In [2]:
```
# purpose: use logrithmic regression on dataset

import os
os.environ['MPLCONFIGDIR'] = os.getcwd() + "/configs/"
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.formula.api as smf
from scipy import stats
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import warnings

warnings.filterwarnings(action='ignore')
pd.options.mode.chained_assignment = None  # default='warn' ---- ignores false warning
```

In [3]:
```
# write csv into datafile and select columns to analyze
df = pd.read_csv('medical_clean.csv')
```

In [4]:
```
df = df[['Income','VitD_levels','Doc_visits','Full_meals_eaten','vitD_supp','Soft_drink
#opting to not remove duplicates as it is likely an inturpretation error
print(df.loc[df.duplicated()])
print(df.isnull().sum())
```

```
Empty DataFrame
Columns: [Income, VitD_levels, Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, High
Blood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Aller
gic_rhinitis, Reflux_esophagitis, Asthma]
Index: []
Income                0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
vitD_supp             0
Soft_drink            0
HighBlood             0
Stroke                0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
dtype: int64
```

In [5]:
```
# check for outliers and remove
print(df.shape)
df = df[(np.abs(stats.zscore(df.select_dtypes(include=np.number))) < 3).all(axis=1)]
print(df.shape)
```

```
(10000, 17)
(9727, 17)
```

In [6]:

```
di = {'Yes': 1, 'No': 0}
df = df.replace({'Soft_drink': di,'HighBlood': di,'Stroke': di,'Overweight': di,'Arthri
print(df.head())
```

```
     Income  VitD_levels  Doc_visits  Full_meals_eaten  vitD_supp  Soft_drink  \
0  86575.93    19.141466           6                 0          0           0
1  46805.99    18.940352           4                 2          1           0
2  14370.14    18.057507           4                 1          0           0
3  39741.49    16.576858           4                 1          0           0
4   1209.56    17.439069           5                 0          2           1

   HighBlood  Stroke  Overweight  Arthritis  Diabetes  Hyperlipidemia  \
0          1       0           0          1         1               0
1          1       0           1          0         0               0
2          1       0           1          0         1               0
3          0       1           0          1         0               0
4          0       0           0          0         0               1

   BackPain  Anxiety  Allergic_rhinitis  Reflux_esophagitis  Asthma
0         1        1                  1                   0       1
1         0        0                  0                   1       0
2         0        0                  0                   0       0
3         0        0                  0                   1       1
4         0        0                  1                   0       0
```

In [7]:

```
# bivariate analysis heatmap
ax = plt.subplots(figsize=(16,16))
ax = sns.heatmap(df.corr(), annot=True)
plt.savefig('heatmap_initial.jpg')
plt.close()
print('Initial heatmap done')
```

```
Initial heatmap done
```

In [8]:

```
# list to iterate through
testList = ['HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidem
for i in testList:
  df[['Soft_drink', i]].value_counts().plot(kind='barh')
  plt.savefig('barh%s.jpg' % (i))
  plt.close()
print('scatterplots done')
```

```
scatterplots done
```

In [9]:

```
df.hist(figsize = (16,16))
plt.savefig('hospital_pyplot.jpg')
plt.tight_layout()
plt.close()
print('Histogram done')
```

```
Histogram done
```

In [10]:

```
test = smf.logit(formula = 'Soft_drink ~ Income + VitD_levels + Doc_visits + Full_meals
print(test.summary())
```

```
Optimization terminated successfully.
```

```
         Current function value: 0.569385
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:             Soft_drink   No. Observations:                 9727
Model:                          Logit   Df Residuals:                     9710
Method:                           MLE   Df Model:                           16
Date:                Sat, 23 Oct 2021   Pseudo R-squ.:                0.002071
Time:                        22:15:14   Log-Likelihood:                -5538.4
converged:                       True   LL-Null:                       -5549.9
Covariance Type:            nonrobust   LLR p-value:                    0.1140
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            -1.4062      0.251     -5.597      0.000      -1.899      -0.914
Income            7.586e-08   9.14e-07      0.083      0.934   -1.72e-06    1.87e-06
VitD_levels          0.0074      0.012      0.632      0.527      -0.016       0.030
Doc_visits           0.0266      0.022      1.188      0.235      -0.017       0.070
Full_meals_eaten     0.0488      0.024      2.072      0.038       0.003       0.095
vitD_supp           -0.0504      0.040     -1.260      0.208      -0.129       0.028
HighBlood           -0.0233      0.047     -0.493      0.622      -0.116       0.069
Stroke               0.0231      0.058      0.400      0.689      -0.090       0.137
Overweight          -0.0280      0.051     -0.548      0.584      -0.128       0.072
Arthritis           -0.0393      0.049     -0.808      0.419      -0.134       0.056
Diabetes             0.0710      0.052      1.376      0.169      -0.030       0.172
Hyperlipidemia       0.0943      0.049      1.931      0.053      -0.001       0.190
BackPain             0.0685      0.047      1.453      0.146      -0.024       0.161
Anxiety              0.0752      0.049      1.524      0.127      -0.022       0.172
Allergic_rhinitis   -0.0812      0.048     -1.701      0.089      -0.175       0.012
Reflux_esophagitis   0.0008      0.047      0.018      0.986      -0.092       0.093
Asthma               0.0481      0.051      0.943      0.345      -0.052       0.148
=====================================================================================
```

