Logarithmic Regression D208

January 4, 2022

1 D208 Project 2

1.1 Malcolm Mikkelsen

1.2 D208 Predictive Modeling

1.3 Dr. Sewell

2 Part 1: Research Question

The research question that we are investigating is: Can we create a model, with both categorical and continous variables, that can identify patients that are at high risk of readmission within 30 days of their initial discharge? Our goals of this analyses are to provide a cleaned data set that is prepared for a logistic regression model, and to create both an initial and reduced logistic model.

3 Part 2: Method Justification

3.1 B1: Assumptions of Logistic Regressions

When utilizing a logistic regression model there are five assumptions being made. First, the dependent variable that we are investigating must be a binary variable, in this case either the patient was readmitted, or they were not. Next, our predictor variables must be independent of each other, meaning that we cannot use variables that come from repeated data. The data must also contain little multicollinearity and that predictor variables have a linear relationship with log odds. Lastly, the data set must be large (Statistics Solutions, 2021). ## B2: Python Tool Justification Python was selected due to its ability to be versatile and the large number of libraries already available to users. We will be able to import and utilize these tools to streamline the analytics process, including the data wrangling steps.(Larose, 2019). ## B3: Selection of Logistic Regression Logistic regression was selected for this model because we are exploring a categorical dependent variable. While linear regression allows for only continuous data to be used as dependent variables, logistic regression only allows for categorical. Logistic regression also allows for us to use multiple predictor variables, as long as they meet the assumptions listed above.

4 Part 3: Data Preparation

4.1 C1 and C3Data Preparation Goals and Steps to Achieve Them

To prepare out data for the modeling we will need to accomplish a handful of steps. First, we will need to import the data into python from its text file and save it as a data frame. We will then

need to recast categorical data from its current state into one that is easily usable. For example, for yes or no questions we will cast no as 0 and yes as 1. Each dictionary we are using for this is identified within the code below. Then we will fill null values of continuous data using their means, and for categorical data we will assume them leaving the answer blank means that it does not apply to them. Thus, we will fill those null values with 0. We will allow for the outliers in both vitamin d supplementation and number of meals eaten, since both are within a realistic number dependent on the length of a patients stay. (Pandas Development Team, 2008).

Following these steps, we will create a new data frame with only the columns identified as necessary for the initial model.

4.2 Identify Summary Statistics and Predictor Variable Selection

There will be three types of predictor variables in the initial model. For continuous predictors we have 4: • Number of doctor visits during initial stay • Number of full meals eaten • Number of Vitamin D supplementations administered • Number of days for initial admission

For our categorical predictors we have 12 variables: • Soft drink consumption • High blood pressure • Previous stroke • Arthritis • Overweight • Diabetes • Hyperlipidemia • Back pain • Anxiety • Allergic Rhinitis • Reflux esophagitis • Asthma

Lastly, we have a survey that the patient filled out at the end of their initial admission. This provides us with ordinal data type predictor variables. They are: • Timely admission • Timely treatment • Timely visits by provider • Reliability • Options of treatment • Hours of treatment • Courteous staff • Active listening by provider

The summary statistics for all 24 predictor variables and the target variable are provided below.

4.3 Univariate and Bivariate Visualizations

For the predictor variables we will be using histograms and box plots for their univariate statistics. Any outliers will be identified and addressed as discussed in the data preparation steps above. We will then make a scatter plot of each predictor with the target variable.

5 Importing Data and Recasting Categorical Columns

[141]:
```
# I need to import the csv file and the libraries necessary for data cleaning
import pandas as pd
import numpy as np
import seaborn as sns

md = pd.read_csv('medical_raw_data.csv')
md.head
```

[141]:
```
<bound method NDFrame.head of       Unnamed: 0  CaseOrder Customer_id
Interaction  \
0              1          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1              2          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2              3          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
```

```
3              4          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4              5          5     C544523  5885f56b-d6da-43a3-8760-83583af94266
…            …          …       …                                      …
9995        9996       9996     B863060  a25b594d-0328-486f-a9b9-0567eb0f9723
9996        9997       9997     P712040  70711574-f7b1-4a17-b15f-48c54564b70f
9997        9998       9998     R778890  1d79569d-8e0f-4180-a207-d67ee4527d26
9998        9999       9999     E344109  f5a68e69-2a60-409b-a92f-ac0847b27db0
9999       10000      10000     I569847  bc482c02-f8c9-4423-99de-3db5e62a18d5
```

```
                                    UID          City State        County  \
0     3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan
1     176354c5eef714957d486009feabf195      Marianna    FL       Jackson
2     e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha
3     cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
4     d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William
…                                    …           …   …           …
9995  39184dc28cc038871912ccc4500049e5       Norlina    NC        Warren
9996  3cd124ccd43147404292e883bf9ec55c        Milmay    NJ      Atlantic
9997  41b770aeee97a5b9e7f69c906a8119d7     Southside    TN    Montgomery
9998  2bb491ef5b1beb1fed758cc6885c167a         Quinn    SD    Pennington
9999  95663a202338000abdf7e09311c2a8a1    Coraopolis    PA     Allegheny

        Zip       Lat  …  TotalCharge  Additional_charges Item1 Item2 Item3  \
0     35621  34.34960  …  3191.048774        17939.403420     3     3     2
1     32446  30.84513  …  4214.905346        17612.998120     3     4     3
2     57110  43.54321  …  2177.586768        17505.192460     2     4     4
3     56072  43.89744  …  2465.118965        12993.437350     3     5     5
4     23181  37.59894  …  1885.655137         3716.525786     2     1     3
…     …        …  …          …                  …   …   …   …
9995  27563  36.42886  …  6651.241294         8927.642189     3     2     2
9996   8340  39.43609  …  7851.522660        28507.147340     3     3     4
9997  37171  36.36655  …  7725.953391        15281.214660     3     3     3
9998  57775  44.10354  …  8462.831883         7781.678412     5     5     3
9999  15108  40.49998  …  8700.856021        11643.189930     4     3     3

      Item4  Item5 Item6 Item7  Item8
0         2      4     3     3      4
1         4      4     4     3      3
2         4      3     4     3      3
3         3      4     5     5      5
4         3      5     3     4      3
…     …    …   …   …    …
9995      3      4     3     4      2
9996      2      5     3     4      4
9997      4      4     2     3      2
9998      4      4     3     4      3
9999      2      3     6     4      3
```

```
[10000 rows x 53 columns]>
```

[142]:
```
#Drop columns described in cleaning plan then call columns to show that it has␣
 ↪been succesfully dropped
cmd = md.drop(columns = ['CaseOrder', 'Employment', 'Marital', 'Job',␣
 ↪'Interaction', 'UID', 'Lat', 'Lng', 'Income', 'City',
                          'State', 'County', 'Zip', 'Population', 'Area',␣
 ↪'Timezone', 'Children', 'Age', 'Education', 'Gender',
                          'Services', 'TotalCharge', 'Additional_charges',␣
 ↪'VitD_levels', 'Initial_admin', 'Complication_risk'])

cmd.drop(columns=cmd.columns[0], axis=1, inplace=True)
```

[143]:
```
#I will be renaming each 'item' column to better reflect what question was␣
 ↪being asked without having to reference the data sheet
cmd.rename(columns={'Item1': 'AdmissionTime', 'Item2': 'TreatmentTime', 'Item3':
 ↪ 'Visits', 'Item4': 'Reliability', 'Item5': 'Options', 'Item6': 'Hours',
           'Item7': 'Courteous', 'Item8': 'ActiveListening'}, inplace = True)
cmd.dtypes
```

[143]:
```
Customer_id            object
ReAdmis                object
Doc_visits              int64
Full_meals_eaten        int64
VitD_supp               int64
Soft_drink             object
HighBlood              object
Stroke                 object
Overweight            float64
Arthritis              object
Diabetes               object
Hyperlipidemia         object
BackPain               object
Anxiety               float64
Allergic_rhinitis      object
Reflux_esophagitis     object
Asthma                 object
Initial_days          float64
AdmissionTime           int64
TreatmentTime           int64
Visits                  int64
Reliability             int64
Options                 int64
Hours                   int64
Courteous               int64
ActiveListening         int64
```

```
dtype: object
```

[144]:
```
#This will be a similar dictionary to replace each yes/no questions. This will␣
 ↪be for ReAdmis column.
dict_admis = {'ReAdmis':{ 'Yes' : 1, 'No': 0}}
cmd.replace(dict_admis, inplace = True)
# Soft drink intake
dict_soft = {'Soft_drink':{ 'Yes' : 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_soft, inplace = True)
#High blood pressure
dict_bp = {'HighBlood':{ 'Yes' : 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_bp, inplace = True)
#Stroke
dict_stroke = {'Stroke':{ 'Yes' : 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_stroke, inplace = True)
# Athritis
dict_ath = {'Arthritis':{'Yes' : 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_ath, inplace = True)
#Diabetes
dict_dia = {'Diabetes':{'Yes' : 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_dia, inplace = True)
#Hyperlipidemia
dict_hype = {'Hyperlipidemia':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_hype, inplace = True)
#Back Pain
dict_back = {'BackPain':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_back, inplace = True)
#Allergic Rhinitis
dict_aller = {'Allergic_rhinitis':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_aller, inplace = True)
# Reflux Esophagitis
dict_ref = {'Reflux_esophagitis':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_ref, inplace = True)
#Asthma
dict_ast = {'Asthma': {'Yes': 1, 'No': 0, 'NA': np.NaN}}
cmd.replace(dict_ast, inplace = True)
# Services
dict_ser = {'Services':{ 'Blood Work':1, 'Intravenous': 2, 'CT Scan': 3, 'MRI':␣
 ↪4}}
cmd.replace(dict_ser, inplace = True)
```

6 Identify and Addressing Null Values

[145]:
```
null_columns=cmd.columns[cmd.isnull().any()]
cmd[null_columns].isnull().sum()
```

[145]:
```
Soft_drink      2467
Overweight       982
Anxiety          984
Initial_days    1056
dtype: int64
```

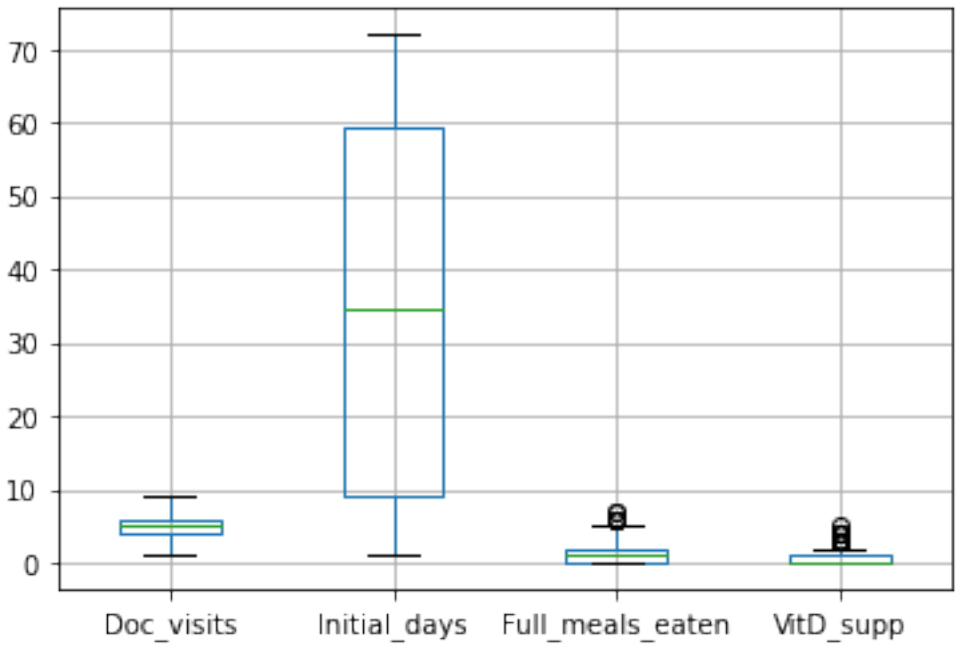
[146]:
```
# First we will fill the null values that we are planning to 0 out
cmd.Soft_drink.fillna(0, inplace = True)
cmd.Anxiety.fillna(0, inplace = True)
cmd.Overweight.fillna(0, inplace = True)

# Next is the Initial days which will be replaced with mean values
cmd['Initial_days'] = cmd['Initial_days'].fillna((cmd['Initial_days'].mean()))

#Check to make sure the nulls were replaced
null_columns=cmd.columns[cmd.isnull().any()]
cmd[null_columns].isnull().sum()
```

[146]:
```
Series([], dtype: float64)
```

7 Detecting and Addressing Outliers (Continous Univariate Statistics)

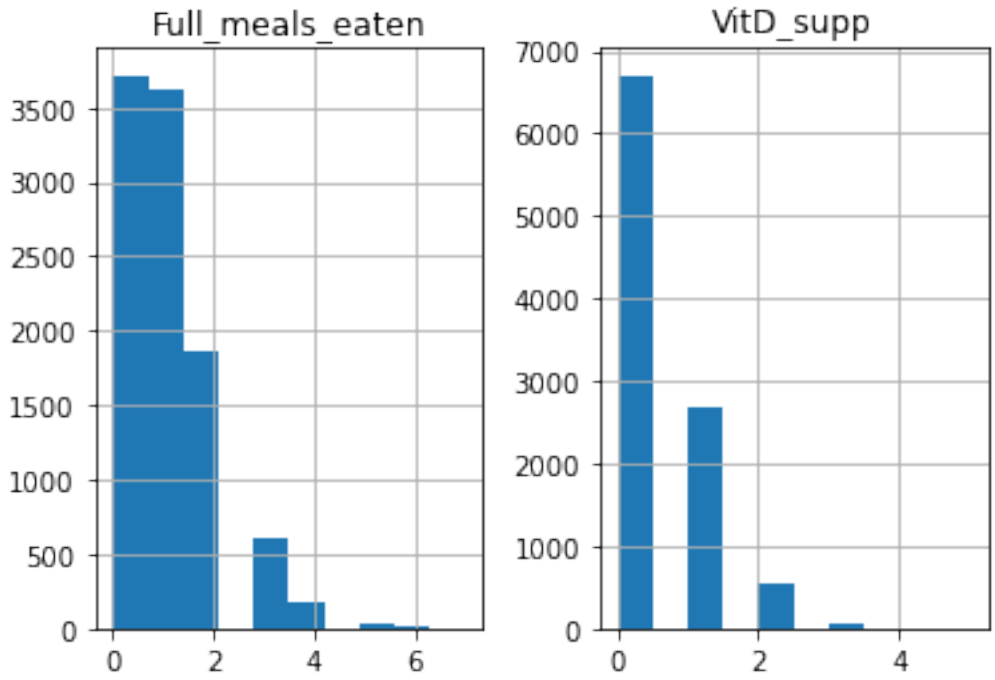
[147]:
```
#Using a boxplot to identify outliers in the rows most likely to contain them
cmd.boxplot(['Doc_visits', 'Initial_days', 'Full_meals_eaten', 'VitD_supp'])
```

[147]:
```
<AxesSubplot:>
```



[148]:
```
# Taking a closer look at the full meals eaten and vitamin d supplementation
cmd.hist(['Full_meals_eaten', 'VitD_supp'])
```

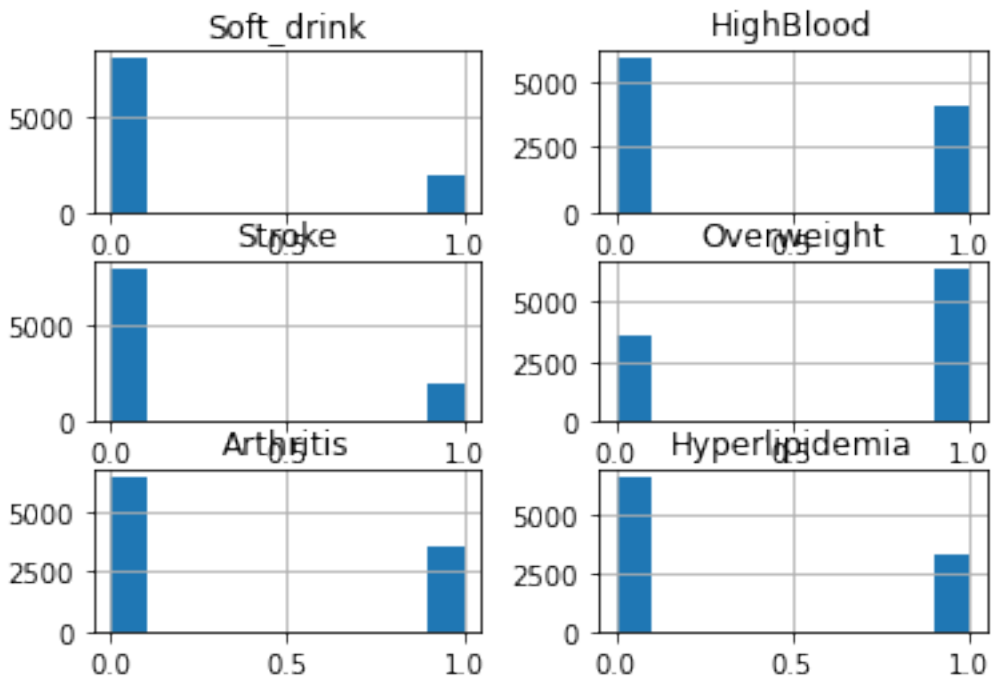
[148]:
```
array([[<AxesSubplot:title={'center':'Full_meals_eaten'}>,
        <AxesSubplot:title={'center':'VitD_supp'}>]], dtype=object)
```



8 Univariate Statistics for Remaining Variables

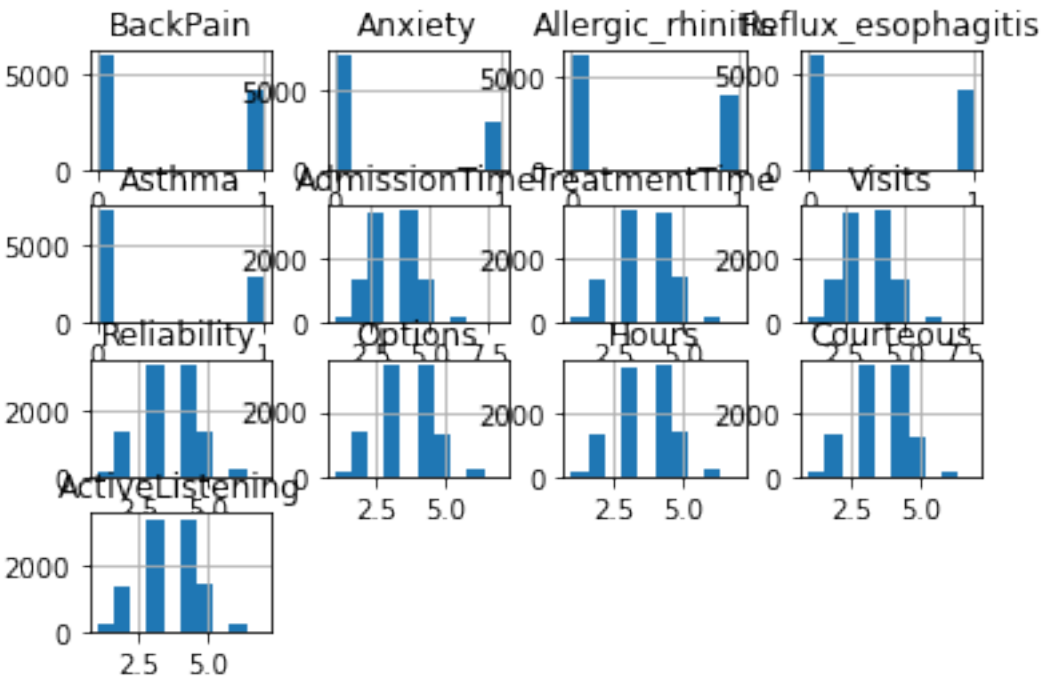
[149]:
```
# Running a histogram for remaining predictors
cmd.hist(['Soft_drink', 'HighBlood', 'Stroke','Overweight', 'Arthritis',␣
↪'Hyperlipidemia'])
```

[149]:
```
array([[<AxesSubplot:title={'center':'Soft_drink'}>,
        <AxesSubplot:title={'center':'HighBlood'}>],
       [<AxesSubplot:title={'center':'Stroke'}>,
        <AxesSubplot:title={'center':'Overweight'}>],
       [<AxesSubplot:title={'center':'Arthritis'}>,
        <AxesSubplot:title={'center':'Hyperlipidemia'}>]], dtype=object)
```



[150]:
```
cmd.hist(['BackPain', 'Anxiety', 'Allergic_rhinitis',␣
↪'Reflux_esophagitis','Asthma',␣
↪'AdmissionTime','TreatmentTime','Visits','Reliability',␣
↪'Options','Hours','Courteous','ActiveListening'])
```

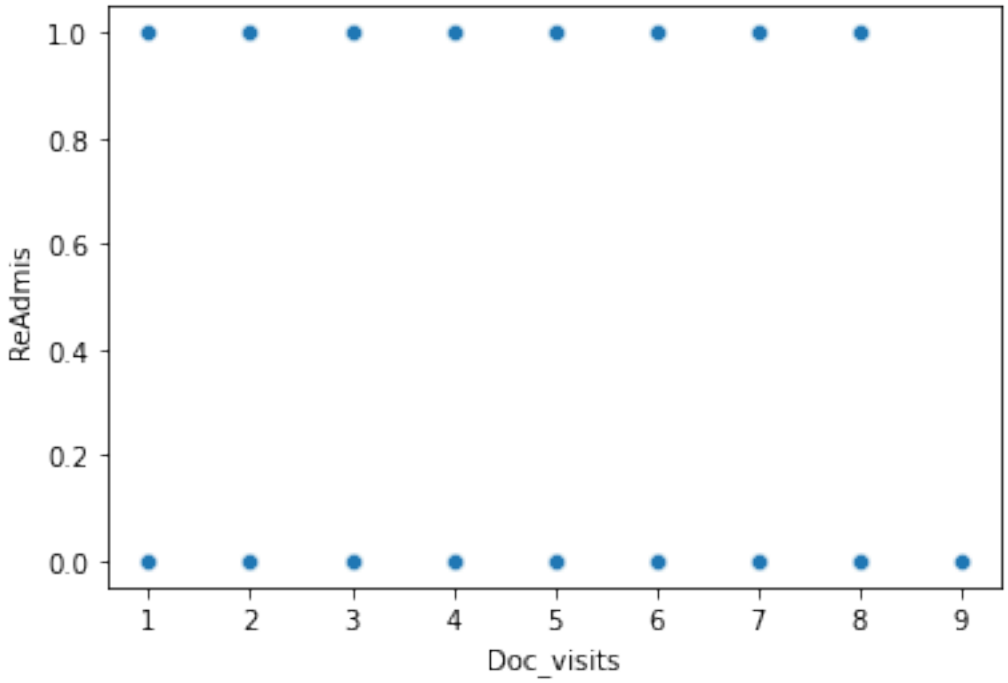
[150]:
```
array([[<AxesSubplot:title={'center':'BackPain'}>,
        <AxesSubplot:title={'center':'Anxiety'}>,
        <AxesSubplot:title={'center':'Allergic_rhinitis'}>,
        <AxesSubplot:title={'center':'Reflux_esophagitis'}>],
       [<AxesSubplot:title={'center':'Asthma'}>,
        <AxesSubplot:title={'center':'AdmissionTime'}>,
        <AxesSubplot:title={'center':'TreatmentTime'}>,
        <AxesSubplot:title={'center':'Visits'}>],
       [<AxesSubplot:title={'center':'Reliability'}>,
        <AxesSubplot:title={'center':'Options'}>,
        <AxesSubplot:title={'center':'Hours'}>,
        <AxesSubplot:title={'center':'Courteous'}>],
       [<AxesSubplot:title={'center':'ActiveListening'}>, <AxesSubplot:>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



9 Bivariate Statistics

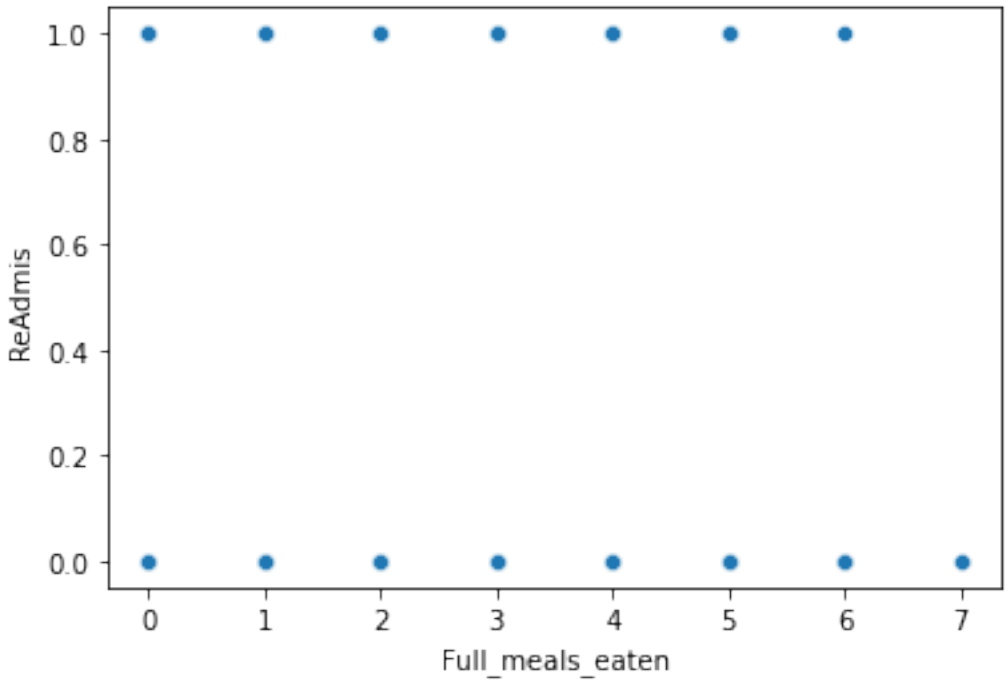
[151]:
```
# Run a scatterplot for each predictor vs the target variable
sns.scatterplot(x = 'Doc_visits', y ='ReAdmis', data = cmd)
```

[151]: <AxesSubplot:xlabel='Doc_visits', ylabel='ReAdmis'>



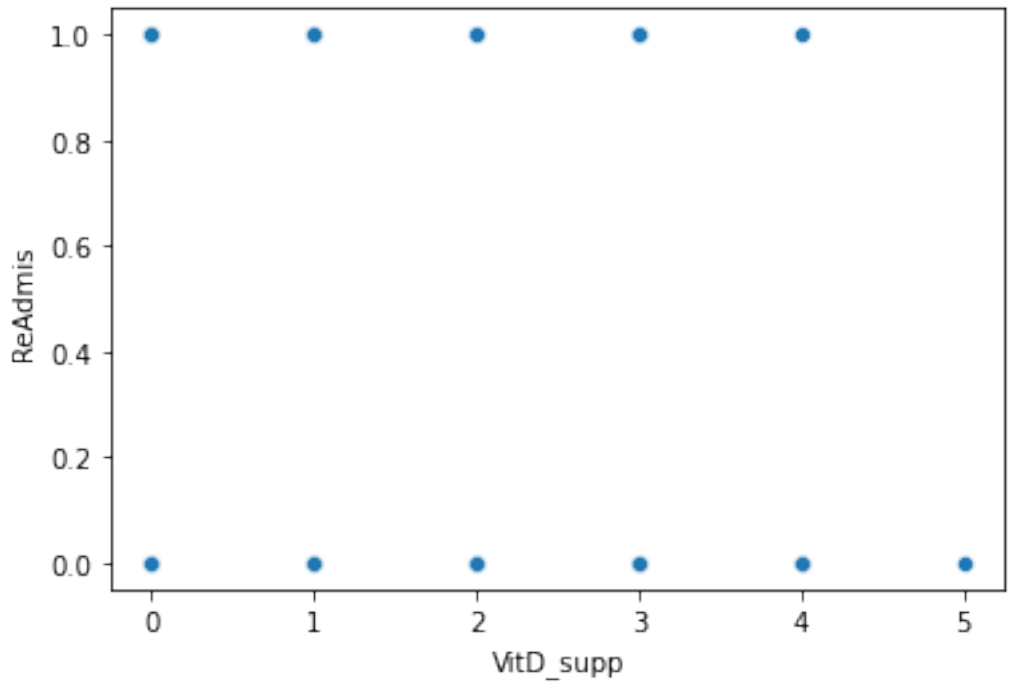
[152]:
```
sns.scatterplot(x = 'Full_meals_eaten', y ='ReAdmis', data = cmd)
```

[152]: <AxesSubplot:xlabel='Full_meals_eaten', ylabel='ReAdmis'>



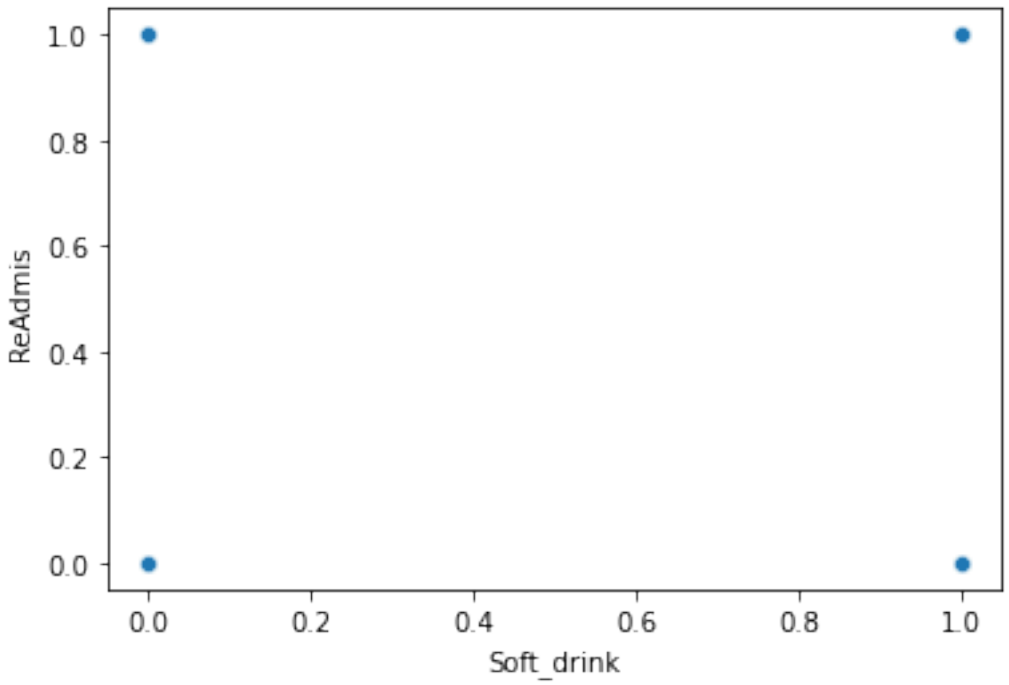
[153]:
```
sns.scatterplot(x = 'VitD_supp', y ='ReAdmis', data = cmd)
```

[153]: <AxesSubplot:xlabel='VitD_supp', ylabel='ReAdmis'>



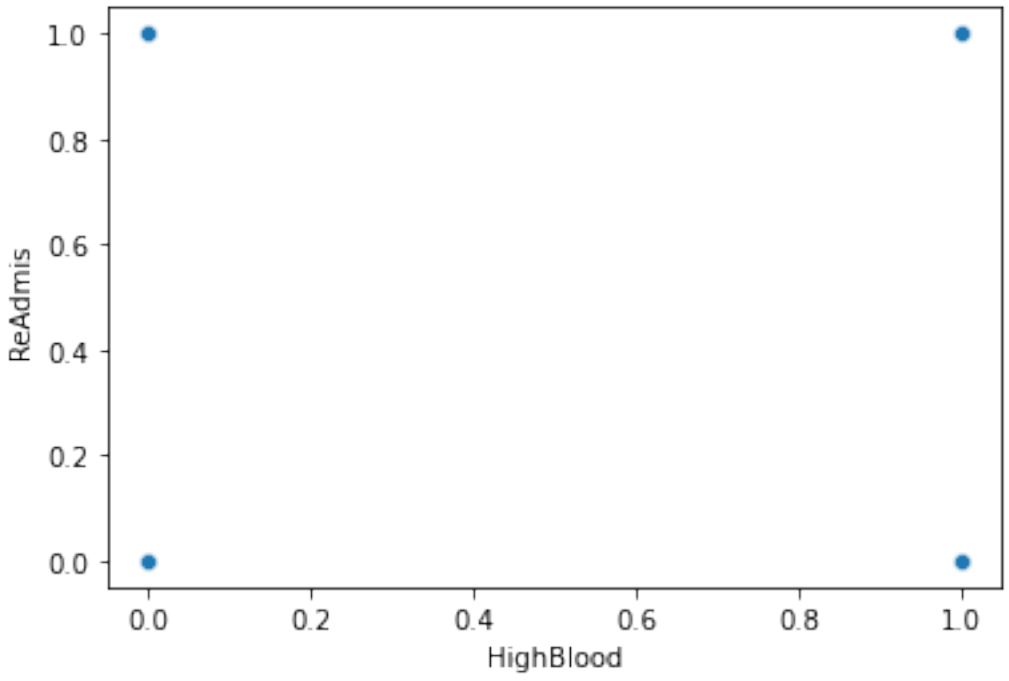
[154]:
```
sns.scatterplot(x = 'Soft_drink', y ='ReAdmis', data = cmd)
```

[154]: <AxesSubplot:xlabel='Soft_drink', ylabel='ReAdmis'>



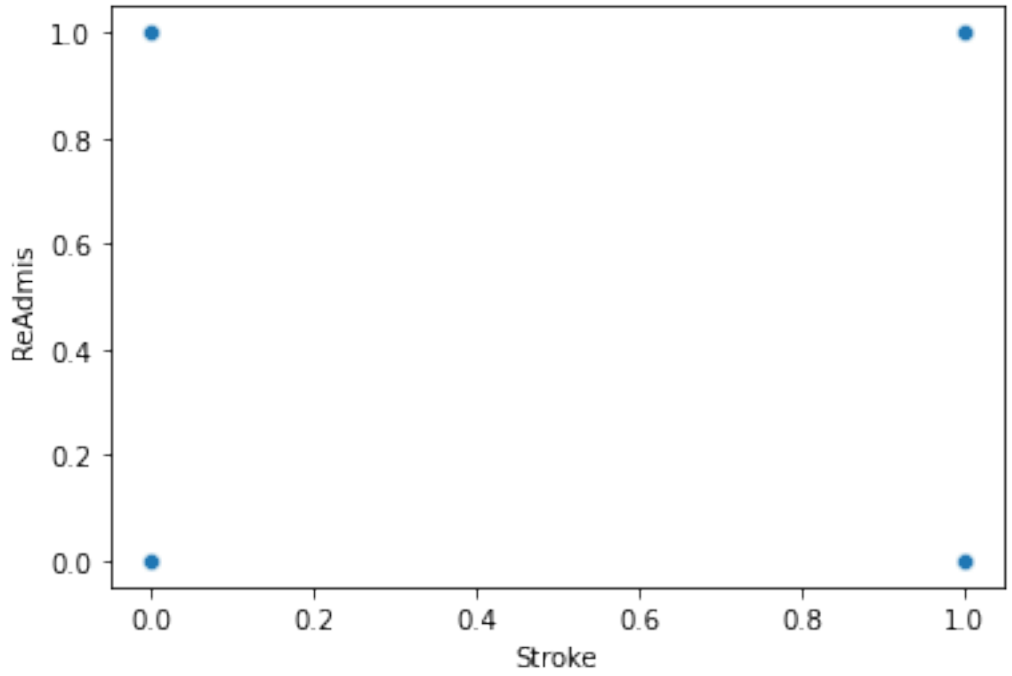
[155]:
```
sns.scatterplot(x = 'HighBlood', y ='ReAdmis', data = cmd)
```

[155]: <AxesSubplot:xlabel='HighBlood', ylabel='ReAdmis'>



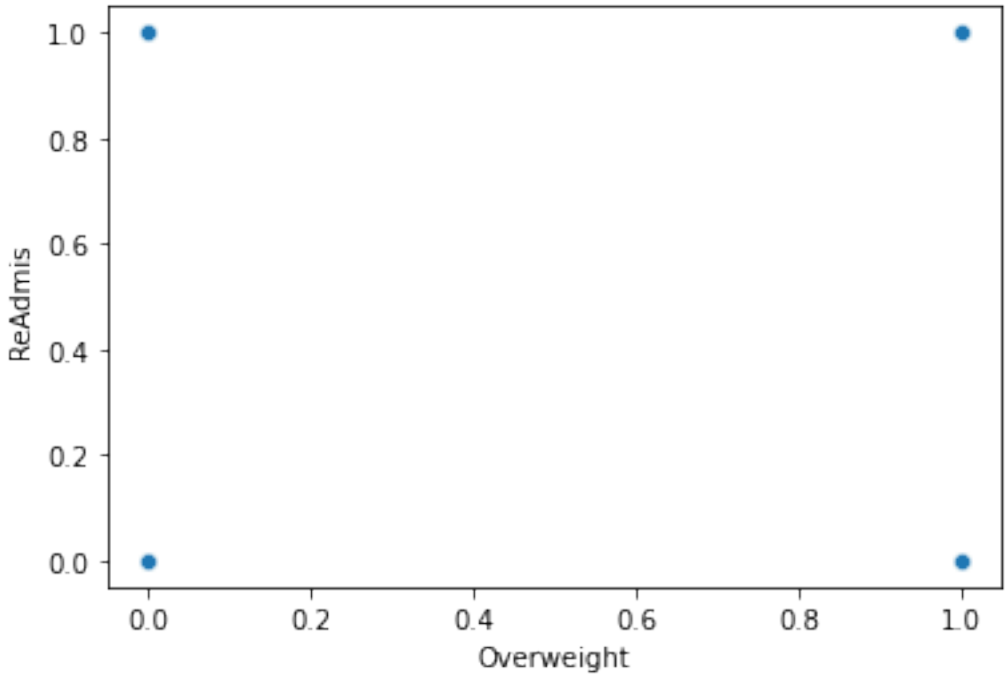
[156]:
```
sns.scatterplot(x = 'Stroke', y ='ReAdmis', data = cmd)
```

[156]: <AxesSubplot:xlabel='Stroke', ylabel='ReAdmis'>



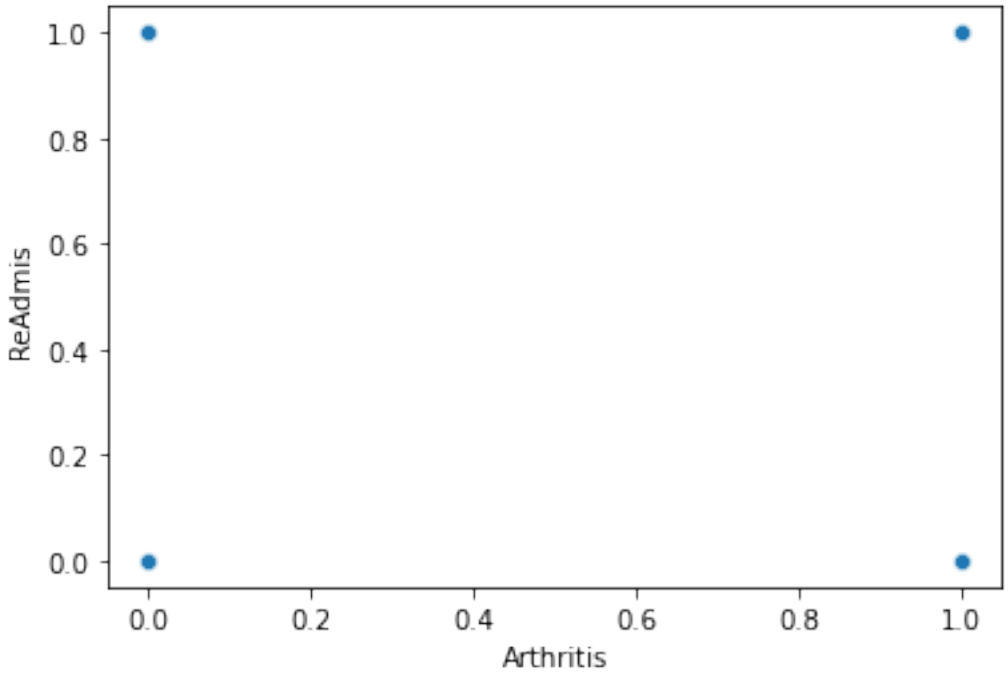
[157]:
```
sns.scatterplot(x = 'Overweight', y ='ReAdmis', data = cmd)
```

[157]: <AxesSubplot:xlabel='Overweight', ylabel='ReAdmis'>



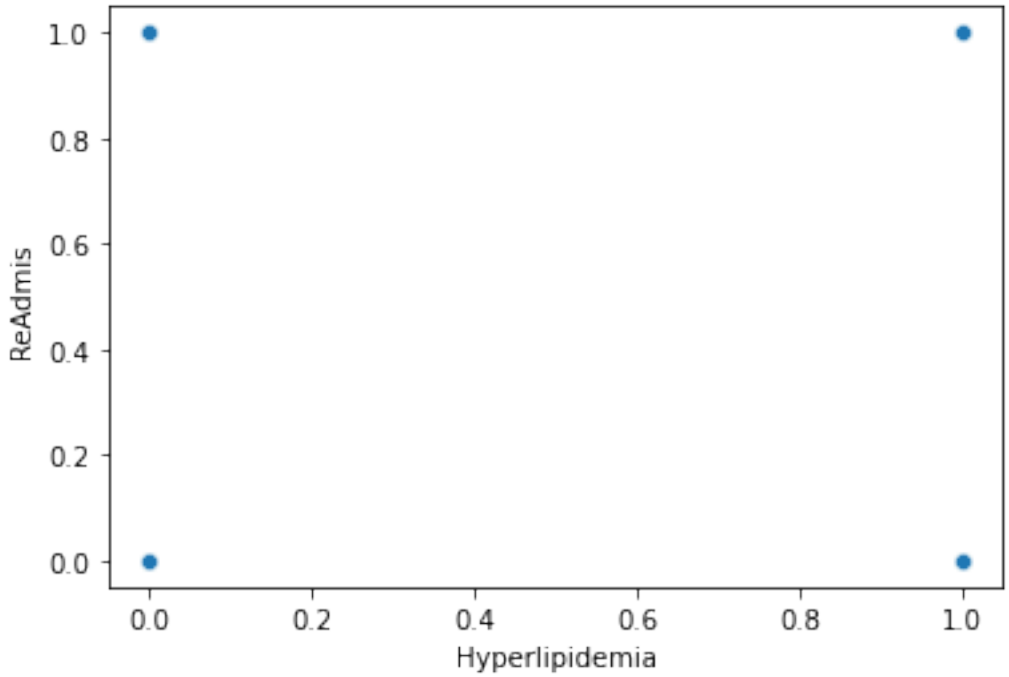
[158]:
```
sns.scatterplot(x = 'Arthritis', y ='ReAdmis', data = cmd)
```

[158]: <AxesSubplot:xlabel='Arthritis', ylabel='ReAdmis'>



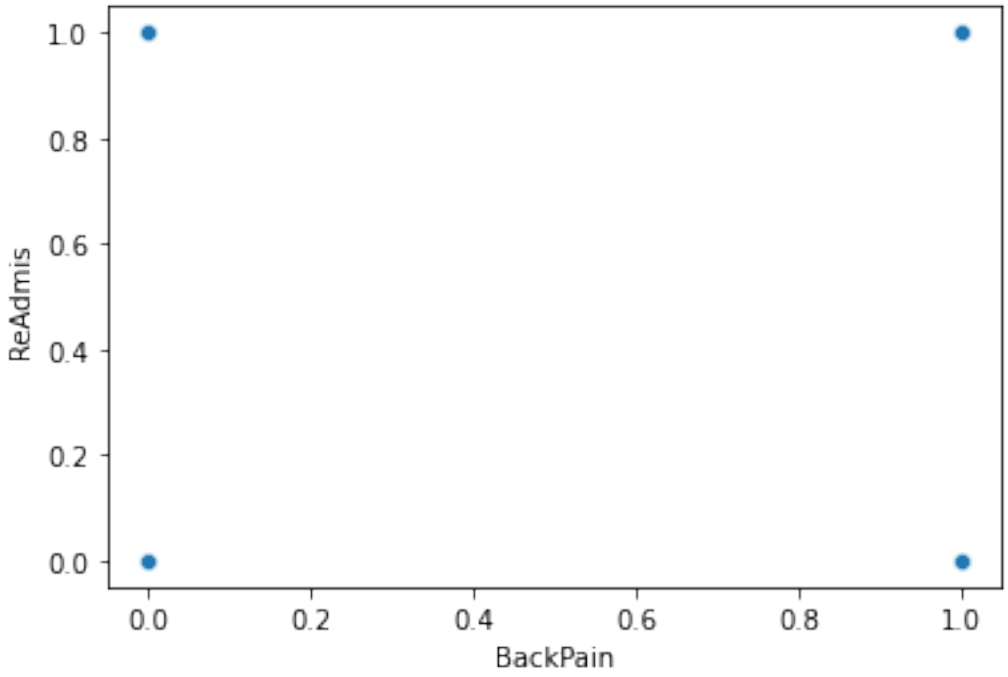
[159]:
```
sns.scatterplot(x = 'Hyperlipidemia', y ='ReAdmis', data = cmd)
```

[159]: <AxesSubplot:xlabel='Hyperlipidemia', ylabel='ReAdmis'>



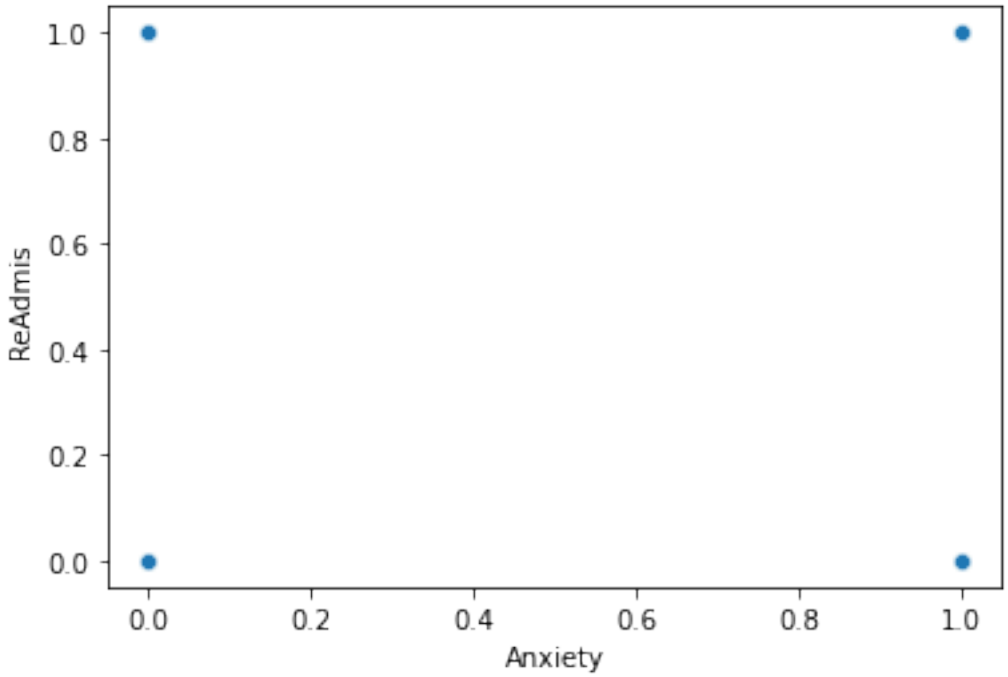
[160]:
```
sns.scatterplot(x = 'BackPain', y ='ReAdmis', data = cmd)
