IUPUI
SCHOOL OF INFORMATICS AND COMPUTING

Date: December 3, 2022
Course: Intro to Informatics
Group 12 Project Presentation

A Comparative Analysis and Predicative Model for Reporting Depression in U.S. Adults

Kiran Mai Jaiswal Charpuria, Kruthika Gaddam, Mohith Surya Kiran Kasula, Bala Samantula, Sri Harsha Sudalagunta, April Taylor, Alan Varkey

Indiana University-Purdue University, Indianapolis, USA

InformaticsIUPUI @IUPUInformatics soicindy

Background

standardized screening increases diagnosis in disparaged groups[5]

1. World Health Organization. (2022
2. American Academy of Pediatrics, American Academy of Child, and Adolescent Psychiatry, and Children's Hospital Association,. (2021, 10/19/2021).
3. Hansson, L., Jormfeldt, H., Svedberg, P., & Svensson, B. (2013).
4. Knaak, S., Mantler, E., & Szeto, A. (2017).
5. Garcia, M. E., Hinton, L., Neuhaus, J., Feldman, M., Livaudais-Toman, J., & Karliner, L. S. (2022).

Project Scope

Aims

To evaluate any significant mean difference between self-reported and provider-reported cases of depression in adults in the U.S.

To identify and correlate the main factors impacting the reporting of depression by self-reporters and providers in adults in the U.S. and identify the groups that report depression the least.

To develop a multi-variant model that predicts the diagnosis of depression cases from the features of age, gender, income, co-morbidity, medical insurance, education level, and race.

Purpose

1. Investigate discrepancies between:
   - Self-reporting of depression and
   - Healthcare professional reporting of depression.

2. Predict specific groups more likely to:
   - Experience depression and
   - Be undiagnosed with depression.

The results can be used to improve screening standards for healthcare professionals to increase the diagnosis of depression in the groups that are currently underdiagnosed.

Hypotheses

- **Null Hypothesis:** There will be no increase in the reporting of depression by adults as compared to the reporting of major depressive disorder by healthcare providers in medical records in the US.

- **Alternate Hypothesis:** There will be an increase in the reporting of depression by adults as compared to the reporting of major depressive disorder by healthcare providers in medical records in the US.

Methodology

- **Comparative analysis** between adults and healthcare providers using two datasets to compare depression indicators.
  - Datasets:
    - National Health and Nutrition Examination Survey (NHANES)[6]
    - National Ambulatory Medical Care Survey (NAMCS)[7]
- **Modeling** to predict depression and the factors impacting the reporting of depression by 1) adults and 2) healthcare providers.
  - Models:
    - XGBoost
    - Random Forest
    - SVM

6. National Center for Health Statistics. (2022b). National Health and Nutrition Examination Survey. National Center for Health Statistics.
7. National Center for Health Statistics. (2018). National Ambulatory Medical Care Survey

Data Collection

- Secondary data from National Center for Health Statistics (NCHS):
- Surveys and physical examinations
- United States (U.S.)
- 2017 to 2018 (most comprehensive set published since covid pandemic)

NHANES[6]

"Mobile examination center" collects data directly regarding:

- Nutrition
- Diseases
- Health behaviors
- Demographics

NAMCS[7]

- Healthcare providers are surveyed and charts examined regarding:
  - direct patient care received in ambulatory care centers

6. National Center for Health Statistics. (2022b). National Health and Nutrition Examination Survey. National Center for Health Statistics.
7. National Center for Health Statistics. (2018). National Ambulatory Medical Care Survey

Data Storage, Extraction, Cleaning

- Extraction
  - NHANES multiple sub-files in XPT format
    - Downloaded to SAS converter per NHIS recommendation. Saved to CSV file.
    - Merged per Pandas in Python and extracted columns from large set
  - NAMCS
    - Downloaded TXT file from site
    - Converted with Python to CSV

Cleaning

- Missing data calculation
  - No data over the 10% cutoff per proposal
  - PHQ questions were calculated, and all had very low missing, refused or unanswered
  - Team chose to leave them in the dataset due to dropping those rows would lose data in 40+ other variables and other DI measures
- Categorical encoding
  - Categories converted to Boolean
- Variable transformation
  - Some variables grouped for interpretation or ease in use in analysis
  - Diabetes Indicator (DI)
    - Boolean feature created for each sets
    - Compilation of any variables that could indicate depression for each set
    - PHQ total calculated and added to DI for the NHANES

Data Description

- variables after data extraction, cleaning and engineering
  - NHANES-47
  - NAMCS-39

Dataset of NHANES

NHANES DATASET

```python
# Reading the data from CSV file
import pandas as pd
import numpy as np
import statistics as st

df = pd.read_csv("nhanes_features_FINAL.csv")
print(df.shape)
print(df.info())
```

```python
df.head()
```

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	INSNOCH	SEXML	RACEWH	RACEBL	RACEOT	INSSTATE
0	93705	0	0	0	0	0	0	0	0	0	...	0	1	0	1	0	1
1	93706	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0
2	93708	0	0	0	0	0	0	0	0	0	...	0	1	0	0	1	0
3	93709	0	0	0	0	0	0	0	0	0	...	0	1	0	1	0	1
4	93711	1	0	1	0	0	0	0	0	0	...	0	0	0	0	1	0

5 rows × 40 columns

Dataset of NAMCS

NAMCS DATASET

```python
# Reading the data from CSV file
import pandas as pd
import numpy as np
import statistics as st

df = pd.read_csv("namcs_features_FINAL.csv")
print(df.shape)
print(df.info())
```

```python
df.head()
```

	SEQN	AGE	SEXML	RACEWH	RACEBL	RACEOT	NOPROVID	PHYS	PHYSASST	NPNMW	...	DGRP12	DGRP13	DGRP14	DGRP15	DGRP16	DEPRN
0	11002	18	1	1	0	0	0	1	0	0	...	0	0	0	0	0	0
1	11003	18	1	1	0	0	0	1	0	0	...	0	0	0	0	1	0
2	11007	18	1	1	0	0	0	1	0	0	...	0	0	0	0	1	0
3	11009	18	1	1	0	0	0	1	0	0	...	0	0	0	0	0	0
4	11010	18	1	1	0	0	0	1	0	0	...	0	0	0	0	1	0

5 rows × 48 columns

Descriptive Statistics

Normality Test using Python

- Measures of central tendency with variance & range (each variable)
  - Skewness and Kurtosis was analyzed. (see appendix slide 33)
  - According to Hair et al. (2010) and Bryne (2010), data is deemed normal if the skewness is between 2 and +2 and the kurtosis is between 7 and +7.

CATEGORICAL DATA

NHANES

VARIABLE	MOD	STD	VARIANCE	IQR	SKEWNESS	KURTOSIS
Uninsured	0	0.01	0.00	0	76.52	5856.00
CHF	0	0.18	0.03	0	5.12	24.19
Vascular	0	0.21	0.04	0	4.38	17.16
Chest Pain	0	0.16	0.03	0	5.78	31.43
Heart attack	0	0.21	0.04	0	4.33	16.75
Stroke	0	0.21	0.04	0	4.30	16.51
Emphysema	0	0.13	0.02	0	7.23	50.31
Bronchitis	0	0.25	0.06	0	3.45	9.91
COPD	0	0.22	0.05	0	4.13	15.05
Liver	0	0.22	0.05	0	4.12	14.99
Thyroid	0	0.32	0.10	0	2.46	4.07
Cancer	0	0.30	0.09	0	2.66	5.08
DPQ1	0	0.72	0.52	0	2.31	4.74
DPQ2	0	0.68	0.46	0	2.48	5.84
DPQ5	0	0.74	0.55	0	2.32	4.68
DPQ6	0	0.59	0.34	0	3.18	10.30
DPQ8	0	0.51	0.26	0	4.01	16.56
DPQ9	0	0.28	0.08	0	7.38	61.53
PHQ Score	0	4.11	16.89	4	2.08	4.67
DPRN Meds	2	0.39	0.15	0	-2.99	8.73

NAMCS

VARIABLE	MODE	STD	VARIANCE	IQR	SKEWNESS	KURTOSIS
Black	0	0.26	0.07	0	3.25	8.60
Other Race	0	0.23	0.05	0	3.77	12.20
Provider-withheld	0	0.03	0.00	0	31.05	962.33
NPNMW	0	0.11	0.01	0	8.98	78.72
MHP	0	0.11	0.01	0	9.22	82.95
MD/DO	1	0.11	0.01	0	-8.49	70.06
PA	0	0.18	0.03	0	5.12	24.25
No provider	0	0.02	0.00	0	46.62	2171.50
Hospital Owner	0	0.21	0.05	0	4.21	15.73
Owner-withheld	0	0.06	0.00	0	17.87	317.45
Uninsured	0	0.15	0.02	0	6.17	36.09
Infection	0	0.04	0.00	0	23.26	539.12
Urinary	0	0.23	0.05	0	3.88	13.08
Womens Health	0	0.12	0.01	0	8.10	63.56
Genetic	0	0.05	0.00	0	19.82	390.73
Injury	0	0.15	0.02	0	6.38	38.77
Cancer/Blood	0	0.18	0.03	0	5.16	24.65
Metabolic	0	0.21	0.05	0	4.27	16.23
Neurologic	0	0.17	0.03	0	5.49	28.13
Heart/Lung	0	0.26	0.07	0	3.20	8.27
Digestive	0	0.17	0.03	0	5.37	26.80
Bones	0	0.25	0.06	0	3.47	10.05
Therapy Referral	0	0.14	0.02	0	6.97	46.53
MHP Referral	0	0.14	0.02	0	6.76	43.66
DPRN Screen	0	0.20	0.04	0	4.58	19.02
Nurse	0	0.30	0.09	0	2.65	5.05
Insurance Owner	0	0.32	0.10	0	2.36	3.57
Skin	0	0.28	0.08	0	2.89	6.37
Chronic DPRN	0	0.30	0.09	0	2.61	4.81

Descriptive Statistics

Descriptive Statistics of NHANES

```
#Descriptive statistics for continuous data
df1= df[['AGE','PHQSCR']]
df1.describe()
```

	AGE	PHQSCR
count	5856.000000	5856.000000
mean	49.886100	2.819672
std	18.775991	4.109342
min	18.000000	0.000000
25%	33.000000	0.000000
50%	51.000000	1.000000
75%	65.000000	4.000000
max	80.000000	25.000000

```
print('median of AGE:', df.loc[:,'AGE'].median())
print('median of PHQ_SCORE:', df.loc[:,'PHQSCR'].median())
```

```
median of AGE: 51.0
median of PHQ_SCORE: 1.0
```

Descriptive Statistics of NAMCS

```
#Descriptive statistics for continuous data
df1= df['AGE']
df1.describe()
```

```
count    8701.000000
mean       58.104586
std        18.441697
min        18.000000
25%        45.000000
50%        61.000000
75%        72.000000
max        93.000000
Name: AGE, dtype: float64
```

```
print('median of AGE:', df.loc[:,'AGE'].median())
```

```
median of AGE: 61.0
```

Data Analysis

- Following the descriptive statistics, visualizations were utilized to present data across groups and to determine the influence of the depression indicator on distinct groups.
- A stacked bar was used to visualize the relationship between
    1. Depression Indicator vs. Age
    2. Depression Indicator vs. Relation to Race
    3. Depression Indicator vs. Gender
    4. Depression Indicator vs. Education
    5. Depression Indicator vs. Insurance Types
    6. Depression Indicator vs. Types of Providers
    7. Depression Indicator vs. relation to co-morbid conditions
- The pie chart was used to visualize the percentages of gender, depression, and education.
- Histograms of Age, income, education, and gender were plotted to determine each distribution.
- A heatmap was created to view and illustrate the correlations between variables in both datasets.

Gender in NHANES

The gender distribution was found to be 59.4% for Females and 40.6% for Males in NHANES.

Depression Indicator in NHANES

The Depression Indicator in NHANES showed a total of 24.5% had depression and 75.5% had no depression.

Education percentage in NHANES

This pie chart shows the Distribution of Education background with respect to AGE.

Education Histogram in NHANES

This Histogram shows educational distribution of the sample from patients with no educational background to patients with higher studies.

AGE Histogram in NHANES

This age distribution NHANES shows that higher patient population were around 80 years of age and the lowest around 70 years.

Income in NHANES

GENDER NAMCS

A total of 33.3% Males and 66.7% Females were involved in NAMCS dataset.

AGE Histogram in NAMCS

The age distribution in NAMCS shows a larger patient group was aged around 70 years and the lowest with 90 years of age.

Exploratory Data Analysis

Depression Indicator vs Race in NHANES

Depression Indicator vs Race in NAMCS

Depression Indicator vs Age in NHANES
DI
0
1
Age

Depression Indicator vs Age in NAMCS
DI
0
1
Age

Depression Indicator vs Gender in NHANES

Depression Indicator vs Gender in NAMCS

Depression Indicator vs Insurance in NHANES

Depression Indicator vs Insurance in NAMCS

Depression Indicator vs Income in NHANES

Depression Indicator vs Education in NHANES

Depression Indicator vs Provider type in NAMCS

Depression Indicator vs Co-Morbid conditions in NHANES

Depression Indicator vs Co-Morbid conditions in NAMCS

NHANES Heatmap

	Age	Income	Asthma	Overweight	Arthritis	CHF	Vascular	Chest Pain	Heart attack	Stroke	Thyroid	Emphysema	Bronchitis	COPD	Cancer	Commercial	Medicare	Uninsured	Gender	White	Black	Other Race	State-funded	Education	Liver	DI
Age	1	-0.044	-0.049	0.051	0.44	0.18	0.22	0.15	0.2	0.19	0.21	0.12	0.11	0.19	0.3	0.031	0.67	-0.0097	-0.018	0.14	0.0073	-0.14	-0.084	-0.089	0.099	0.062
Income	-0.044	1	-0.039	0.021	-0.081	-0.052	-0.016	-0.041	-0.042	-0.062	-0.013	-0.06	-0.078	-0.096	0.0099	0.34	-0.11	-0.0077	-0.02	0.052	-0.088	0.024	-0.23	-0.15	-0.0097	-0.14
Asthma	-0.049	-0.039	1	0.086	0.12	0.049	0.025	0.048	0.03	0.045	0.038	0.12	0.25	0.19	0.018	-0.046	0.018	-0.0055	0.042	0.0086	0.049	-0.05	0.066	0.055	0.036	0.11
Overweight	0.051	0.021	0.086	1	0.18	0.086	0.061	0.045	0.044	0.035	0.067	0.0017	0.085	0.033	0.039	0.023	0.044	-0.01	0.086	0.035	0.037	-0.066	0.013	0.065	0.063	0.12
Arthritis	0.44	-0.081	0.12	0.18	1	0.15	0.16	0.14	0.16	0.15	0.17	0.12	0.2	0.18	0.16	-0.023	0.36	-0.0083	0.073	0.14	0.018	-0.15	0.0092	0.046	0.1	0.18
CHF	0.18	-0.052	0.049	0.086	0.15	1	0.34	0.23	0.37	0.18	0.094	0.12	0.13	0.22	0.074	-0.03	0.2	-0.0025	-0.042	0.058	0.022	-0.074	0.034	0.035	0.043	0.1
Vascular	0.22	-0.016	0.025	0.061	0.16	0.34	1	0.32	0.52	0.17	0.084	0.051	0.092	0.18	0.11	0.0068	0.21	-0.0028	-0.11	0.12	-0.046	-0.074	0.0013	0.0082	0.029	0.078
Chest Pain	0.15	-0.041	0.048	0.045	0.14	0.23	0.32	1	0.31	0.13	0.083	0.095	0.11	0.12	0.072	-0.032	0.13	-0.0022	-0.027	0.073	-0.045	-0.032	0.041	0.0074	0.052	0.098
Heart attack	0.2	-0.042	0.03	0.044	0.16	0.37	0.52	0.31	1	0.21	0.046	0.12	0.084	0.19	0.078	-0.036	0.19	-0.0029	-0.098	0.081	-0.021	-0.06	0.014	0.026	0.039	0.081
Stroke	0.19	-0.062	0.045	0.035	0.15	0.18	0.17	0.13	0.21	1	0.065	0.073	0.086	0.11	0.082	-0.055	0.21	-0.0029	0.0058	0.043	0.053	-0.086	0.039	0.029	0.038	0.12
Thyroid	0.21	-0.013	0.038	0.067	0.17	0.094	0.084	0.083	0.046	0.065	1	0.021	0.09	0.075	0.098	0.032	0.18	-0.0046	0.17	0.093	-0.063	-0.036	-0.0052	-0.021	0.038	0.081
Emphysema	0.12	-0.06	0.12	0.0017	0.12	0.12	0.051	0.095	0.12	0.073	0.021	1	0.22	0.48	0.1	-0.027	0.14	-0.0018	-0.037	0.081	-0.022	-0.059	0.032	0.025	0.063	0.086
Bronchitis	0.11	-0.078	0.25	0.085	0.2	0.13	0.092	0.11	0.084	0.086	0.09	0.22	1	0.34	0.08	-0.066	0.13	-0.0035	0.031	0.094	-0.0091	-0.083	0.039	0.053	0.069	0.15
COPD	0.19	-0.096	0.19	0.033	0.18	0.22	0.18	0.12	0.19	0.11	0.075	0.48	0.34	1	0.13	-0.052	0.21	-0.003	-0.02	0.14	-0.025	-0.11	0.082	0.052	0.062	0.12
Cancer	0.3	0.0099	0.018	0.039	0.16	0.074	0.11	0.072	0.078	0.082	0.098	0.1	0.08	0.13	1	0.049	0.28	-0.0044	0.01	0.17	-0.051	-0.12	-0.025	0.00067	0.035	0.055
Commercial	0.031	0.34	-0.046	0.023	-0.023	-0.03	0.0068	-0.032	-0.036	-0.055	0.032	-0.027	-0.066	-0.052	0.049	1	-0.064	-0.013	0.017	0.11	-0.07	-0.047	-0.39	-0.095	-0.061	-0.15
Medicare	0.67	-0.11	0.018	0.044	0.36	0.2	0.21	0.13	0.19	0.21	0.18	0.14	0.13	0.21	0.28	-0.064	1	0.0079	-0.036	0.2	-0.0016	-0.19	-0.015	0.0059	0.044	0.1
Uninsured	-0.0097	-0.0077	0.0055	-0.01	-0.0083	0.0025	0.0028	0.0022	0.0029	0.0029	0.0046	0.0018	0.0035	-0.003	-0.0044	-0.013	-0.0079	1	-0.013	-0.0095	0.024	-0.011	-0.006	-0.011	-0.003	-0.0074
Gender	-0.018	-0.02	0.042	0.086	0.073	-0.042	-0.11	-0.027	-0.098	-0.0058	0.17	-0.037	0.031	-0.02	0.01	0.017	-0.036	-0.013	1	-0.021	0.014	0.0084	0.065	0.016	-0.019	0.075
White	0.14	0.052	0.0086	0.035	0.14	0.058	0.12	0.073	0.081	0.043	0.093	0.081	0.094	0.14	0.17	0.11	0.2	-0.0095	-0.021	1	-0.4	-0.62	-0.066	0.092	-0.0033	0.1
Black	0.0073	-0.088	0.049	0.037	0.018	0.022	-0.046	-0.045	-0.021	0.053	-0.063	-0.022	-0.0091	-0.025	-0.051	-0.07	-0.0016	0.024	0.014	-0.4	1	-0.47	0.037	0.11	-0.051	-0.022
Other Race	-0.14	0.024	-0.05	-0.066	-0.15	-0.074	-0.074	-0.032	-0.06	-0.086	-0.036	-0.059	-0.083	-0.11	-0.12	-0.047	-0.19	-0.011	0.0084	-0.62	-0.47	1	0.032	-0.18	0.047	-0.077
State-funded	-0.084	-0.23	0.066	0.013	0.0092	0.034	0.0013	0.041	0.014	0.039	-0.0052	0.032	0.039	0.082	-0.025	-0.39	-0.015	-0.006	0.065	-0.066	0.037	0.032	1	0.069	0.057	0.14
Education	-0.089	-0.15	0.055	0.065	0.046	0.035	0.0082	0.0074	0.026	0.029	-0.021	0.025	0.053	0.052	-0.00067	-0.095	0.0059	-0.011	0.016	0.092	0.11	-0.18	0.069	1	-0.0042	0.068
Liver	0.099	-0.0097	0.036	0.063	0.1	0.043	0.029	0.052	0.039	0.038	0.038	0.063	0.069	0.062	0.035	-0.061	0.044	-0.003	-0.019	-0.0033	-0.051	0.047	0.057	-0.0042	1	0.08
DI	0.062	-0.14	0.11	0.12	0.18	0.1	0.078	0.098	0.081	0.12	0.081	0.086	0.15	0.12	0.055	-0.15	0.1	-0.0074	0.075	0.1	-0.022	-0.077	0.14	0.068	0.08	1