In [11]:

```
conf = test.conf_int()
conf['OR'] = test.params # create odds ratio
conf.columns = ['2.5%', '97.5%', 'OR']
print(np.exp(conf))
```

```
                       2.5%     97.5%        OR
Intercept          0.149772  0.400987  0.245065
Income             0.999998  1.000002  1.000000
VitD_levels        0.984618  1.030719  1.007405
Doc_visits         0.982856  1.073024  1.026951
Full_meals_eaten   1.002635  1.099608  1.050003
vitD_supp          0.879065  1.028398  0.950804
HighBlood          0.890438  1.071850  0.976942
Stroke             0.913673  1.146348  1.023419
Overweight         0.879851  1.074748  0.972429
Arthritis          0.874177  1.057562  0.961507
Diabetes           0.970312  1.187960  1.073635
Hyperlipidemia     0.998597  1.209367  1.098940
BackPain           0.976410  1.174488  1.070879
Anxiety            0.978721  1.187664  1.078143
Allergic_rhinitis  0.839669  1.012418  0.922006
Reflux_esophagitis 0.912436  1.097809  1.000840
Asthma             0.949520  1.159450  1.049248
```

In [12]:

```
# for reduced, use Diabetes, Hyperlipidemia, and BackPain since P < 0.1
testReduced = smf.logit(formula = 'Soft_drink ~ Full_meals_eaten + Hyperlipidemia + All
print(testReduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.570010
```

```
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:             Soft_drink   No. Observations:                 9727
Model:                          Logit   Df Residuals:                     9723
Method:                           MLE   Df Model:                            3
Date:                Sat, 23 Oct 2021   Pseudo R-squ.:               0.0009753
Time:                        22:15:47   Log-Likelihood:                -5544.5
converged:                       True   LL-Null:                       -5549.9
Covariance Type:            nonrobust   LLR p-value:                   0.01271
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept         -1.1074      0.041    -26.728      0.000      -1.189      -1.026
Full_meals_eaten   0.0498      0.024      2.118      0.034       0.004       0.096
Hyperlipidemia     0.0914      0.049      1.874      0.061      -0.004       0.187
Allergic_rhinitis -0.0818      0.048     -1.716      0.086      -0.175       0.012
==============================================================================
```

In [13]:

```
## Plotting multiple plots same figure
fig, (axL, axR) = plt.subplots(2, figsize=(15, 15))
# Deviance Residuals
sns.regplot(test.fittedvalues, test.resid_dev, ax= axL, color="black", scatter_kws={"s"
sns.regplot(testReduced.fittedvalues, testReduced.resid_pearson, ax= axR, color="black"
plt.savefig('residual_plot.jpg')
plt.close()
```

In [14]:

```
df.to_csv('initial_cleaned_data.csv')
dfReduced = df[['Full_meals_eaten','Soft_drink', 'Hyperlipidemia', 'Allergic_rhinitis']
dfReduced.to_csv('reduced_cleaned_data.csv')
```

In [15]:

```
# bivariate analysis heatmap
ax = plt.subplots(figsize=(12,12))
ax = sns.heatmap(dfReduced.corr(), annot=True)
plt.savefig('heatmap_reduced.jpg')
plt.close()
print('Reduced heatmap done')
```

```
Reduced heatmap done
```

In [16]:

```
# confusion matrix
X = df.loc[:, df.columns != 'Soft_drink']
y = df.loc[:, df.columns == 'Soft_drink']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on initial set: {:.2f}'.format(logreg
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)

#confusion matrix accuracy stats
print('Accuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred)))
print('Classification Report\n')
print(classification_report(y_test, y_pred, target_names=['Class 1', 'Class 2']))
```

```
Accuracy of logistic regression classifier on initial set: 0.73
```

```
[[1787    0]
 [ 645    0]]
Accuracy: 0.73

Classification Report

              precision    recall  f1-score   support

     Class 1       0.73      1.00      0.85      1787
     Class 2       0.00      0.00      0.00       645

    accuracy                           0.73      2432
   macro avg       0.37      0.50      0.42      2432
weighted avg       0.54      0.73      0.62      2432
```

In [ ]: