the cells with the nearest value algorithm. The first approach of dropping the rows is chosen for dealing with categorical data.

Percentage of missing data from the dataset :

row\_miss = len(dataset.axes[0])

col\_miss = len(dataset.axes[1])

tot\_cell = row\_miss \* col\_miss

per\_miss = (tot\_miss\*100)/tot\_cell

per\_miss

The % of missing data is around 15%. Since 15% of 15 lakhs is considerably less, we can consider doing analysis on this data.

STEP 3 : Dealing with the missing values

1. DROPPING THE ROWS :

dataset1 = dataset.drop(["essay0","essay1","essay2","essay3","essay4","essay5","essay6","essay7","essay8","essay9"],axis=1)

all those rows which have more than 15% of missing values are dropped from the data

"essay0", "essay1" are dropped even though less than 15% becuase without the other "essay" variables, their impact on the model cnt be determined

dataset2=dataset1.drop(["diet","drugs","location","offspring","religion","speaks","pets","job","sign"], axis=1)

1. Replacing the missing data in gender as ‘no gender’ :

replacing the missing values in the sex as no gender insted of imputing them with the nearest values (KNN algorithm)

since only 1% if the data is missing in that row, those respondents might belong to another category for which the provision isnt given in the quessionare

dataset2["sex"].fillna("No Gender", inplace = True)

dataset2[dataset2['sex']=='No Gender']

1. Imputing the missing data of row AGE with mean.

it is detected(when age is executed for missing data error stated it )that the cell 18843 has a string in the age coloumn.

so that cell is dropped from the dataset

age body\_type drinks education ethnicity height income last\_online orientation sex smokes status

18843 /interests?i=nightperson"">nightperson</a>." <a class="ilink" href="/interests?i=humor%21">... me,<br />\n50 years ahead of my time. &lt;--pe... it's any way you like it... m pisces no single NaN No Gender NaN NaN

dataset2['age']=dataset2['age'].astype(float)

dataset2.age.fillna(dataset2.age.mean())

Age column is imputed with mean values

0 22.0

1 35.0

2 38.0

3 23.0

4 29.0

...

59944 59.0

59945 24.0

59946 42.0

59947 27.0

59948 39.0

Name: age, Length: 59948, dtype: float64

1. Imputing Income row with mean

It is to be noted that the Income row has NaNs(missing cells) as well as cells with -1s, which are outliers. The percentage of -1 in the income row is calculated as follows

# total % of '-1's appearing in the income coloumn

income\_permiss = ff\*100/len(dataset2.income)

income\_permiss

output : 36.70119601661412

So first we find the mean of income, replace -1,s and NaNs with the calculated mean.

dataset2['income']=dataset2['income'].astype(float)

avg\_income= dataset2['income'].mean()

avg\_income

dataset['income']=dataset2.income.replace('-1',20033.89)

dataset2.income

1. Since in the rest of the rows the missing value percentages are very less, the missing value coloumns are dropped.

we are not using how='all' inside dropna() for simplification of the process

dataset2 = dataset2[dataset2['body\_type'].notna()]

the above code is repeated for the rest of the variables. Now we get a clean data. The output is as follows

dataset2.isnull().sum()

age 0

body\_type 0

drinks 0

education 0

ethnicity 0

height 0

income 0

last\_online 0

orientation 0

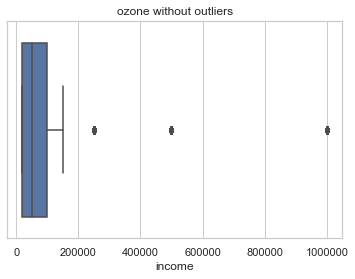
sex 0

smokes 0

status 0

1. It has to be noted that even though the -1s from income are imputed with mean, but still thee are 3 outliers . It is represented in a graph. Since the outliers are only 3 , they can be ignored and be carried on with further analysis.

Graph1 : Box-plot of outliers on income after imputing.



STEP 4 : Descriptive statistics for the two metric variables.

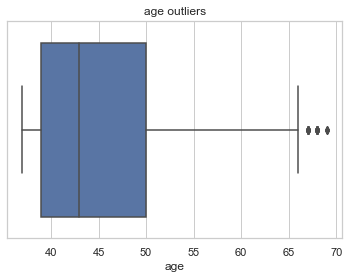
Descriptive statistics cnt be generated for categorical variables. Even if the variables can be encoded and calculated, it wouldn’t make much sense as they will be encoded as ordinal or nominal data..

| **age** |  |
| --- | --- |
| count | 41846.000000 |
| mean | 32.690580 |
| std | 9.698568 |
| min | 18.000000 |
| 25% | 26.000000 |
| 50% | 30.000000 |
| 75% | 37.000000 |
| max | 69.000000 |

From the above statistics it can be seen even the age has outliers . Since the minimum value 18 is less than the first quartile and maximum value is greater than 3rd quartile, all the values greater than 26 and less than 37 are considered outliers .

The outliers are not deleted as 3 outloers will have less or no impact in model development..

GRAPH 2: box plot for outliers of age



STEP 5 : DATA Exploration.