NAMCS Heatmap

Hypothesis Testing

- The chi-square test was utilized for hypothesis testing because both of the data sets contained categorical data.
- Even though we mentioned in the draft proposal using a two-sample t test, we employed the chi square test because the data we had was categorical. So we chose Chi Square over the two sample tests.
- With the use of the chi-square contingency test, we identified the significant factors from both datasets have an association with the depression indicator.
- The significant factors for the depression indicator for both datasets was used to forecast depression using XGBoost, Random Forest and SVM models.

Significant Features found after the Chi Squared test

- namcs = ['AGE', 'GENDER', 'RACE-WHITE', 'Physician', 'Physician assistant', 'Mental health provider', 'Other provider', 'Medicare', 'State Insurance', 'Private Insurance', 'Specialty:Primary care', 'Specialty:Surgical care', 'Specialty:Medical care', 'Practice_owner:Unknown', 'Practice_owner:Refused to answer question', 'Practice_owner:Physician or physician group', 'Practice_owner:Medical/Academic health center; other hospital', 'Diagnosis: No chronic conditions', 'Diagnosis: Cancer/Blood', 'Diagnosis: Metabolic', 'Diagnosis: Neurologic', 'Diagnosis: Eyes/Ears', 'Diagnosis: Heart/Lung', 'Diagnosis: Digestive', 'Diagnosis: Skin', 'Diagnosis: Bones', 'Diagnosis: Women\'s Health', 'Diagnosis: Injury', 'Diagnosis: Public Health']
- nhanes = ['AGE', 'Month family income', 'MedHistory: Asthma', 'MedHistory: Overweight', 'MedHistory: Arthritis', 'MedHistory: Congestive heart failure', 'MedHistory: Coronary heart disease', 'MedHistory: Angina/Angina Pectoris', 'MedHistory: Heart attack', 'MedHistory: Stroke', 'MedHistory: Thyroid problem', 'MedHistory: Emphysema', 'MedHistory: Chronic Bronchitis', 'MedHistory: COPD', 'MedHistory: Cancer or Malignancy', 'Private Insurance', 'Medicare', 'GENDER', 'RACE-WHITE', 'RACE-OTHERS', 'State Insurance', 'EDUCATION LEVEL', 'MedHistory: Liver issues']

Binary Classification Models

- XG BOOST was applied for both significant variables of the datasets NHANES and NAMCS. We employed SMOTE to balance the dataset and to achieve better recall for the model.
- We employ Random Forest model too for testing out the datasets for better explaining the pattern in it.

Using XG boost model

XGBOOST CLASSIFIER

```
In [6]: xgb_cl = xgb.XGBClassifier()
```

namcs dataset

```
In [7]: Xnam_train, Xnam_test, ynam_train, ynam_test = train_test_split(X_nam, y_nam, train_size=0.8, random_state=1000000000)
```

Model scores for namcs

```
In [12]: skmets(ynam_test, prednam)

Out[12]: {'Accuracy': 0.8345778288340034,
          'Precision': 0.4470198675496689,
          'Sensitivity_recall': 0.52734375,
          'Specificity': 0.8875420875420875,
          'F1_score': 0.48387096774193555}
```

Feature Importance for predicting Depression for namcs

In [13]:
```
feat_importances = pd.Series(xgb_cl.feature_importances_, index=namcs)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[13]: <AxesSubplot:>

FEATURE IMPORTANCE

- Feature Importance depicts the columns that contributed towards the Depression indication and according to the chart, the diagnosis of the patient's medical history has the greater feature importance out of all the NAMCS features provided.

Performance Analysis of ML models

Confusion matrix for namcs

In [14]:
```
cn_matrix_nam = metrics.confusion_matrix(ynam_test, prednam)
cm_display_nam = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nam , display_labels = [False, True])
cm_display_nam.plot()
plt.show()
```

XG BOOST namcs dataset Report

In [15]:
```
print(classification_report(ynam_test,prednam))
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90      1485
           1       0.45      0.53      0.48       256

    accuracy                           0.83      1741
   macro avg       0.68      0.71      0.69      1741
weighted avg       0.85      0.83      0.84      1741
```

ROC Curve for namcs

nhanes dataset

In [18]:
```
Xnh_train, Xnh_test, ynh_train, ynh_test = train_test_split(X_nh, y_nh, train_size=0.8, random_state=1000000000)
```

In [19]:
```
sm = SMOTE()
Xnh_train, ynh_train = sm.fit_resample(Xnh_train, ynh_train)
```

In [20]:
```
# nhanes Fit
xgb_cl.fit(Xnh_train, ynh_train)
modelnh_fit = xgb_cl.fit(Xnh_train, ynh_train)
# Predict
prednh = xgb_cl.predict(Xnh_test)
```

Model scores for nhanes

In [22]:
```
skmets(ynh_test, prednh)
```

Out[22]:
```
{'Accuracy': 0.6399317406143344,
 'Precision': 0.3209549071618037,
 'Sensitivity_recall': 0.42160278745644597,
 'Specificity': 0.7107344632768362,
 'F1_score': 0.36445783132530113}
```

Confusion matrix for nhanes

In [23]:
```
cn_matrix_nh = metrics.confusion_matrix(ynh_test, prednh)
cm_display_nh = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nh , display_labels = [False, True])
cm_display_nh.plot()
plt.show()
```

XG BOOST nhanes dataset Report

In [24]:
```
print("XG BOOST nhanes dataset Report")
print(classification_report(ynh_test,prednh))
```

```
XG BOOST nhanes dataset Report
              precision    recall  f1-score   support

           0       0.79      0.71      0.75       885
           1       0.32      0.42      0.36       287

    accuracy                           0.64      1172
   macro avg       0.56      0.57      0.56      1172
weighted avg       0.68      0.64      0.65      1172
```

Feature Importance for predicting Depression for nhanes

In [25]:
```
feat_importances = pd.Series(xgb_cl.feature_importances_, index=nhanes)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[25]: <AxesSubplot:>

Precision-Recall Curve for namcs
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.60)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Random Forest Classifier – NAMCS dataset

RANDOM FOREST CLASSIFIER

In [28]:
```
clf = RandomForestClassifier(criterion = "gini", random_state=1)
```

Cross Validation for namcs

In [32]:
```
scores = cross_val_score(clf, Xnam_train, ynam_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.67 with a standard deviation of 0.02
```

Model scores for namcs

In [33]:
```
skmets(ynam_test, prednam)
```

Out[33]:
```
{'Accuracy': 0.8575531303848363,
 'Precision': 0.5204081632653061,
 'Sensitivity_recall': 0.3984375,
 'Specificity': 0.9367003367003367,
 'F1_score': 0.4513274336283186}
```

Feature Importance for predicting Depression for namcs

In [34]:
```
feat_importances = pd.Series(clf.feature_importances_, index=namcs)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[34]: <AxesSubplot:>

Confusion matrix for namcs

In [35]:
```
cn_matrix_nam = metrics.confusion_matrix(ynam_test, prednam)
cm_display_nam = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nam , display_labels = [False, True])
cm_display_nam.plot()
plt.show()
```

RANDOM FOREST namcs dataset Report

In [36]:
```
print(classification_report(ynam_test,prednam))
```

```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      1485
           1       0.52      0.40      0.45       256

    accuracy                           0.86      1741
   macro avg       0.71      0.67      0.68      1741
weighted avg       0.84      0.86      0.85      1741
```

1.0
0.8
0.6
0.4
0.2
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.80)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

nhanes dataset

In [39]:
```
Xnh_train, Xnh_test, ynh_train, ynh_test = train_test_split(X_nh, y_nh, train_size=0.8, random_state=1000000000)
```

In [40]:
```
sm = SMOTE()
Xnh_train, ynh_train = sm.fit_resample(Xnh_train, ynh_train)
```

In [41]:
```
# nhanes
clf.fit(Xnh_train, ynh_train)
prednh = clf.predict(Xnh_test)
```

Cross Validation for nhanes

In [42]:
```
scores = cross_val_score(clf, Xnh_train, ynh_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.75 with a standard deviation of 0.06
```

Model scores for nhanes

In [43]:
```
skmets(ynh_test, prednh)
```

Out[43]:
```
{'Accuracy': 0.6168941979522184,
 'Precision': 0.2798913043478261,
 'Sensitivity_recall': 0.3588850174216028,
 'Specificity': 0.7005649717514124,
 'F1_score': 0.31450381679389317}
```

Feature Importance for predicting Depression for nhanes

In [44]:
```
feat_importances = pd.Series(clf.feature_importances_, index=nhanes)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[44]: <AxesSubplot:>

Confusion matrix for nhanes

In [45]:
```
cn_matrix_nh = metrics.confusion_matrix(ynh_test, prednh)
cm_display_nh = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nh , display_labels = [False, True])
cm_display_nh.plot()
plt.show()
```

RANDOM FOREST nhanes dataset Report

In [46]:
```
print(classification_report(ynh_test, prednh))
```

```
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       885
           1       0.28      0.36      0.31       287

    accuracy                           0.62      1172
   macro avg       0.53      0.53      0.52      1172
weighted avg       0.65      0.62      0.63      1172
```

Precision-Recall Curve for nhanes
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.59)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Findings

- Both models were able to provide good classification for the nhanes and namcs datasets.
- The namcs data was able to provide better classification for the Depression as compared to NHANES survey data.
- The diagnosis codes were able to provide a good deal of information to the model to enable the Depression Indicator.
- There was found to be a decrease in the depression cases in the nhanes dataset as compared to namcs dataset. Hence, we reject the null hypothesis as there is substantiating evidence to do so.
- A person in the NAMCS dataset has higher chance of being detected for depression compared to a person who was surveyed in NHANES.

Conclusion

- Hypothesis testing supported the rejection of the null hypothesis
- This project was able to demonstrate multiple factors impacting depression and the providers who report depression.
- A model was established to support the further development of a survey that can be used to further develop a depression indicator.

Limitations

- We used XGBoost and Random Forest instead of SVM because they provide more explanatory power for the datasets we used.
- Findings are from two datasets taken from two populations. Conclusions from each set cannot be extrapolated to the other set without further testing
- The datasets are from 2018, collected in 2017. Covid limited more recent data collection. Findings may not represent most resent environments and populations.

Project Management

- 7-member team
- Sub-teams-all members were lead of some aspect of a sub-team though-out the process
  - Data Discovery & Extractions
  - Data Cleaning & Engineering
  - Machine Learning/Modeling
  - Visualization
  - Project Management

- Process
  - Weekly Teams meetings 11am Fridays
  - Teams / Canvas for file storage
  - WhatsApp for daily direct communications
  - Canvas for Official communications
    - Announcements
    - Communications with course staff/professor
  - phpAdmin(mySQL)/Colab- file storage
  - Wrike Software for Project timeline development and tracking

Challenges:

- Datasets complex and difficult to convert from XPT
- Two datasets increased complexity
- Most variables were categorical challenging the analysis

Appendix

CATEGORICAL DATA

NHANES

VARIABLE	MOD	STD	VARIANCE	IQR	SKEWNESS	KURTOSIS
Gender	1	0.50	0.25	1	-0.06	-2.00
Black	0	0.42	0.18	0	1.29	-0.34
Other Race	0	0.49	0.24	1	0.31	-1.91
White	0	0.48	0.23	1	0.64	-1.59
Education	4	1.55	2.40	3	-0.44	-1.33
Income	15	4.98	24.81	10	0.04	-1.17
Medicare	0	0.44	0.20	1	1.04	-0.91
Uninsured	0	0.01	0.00	0	76.52	5856.00
Commercial	0	0.50	0.25	1	0.05	-2.00
State-funded	0	0.38	0.14	0	1.71	0.93
Asthma	0	0.36	0.13	0	1.95	1.79
Overweight	0	0.49	0.24	1	0.49	-1.76
Arthritis	0	0.45	0.21	1	0.93	-1.14
CHF	0	0.18	0.03	0	5.12	24.19
Vascular	0	0.21	0.04	0	4.38	17.16
Chest Pain	0	0.16	0.03	0	5.78	31.43
Heart attack	0	0.21	0.04	0	4.33	16.75
Stroke	0	0.21	0.04	0	4.30	16.51
Emphysema	0	0.13	0.02	0	7.23	50.31
Bronchitis	0	0.25	0.06	0	3.45	9.91
Thyroid	0	0.32	0.10	0	2.46	4.07
COPD	0	0.22	0.05	0	4.13	15.05
Cancer	0	0.30	0.09	0	2.66	5.08
Liver	0	0.22	0.05	0	4.12	14.99
DPQ1	0	0.72	0.52	0	2.31	4.74
DPQ2	0	0.68	0.46	0	2.48	5.84
DPQ3	0	0.91	0.83	1	1.60	1.44
DPQ4	0	0.91	0.83	1	1.33	0.83
DPQ5	0	0.74	0.55	0	2.32	4.68
DPQ6	0	0.59	0.34	0	3.18	10.30
DPQ7	0	0.64	0.41	0	3.10	9.27
DPQ8	0	0.51	0.26	0	4.01	16.56
DPQ9	0	0.28	0.08	0	7.38	61.53
PHQ Score	0	4.11	16.89	4	2.08	4.67
Feels DPRN	5	1.27	1.61	1	-1.40	1.19
DPRN Meds	2	0.39	0.15	0	-2.99	8.73
DPRN Level	0	1.15	1.31	2	0.80	-0.86
DI	0	0.43	0.19	0	1.18	-0.60

NAMCS

VARIABLE	MODE	STD	VARIANCE	IQR	SKEWNESS	KURTOSIS
Gender	0	0.49	0.24	1	0.31	-1.90
Black	0	0.26	0.07	0	3.25	8.60
Other Race	0	0.23	0.05	0	3.77	12.20
White	1	0.34	0.11	0	-2.17	2.71
Speciality Care	0	0.46	0.21	1	0.84	-1.29
Primary Care	0	0.44	0.20	1	1.05	-0.91
Surgery Care	0	0.49	0.24	1	0.30	-1.91
Nurse	0	0.30	0.09	0	2.65	5.05
Provider-withheld	0	0.03	0.00	0	31.05	962.33
NPNMW	0	0.11	0.01	0	8.98	78.72
MHP	0	0.11	0.01	0	9.22	82.95
Other provider	0	0.46	0.21	1	0.83	-1.31
MD/DO	1	0.11	0.01	0	-8.49	70.06
PA	0	0.18	0.03	0	5.12	24.25
No provider	0	0.02	0.00	0	46.62	2171.50
Hospital Owner	0	0.21	0.05	0	4.21	15.73
Insurance Owner	0	0.32	0.10	0	2.36	3.57
Phys.Owner	1	0.41	0.17	0	-1.41	-0.02
Owner-withheld	0	0.06	0.00	0	17.87	317.45
Medicare	0	0.48	0.23	1	0.60	-1.64
Uninsured	0	0.15	0.02	0	6.17	36.09
Commercial	0	0.50	0.25	1	0.26	-1.93
State-funded	0	0.27	0.07	0	3.09	7.58
Infection	0	0.04	0.00	0	23.26	539.12
Urinary	0	0.23	0.05	0	3.88	13.08
Womens Health	0	0.12	0.01	0	8.10	63.56
Genetic	0	0.05	0.00	0	19.82	390.73
Other Disorder	0	0.28	0.08	0	3.01	7.03
Injury	0	0.15	0.02	0	6.38	38.77
Morbidity	0	0.00	0.00	0	0.00	0.00
Public health	0	0.37	0.13	0	1.87	1.50
Cancer/Blood	0	0.18	0.03	0	5.16	24.65
Metabolic	0	0.21	0.05	0	4.27	16.23
Neurologic	0	0.17	0.03	0	5.49	28.13
Eyes/Ears	0	0.35	0.12	0	2.07	2.30
Heart/Lung	0	0.26	0.07	0	3.20	8.27
Digestive	0	0.17	0.03	0	5.37	26.80
Skin	0	0.28	0.08	0	2.89	6.37
Bones	0	0.25	0.06	0	3.47	10.05
No illness	0	0.47	0.22	1	0.70	-1.50
Chronic DPRN	0	0.30	0.09	0	2.61	4.81
Therapy Referral	0	0.14	0.02	0	6.97	46.53
MHP Referral	0	0.14	0.02	0	6.76	43.66
DPRN Screen	0	0.20	0.04	0	4.58	19.02
DI	0	0.36	0.13	0	1.91	1.66

CONTINUOUS DATA

DATASET	n	VARIABLE	MEAN	MEDIAN	MODE	STD	VARIANCE	IQR	SKEWNESS	KURTOSIS
NAMCS	8701	Age	58	61	71	18.44	340.1	27	0	-0.74
NHANES	5856	Age	50	51	80	18.78	352.54	32	-0.06	-1.17
NHANES	5856	PHQ Score	2.82	1	0	4.11	16.89	4	2.08	4.67

References

1. World Health Organization. (2022). *Mental Health and COVID-19: Early evidence of the pandemic's impact* (WHO/2019-nCoV/Sci_Brief/Mental_health/2022.). https://www.who.int/publications/i/item/WHO-2019-nCoV-Sci_Brief-Mental_health-2022.1

2. American Academy of Pediatrics, American Academy of Child, and Adolescent Psychiatry, and Children's Hospital Association,. (2021, 10/19/2021). *AAP-AACAP-CHA Declaration of a National Emergency in Child and Adolescent Mental Health*. Retrieved 09/30/2022 from https://www.aap.org/en/advocacy/child-and-adolescent-healthy-mental-development/aap-aacap-cha-declaration-of-a-national-emergency-in-child-and-adolescent-mental-health/

3. Hansson, L., Jormfeldt, H., Svedberg, P., & Svensson, B. (2013). Mental health professionals' attitudes towards people with mental illness: do they differ from attitudes held by people with mental illness? *Int J Soc Psychiatry*, *59*(1), 48-54. https://doi.org/10.1177/0020764011423176

4. Knaak, S., Mantler, E., & Szeto, A. (2017). Mental illness-related stigma in healthcare. *Healthcare Management Forum*, *30*(2), 111-116. https://doi.org/10.1177/0840470416679413

5. Garcia, M. E., Hinton, L., Neuhaus, J., Feldman, M., Livaudais-Toman, J., & Karliner, L. S. (2022). Equitability of Depression Screening After Implementation of General Adult Screening in Primary Care. *JAMA Network Open*, *5*(8), e2227658-e2227658. https://doi.org/10.1001/jamanetworkopen.2022.27658

6. Hair, J., Black, W. C., Babin, B. J. & Anderson, R. E. (2010) Multivariate data analysis (7th ed.). Upper Saddle River, New Jersey: Pearson Educational International.

Codes

Codebook

NAMCS		NHANES	
CODE	LABEL	CODE	LABEL
AGE	Age	AGE	Age
SEXML	Gender	SEXML	Gender
RACEBL	Black	RACEBL	Black
RACEOT	Other Race	RACEOT	Other Race
RACHWH	White	RACEWH	White
SPMEC	Specialty Care	EDUC	Education
SPPRC	Primary Care	INDHHIN2	Income
SPSUR	Surgery Care	INSMCARE	Medicare
RNLPN	Nurse	INNOCH	Uninsured
PHYS	MD/DO	INSPRVT	Commercial
PHYSASST	PA	INSSTATE	State-funded
NPNMW	NPNMW	MCQ160O	COPD
MHP	MHP	MCQ160A	Arthritis
PROVNONE	No provider	MCQ01O	Asthma
OTHPROV	Other provider	MCQ160K	Bronchitis
NOPROVID	Provider-withheld	MCQ160B	CHF
OWNPHYS	Phys. Owner	MCQ160C	Vascular
OWNHOSP	Hospital Owner	MCQ160D	Chest Pain
OWNUNK	Owner-withheld	MCQ080	Overweight
OWNINSR	Insurance Owner	MCQ160E	Heart attack
OWNUNKN	Unk Practice	MCQ160F	Stroke
INSPRVT	Commercial	MCQ160G	Emphysema
INSMCARE	Medicare	MCQ160M	Thyroid
INSSTATE	State-funded	MCQ700	Liver
INSNOCH	Uninsured	MCQ220	Cancer
DGRP1	Infection	DPQ010	DPQ1
DGRP10	Urinary	DPQ020	DPQ2
DGRP11	Womens Health	DPQ030	DPQ3
DGRP12	Genetic	DPQ040	DPQ4
DGRP13	Other Disorder	DPQ050	DPQ5
DGRP14	Injury	DPQ060	DPQ6
DGRP15	Morbidity	DPQ070	DPQ7
DGRP16	Public health	DPQ080	DPQ8
DGRP2	Cancer/Blood	DPQ090	DPQ9
DGRP3	Metabolic	PHQSCR	PHQ Score
DGRP4	Neurologic	DLQ140	Feels DPRN
DGRP5	Eyes/Ears	DLQ150	DPRN Meds
DGRP6	Heart/Lung	DLQ170	DPRN Level
DGRP7	Digestive	DI	DI
DGRP8	Skin		
DGRP9	Bones		
NOCHRON	No illness		
DPRN	Chronic DPRN		
MENTAL	Therapy Referral		
PSYCHOTH	MHP Referral		
DEPRESS	DPRN Screen		
DI	DI		

Finding Missing values in NAMCS

```
import pandas as pd
df = pd.read_csv('NAMCS_limited3.csv')
```

```
df.dtypes
```

```
print('% of missing values in AGER:',len(df[df.AGER == -9])/len(df.AGER)*100)
print('% of missing values in SEX:',len(df[df.SEX == -9])/len(df.SEX)*100)
print('% of missing values in RACEUN:',len(df[df.RACEUN == -9])/len(df.RACEUN)*100)
print('% of missing values in PAYTYPER:',len(df[df.PAYTYPER == -9])/len(df.PAYTYPER)*100)
print('% of missing values in DIAG1:',len(df[df.DIAG1 == '-9'])/len(df.DIAG1)*100)
print('% of missing values in DIAG2:',len(df[df.DIAG2 == '-9'])/len(df.DIAG2)*100)
print('% of missing values in DIAG3:',len(df[df.DIAG3 == '-9'])/len(df.DIAG3)*100)
print('% of missing values in DIAG4:',len(df[df.DIAG4 == '-9'])/len(df.DIAG4)*100)
print('% of missing values in DIAG5:',len(df[df.DIAG5 == '-9'])/len(df.DIAG5)*100)
print('% of missing values in DEPRN:',len(df[df.DEPRN == '-9'])/len(df.DEPRN)*100)
print('% of missing values in NOCHRON:',len(df[df.NOCHRON == -9])/len(df.NOCHRON)*100)
print('% of missing values in DEPRESS:',len(df[df.DEPRESS == -9])/len(df.DEPRESS)*100)
print('% of missing values in MENTAL:',len(df[df.MENTAL == -9])/len(df.MENTAL)*100)
print('% of missing values in PSYCHOTH:',len(df[df.PSYCHOTH == -9])/len(df.PSYCHOTH)*100)
print('% of missing values in NOPROVID:',len(df[df.NOPROVID == -9])/len(df.NOPROVID)*100)
print('% of missing values in PHYS:',len(df[df.PHYS == -9])/len(df.PHYS)*100)
print('% of missing values in PHYSASST:',len(df[df.PHYSASST == -9])/len(df.PHYSASST)*100)
print('% of missing values in NPNWM:',len(df[df.NPNMW == -9])/len(df.NPNMW)*100)
print('% of missing values in RNLPN:',len(df[df.RNLPN == -9])/len(df.RNLPN)*100)
print('% of missing values in MHP:',len(df[df.MHP == -9])/len(df.MHP)*100)
print('% of missing values in OTHPROV:',len(df[df.OTHPROV == -9])/len(df.OTHPROV)*100)
print('% of missing values in PROVNONE:',len(df[df.PROVNONE == -9])/len(df.PROVNONE)*100)
print('% of missing values in SPECCAT:',len(df[df.SPECCAT == -9])/len(df.SPECCAT)*100)
print('% of missing values in OWNSR:',len(df[df.OWNSR == -9])/len(df.OWNSR)*100)
```

```
% of missing values in AGER: 0.0
% of missing values in SEX: 0.0
% of missing values in RACEUN: 28.564251984326333
% of missing values in PAYTYPER: 2.5318999296694464
% of missing values in DIAG1: 0.4119360996684417
% of missing values in DIAG2: 34.20074349442379
% of missing values in DIAG3: 57.87199839244449
% of missing values in DIAG4: 73.103586858233 7
% of missing values in DIAG5: 84.29619210288355
% of missing values in DEPRN: 0.0
% of missing values in NOCHRON: 0.0
% of missing values in DEPRESS: 0.0
% of missing values in MENTAL: 0.0
% of missing values in PSYCHOTH: 0.0
% of missing values in NOPROVID: 0.0
% of missing values in PHYS: 0.0
% of missing values in PHYSASST: 0.0
% of missing values in NPNWM: 0.0
% of missing values in RNLPN: 0.0
% of missing values in MHP: 0.0
% of missing values in OTHPROV: 0.0
% of missing values in PROVNONE: 0.0
% of missing values in SPECCAT: 0.0
% of missing values in OWNSR: 4.450919320807797
```

Limit NAMCS Source File to Project Variables

```
In [1]: import pandas as pd
        df = pd.read_csv('namcs2018.csv',
            usecols=['AGE', 'SEX', 'RACER', 'PAYTYPER', 'DIAG1','DIAG2','DIAG3', 'DIAG4','DIAG5', 'DEPRN', 'NOCHRON', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID','PHYS','PHYSASST','NPNMW','RNLPN', 'MHP', 'OTHPROV','PROVNONE', 'SPECCAT', 'OWNSR'
```

```
In [2]: df.head()
```

Out[2]:

	AGE	SEX	RACER	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	...	NOPROVID	PHYS	PHYSASST	NPNMW	RNLPN	MHP	OTHPROV	PROVNONE	SPECCAT	OWNSR
0	69	2	1	1	R918	J439	R599	J449	R090	0	...	0	1	0	0	0	0	0	0	3	1
1	64	1	1	1	J449	Z720	R918	R911	-9	0	...	0	1	0	0	0	0	0	0	3	1
2	81	1	1	2	G473	I489	I10-	-9	-9	0	...	0	1	0	0	0	0	0	0	3	1
3	66	1	1	2	J449	J961	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	1
4	44	1	1	1	J449	R911	F172	I503	I340	0	...	0	1	0	0	0	0	0	0	3	1

5 rows × 24 columns

```
In [3]: df.shape
```

```
Out[3]: (9953, 24)
```

```
In [4]: print(df.columns)
```

```
Index(['AGE', 'SEX', 'RACER', 'PAYTYPER', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4',
       'DIAG5', 'DEPRN', 'NOCHRON', 'DEPRESS', 'MENTAL', 'PSYCHOTH',
       'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV',
       'PROVNONE', 'SPECCAT', 'OWNSR'],
      dtype='object')
```

```
In [5]: df1 = df[df['AGE'] > 17]
```

```
In [6]: df1.head()
```

Out[6]:

	AGE	SEX	RACER	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	...	NOPROVID	PHYS	PHYSASST	NPNMW	RNLPN	MHP	OTHPROV	PROVNONE	SPECCAT	OWNSR
0	69	2	1	1	R918	J439	R599	J449	R090	0	...	0	1	0	0	0	0	0	0	3	1
1	64	1	1	1	J449	Z720	R918	R911	-9	0	...	0	1	0	0	0	0	0	0	3	1
2	81	1	1	2	G473	I489	I10-	-9	-9	0	...	0	1	0	0	0	0	0	0	3	1
3	66	1	1	2	J449	J961	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	1
4	44	1	1	1	J449	R911	F172	I503	I340	0	...	0	1	0	0	0	0	0	0	3	1

5 rows × 24 columns

```
In [7]: df1.to_csv('NAMCS2.csv', encoding='utf-8', index=False)
```

Merging NHANES Source

In [2]:
```
import pandas as pd
from functools import reduce
df1=pd.read_csv('nhanes_depression_screeners.csv')
df2=pd.read_csv('nhanes_demo.csv')
df3=pd.read_csv('nhanes_medical_conditions.csv')
df4=pd.read_csv('nhanes_disability.csv')
df5=pd.read_csv('nhanes_health_insurance.csv')
#define list of DataFrames
dfs = [df1, df2, df3, df4, df5]

#merge all DataFrames into one
final_df = reduce(lambda  left,right: pd.merge(left,right,on=['SEQN'],
                                               how='outer'), dfs)
```

5 rows × 158 columns

In [10]:
```
final_df.drop_duplicates(keep=False)
```

Out[10]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	HIQ031E	HIQ031F	HIQ031H	HIQ031I	HIQ031J	HIQ031AA	HIQ260	HIQ105	HIQ270	HIQ210
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	0	1	2
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2

3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2

5 rows × 158 columns

In [10]:
```
final_df.drop_duplicates(keep=False)
```


Out[10]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	HIQ031E	HIQ031F	HIQ031H	HIQ031I	HIQ031J	HIQ031AA	HIQ260	HIQ105	HIQ270	HIQ210
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	0	1	2
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	1	1	2
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	21	0	23	0	0	0	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
36987	102942.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0	0	0	0	0	0	0	0	1	2
36988	102945.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0	0	21	0	0	0	0	0	1	2
36989	102950.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0	19	21	0	0	0	0	0	1	2
36990	102951.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0	0	0	0	0	0	0	0	1	2

NAMCS Cleaning and Feature Engineering

In [1]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('namcs2.csv', usecols=["AGE","SEX","RACER","PAYTYPER","DIAG1","DIAG2","DIAG3","DIAG4","DIAG5","DEPRN","NOCHRON","DEPRESS","MENTAL",
                                      "PSYCHOTH","NOPROVID","PHYS","PHYSASST","NPNMW","RNLPN","MHP","OTHPROV","PROVNONE","SPECCAT","OWNSR"],
                 na_values = ['na', '-', '.', '', 'NaN'])

df.head()
```

Out[1]:

	AGE	SEX	RACER	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	...	NOPROVID	PHYS	PHYSASST	NPNMW	RNLPN	MHP	OTHPROV	PROVNONE	SPECCAT	OWNS
0	18	1	1	1	Z000	F419	E739	N832	-9	0	...	0	1	0	0	1	0	0	0	1	
1	18	1	1	1	L409	-9	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	
2	18	2	1	1	L989	-9	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	
3	18	2	1	1	Z768	L988	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	
4	18	1	1	1	E669	-9	-9	-9	-9	0	...	0	1	0	0	0	0	0	0	3	

5 rows × 24 columns

In [2]:
```
df['SEQN'] = df.index + 11000
df.columns
```

Out[2]:
```
Index(['AGE', 'SEX', 'RACER', 'PAYTYPER', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4',
       'DIAG5', 'DEPRN', 'NOCHRON', 'DEPRESS', 'MENTAL', 'PSYCHOTH',
       'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV',
       'PROVNONE', 'SPECCAT', 'OWNSR', 'SEQN'],
      dtype='object')
```

In [3]:
```
df.reindex(columns=['SEQN', 'AGE', 'SEX', 'RACER', 'PAYTYPER', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4',
       'DIAG5', 'DEPRN', 'NOCHRON', 'DEPRESS', 'MENTAL', 'PSYCHOTH',
       'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV',
       'PROVNONE', 'SPECCAT', 'OWNSR' ])
```

Out[3]:

	SEQN	AGE	SEX	RACER	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	...	NOPROVID	PHYS	PHYSASST	NPNMW	RNLPN	MHP	OTHPROV	PROVNONE	SPECCAT	O
0	11000	18	1	1	1	Z000	F419	E739	N832	-9	...	0	1	0	0	1	0	0	0	1	
1	11001	18	1	1	1	L409	-9	-9	-9	-9	...	0	1	0	0	0	0	0	0	3	
2	11002	18	2	1	1	L989	-9	-9	-9	-9	...	0	1	0	0	0	0	0	0	3	
3	11003	18	2	1	1	Z768	L988	-9	-9	-9	...	0	1	0	0	0	0	0	0	3	
4	11004	18	1	1	1	E669	-9	-9	-9	-9	...	0	1	0	0	0	0	0	0	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8696	19696	93	1	1	2	H041	H11-	H11-	Z961	H353	...	0	1	0	0	0	0	1	0	2	
8697	19697	93	1	1	2	M259	M843	-9	-9	-9	...	0	1	0	0	0	0	0	0	2	
8698	19698	93	1	1	2	H353	H353	H041	H333	Z961	...	0	1	0	0	0	0	0	0	2	
8699	19699	93	2	1	2	H401	H353	H438	H449	H330	...	0	1	0	0	0	0	0	0	2	
8700	19700	93	1	1	2	K589	-9	-9	-9	-9	...	0	1	0	0	0	0	0	0	3	

In [4]:

```
# use of 'AGE' code for NAMCS limits dataset to adults over 17 years old. No need to filter for adults.

df.count()
```

Out[4]:

```
AGE          8701
SEX          8701
RACER        8701
PAYTYPER     8701
DIAG1        8701
DIAG2        8701
DIAG3        8701
DIAG4        8701
DIAG5        8701
DEPRN        8701
NOCHRON      8701
DEPRESS      8701
MENTAL       8701
PSYCHOTH     8701
NOPROVID     8701
PHYS         8701
PHYSASST     8701
NPNMW        8701
RNLPN        8701
MHP          8701
OTHPROV      8701
PROVNONE     8701
SPECCAT      8701
OWNSR        8701
SEQN         8701
```

```
OTHPROV     0
PROVNONE    0
SPECCAT     0
OWNSR       0
SEQN        0
dtype: int64
```

In [6]:
```
df=pd.get_dummies(df, columns=['SEX'], drop_first=True)
```

In [7]:
```
df=df.fillna(0)
```

In [8]:
```
df.rename(columns = {'SEX_2':'SEXML'}, inplace = True)
df.columns
```

Out[8]:
```
Index(['AGE', 'RACER', 'PAYTYPER', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5',
       'DEPRN', 'NOCHRON', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SPECCAT',
       'OWNSR', 'SEQN', 'SEXML'],
      dtype='object')
```

In [9]:
```
df=pd.get_dummies(df, columns=['NOCHRON'], drop_first=False)
```

In [10]:
```
df2=df.fillna(0)
```

In [11]:
```
s=df2.pop('NOCHRON_0')
s=df2.pop('NOCHRON_2')
df2.columns
```

Out[11]:
```
Index(['AGE', 'RACER', 'PAYTYPER', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5',
       'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SPECCAT',
       'OWNSR', 'SEQN', 'SEXML', 'NOCHRON_1'],
      dtype='object')
```

In [12]:
```
df2=pd.get_dummies(df2, columns=['RACER'], drop_first=False)
```

In [13]:
```
df3=df2.fillna(0)
```

In [14]:
```
df3.head()
```

Out[14]:

	AGE	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	...	OTHPROV	PROVNONE	SPECCAT	OWNSR	SEQN	SEXML	NOCHRON_1	RACER_1	RACER_2	RACER_3
0	18	1	Z000	F419	E739	N832	-9	0	1	0	...	0	0	1	1	11000	0	1	1	0	0
1	18	1	L409	-9	-9	-9	-9	0	0	0	...	0	0	3	1	11001	0	1	1	0	0

5 rows × 27 columns

In [15]:
```
df3.rename(columns = {'NOCHRON_1':'NOCHRON','RACER_1':'RACEWH', 'RACER_2':'RACEBL', 'RACER_3':'RACEOT' }, inplace = True)
```

In [16]:
```
df3.head()
```

Out[16]:

	AGE	PAYTYPER	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	...	OTHPROV	PROVNONE	SPECCAT	OWNSR	SEQN	SEXML	NOCHRON	RACEWH	RACEBL	RACEOT
0	18	1	Z000	F419	E739	N832	-9	0	1	0	...	0	0	1	1	11000	0	1	1	0	0
1	18	1	L409	-9	-9	-9	-9	0	0	0	...	0	0	3	1	11001	0	1	1	0	0
2	18	1	L989	-9	-9	-9	-9	0	0	0	...	0	0	3	1	11002	1	1	1	0	0
3	18	1	Z768	L988	-9	-9	-9	0	0	0	...	0	0	3	1	11003	1	1	1	0	0
4	18	1	E669	-9	-9	-9	-9	0	0	0	...	0	0	3	3	11004	0	0	1	0	0

5 rows × 27 columns

In [17]:
```
df4=pd.get_dummies(df3, columns=['PAYTYPER'], drop_first=False)
```

In [18]:
```
df4=df4.replace(r'^\s*$', np.nan, regex=True)
```

In [19]:
```
df5=df4.fillna(0)
```

In [20]:
```
df5.rename(columns = {'PAYTYPER_2': "INSMCARE", 'PAYTYPER_3': 'INSSTATE'}, inplace = True)
```

In [21]:
```
def combinePayColumns(row):
    if row['PAYTYPER_-9'] ==0:
        return '0'
    else:
         return '1'
    if row['PAYTYPER_-8'] ==0:
        return '0'
    else:
        return '1'
    if row['PAYTYPER_5'] ==0:
        return '0'
    else:
        return '1'
    if row['PAYTYPER_6'] ==0:
        return '0'
    else:
        return '1'
    if row['PAYTYPER_7'] ==0:
        return '0'
    else:
        return '1'
```

In [22]:
```
df5['INSNOCH'] =df5.apply(lambda row: combinePayColumns(row), axis =1)
```

In [23]:
```
df5.columns
```

Out[23]:
```
Index(['AGE', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5', 'DEPRN', 'DEPRESS',
       'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN',
       'MHP', 'OTHPROV', 'PROVNONE', 'SPECCAT', 'OWNSR', 'SEQN', 'SEXML',
       'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'PAYTYPER_-9', 'PAYTYPER_-8',
       'PAYTYPER_1', 'INSMCARE', 'INSSTATE', 'PAYTYPER_4', 'PAYTYPER_5',
       'PAYTYPER_6', 'PAYTYPER_7', 'INSNOCH'],
      dtype='object')
```

In [24]:
```
s=df5.pop('PAYTYPER_-9')
s=df5.pop('PAYTYPER_-8')
s=df5.pop('PAYTYPER_5')
s=df5.pop('PAYTYPER_6')
s=df5.pop('PAYTYPER_7')
#dropping individual groups for other payers
df5.columns
```

Out[24]:
```
Index(['AGE', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5', 'DEPRN', 'DEPRESS',
       'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN',
       'MHP', 'OTHPROV', 'PROVNONE', 'SPECCAT', 'OWNSR', 'SEQN', 'SEXML',
       'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'PAYTYPER_1', 'INSMCARE',
       'INSSTATE', 'PAYTYPER_4', 'INSNOCH'],
      dtype='object')
```

In [25]:
```
def combinePrvtColumns(row):
    if row['PAYTYPER_1'] ==0:
        return '0'
    else:
        return '1'
    if row['PAYTYPER_4'] ==0:
        return '0'
    else:
        return '1'
```

In [26]:
```
df5['INSPRVT'] =df5.apply(lambda row: combinePrvtColumns(row), axis =1)
```

In [27]:
```
s=df5.pop('PAYTYPER_1')
s=df5.pop('PAYTYPER_4')
```

In [28]:
```
df5.head()
```

Out[28]:

	AGE	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	PSYCHOTH	...	SEQN	SEXML	NOCHRON	RACEWH	RACEBL	RACEOT	INSMCARE	INSSTATE	INSNOCH	INSPRVT
0	18	Z000	F419	E739	N832	-9	0	1	0	0	...	11000	0	1	1	0	0	0	0	0	1
1	18	L409	-9	-9	-9	-9	0	0	0	0	...	11001	0	1	1	0	0	0	0	0	1
2	18	L989	-9	-9	-9	-9	0	0	0	0	...	11002	1	1	1	0	0	0	0	0	1
3	18	Z768	L988	-9	-9	-9	0	0	0	0	...	11003	1	1	1	0	0	0	0	0	1
4	18	E669	-9	-9	-9	-9	0	0	0	0	...	11004	0	0	1	0	0	0	0	0	1

5 rows × 30 columns

In [29]:
```
df6=pd.get_dummies(df5, columns=['SPECCAT', 'OWNSR'], drop_first=False)
df6.head()
```

Out[29]:

	AGE	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	PSYCHOTH	...	INSNOCH	INSPRVT	SPECCAT_1	SPECCAT_2	SPECCAT_3	OWNSR_-9	OWNSR_-8	OWNSR_1	OWNSR_2	OWNSR_3
0	18	Z000	F419	E739	N832	-9	0	1	0	0	...	0	1	1	0	0	0	0	1	0	0
1	18	L409	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
2	18	L989	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
3	18	Z768	L988	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
4	18	E669	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	0	0	1

5 rows × 36 columns

In [30]:
```
df6.rename(columns = {'SPECCAT_1':'SPPRC', 'SPECCAT_2':'SPSUR', 'SPECCAT_3':'SPMEC', 'OWNSR_-9': "OWNMISS", 'OWNSR_-8': 'OWNUNKN', 'OWNSR_1': 'OWNPHYS', 'OWNSR_2': 'OWNHOSP', 'OWNSR_3': 'OWNINSR'}
#OWNMISS = missing response
```

In [31]:
```
df6.columns
```

Out[31]:
```
Index(['AGE', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5', 'DEPRN', 'DEPRESS',
       'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN',
       'MHP', 'OTHPROV', 'PROVNONE', 'SEQN', 'SEXML', 'NOCHRON', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSMCARE', 'INSSTATE', 'INSNOCH', 'INSPRVT',
       'SPPRC', 'SPSUR', 'SPMEC', 'OWNMISS', 'OWNUNKN', 'OWNPHYS', 'OWNHOSP',
       'OWNINSR'],
      dtype='object')
```

In [32]:
```
df6.head()
```

Out[32]:

	AGE	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	PSYCHOTH	...	INSNOCH	INSPRVT	SPPRC	SPSUR	SPMEC	OWNMISS	OWNUNKN	OWNPHYS	OWNHOSP	OWNINSR
0	18	Z000	F419	E739	N832	-9	0	1	0	0	...	0	1	1	0	0	0	0	1	0	0
1	18	L409	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
2	18	L989	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
3	18	Z768	L988	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	1	0	0
4	18	E669	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	0	0	1

5 rows × 36 columns

In [33]:
```
df0=(df6['OWNMISS'].value_counts(normalize=True))
df0
```

Out[33]:
```
0    0.957936
1    0.042064
Name: OWNMISS, dtype: float64
```

In [34]:
```
s2=df0.round(2)
print('Missing Provider Ownership Type df =', (s[1]*100),'%')
```

```
Missing Provider Ownership Type df = 0 %
```

In [35]:
```
s3 = df6.pop('OWNMISS')
#dropping OWNMISS df since less than 10% missing
```

In [36]:
```
df6.columns
```

Out[36]:
```
Index(['AGE', 'DIAG1', 'DIAG2', 'DIAG3', 'DIAG4', 'DIAG5', 'DEPRN', 'DEPRESS',
       'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN',
       'MHP', 'OTHPROV', 'PROVNONE', 'SEQN', 'SEXML', 'NOCHRON', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSMCARE', 'INSSTATE', 'INSNOCH', 'INSPRVT',
       'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN', 'OWNPHYS', 'OWNHOSP', 'OWNINSR'],
      dtype='object')
```

In [37]:
```
df6['DIAGALL'] = df6['DIAG1'] + " " + df6['DIAG2'] + " " + df6['DIAG3'] + " " + df6['DIAG4'] + " " + df6['DIAG5']

df6.head()
```

Out[37]:

	AGE	DIAG1	DIAG2	DIAG3	DIAG4	DIAG5	DEPRN	DEPRESS	MENTAL	PSYCHOTH	...	INSNOCH	INSPRVT	SPPRC	SPSUR	SPMEC	OWNUNKN	OWNPHYS	OWNHOSP	OWNINSR	DIAGALL
0	18	Z000	F419	E739	N832	-9	0	1	0	0	...	0	1	1	0	0	0	1	0	0	Z000 F419 E739 N832 -9
1	18	L409	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	1	0	0	L409 -9 -9 -9 -9
2	18	L989	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	1	0	0	L989 -9 -9 -9 -9
3	18	Z768	L988	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	1	0	0	Z768 L988 -9 -9 -9
4	18	E669	-9	-9	-9	-9	0	0	0	0	...	0	1	0	0	1	0	0	0	1	E669 -9 -9 -9 -9

5 rows × 36 columns

In [38]:
```
s4=df6.pop('DIAG1')
s4=df6.pop('DIAG2')
s4=df6.pop('DIAG3')
s4=df6.pop('DIAG4')
s4=df6.pop('DIAG5')
```

In [39]:
```
import re
```

In [40]:
```
import numpy as np
df6['DGRP1']=df6['DIAGALL'].str.extract(r'(^[A,B]{1})')
df6['DGRP2']=df6['DIAGALL'].str.extract(r'(^[C,D]{1})')
df6['DGRP3']=df6['DIAGALL'].str.extract(r'(^[E]{1})')
df6['DGRP4']=df6['DIAGALL'].str.extract(r'(^[G]{1})')
df6['DGRP5']=df6['DIAGALL'].str.extract(r'(^[H]{1})')
df6['DGRP6']=df6['DIAGALL'].str.extract(r'(^[I,J]{1})')
df6['DGRP7']=df6['DIAGALL'].str.extract(r'(^[K]{1})')
df6['DGRP8']=df6['DIAGALL'].str.extract(r'(^[L]{1})')
df6['DGRP9']=df6['DIAGALL'].str.extract(r'(^[M]{1})')
df6['DGRP10']=df6['DIAGALL'].str.extract(r'(^[N]{1})')
df6['DGRP11']=df6['DIAGALL'].str.extract(r'(^[O,P]{1})')
df6['DGRP12']=df6['DIAGALL'].str.extract(r'(^[Q]{1})')
df6['DGRP13']=df6['DIAGALL'].str.extract(r'(^[R]{1})')
df6['DGRP14']=df6['DIAGALL'].str.extract(r'(^[S,T]{1})')
df6['DGRP15']=df6['DIAGALL'].str.extract(r'(^[V,W,X,Y]{1})')
df6['DGRP16']=df6['DIAGALL'].str.extract(r'(^[Z]{1})')
df6.columns
```

Out[40]:
```
Index(['AGE', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SEQN',
       'SEXML', 'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'DGRP1', 'DGRP2', 'DGRP3',
       'DGRP4', 'DGRP5', 'DGRP6', 'DGRP7', 'DGRP8', 'DGRP9', 'DGRP10',
       'DGRP11', 'DGRP12', 'DGRP13', 'DGRP14', 'DGRP15', 'DGRP16'],
      dtype='object')
```

In [41]:
```
df7=pd.get_dummies(df6, columns=['DGRP1', 'DGRP2', 'DGRP3', 'DGRP4', 'DGRP5',
       'DGRP6', 'DGRP7', 'DGRP8', 'DGRP9', 'DGRP10', 'DGRP11',
       'DGRP12', 'DGRP13', 'DGRP14','DGRP16'], drop_first=False)
```

In [42]:
```
df7.columns
```

Out[42]:
```
Index(['AGE', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SEQN',
       'SEXML', 'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'DGRP15', 'DGRP1_A',
       'DGRP1_B', 'DGRP2_C', 'DGRP2_D', 'DGRP3_E', 'DGRP4_G', 'DGRP5_H',
       'DGRP6_I', 'DGRP6_J', 'DGRP7_K', 'DGRP8_L', 'DGRP9_M', 'DGRP10_N',
       'DGRP11_O', 'DGRP12_Q', 'DGRP13_R', 'DGRP14_S', 'DGRP14_T', 'DGRP16_Z'],
      dtype='object')
```

In [43]:
```
s5=df7.pop('DGRP1_B')
s5=df7.pop('DGRP2_D')
s5=df7.pop('DGRP6_J')
s5=df7.pop('DGRP14_T')
#dropping first columns for each group
df7.columns
```

Out[43]:
```
Index(['AGE', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SEQN',
       'SEXML', 'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'DGRP15', 'DGRP1_A',
       'DGRP2_C', 'DGRP3_E', 'DGRP4_G', 'DGRP5_H', 'DGRP6_I', 'DGRP7_K',
       'DGRP8_L', 'DGRP9_M', 'DGRP10_N', 'DGRP11_O', 'DGRP12_Q', 'DGRP13_R',
       'DGRP14_S', 'DGRP16_Z'],
      dtype='object')
```

In [44]:
```
df7['temp']=df7['DIAGALL'].str.extract(r'(^[F32*,F33*,F43*])')

df7.columns
```

Out[44]:
```
Index(['AGE', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SEQN',
       'SEXML', 'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'DGRP15', 'DGRP1_A',
       'DGRP2_C', 'DGRP3_E', 'DGRP4_G', 'DGRP5_H', 'DGRP6_I', 'DGRP7_K',
       'DGRP8_L', 'DGRP9_M', 'DGRP10_N', 'DGRP11_O', 'DGRP12_Q', 'DGRP13_R',
       'DGRP14_S', 'DGRP16_Z', 'temp'],
      dtype='object')
```

In [45]:
```
df8=df7.fillna(0)
```

```
df8.head()
```

Out[48]:

	AGE	DEPRN	DEPRESS	MENTAL	PSYCHOTH	NOPROVID	PHYS	PHYSASST	NPNMW	RNLPN	...	DGRP8	DGRP9	DGRP10	DGRP11	DGRP12	DGRP13	DGRP14	DGRP16	temp	DI
0	18	0	1	0	0	0	1	0	0	1	...	0	0	0	0	0	0	0	1	0	1
1	18	0	0	0	0	0	1	0	0	0	...	1	0	0	0	0	0	0	0	0	0
2	18	0	0	0	0	0	1	0	0	0	...	1	0	0	0	0	0	0	0	0	0
3	18	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
4	18	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 49 columns

In [49]:
```
df8.value_counts('DI')
```

Out[49]:
```
DI
0    7357
1    1344
dtype: int64
```

In [50]:
```
s6=df8.pop('temp')
```

In [51]:
```
df8.columns
```

Out[51]:
```
Index(['AGE', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE', 'SEQN',
       'SEXML', 'NOCHRON', 'RACEWH', 'RACEBL', 'RACEOT', 'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'DGRP15', 'DGRP1', 'DGRP2',
       'DGRP3', 'DGRP4', 'DGRP5', 'DGRP6', 'DGRP7', 'DGRP8', 'DGRP9', 'DGRP10',
       'DGRP11', 'DGRP12', 'DGRP13', 'DGRP14', 'DGRP16', 'DI'],
      dtype='object')
```

In [60]:
```
df9=df8.reindex(columns=['SEQN','AGE', 'SEXML', 'RACEWH', 'RACEBL', 'RACEOT','NOPROVID', 'PHYS',
       'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE',  'INSMCARE',
       'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC', 'OWNUNKN',
       'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'NOCHRON',  'DGRP1', 'DGRP2',
       'DGRP3', 'DGRP4', 'DGRP5', 'DGRP6', 'DGRP7', 'DGRP8', 'DGRP9', 'DGRP10',
       'DGRP11', 'DGRP12', 'DGRP13', 'DGRP14', 'DGRP15','DGRP16', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH','DI'])
```

In [61]:
```
df9.columns
```

Out[61]:
```
Index(['SEQN', 'AGE', 'SEXML', 'RACEWH', 'RACEBL', 'RACEOT', 'NOPROVID',
       'PHYS', 'PHYSASST', 'NPNMW', 'RNLPN', 'MHP', 'OTHPROV', 'PROVNONE',
       'INSMCARE', 'INSSTATE', 'INSNOCH', 'INSPRVT', 'SPPRC', 'SPSUR', 'SPMEC',
       'OWNUNKN', 'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'DIAGALL', 'NOCHRON',
       'DGRP1', 'DGRP2', 'DGRP3', 'DGRP4', 'DGRP5', 'DGRP6', 'DGRP7', 'DGRP8',
       'DGRP9', 'DGRP10', 'DGRP11', 'DGRP12', 'DGRP13', 'DGRP14', 'DGRP15',
       'DGRP16', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'DI'],
      dtype='object')
```

In [63]:
```
df9.to_csv('namcs_features_FINAL.csv', encoding='utf-8', index=False)
```

Finding Missing values in namcs.csv

```
print('% of missing values in AGER:',len(df[df.AGER == -9])/len(df.AGER)*100)
print('% of missing values in SEX:',len(df[df.SEX == -9])/len(df.SEX)*100)
print('% of missing values in RACEUN:',len(df[df.RACEUN == -9])/len(df.RACEUN)*100)
print('% of missing values in PAYTYPER:',len(df[df.PAYTYPER == -9])/len(df.PAYTYPER)*100)
print('% of missing values in DIAG1:',len(df[df.DIAG1 == '-9'])/len(df.DIAG1)*100)
print('% of missing values in DIAG2:',len(df[df.DIAG2 == '-9'])/len(df.DIAG2)*100)
print('% of missing values in DIAG3:',len(df[df.DIAG3 == '-9'])/len(df.DIAG3)*100)
print('% of missing values in DIAG4:',len(df[df.DIAG4 == '-9'])/len(df.DIAG4)*100)
print('% of missing values in DIAG5:',len(df[df.DIAG5 == '-9'])/len(df.DIAG5)*100)
print('% of missing values in DEPRN:',len(df[df.DEPRN == '-9'])/len(df.DEPRN)*100)
print('% of missing values in NOCHRON:',len(df[df.NOCHRON == -9])/len(df.NOCHRON)*100)
print('% of missing values in DEPRESS:',len(df[df.DEPRESS == -9])/len(df.DEPRESS)*100)
print('% of missing values in MENTAL:',len(df[df.MENTAL == -9])/len(df.MENTAL)*100)
print('% of missing values in PSYCHOTH:',len(df[df.PSYCHOTH == -9])/len(df.PSYCHOTH)*100)
print('% of missing values in NOPROVID:',len(df[df.NOPROVID == -9])/len(df.NOPROVID)*100)
print('% of missing values in PHYS:',len(df[df.PHYS == -9])/len(df.PHYS)*100)
print('% of missing values in PHYSASST:',len(df[df.PHYSASST == -9])/len(df.PHYSASST)*100)
print('% of missing values in NPNWM:',len(df[df.NPNMW == -9])/len(df.NPNMW)*100)
print('% of missing values in RNLPN:',len(df[df.RNLPN == -9])/len(df.RNLPN)*100)
print('% of missing values in MHP:',len(df[df.MHP == -9])/len(df.MHP)*100)
print('% of missing values in OTHPROV:',len(df[df.OTHPROV == -9])/len(df.OTHPROV)*100)
print('% of missing values in PROVNONE:',len(df[df.PROVNONE == -9])/len(df.PROVNONE)*100)
print('% of missing values in SPECCAT:',len(df[df.SPECCAT == -9])/len(df.SPECCAT)*100)
print('% of missing values in OWNSR:',len(df[df.OWNSR == -9])/len(df.OWNSR)*100)
```

NHANES Cleaning and Feature Engineering

```
import numpy as np
import pandas as pd
df=pd.read_csv('NHANES_merged.csv', usecols=['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060', 'DPQ070', 'DPQ080', 'DPQ090', 'RIAGENDR', '
                                             'DMDEDUC2', 'DLQ140', 'DLQ150', 'DLQ170','MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ16
                                             'MCQ160F', 'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ160L', 'MCQ170L', 'MCQ500', 'MCQ510A', 'MCQ510B', 'MC
                                             'MCQ510C', 'MCQ510D', 'MCQ510E', 'MCQ510F', 'MCQ220', 'INDHHIN2', 'HIQ031A', 'HIQ031B', 'HIQ031D', 'HIQ031AA',
                na_values = ['na', '-', '.', '', 'NaN'])
```

```
df.head()
```

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	DLQ140	DLQ150	DLQ170	HIQ031A	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HI
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.0	2.0	1.0	0.0	15.0	17.0	0	0	
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	14.0	0.0	0.0	0	0	
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	15.0	0.0	0	0	
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	1.0	0.0	15.0	17.0	0	0	
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	0.0	0.0	0	21	

5 rows × 48 columns

```
#remove SEQN values (inf) not found in source dfhh for variable (confirmed total of 9254 of valid dfhh in NHANES codebook [a.taylor])
s=df.drop(range(9254,9668))
```

```
#limiting dataset to age adults only per proposal
df=df[df['RIDAGEYR'] > 17]
df.count()
```

[4]:

```
SEQN        5856
DPQ010      5533
DPQ020      5533
DPQ030      5533
DPQ040      5533
DPQ050      5533
DPQ060      5533
DPQ070      5533
DPQ080      5533
DPQ090      5533
RIAGENDR    5856
RIDAGEYR    5856
RIDRETH3    5856
DMDEDUC3    5856
DMDEDUC2    5856
INDHHIN2    5856
MCQ010      5856
MCQ080      5856
MCQ160A     5856
MCQ160B     5856
MCQ160C     5856
MCQ160D     5856
MCQ160E     5856
MCQ160F     5856
MCQ160M     5856
MCQ160G     5856
MCQ160K     5856
MCQ160O     5856
MCQ160L     5856
MCQ170L     5856
MCQ500      5856
MCQ510A     5856
MCQ510B     5856
MCQ510C     5856
MCQ510D     5856
MCQ510E     5856
MCQ510F     5856
MCQ220      5856
DLQ140      5856
DLQ150      5856
DLQ170      5856
HIQ031A     5856
HIQ031B     5856
HIQ031D     5856
HIQ031E     5856
HIQ031H     5856
HIQ031I     5856
HIQ031AA    5856
dtype: int64
```

```
5]: df.isna().sum()
```

```
5]: SEQN          0
    DPQ010      323
    DPQ020      323
    DPQ030      323
    DPQ040      323
    DPQ050      323
    DPQ060      323
    DPQ070      323
    DPQ080      323
    DPQ090      323
    RIAGENDR      0
    RIDAGEYR      0
    RIDRETH3      0
    DMDEDUC3      0
    DMDEDUC2      0
    INDHHIN2      0
    MCQ010        0
    MCQ080        0
    MCQ160A       0
    MCQ160B       0
    MCQ160C       0
    MCQ160D       0
    MCQ160E       0
    MCQ160F       0
    MCQ160M       0
    MCQ160G       0
    MCQ160K       0
    MCQ160O       0
    MCQ160L       0
    MCQ170L       0
    MCQ500        0
    MCQ510A       0
```

```
DLQ150        0
DLQ170        0
HIQ031A       0
HIQ031B       0
HIQ031D       0
HIQ031E       0
HIQ031H       0
HIQ031I       0
HIQ031AA      0
dtype: int64
```

6]:
```
#replacing blank values with nan
df.replace(r'^\s*$', np.nan, regex=True)
```

6]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	DLQ140	DLQ150	DLQ170	HIQ031A	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HIQ031I	HIQ031AA
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.0	2.0	1.0	0.0	15.0	17.0	0	0	0	0
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	14.0	0.0	0.0	0	0	0	0
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	15.0	0.0	0	0	0	0
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	1.0	0.0	15.0	17.0	0	0	0	0
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	0.0	0.0	0	21	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
36913	102751.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	5.0	2.0	0.0	14.0	0.0	0.0	0	0	0	0
36931	102804.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	4.0	2.0	1.0	0.0	0.0	0.0	0	0	0	0
36941	102825.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	1.0	3.0	0.0	15.0	0.0	0	0	0	0
36985	102940.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	4.0	2.0	1.0	0.0	15.0	0.0	0	0	0	0
36989	102950.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	2.0	1.0	0.0	0.0	0.0	0	21	0	0

5856 rows × 48 columns

7]:
```
df2=df.fillna(0)
```

8]:
```
df2.isna().sum()
```

8]:
```
SEQN        0
DPQ010      0
DPQ020      0
DPQ030      0
DPQ040      0
DPQ050      0
DPQ060      0
```

```
HIQ031A     0
HIQ031B     0
HIQ031D     0
HIQ031E     0
HIQ031H     0
HIQ031I     0
HIQ031AA    0
dtype: int64
```

In [9]:
```
df2.head()
```

Out[9]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	DLQ140	DLQ150	DLQ170	HIQ031A	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HIQ031I	HIQ031AA
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.0	2.0	1.0	0.0	15.0	17.0	0	0	0	0
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	14.0	0.0	0.0	0	0	0	0
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	15.0	0.0	0	0	0	0
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	1.0	0.0	15.0	17.0	0	0	0	0
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.0	2.0	0.0	0.0	0.0	0.0	0	21	0	0

5 rows × 48 columns

In [10]:
```
df2=pd.get_dummies(df2, columns=['RIAGENDR'], drop_first=True)
```

In [11]:
```
df3=df2.fillna(0)
```

In [12]:
```
df3.rename(columns = {'RIAGENDR_2.0':'SEXML', 'RIDAGEYR': 'AGE'}, inplace = True)
df3.columns
```

Out[12]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'RIDRETH3', 'DMDEDUC3', 'DMDEDUC2',
       'INDHHIN2', 'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C',
       'MCQ160D', 'MCQ160E', 'MCQ160F', 'MCQ160M', 'MCQ160G', 'MCQ160K',
       'MCQ160O', 'MCQ160L', 'MCQ170L', 'MCQ500', 'MCQ510A', 'MCQ510B',
       'MCQ510C', 'MCQ510D', 'MCQ510E', 'MCQ510F', 'MCQ220', 'DLQ140',
       'DLQ150', 'DLQ170', 'HIQ031A', 'HIQ031B', 'HIQ031D', 'HIQ031E',
       'HIQ031H', 'HIQ031I', 'HIQ031AA', 'SEXML'],
      dtype='object')
```

In [13]:
```
df4=pd.get_dummies(df3, columns=['RIDRETH3'], drop_first=False, dtype=str)
df4.columns
```

Out[13]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'DMDEDUC3', 'DMDEDUC2', 'INDHHIN2',
       'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D',
       'MCQ160E', 'MCQ160F', 'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O',
       'MCQ160L', 'MCQ170L', 'MCQ500', 'MCQ510A', 'MCQ510B', 'MCQ510C',
       'MCQ510D', 'MCQ510E', 'MCQ510F', 'MCQ220', 'DLQ140', 'DLQ150', 'DLQ170',
       'HIQ031A', 'HIQ031B', 'HIQ031D', 'HIQ031E', 'HIQ031H', 'HIQ031I',
       'HIQ031AA', 'SEXML', 'RIDRETH3_1.0', 'RIDRETH3_2.0', 'RIDRETH3_3.0',
       'RIDRETH3_4.0', 'RIDRETH3_6.0', 'RIDRETH3_7.0'],
```

```
       dtype='object')
```

In [14]:
```
df4['RACEOT']=df4['RIDRETH3_1.0'] + " " + df4['RIDRETH3_2.0'] + " " + df4['RIDRETH3_6.0'] + " " + df4['RIDRETH3_7.0']
```

In [15]:
```
s2=df4.pop('RIDRETH3_1.0')
s2=df4.pop('RIDRETH3_2.0')
s2=df4.pop('RIDRETH3_6.0')
s2=df4.pop('RIDRETH3_7.0')
#dropping individual groups for other races
df4.columns
```

Out[15]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'DMDEDUC3', 'DMDEDUC2', 'INDHHIN2',
       'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D',
       'MCQ160E', 'MCQ160F', 'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O',
       'MCQ160L', 'MCQ170L', 'MCQ500', 'MCQ510A', 'MCQ510B', 'MCQ510C',
       'MCQ510D', 'MCQ510E', 'MCQ510F', 'MCQ220', 'DLQ140', 'DLQ150', 'DLQ170',
       'HIQ031A', 'HIQ031B', 'HIQ031D', 'HIQ031E', 'HIQ031H', 'HIQ031I',
       'HIQ031AA', 'SEXML', 'RIDRETH3_3.0', 'RIDRETH3_4.0', 'RACEOT'],
      dtype='object')
```

In [16]:
```
df4.rename(columns = {'RIDRETH3_3.0':'RACEWH', 'RIDRETH3_4.0':'RACEBL'}, inplace = True)
```

In [17]:
```
df4.head()
```

Out[17]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HIQ031I	HIQ031AA	SEXML	RACEWH	RACEBL	RACEOT
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	15.0	17.0	0	0	0	0	1		1	
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0	0	0	0	0			1
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	15.0	0.0	0	0	0	0	1			1
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	15.0	17.0	0	0	0	0	1		1	
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0	21	0	0	0			1

5 rows × 50 columns

In [18]:
```
df5=df4.replace(r'^\s*$', np.nan, regex=True)
```

In [19]:
```
df5=df5.fillna(0)
```

In [20]:
```
dfhO=(df5['INDHHIN2'].value_counts(normalize=True))
dfhO
```

Out[20]:
```
15.0    0.169570
6.0     0.099727
7.0     0.097848
14.0    0.086236
8.0     0.066940
0.0     0.059597
5.0     0.059085
4.0     0.057889
9.0     0.057719
10.0    0.047302
3.0     0.045253
12.0    0.037910
2.0     0.028347
1.0     0.028176
99.0    0.024761
77.0    0.020833
13.0    0.012807
Name: INDHHIN2, dtype: float64
```

In [21]:
```
# missing (2%) and refused to answer(2.5%); values <10%; will convert to 0 according to proposal
```

In [22]:
```
#missing not calculated for the co-morbid conditions since these would not be a required field for all responders
df6= df5.replace({'MCQ010':{7:0, 9:0, 2:0}, 'MCQ080':{7:0, 9:0, 2:0}, 'MCQ160A':{7:0, 9:0, 2:0}, 'MCQ160B':{7:0, 9:0, 2:0}, 'MCQ160C':{7:0, 9:0, 2:0}, 'MCQ160D':{7:0, 9:0, 2:0}, 'MCQ160E':{7:0, 9:0, 2:0},
                  'MCQ160F':{7:0, 9:0, 2:0}, 'MCQ160M':{7:0, 9:0, 2:0}, 'MCQ160G':{7:0, 9:0, 2:0}, 'MCQ160K':{7:0, 9:0, 2:0}, 'MCQ160O':{7:0, 9:0, 2:0}, 'MCQ160L':{7:0, 9:0, 2:0}, 'MCQ170L':{7:0, 9:0, 2:0}, 'MCQ500':{7:0,
                  'MCQ510C':{3:1}, 'MCQ510D':{3:1}, 'MCQ510E':{5:1}, 'MCQ510F':{6:1}, 'MCQ220':{7:0, 9:0, 2:0}})
```