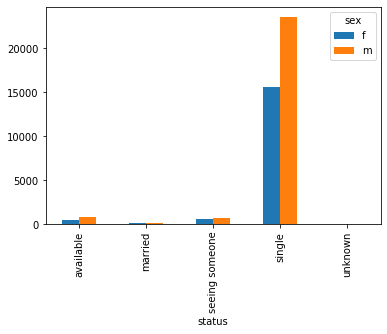
1. Gender – status data spread

to view the gender-status data

here "no gender" values has not been displayed as they are just 3 values and might have been dropped in relation to other missing data

d= dataset2.groupby("status").sex.value\_counts()

GRAPH 3 : bar plot for gender vs status



OBSERVATION: most of the respondents seems to be single and most of them are male. So a business opportunity of match making application can be assumed.

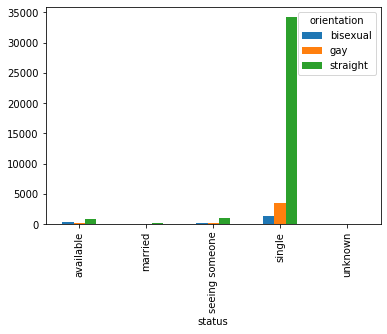
1. To view Orientation- status data

e= dataset2.groupby("status").orientation.value\_counts()

#graph for oreintation vs status

e.unstack().plot(kind='bar')

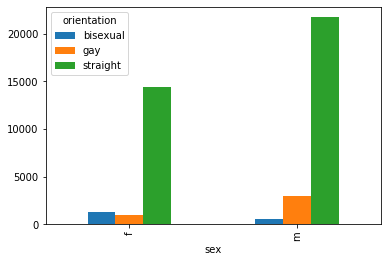
GRAPH 4: Bar graph for Orientation vs status



OBSERVATION : There is not much diversity in the Orientation as most of the single people seem to be straight. But Gay and bisexual persons form a significant ‘single’ status. So the business opportunity should be to including the gay and bisexual people also in the match making application.

1. Orientation vs Sex

Graph 7:



OBSERVATION : Female seem to be more of bisexual than men in this sample. Even though the total of men are more than women in this sample, but the bisexual female are more than men. Which means, the match making application should give more importance to bisexual female in their marketing and advertising practises. The percentage of gay female in total women population and percentage of gay men in total men population is same (i.e 1% approximately)

1. gender vs agr group vs smoking status

sex f m

agegroup orientation

18-30 bisexual 824 310

gay 556 1507

straight 6645 11720

31-40 bisexual 311 124

gay 217 783

straight 4498 6433

41-50 bisexual 99 53

gay 131 457

straight 1879 2327

51-60 bisexual 24 11

gay 43 139

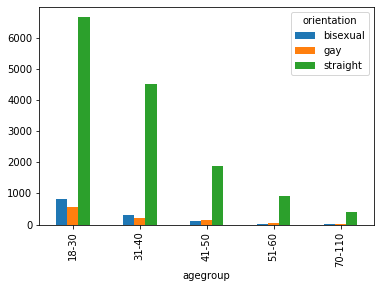
straight 910 868

70-110 bisexual 6 2

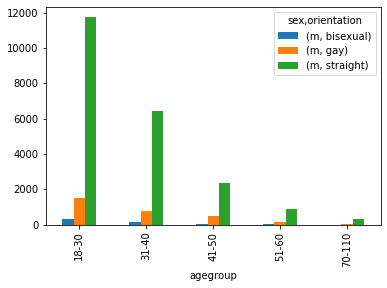
gay 18 30

straight 398 324

GRAPH 8 :orientation vs age group for FEMALE



GRAPH 9 : orientation vs age group for MEN



OBSERVATION : weather in mean or women and wheat ever is the orientation , 18-30 age group followed by 31-40 should be target by the match making application company, for higher reach.

STEP 6 : MODEL BUILDING .

Since most of the data is categorical Random forest classification/prediction is chosen.

Load the randomforestclassifier.

[feature selection isn’t done for categorical data, as the skills right now are limited to qualitative data]

AIM : to predict the status depending on available features.

1. Divide the data into test and training set of 3:7

Hot Encode the categorical variables so as to convert into metric data . Later it can be reverse encoded.

data['body\_type'] = number.fit\_transform(data.body\_type)

the above code for all other categorical variables.

1. dividing the training data into train and validate data in 3:1 ration respectively

train['is\_train'] = np.random.uniform(0, 1, len(train)) <= .75

train, validate = train[train['is\_train']==True], train[train['is\_train']==False]

1. selecting the input features

features=['age','body\_type','drinks','education','ethnicity','orientation','sex','smokes','income']

1. # selecting the target feature i.e Status.

x\_train = train[list(features)].values

x\_validate=validate[list(features)].values

y\_train = train['status'].values

y\_validate = validate['status'].values

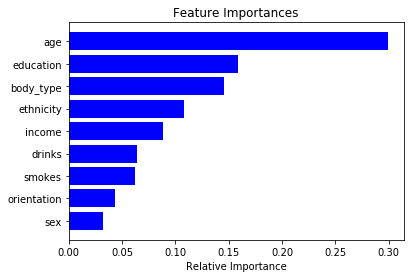
x\_test=test[list(features)].values

1. Building the model

rf = RandomForestClassifier(n\_estimators=1000)

rf.fit(x\_train, y\_train)

1. Generating feature importance graph



Relative importance of each feature wrt the status.

1. Cross validating, reverse encoding of the data and generating output.

[PS: got syntax error for reverse encoding. So output is obtained in metric form].

1. Output in csv file : Randomforsestsolution.