In [23]:
```
dfI=(df6[['HIQ031A', 'HIQ031B', 'HIQ031D', 'HIQ031E', 'HIQ031H', 'HIQ031I', 'HIQ031AA']].apply(pd.Series.value_counts,normalize=True))
dfI
```

Out[23]:

	HIQ031A	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HIQ031I	HIQ031AA
0.0	0.506148	0.731387	0.824966	0.999658	0.960212	0.962432	0.999829
14.0	0.486339	NaN	NaN	NaN	NaN	NaN	NaN
15.0	NaN	0.268613	NaN	NaN	NaN	NaN	NaN
17.0	NaN	NaN	0.175034	NaN	NaN	NaN	NaN
18.0	NaN	NaN	NaN	0.000342	NaN	NaN	NaN
21.0	NaN	NaN	NaN	NaN	0.039788	NaN	NaN
22.0	NaN	NaN	NaN	NaN	NaN	0.037568	NaN
40.0	NaN	NaN	NaN	NaN	NaN	NaN	0.000171
77.0	0.000512	NaN	NaN	NaN	NaN	NaN	NaN
99.0	0.007001	NaN	NaN	NaN	NaN	NaN	NaN

In [24]:
```
# missing (0.7% and refused to answer(0.05%); values <10%; will convert to 0 according to proposal
```

In [25]:
```
df7=df6.replace({'HIQ031A':{14:1, 99:0, 77:0}, 'HIQ031B':{15:1}, 'HIQ031D':{17:1}, 'HIQ031AA':{40:1}, 'HIQ031E':{18:1}, 'HIQ031H':{21:1}, 'HIQ031I':{22:1}, 'INDHHIN2':{77:0, 99:0}})
```

In [26]:
```
df7.head()
```

Out[26]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	HIQ031B	HIQ031D	HIQ031E	HIQ031H	HIQ031I	HIQ031AA	SEXML	RACEWH	RACEBL	RACEOT
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	0	0	0	0	1	0	1	0
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0	0	0	0	0	0	0	1
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0	0	0	0	1	0	0	1
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	0	0	0	0	1	0	1	0
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0	1	0	0	0	0	0	1

5 rows × 50 columns

In [27]:
```
def combineColumns(row):
    if row['HIQ031D'] ==0:
        return '0'
    else:
        return '1'

    if row['HIQ031E'] ==0:
        return '0'
    else:
        return '1'

    if row['HIQ031H'] ==0:
        return '0'
    else:
        return '1'

    if row['HIQ031I'] ==0:
        return '0'
    else:
        return '1'
```

In [28]:
```
df7['INSSTATE'] =df7.apply(lambda row: combineColumns(row), axis =1)
```

```
dtype='object')
```

In [30]:
```
df7.rename(columns = {'HIQ031A':'INSPRVT', 'HIQ031B':'INSMCARE', 'HIQ031AA':'INSNOCH'}, inplace = True)
```

In [31]:
```
dfD=(df7['DMDEDUC2'].value_counts(normalize=True))
dfD
```

Out[31]:
```
4.0    0.303620
5.0    0.228142
3.0    0.226264
2.0    0.108948
1.0    0.081796
0.0    0.049010
9.0    0.001878
7.0    0.000342
Name: DMDEDUC2, dtype: float64
```

In [32]:
```
dfE=(df7['DMDEDUC3'].value_counts(normalize=True))
dfE
```

Out[32]:
```
0.0     0.950990
13.0    0.024249
15.0    0.011954
11.0    0.007172
12.0    0.002220
10.0    0.001537
14.0    0.000683
9.0     0.000683
66.0    0.000512
Name: DMDEDUC3, dtype: float64
```

In [33]:
```
# missing, unknown or refused for Monthly Income (INDHHIN2), Education levels (DMDEDUC2, DMDEDUC3) values <10%, will convert to 0 according to proposal.
```

In [34]:
```
df8=df7.replace({'DMDEDUC3':{55.0:1, 66.0:1, 77.0:0, 99.0:0, 15.0:4, 14.0:3, 13.0:3, 12.0:3, 11.0:2, 10.0:2, 9.0:2, 8.0:1, 7.0:1, 6.0:1, 5.0:1, 4.0:1, 3.0:1, 2.0:1}})

df8.value_counts("DMDEDUC3")
```

Out[34]:
```
DMDEDUC3
0.0    5569
3.0     159
4.0      70
2.0      55
1.0       3
dtype: int64
```

```
dtype: int64
```

In [35]:
```
dfP=(df8[['DLQ140','DLQ150','DLQ170']].apply(pd.Series.value_counts,normalize=True))
dfP
```

Out[35]:

	DLQ140	DLQ150	DLQ170
0.0	0.016052	0.016052	0.481387
1.0	0.054474	0.099556	0.248805
2.0	0.070014	0.883538	0.080772
3.0	0.081113	NaN	0.186988
4.0	0.310451	NaN	NaN
5.0	0.465335	NaN	NaN
7.0	0.000171	NaN	NaN
9.0	0.002391	0.000854	0.002049

In [36]:
```
# missing (0.2%) and refused to answer(2%); values <10%; will convert to 0 according to proposal
```

In [37]:
```
dfP=(df8[['DPQ010','DPQ020','DPQ030','DPQ040','DPQ050','DPQ060','DPQ070','DPQ080','DPQ090']].apply(pd.Series.value_counts,normalize=True))
dfP
```

Out[37]:

	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090
0.0	0.777664	0.790301	0.662910	0.571721	0.781421	0.854167	0.854850	0.906079	0.966359
1.0	0.139344	0.141906	0.194843	0.275273	0.132855	0.098190	0.088286	0.056011	0.023224
2.0	0.047814	0.038251	0.065403	0.078040	0.047131	0.025102	0.026639	0.020492	0.005464
3.0	0.033811	0.028518	0.075649	0.073770	0.037739	0.021175	0.029372	0.016223	0.004098
7.0	0.000683	0.000342	0.000512	0.000512	0.000342	0.000512	0.000342	0.000342	0.000342
9.0	0.000683	0.000683	0.000683	0.000683	0.000512	0.000854	0.000512	0.000854	0.000512

In [38]:
```
# 'dont know'(9.0) values for each question range from 0.5-0.9% and refused (7.0) values for each question range from 0.4-0.7%; values <10%, will convert to 0 according to proposal
```

In [37]:
```
dfP=(df8[['DPQ010','DPQ020','DPQ030','DPQ040','DPQ050','DPQ060','DPQ070','DPQ080','DPQ090']].apply(pd.Series.value_counts,normalize=True))
dfP
```

Out[37]:

	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090
0.0	0.777664	0.790301	0.662910	0.571721	0.781421	0.854167	0.854850	0.906079	0.966359
1.0	0.139344	0.141906	0.194843	0.275273	0.132855	0.098190	0.088286	0.056011	0.023224
2.0	0.047814	0.038251	0.065403	0.078040	0.047131	0.025102	0.026639	0.020492	0.005464
3.0	0.033811	0.028518	0.075649	0.073770	0.037739	0.021175	0.029372	0.016223	0.004098
7.0	0.000683	0.000342	0.000512	0.000512	0.000342	0.000512	0.000342	0.000342	0.000342
9.0	0.000683	0.000683	0.000683	0.000683	0.000512	0.000854	0.000512	0.000854	0.000512

In [38]:
```
# 'dont know'(9.0) values for each question range from 0.5-0.9% and refused (7.0) values for each question range from 0.4-0.7%; values <10%, will convert to 0 according to proposal
```

In [39]:
```
df9=df8.replace({'DPQ010':{7:0, 9:0},'DPQ020':{7:0, 9:0},'DPQ030':{7:0, 9:0},'DPQ040':{7:0, 9:0}, 'DPQ050':{7:0, 9:0}, 'DPQ060':{7:0, 9:0}, 'DPQ070':{7:0, 9:0},
                                 'DPQ080':{7:0, 9:0}, 'DPQ090':{7:0, 9:0} ,'DLQ140':{7:0, 9:0}, 'DLQ150':{7:0, 9:0}, 'DLQ170':{7:0, 9:0}})
```

In [40]:
```
def combineColumns(row):
    if row['DMDEDUC3'] ==1:
        return '1'

    if row['DMDEDUC3'] ==2:
        return '2'

    if row['DMDEDUC3'] ==3:
        return '3'

    if row['DMDEDUC3'] ==4:
        return '4'

    if row['DMDEDUC2'] ==1:
        return '1'

    if row['DMDEDUC2'] ==2:
        return '2'

    if row['DMDEDUC2'] ==3:
        return '3'

    if row['DMDEDUC2'] ==4:
        return '4'
    else:
        return '0'
```

In [44]:
```
s4=df9.pop('MCQ160L')
s4=df9.pop('MCQ170L')
s4=df9.pop('MCQ500')
s4=df9.pop('MCQ510A')
s4=df9.pop('MCQ510B')
s4=df9.pop('MCQ510C')
s4=df9.pop('MCQ510D')
s4=df9.pop('MCQ510E')
s4=df9.pop('MCQ510F')
s4=df9.pop('DMDEDUC3')
s4=df9.pop('DMDEDUC2')
#dropping individual groups for other education and liver conditions
df9.columns
```

Out[44]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080',
       'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F',
       'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ220', 'DLQ140',
       'DLQ150', 'DLQ170', 'INSPRVT', 'INSMCARE', 'INSNOCH', 'SEXML', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSSTATE', 'EDUC', 'MCQ700'],
      dtype='object')
```

In [45]:
```
# Create new column for PHQ total

df9['PHQSCR'] = df9[['DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060', 'DPQ070', 'DPQ080', 'DPQ090']].sum(axis=1)

df9.head()
```

Out[45]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	INSMCARE	INSNOCH	SEXML	RACEWH	RACEBL	RACEOT	INSSTATE	EDUC	MCQ70
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0	1	0	1	0	1	2	
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0	0	0	0	1	0	4	
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0	1	0	0	1	0	1	
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0	1	0	1	0	1	4	
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0	0	0	0	1	0	0	

5 rows × 39 columns

In [46]:
```
def DepInd(row):
    if row['PHQSCR'] >= 10:
        return '1'
    elif row['DLQ140'] < 3:
        return '1'
    elif row['DLQ150'] == 1:
        return '1'
    elif row['DLQ170'] == 2:
        return '1'
    else:
        return '0'
```

In [47]:
```
df9['DI'] =df9.apply(lambda row: DepInd(row), axis =1)
```

In [48]:
```
df9.head()
```

Out[48]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	...	INSNOCH	SEXML	RACEWH	RACEBL	RACEOT	INSSTATE	EDUC	MCQ700	PHQSCR	DI
0	93705.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	1	0	1	0	1	2	0	0.0	0
1	93706.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	1	0	4	0	0.0	0
2	93708.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	1	0	0	1	0	1	0	0.0	0
3	93709.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	1	0	1	0	1	4	0	0.0	1
4	93711.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	1	0	0	0	2.0	0

5 rows × 40 columns

In [62]:
```
df9.columns
```

Out[62]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080',
       'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F',
       'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ220', 'DLQ140',
       'DLQ150', 'DLQ170', 'INSPRVT', 'INSMCARE', 'INSNOCH', 'SEXML', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSSTATE', 'EDUC', 'MCQ700', 'PHQSCR', 'DI'],
      dtype='object')
```

In [62]:
```
df9.columns
```

Out[62]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080',
       'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F',
       'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ220', 'DLQ140',
       'DLQ150', 'DLQ170', 'INSPRVT', 'INSMCARE', 'INSNOCH', 'SEXML', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSSTATE', 'EDUC', 'MCQ700', 'PHQSCR', 'DI'],
      dtype='object')
```

In [71]:
```
df9.reindex(columns=['SEQN', 'AGE', 'SEXML','RACEWH', 'RACEBL', 'RACEOT','INDHHIN2', 'INSPRVT', 'INSMCARE',
       'INSNOCH', 'INSSTATE', 'EDUC','MCQ010', 'MCQ080',  'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F','MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ700',
       'MCQ220', 'DLQ140', 'DLQ150', 'DLQ170', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060', 'DPQ070', 'DPQ080', 'DPQ090', 'PHQSCR', 'DI'], fill_value = 0)
```

Out[71]:

	SEQN	AGE	SEXML	RACEWH	RACEBL	RACEOT	INDHHIN2	INSPRVT	INSMCARE	INSNOCH	...	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	PHQSCR	DI
0	93705.0	66.0	1	0	1	0	3.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	93706.0	18.0	0	0	0	1	0.0	1.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	93708.0	66.0	1	0	0	1	6.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	93709.0	75.0	1	0	1	0	2.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	93711.0	56.0	0	0	0	1	15.0	0.0	0.0	0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
36913	102751.0	19.0	1	1	0	0	12.0	1.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36931	102804.0	30.0	0	0	0	1	8.0	0.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36941	102825.0	80.0	1	1	0	0	7.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
36985	102940.0	77.0	1	1	0	0	0.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36989	102950.0	64.0	0	0	1	0	7.0	0.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

5856 rows × 40 columns

In [72]:
```
df9.index.astype(int, copy=True)
```

In [62]:
```
df9.columns
```

Out[62]:
```
Index(['SEQN', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
       'DPQ070', 'DPQ080', 'DPQ090', 'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080',
       'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F',
       'MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ220', 'DLQ140',
       'DLQ150', 'DLQ170', 'INSPRVT', 'INSMCARE', 'INSNOCH', 'SEXML', 'RACEWH',
       'RACEBL', 'RACEOT', 'INSSTATE', 'EDUC', 'MCQ700', 'PHQSCR', 'DI'],
      dtype='object')
```

In [71]:
```
df9.reindex(columns=['SEQN', 'AGE', 'SEXML','RACEWH', 'RACEBL', 'RACEOT','INDHHIN2', 'INSPRVT', 'INSMCARE',
       'INSNOCH', 'INSSTATE', 'EDUC','MCQ010', 'MCQ080',  'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F','MCQ160M', 'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ700',
       'MCQ220', 'DLQ140', 'DLQ150', 'DLQ170', 'DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060', 'DPQ070', 'DPQ080', 'DPQ090', 'PHQSCR', 'DI'], fill_value = 0)
```

Out[71]:

	SEQN	AGE	SEXML	RACEWH	RACEBL	RACEOT	INDHHIN2	INSPRVT	INSMCARE	INSNOCH	...	DPQ020	DPQ030	DPQ040	DPQ050	DPQ060	DPQ070	DPQ080	DPQ090	PHQSCR	DI
0	93705.0	66.0	1	0	1	0	3.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	93706.0	18.0	0	0	0	1	0.0	1.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	93708.0	66.0	1	0	0	1	6.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	93709.0	75.0	1	0	1	0	2.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	93711.0	56.0	0	0	0	1	15.0	0.0	0.0	0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
36913	102751.0	19.0	1	1	0	0	12.0	1.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36931	102804.0	30.0	0	0	0	1	8.0	0.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36941	102825.0	80.0	1	1	0	0	7.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
36985	102940.0	77.0	1	1	0	0	0.0	0.0	1.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
36989	102950.0	64.0	0	0	1	0	7.0	0.0	0.0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

5856 rows × 40 columns

In [72]:
```
df9.index.astype(int, copy=True)
```

Out[72]:
```
Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,
                9,
            ...
            36849, 36888, 36897, 36905, 36907, 36913, 36931, 36941, 36985,
            36989],
           dtype='int64', length=5856)
```

In [73]:
```
df9.to_csv('nhanes_features_FINAL.csv', encoding='utf-8', index=False)
```

Descriptive Statistics of NHANES

NHANES DATASET

```
# Reading the data from CSV file
import pandas as pd
import numpy as np
import statistics as st

df = pd.read_csv("nhanes_features_FINAL.csv")
print(df.shape)
print(df.info())
```

```
df.head()
```

Descriptive Statistics of NHANES

```
#Descriptive statistics for continuous data
df1= df[['AGE','PHQSCR']]
df1.describe()
```

```
print('median of AGE:', df.loc[:,'AGE'].median())
print('median of PHQ_SCORE:', df.loc[:,'PHQSCR'].median())
```

```
# Calculating descriptive statistics for int datatypes
cols=set(df.columns)- {'SEQN'}
df2 = df[list(cols)]
```

```
df2.mode().T
```

```
df2.std()
```

```
df2.var()
```

```
#Calculating IQR
from scipy.stats import iqr
def find_iqr(x):
    return np.subtract(*np.percentile(x, [75, 25]))
df.loc[:, ~df.columns.isin(['SEQN'])].apply(find_iqr)
```

```
df2.skew()
```

```
df2.kurt()
```

Descriptive Statistics of NAMCS

NAMCS DATASET

```
# Reading the data from CSV file
import pandas as pd
import numpy as np
import statistics as st

df = pd.read_csv("namcs_features_FINAL.csv")
print(df.shape)
print(df.info())
```

```
df.head()
```

Descriptive Statistics of NAMCS

```
#Descriptive statistics for continuous data
df1= df['AGE']
df1.describe()
```

```
print('median of AGE:', df.loc[:,'AGE'].median())
```

```
# Calculating descriptive statistics for int datatypes
cols=set(df.columns)- {'SEQN','DIAGALL'}
df2 = df[list(cols)]
```

```
df2.mode().T
```

```
df2.std()
```

```
df2.var()
```

```
df2.skew()
```

```
df2.kurt()
```

```
#Calculating IQR
from scipy.stats import iqr
def find_iqr(x):
    return np.subtract(*np.percentile(x, [75, 25]))
df.loc[:, ~df.columns.isin(['SEQN','DIAGALL'])].apply(find_iqr)
```

EXPLORATORY DATA ANALYSIS

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
data_namcs=pd.read_csv('namcs_features_FINAL.csv')
```

```
data_namcs.rename(columns={'AGE': 'Age','SEXML': 'Gender','RACEWH': 'White', 'RACEBL': 'Black', 'RACEOT': 'Other Race','SPMEC':
```

```
datanam = data_namcs.drop(['SEQN','Chronic DPRN','DPRN Screen','Therapy Referral','MHP Referral','Infection','Urinary','Womens He
g = sns.heatmap(datanam.corr(),annot=True, annot_kws={"fontsize":15})
g.set_yticklabels(g.get_yticklabels(),size=25, rotation=0)
g.set_xticklabels(g.get_xticklabels(),size=25, rotation=90)
g.set_title('NAMCS Heatmap',size=35)
sns.set(rc = {'figure.figsize':(30,30)})
plt.tight_layout()
plt.savefig('NAMCS_Heatmap.png')
plt.show()
```

```
data_nhanes=pd.read_csv('nhanes_features_FINAL.csv')
```

```
','DPQ090':'DPQ9','PHQSCR':'PHQ Score','DLQ140':'Feels DPRN','DLQ150':'DPRN Meds','DLQ170':'DPRN Level','DI':'DI'}, inplace=True)
```

```
datanhan = data_nhanes.drop(['SEQN','PHQ Score','DPRN Level','Feels DPRN','DPRN Meds','DPQ1','DPQ2','DPQ3','DPQ4','DPQ5','DPQ6',
g = sns.heatmap(datanhan.corr(),annot=True, annot_kws={"fontsize":17})
g.set_yticklabels(g.get_yticklabels(),size=25, rotation=0)
g.set_xticklabels(g.get_xticklabels(),size=25, rotation=90)
g.set_title('NHANES Heatmap',size=35)
sns.set(rc = {'figure.figsize':(30,30)})
plt.tight_layout()
plt.savefig('NHANES_Heatmap.png')
plt.show()
```

```
ct1 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Asthma"])
ct2 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Overweight"])
ct3 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Arthritis"])
ct4 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["CHF"])
ct5 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Vascular"])
ct6 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Chest Pain"])
ct7 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Heart attack"])
ct8 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Stroke"])
ct9 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Thyroid"])
ct10 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Emphysema"])
ct11 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Bronchitis"])
ct12 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["COPD"])
ct13 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Liver"])
ct14 = pd.crosstab(index=data_nhanes["DI"], columns=data_nhanes["Cancer"])
```

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows=7, ncols=2)
ct1.plot(kind="bar",stacked=True, ax=axes[0,0])
ct2.plot(kind="bar",stacked=True, ax=axes[0,1])
ct3.plot(kind="bar",stacked=True, ax=axes[1,0])
ct4.plot(kind="bar",stacked=True, ax=axes[1,1])
ct5.plot(kind="bar",stacked=True, ax=axes[2,0])
ct6.plot(kind="bar",stacked=True, ax=axes[2,1])
ct7.plot(kind="bar",stacked=True, ax=axes[3,0])
ct8.plot(kind="bar",stacked=True, ax=axes[3,1])
ct9.plot(kind="bar",stacked=True, ax=axes[4,0])
ct10.plot(kind="bar",stacked=True, ax=axes[4,1])
ct11.plot(kind="bar",stacked=True, ax=axes[5,0])
ct12.plot(kind="bar",stacked=True, ax=axes[5,1])
ct13.plot(kind="bar",stacked=True, ax=axes[6,0])
ct14.plot(kind="bar",stacked=True, ax=axes[6,1])
plt.savefig('comorbid_stack_nhanes.png')
```

[153]:
```
cnm1 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Infection'])
cnm2 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Cancer/Blood'])
cnm3 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Metabolic'])
cnm4 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Neurologic'])
cnm5 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Eyes/Ears'])
cnm6 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Heart/Lung'])
cnm7 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Digestive'])
cnm8 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Skin'])
cnm9 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Bones'])
cnm10 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Urinary'])
cnm11 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Womens Health'])
cnm12 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Genetic'])
cnm13 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Other Disorder'])
cnm14 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Injury'])
cnm15 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Morbidity'])
cnm16 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Public health'])
```

[160]:
```
fig, axes = plt.subplots(nrows=4, ncols=4)
cnm1.plot(kind="bar", stacked=True, ax=axes[0,0])
cnm2.plot(kind="bar", stacked=True, ax=axes[0,1])
cnm3.plot(kind="bar", stacked=True, ax=axes[0,2])
cnm4.plot(kind="bar", stacked=True, ax=axes[0,3])
cnm5.plot(kind="bar", stacked=True, ax=axes[1,0])
cnm6.plot(kind="bar", stacked=True, ax=axes[1,1])
cnm7.plot(kind="bar", stacked=True, ax=axes[1,2])
cnm8.plot(kind="bar", stacked=True, ax=axes[1,3])
cnm9.plot(kind="bar", stacked=True, ax=axes[2,0])
cnm10.plot(kind="bar",stacked=True, ax=axes[2,1])
cnm11.plot(kind="bar", stacked=True, ax=axes[2,2])
cnm12.plot(kind="bar", stacked=True, ax=axes[2,3])
cnm13.plot(kind="bar", stacked=True, ax=axes[3,0])
cnm14.plot(kind="bar", stacked=True, ax=axes[3,1])
cnm15.plot(kind="bar", stacked=True, ax=axes[3,2])
cnm16.plot(kind="bar", stacked=True, ax=axes[3,3])
plt.savefig('DI vs Comorbid in NAMCS.png')
```

```
insnam1 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs["Commercial"])
insnam2 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs["Medicare"])
insnam3 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['State-funded'])
insnam4 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Uninsured'])
```

```
fig, axes = plt.subplots(nrows=2, ncols=2)

insnam1.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,0])
insnam2.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,1])
insnam3.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,0])
insnam4.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,1])
plt.savefig('DI vs Insurance in NAMCS.png')
```

```
sxnam = pd.crosstab(index=data_namcs["DI"], columns=data_namcs["Gender"])
sxnam.plot(kind="bar", figsize=(8,8),stacked=True)
plt.savefig('namcs_sex_stack.png')
```

```
pr1 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs["Provider-withheld "])
pr2 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs["MD/DO"])
pr3 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['PA'])
pr4 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['NPNMW'])
pr5 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Nurse'])
pr6 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['MHP'])
pr7 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['Other provider'])
pr8 = pd.crosstab(index=data_namcs["DI"],columns= data_namcs['No provider'])
```

```
fig, axes = plt.subplots(nrows=4, ncols=2)

pr1.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,0])
pr2.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,1])
pr3.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,0])
pr4.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,1])
pr5.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[2,0])
pr6.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[2,1])
pr7.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[3,0])
pr8.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[3,1])
plt.savefig('DI vs PROVIDER in NAMCS.png')
```

```python
# Crosstabing Depression with Race
depracewh = pd.crosstab(index=df["DI"],columns= df["White"])
depracebl = pd.crosstab(index=df["DI"], columns=df["Black"])
depraceot = pd.crosstab(index=df["DI"],columns= df["Other Race"])
```

```python
import matplotlib.pyplot as plt

#defining subplot layout
fig, axes = plt.subplots(nrows=3, ncols=1)

#Adding DataFrames to subplots
depracewh.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0])
depracebl.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1])
depraceot.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[2])
plt.savefig('Race in NHANES.png')
```

```python
# Crosstabing Depression with Insurance
deprprvt = pd.crosstab(index=df["DI"],columns= df["Commercial"])
deprcare = pd.crosstab(index=df["DI"],columns= df["Medicare"])
deprnoch = pd.crosstab(index=df["DI"],columns= df["Uninsured"])
deprstate = pd.crosstab(index=df["DI"],columns= df["State-funded"])
```

```python
fig, axes = plt.subplots(nrows=2, ncols=2)

deprprvt.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,0])
deprcare.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0,1])
deprnoch.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,0])
deprstate.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1,1])
plt.savefig('Insurance in NHANES.png')
```

```
depredu = pd.crosstab(index=df["DI"],columns= df["Education"])
deprage = pd.crosstab(index=df["DI"],columns= df["Age"])
deprincome = pd.crosstab(index=df["DI"],columns= df["Income"])
deprgendr = pd.crosstab(index=df["DI"],columns= df["Gender"])
```

```
depredu.T.plot(kind="bar",figsize=(10,10),stacked=True)
plt.savefig('Educ in NHANES.png')
```

```
depredu.plot(kind="bar",figsize=(10,10),stacked=True)
plt.savefig(' DI vs Educ in NHANES.png')
```

```
deprage.T.plot(kind="bar",figsize=(18,9),stacked=True)
plt.savefig('Age in NHANES.png')
```

```
deprincome.T.plot(kind="bar",figsize=(18,9),stacked=True)
plt.savefig('Income in NHANES.png')
```

```
deprgendr.plot(kind="bar",figsize=(10,10),stacked=True)
plt.savefig('GenderF in NHANES.png')
```

```
#Plotting the distribution of DI vs AGE
import matplotlib.pyplot as plt
deprage.T.plot(kind="bar",figsize=(18,9),stacked=True)
plt.savefig('Age in NAMCS.png')
```

```
# Crosstabing Depression with Race and Age
depracewh = pd.crosstab(index=df["DI"],columns= df["White"])
depracebl = pd.crosstab(index=df["DI"], columns=df["Black"])
depraceot = pd.crosstab(index=df["DI"],columns= df["Other Race"])
deprage = pd.crosstab(index=df["DI"],columns= df["Age"])
```

```
#Plotting the distribution of DI vs RACE
import matplotlib.pyplot as plt
fig, axes = plt.subplots(nrows=3, ncols=1)
depracewh.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[0])
depracebl.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[1])
depraceot.plot(kind="bar", figsize=(10,10),stacked=True, ax=axes[2])
plt.savefig('RaceF in NAMCS.png')
```

Code for Data Visualization (PIE, Histogram) - NHANES

In [ ]:
```
import matplotlib as mlt
import matplotlib.pyplot as plt
df.groupby(['EDUC']).sum().plot(kind='pie', y='AGE',autopct='%1.1f%%')
fig = plt.figure(figsize =(10, 7))
plt.show()
```

In [ ]:
```
ones=df['DI'].sum()
zeros = df.shape[0]-ones
size=[ones,zeros]
lab = ['yes','no']
fig = plt.figure(figsize =(10, 7))
plt.pie(size,labels=lab,autopct='%1.1f%%')
plt.title('Depression Indicator')
plt.show()
```

In [ ]:
```
Male=df['SEXML'].sum()
Female = df.shape[0]-ones
size=[Male,Female]
lab = ['Male','Female']
fig = plt.figure(figsize =(10, 7))
plt.pie(size,labels=lab,autopct='%1.1f%%')
plt.title('Gender')
plt.show()
```

In [ ]:
```
fig=plt.figure(figsize =(10, 7))
plt.hist(df['AGE'], edgecolor='black', linewidth=0.5)
plt.title("Histogram of AGE")
plt.xlabel("AGE")
plt.ylabel("No.of people")
plt.show()
```

In [ ]:
```
fig=plt.figure(figsize =(10, 7))
plt.hist(df['EDUC'])
plt.title("Education")
plt.xlabel("Grade")
plt.ylabel("No.of People")
plt.show()
```

In [ ]:
```
fig=plt.figure(figsize =(10, 7))
plt.hist(df['INDHHIN2'], edgecolor='black', linewidth=0.6)
plt.title("INCOME")
plt.xticks([])
plt.yticks([])
plt.xlabel("Range")
plt.ylabel("Family Income")
plt.show()
```

Code for Data Visualization (PIE, Histogram) - NAMCS

In [ ]:

```
#depression indicator
ones=df['DI'].sum()
zeros = df.shape[0]-ones
size=[ones,zeros]
lab = list(df['DI'].unique())
fig = plt.figure(figsize =(10, 7))
plt.pie(size,labels=lab,autopct='%1.1f%%')
plt.title('Depression Indicator')
plt.show()
```

In [ ]:

```
#gender
Male=df['SEXML'].sum()
Female = df.shape[0]-ones
size=[Male,Female]
lab = ['Male','Female']
fig = plt.figure(figsize =(10, 7))
plt.pie(size,labels=lab,autopct='%1.1f%%')
plt.title('Gender')
plt.show()
```

In [ ]:

```
#Age
fig=plt.figure(figsize =(10, 7))
plt.hist(df['AGE'], edgecolor='black', linewidth=0.5)
plt.title("Histogram of AGE")
plt.xlabel("AGE")
plt.ylabel("No.of people")
plt.show()
```

Hypothesis Testing

Finding Significant variables among the two datasets

In [10]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
from scipy.stats import chisquare
from scipy.stats import chi2
from scipy.stats import chi2_contingency
from colorama import Fore
```

In [11]:
```
# STATISTICALLY SIGNIFICANT VARIABLES
# CALCULATING AND INTERPRETTING BOTH P VALUE AND TEST STATISTIC VALUE
def Stat(d, t):
    S=[]
    alpha = 0.05
    prob = 1-alpha
    for i in d.columns:
        data_table = pd.crosstab(d[i],t)
        stat,p,dof,expected = chi2_contingency(data_table)
        critical = chi2.ppf(prob, dof)

        if p<= alpha:
            print(Fore.GREEN+'p is statistically significant with',i,p)
            print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
            if abs(stat) >= critical:
                print('statistic value is significant  (reject H0)\n')
                S.append(i)

        if p> alpha:
            print(Fore.RED+'p is statistically not significant with',i,p)
            print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
            if abs(stat)< critical:
                print('statistic value is not significant (fail to reject H0)\n')
    return S
```

NHANES Dataset Significant Variables

```
In [5]: # CONSIDERING THE CHI SQUARE FOR DOING THE HYPOTHESIS TESTING AS WE HAVE THE CATEGORICAL DATA
        df_ = pd.read_csv('nhanes_features_FINAL.csv')
        target_ = df_['DI']

        # WE ARE DROPPING THESE COLUMNS 'DLQ140', 'DLQ150', 'DLQ170','PHQSCR','DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060',
        # BECAUSE THEY ARE PART OF DEPRESSION INDICATOR CALCULATION - DI

        df1_ = df_.drop(['SEQN','DI','DLQ140', 'DLQ150', 'DLQ170','PHQSCR','DPQ010', 'DPQ020', 'DPQ030', 'DPQ040', 'DPQ050', 'DPQ060', 'D
```

```
In [6]: nhanes = Stat(df1_, target_)
```

```
p is statistically significant with AGE 0.031169344259373052
probability=0.950, critical=81.381, stat=84.336
statistic value is significant  (reject H0)

p is statistically significant with INDHHIN2 3.936574817639932e-37
probability=0.950, critical=23.685, stat=210.484
statistic value is significant  (reject H0)

p is statistically significant with MCQ010 6.873873609517555e-16
probability=0.950, critical=3.841, stat=65.169
statistic value is significant  (reject H0)

p is statistically significant with MCQ080 3.889666938644342e-19
probability=0.950, critical=3.841, stat=79.925
statistic value is significant  (reject H0)

p is statistically significant with MCQ160A 1.196153517956054e-41
probability=0.950, critical=3.841, stat=182.783
statistic value is significant  (reject H0)
```

NAMCS dataset significant variables

```
In [12]: # CONSIDERING THE CHI SQUARE FOR DOING THE HYPOTHESIS TESTING AS WE HAVE THE CATEGORICAL DATA
         df = pd.DataFrame(pd.read_csv("namcs_features_FINAL.csv"))
         target = df['DI']

         # WE ARE DROPPING THESE COLUMNS 'PSYCHOTH', 'DEPRN', 'DEPRESS', 'MENTAL', 'PSYCHOTH', 'DIAGALL'
         # BECAUSE THEY ARE PART OF DEPRESSION INDICATOR CALCULATION - DI, OR THE DIAGNOSIS GROUPS - DIAGALL

         df1 = df.drop(['SEQN','DI','PSYCHOTH', 'DEPRN', 'DEPRESS', 'MENTAL', 'DIAGALL'], axis=1)
```

```
In [13]: namcs = Stat(df1, target)

         p is statistically significant with AGE 3.8945972750688357e-13
         probability=0.950, critical=96.217, stat=198.706
         statistic value is significant  (reject H0)

         p is statistically significant with SEXML 4.4211211300461036e-05
         probability=0.950, critical=3.841, stat=16.681
         statistic value is significant  (reject H0)

         p is statistically significant with RACEWH 0.04017824866140956
         probability=0.950, critical=3.841, stat=4.210
         statistic value is significant  (reject H0)

         p is statistically not significant with RACEBL 0.11328907163698385
         probability=0.950, critical=3.841, stat=2.508
         statistic value is not significant (fail to reject H0)

         p is statistically not significant with RACEOT 0.2565276432678608
         probability=0.950, critical=3.841, stat=1.287
         statistic value is not significant (fail to reject H0)
```

Significant variables from both the datasets

```
In [9]: print("NAMCS: ",namcs,"\n")
        print("NHANES: ",nhanes)
```

```
NAMCS:  ['AGE', 'SEXML', 'RACEWH', 'PHYS', 'PHYSASST', 'MHP', 'OTHPROV', 'INSMCARE', 'INSSTATE', 'INSPRVT', 'SPPRC', 'SPSUR',
'SPMEC', 'OWNUNKN', 'OWNPHYS', 'OWNHOSP', 'OWNINSR', 'NOCHRON', 'DGRP2', 'DGRP3', 'DGRP4', 'DGRP5', 'DGRP6', 'DGRP7', 'DGRP8',
'DGRP10', 'DGRP11', 'DGRP14', 'DGRP16']

NHANES:  ['AGE', 'INDHHIN2', 'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F', 'MCQ160M',
'MCQ160G', 'MCQ160K', 'MCQ160O', 'MCQ510C', 'MCQ510D', 'MCQ220', 'INSPRVT', 'INSMCARE', 'SEXML', 'RACEWH', 'RACEOT', 'INSSTAT
E', 'EDUC', 'MCQ700']
```

Merging the Significant variables from Both the Datasets

```
In [10]: import pandas as pd
         from functools import reduce

         df3=pd.read_csv('namcs_features_FINAL.csv',  usecols=['SEQN', 'DI', 'AGE', 'SEXML', 'RACEWH', 'PHYS', 'PHYSASST', 'MHP', 'OTHPROV
         df4=pd.read_csv('nhanes_features_FINAL.csv', usecols=['SEQN', 'DI' ,'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B',

         #define list of DataFrames
         dfs = [df3, df4]

         #merge all DataFrames into one
         final_df =  df3.merge(df4, how='outer')

         #view merged DataFrame
         final_df.head()
```

Out[10]:

	SEQN	AGE	SEXML	RACEWH	PHYS	PHYSASST	MHP	OTHPROV	INSMCARE	INSSTATE	...	MCQ160E	MCQ160F	MCQ160M	MCQ160G	MCQ160K	M
0	11002.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	NaN	
1	11003.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	NaN	
2	11007.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	NaN	
3	11009.0	18.0	1	1	1.0	0.0	0.0	1.0	0.0	0	...	NaN	NaN	NaN	NaN	NaN	
4	11010.0	18.0	1	1	1.0	0.0	1.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	NaN	

5 rows × 48 columns

Processing of Merged dataset

```
In [11]: final_df.shape

Out[11]: (14557, 48)
```

```
In [12]: final_df.isna().sum()

Out[12]: SEQN          0
         AGE           0
         SEXML         0
         RACEWH        0
         PHYS       5856
         PHYSASST   5856
         MHP        5856
         OTHPROV    5856
         INSMCARE      0
         INSSTATE      0
         INSPRVT       0
         SPPRC      5856
         SPSUR      5856
         SPMEC      5856
         OWNUNKN    5856
         OWNPHYS    5856
         OWNHOSP    5856
         OWNINSR    5856
         NOCHRON    5856
```

In [13]:
```
#replacing blank values with nan
import numpy as np
final_df.replace(r'^\s*$', np.nan, regex=True)
```

Out[13]:

	SEQN	AGE	SEXML	RACEWH	PHYS	PHYSASST	MHP	OTHPROV	INSMCARE	INSSTATE	...	MCQ160E	MCQ160F	MCQ160M	MCQ160G	MCQ16(
0	11002.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	Na
1	11003.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	Na
2	11007.0	18.0	1	1	1.0	0.0	0.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	Na
3	11009.0	18.0	1	1	1.0	0.0	0.0	1.0	0.0	0	...	NaN	NaN	NaN	NaN	Na
4	11010.0	18.0	1	1	1.0	0.0	1.0	0.0	0.0	0	...	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14552	102751.0	19.0	1	1	NaN	NaN	NaN	NaN	0.0	0	...	0.0	0.0	0.0	0.0	0
14553	102804.0	30.0	0	0	NaN	NaN	NaN	NaN	0.0	0	...	0.0	0.0	0.0	0.0	0
14554	102825.0	80.0	1	1	NaN	NaN	NaN	NaN	1.0	0	...	0.0	0.0	1.0	0.0	1
14555	102940.0	77.0	1	1	NaN	NaN	NaN	NaN	1.0	0	...	0.0	1.0	0.0	0.0	0
14556	102950.0	64.0	0	0	NaN	NaN	NaN	NaN	0.0	0	...	0.0	1.0	0.0	0.0	0

14557 rows × 48 columns

Using Models for answering the Hypothesis Question and selecting best model for prediction of depression

In [1]:
```
import numpy as np
import pandas as pd
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (confusion_matrix, ConfusionMatrixDisplay, accuracy_score, precision_score, f1_score,
                             classification_report, mean_squared_error, auc, recall_score, roc_curve, roc_auc_score,
                             plot_roc_curve, precision_recall_curve)
from sklearn.model_selection import cross_val_score, train_test_split

!pip install imbalanced-learn
from imblearn.over_sampling import SMOTE
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: imbalanced-learn in c:\users\alanv\appdata\roaming\python\python39\site-packages (0.9.1)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (2.2.
0)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\alanv\appdata\roaming\python\python39\site-packages (from imbala
nced-learn) (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.1.0)
```

In [2]:
```
def skmets(y, p):
    Accuracy = accuracy_score(y, p)
    Precision = precision_score(y, p)
    Sensitivity_recall = recall_score(y, p)
    Specificity = recall_score(y, p, pos_label=0)
    F1_score = f1_score(y, p)
    M = {"Accuracy":Accuracy,"Precision":Precision,"Sensitivity_recall":Sensitivity_recall,"Specificity":Specificity,"F1_score":F
    return M
```

In [3]:
```
df_nam = pd.DataFrame(pd.read_excel("namcs_features_FINAL.xlsx"))
df_nam = df_nam[['DI', 'AGE', 'SEXML', 'RACEWH', 'PHYS', 'PHYSASST', 'MHP', 'OTHPROV', 'INSMCARE', 'INSSTATE', 'INSPRVT', 'SPPRC'
namcs = ['AGE', 'GENDER', 'RACE-WHITE', 'Physician', 'Physician assistant', 'Mental health provider', 'Other provider', 'Medicare
df_nh = pd.DataFrame(pd.read_excel("nhanes_features_FINAL.xlsx"))
df_nh = df_nh[['DI', 'AGE', 'INDHHIN2', 'MCQ010', 'MCQ080', 'MCQ160A', 'MCQ160B', 'MCQ160C', 'MCQ160D', 'MCQ160E', 'MCQ160F', 'MC
nhanes = ['AGE', 'Month family income', 'MedHistory: Asthma', 'MedHistory: Overweight', 'MedHistory: Arthritis', 'MedHistory: Cor
```

In [4]:
```
X_nam = df_nam.drop("DI", axis=1)
y_nam = df_nam["DI"]
```

In [5]:
```
X_nh = df_nh.drop("DI", axis=1)
y_nh = df_nh["DI"]
```

XGBOOST CLASSIFIER

In [6]:
```
xgb_cl = xgb.XGBClassifier()
```

namcs dataset

In [7]:
```
Xnam_train, Xnam_test, ynam_train, ynam_test = train_test_split(X_nam, y_nam, train_size=0.8, random_state=1000000000)
```

In [8]:
```
sm = SMOTE()
Xnam_train, ynam_train = sm.fit_resample(Xnam_train, ynam_train)
```

In [9]:
```
# namcs Fit
xgb_cl.fit(Xnam_train, ynam_train)
modelnam_fit = xgb_cl.fit(Xnam_train, ynam_train)
# Predict
prednam = xgb_cl.predict(Xnam_test)
```

In [10]:
```
#metrics.get_scorer_names()
```

Cross Validation Score for namcs

In [11]:
```
scores = cross_val_score(xgb_cl, Xnam_train, ynam_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.89 with a standard deviation of 0.08
```

Model scores for namcs

In [12]:
```
skmets(ynam_test, prednam)
```

Out[12]:
```
{'Accuracy': 0.8345778288340034,
 'Precision': 0.4470198675496689,
 'Sensitivity_recall': 0.52734375,
 'Specificity': 0.8875420875420875,
 'F1_score': 0.48387096774193555}
```

Feature Importance for predicting Depression for namcs

In [13]:
```
feat_importances = pd.Series(xgb_cl.feature_importances_, index=namcs)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[13]: <AxesSubplot:>

Confusion matrix for namcs

In [14]:
```
cn_matrix_nam = metrics.confusion_matrix(ynam_test, prednam)
cm_display_nam = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nam , display_labels = [False, True])
cm_display_nam.plot()
plt.show()
```

XG BOOST namcs dataset Report

In [15]:
```
print(classification_report(ynam_test,prednam))
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90      1485
           1       0.45      0.53      0.48       256

    accuracy                           0.83      1741
   macro avg       0.68      0.71      0.69      1741
weighted avg       0.85      0.83      0.84      1741
```

PR and ROC Curve for namcs

In [16]:
```
#calculate precision and recall
precision, recall, thresholds = precision_recall_curve(ynam_test, prednam)

#create precision recall curve
fig, ax = plt.subplots()
ax.plot(recall, precision, color='red')

#add axis labels to plot
ax.set_title('Precision-Recall Curve for namcs')
ax.set_ylabel('Precision')
ax.set_xlabel('Recall')

#display plot
plt.show()
```

Recall

In [17]:
```
plot_roc_curve(xgb_cl, Xnam_test, ynam_test)
```

```
C:\Users\alanv\AppData\Roaming\Python\Python39\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_roc_
curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class meth
ods: :meth:`sklearn.metrics.RocCurveDisplay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[17]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2319e578dc0>
```

nhanes dataset

In [18]:
```
Xnh_train, Xnh_test, ynh_train, ynh_test = train_test_split(X_nh, y_nh, train_size=0.8, random_state=1000000000)
```

In [19]:
```
sm = SMOTE()
Xnh_train, ynh_train = sm.fit_resample(Xnh_train, ynh_train)
```

In [20]:
```
# nhanes Fit
xgb_cl.fit(Xnh_train, ynh_train)
modelnh_fit = xgb_cl.fit(Xnh_train, ynh_train)
# Predict
prednh = xgb_cl.predict(Xnh_test)
```

Cross Validation Score for nhanes

In [21]:
```
scores = cross_val_score(xgb_cl, Xnh_train, ynh_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.76 with a standard deviation of 0.06
```

Model scores for nhanes

In [22]:
```
skmets(ynh_test, prednh)
```

Out[22]:
```
{'Accuracy': 0.6399317406143344,
 'Precision': 0.3209549071618037,
 'Sensitivity_recall': 0.42160278745644597,
 'Specificity': 0.7107344632768362,
 'F1_score': 0.36445783132530113}
```

Confusion matrix for nhanes

In [23]:
```
cn_matrix_nh = metrics.confusion_matrix(ynh_test, prednh)
cm_display_nh = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nh , display_labels = [False, True])
cm_display_nh.plot()
plt.show()
```

XG BOOST nhanes dataset Report

In [24]:
```
print("XG BOOST nhanes dataset Report")
print(classification_report(ynh_test,prednh))
```

```
XG BOOST nhanes dataset Report
              precision    recall  f1-score   support

           0       0.79      0.71      0.75       885
           1       0.32      0.42      0.36       287

    accuracy                           0.64      1172
   macro avg       0.56      0.57      0.56      1172
weighted avg       0.68      0.64      0.65      1172
```

Feature Importance for predicting Depression for nhanes

In [25]:
```
feat_importances = pd.Series(xgb_cl.feature_importances_, index=nhanes)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[25]: <AxesSubplot:>

PR and ROC Curve for namcs

In [26]:
```
#calculate precision and recall
precision, recall, thresholds = precision_recall_curve(ynh_test, prednh)

#create precision recall curve
fig, ax = plt.subplots()
ax.plot(recall, precision, color='blue')

#add axis labels to plot
ax.set_title('Precision-Recall Curve for namcs')
ax.set_ylabel('Precision')
ax.set_xlabel('Recall')

#display plot
plt.show()
```

In [27]:
```
plot_roc_curve(xgb_cl, Xnh_test, ynh_test)
```

```
C:\Users\alanv\AppData\Roaming\Python\Python39\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_roc_
curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class meth
ods: :meth:`sklearn.metrics.RocCurveDisplay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[27]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2319e81d790>
```

RANDOM FOREST CLASSIFIER

In [28]:
```
clf = RandomForestClassifier(criterion = "gini", random_state=1)
```

namcs dataset

In [29]:
```
Xnam_train, Xnam_test, ynam_train, ynam_test = train_test_split(X_nam, y_nam, train_size=0.8, random_state=1000000000)
```

In [30]:
```
sm = SMOTE()
Xnh_train, ynh_train = sm.fit_resample(Xnh_train, ynh_train)
```

In [31]:
```
# namcs
clf.fit(Xnam_train, ynam_train)
prednam = clf.predict(Xnam_test)
```

Cross Validation for namcs

In [32]:
```
scores = cross_val_score(clf, Xnam_train, ynam_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.67 with a standard deviation of 0.02
```

Model scores for namcs

In [33]:
```
skmets(ynam_test, prednam)
```

Out[33]:
```
{'Accuracy': 0.8575531303848363,
 'Precision': 0.5204081632653061,
 'Sensitivity_recall': 0.3984375,
 'Specificity': 0.9367003367003367,
 'F1_score': 0.4513274336283186}
```

Feature Importance for predicting Depression for namcs

In [34]:
```
feat_importances = pd.Series(clf.feature_importances_, index=namcs)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[34]: <AxesSubplot:>

Confusion matrix for namcs

In [35]:
```
cn_matrix_nam = metrics.confusion_matrix(ynam_test, prednam)
cm_display_nam = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nam , display_labels = [False, True])
cm_display_nam.plot()
plt.show()
```

RANDOM FOREST namcs dataset Report

In [36]:
```
print(classification_report(ynam_test,prednam))
```

```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      1485
           1       0.52      0.40      0.45       256

    accuracy                           0.86      1741
   macro avg       0.71      0.67      0.68      1741
weighted avg       0.84      0.86      0.85      1741
```

PR and ROC Curve for namcs¶

In [37]:
```
#calculate precision and recall
precision, recall, thresholds = precision_recall_curve(ynam_test, prednam)

#create precision recall curve
fig, ax = plt.subplots()
ax.plot(recall, precision, color='blue')

#add axis labels to plot
ax.set_title('Precision-Recall Curve for nhanes')
ax.set_ylabel('Precision')
ax.set_xlabel('Recall')

#display plot
plt.show()
```

```python
# Data to plot precision - recall curve
precision, recall, thresholds = precision_recall_curve(ynam_test, prednam)
# Use AUC function to calculate the area under the curve of precision recall curve
auc_precision_recall = auc(recall, precision)
print(auc_precision_recall)
```

```
0.5163971903767639
```

In [38]:
```
plot_roc_curve(clf, Xnam_test, ynam_test)
```

```
C:\Users\alanv\AppData\Roaming\Python\Python39\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_roc_
curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class meth
ods: :meth:`sklearn.metrics.RocCurveDisplay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[38]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x2319e74eb80>
```

nhanes dataset

In [39]:
```
Xnh_train, Xnh_test, ynh_train, ynh_test = train_test_split(X_nh, y_nh, train_size=0.8, random_state=1000000000)
```

In [40]:
```
sm = SMOTE()
Xnh_train, ynh_train = sm.fit_resample(Xnh_train, ynh_train)
```

In [41]:
```
# nhanes
clf.fit(Xnh_train, ynh_train)
prednh = clf.predict(Xnh_test)
```

Cross Validation for nhanes

In [42]:
```
scores = cross_val_score(clf, Xnh_train, ynh_train, cv=5, scoring='f1_macro')
print("Mean f1 score of %0.2f with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

```
Mean f1 score of 0.75 with a standard deviation of 0.06
```

Model scores for nhanes

In [43]:
```
skmets(ynh_test, prednh)
```

Out[43]:
```
{'Accuracy': 0.6168941979522184,
 'Precision': 0.2798913043478261,
 'Sensitivity_recall': 0.3588850174216028,
 'Specificity': 0.7005649717514124,
 'F1_score': 0.31450381679389317}
```

Feature Importance for predicting Depression for nhanes

In [44]:
```
feat_importances = pd.Series(clf.feature_importances_, index=nhanes)
feat_importances.plot(kind='barh', figsize = (30,30), fontsize=25)
```

Out[44]: <AxesSubplot:>

Confusion matrix for nhanes

In [45]:
```
cn_matrix_nh = metrics.confusion_matrix(ynh_test, prednh)
cm_display_nh = ConfusionMatrixDisplay(confusion_matrix = cn_matrix_nh , display_labels = [False, True])
cm_display_nh.plot()
plt.show()
```

RANDOM FOREST nhanes dataset Report

In [46]:
```
print(classification_report(ynh_test, prednh))
```

```
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       885
           1       0.28      0.36      0.31       287

    accuracy                           0.62      1172
   macro avg       0.53      0.53      0.52      1172
weighted avg       0.65      0.62      0.63      1172
```

PR and ROC Curve for nhanes

In [47]:
```
#calculate precision and recall
precision, recall, thresholds = precision_recall_curve(ynh_test, prednh)

#create precision recall curve
fig, ax = plt.subplots()
ax.plot(recall, precision, color='blue')

#add axis labels to plot
ax.set_title('Precision-Recall Curve for nhanes')
ax.set_ylabel('Precision')
ax.set_xlabel('Recall')

#display plot
plt.show()
```

In [48]:
```
plot_roc_curve(clf, Xnh_test, ynh_test)
```

```
C:\Users\alanv\AppData\Roaming\Python\Python39\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_roc_
curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class meth
ods: :meth:`sklearn.metrics.RocCurveDisplay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[48]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x231a06bdaf0>
```

END