```

[160]: <AxesSubplot:xlabel='BackPain', ylabel='ReAdmis'>



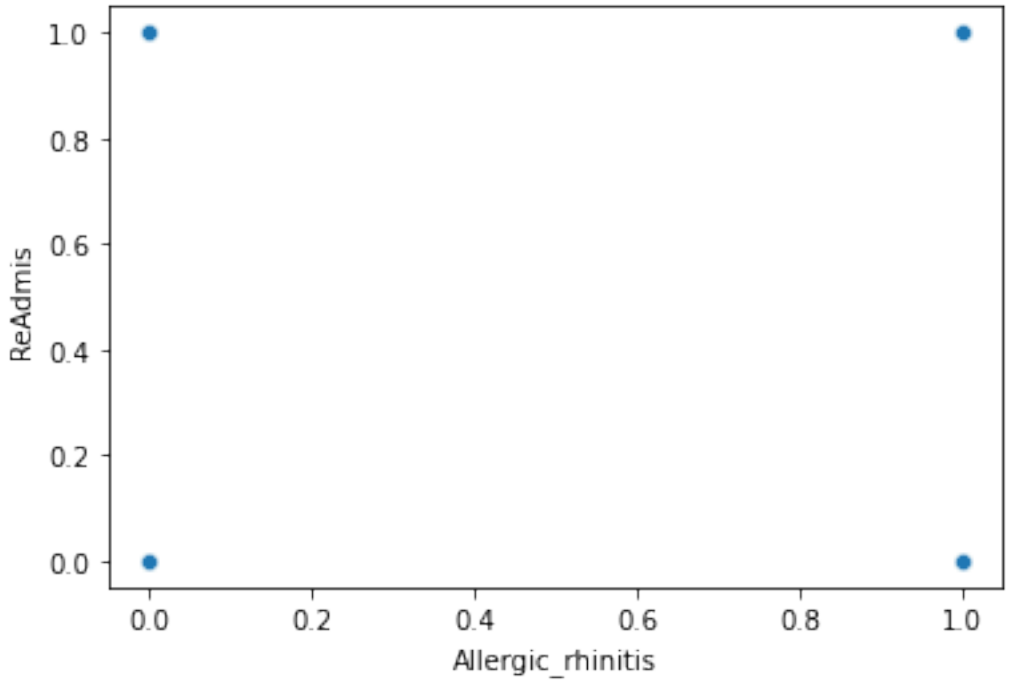
[161]:
```
sns.scatterplot(x = 'Anxiety', y ='ReAdmis', data = cmd)
```

[161]: <AxesSubplot:xlabel='Anxiety', ylabel='ReAdmis'>



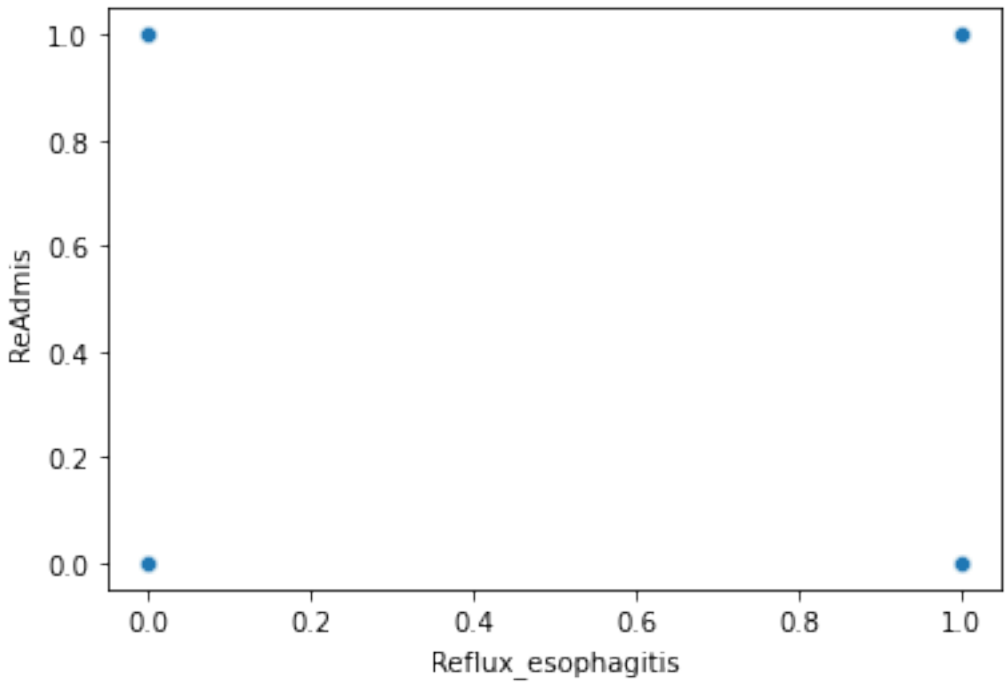
[162]:
```
sns.scatterplot(x = 'Allergic_rhinitis', y ='ReAdmis', data = cmd)
```

[162]: <AxesSubplot:xlabel='Allergic_rhinitis', ylabel='ReAdmis'>



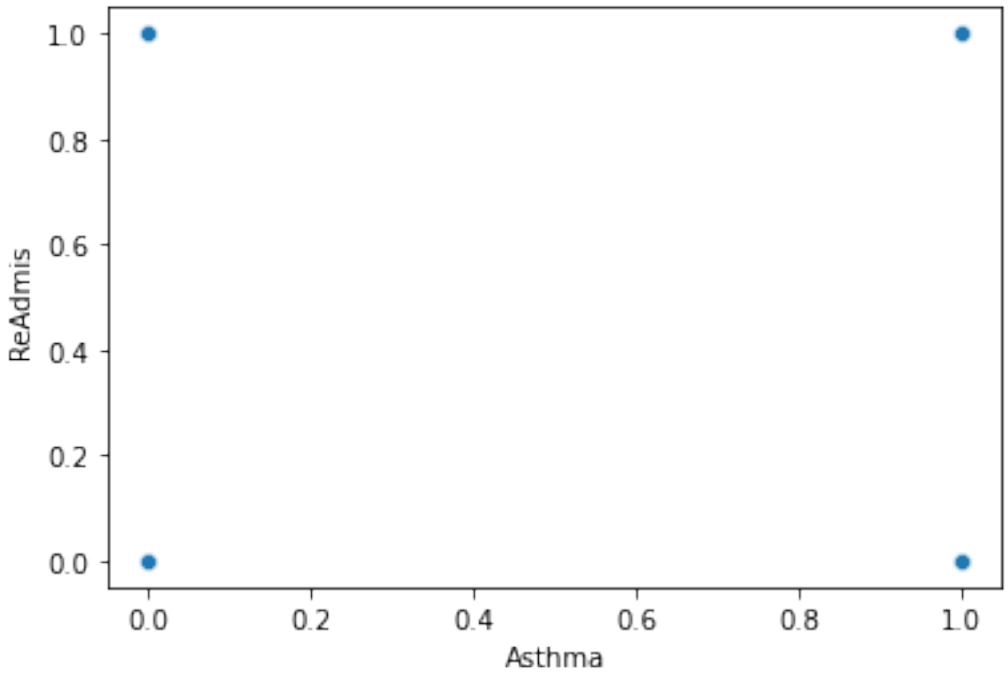
[163]:
```
sns.scatterplot(x = 'Reflux_esophagitis', y ='ReAdmis', data = cmd)
```

[163]: <AxesSubplot:xlabel='Reflux_esophagitis', ylabel='ReAdmis'>



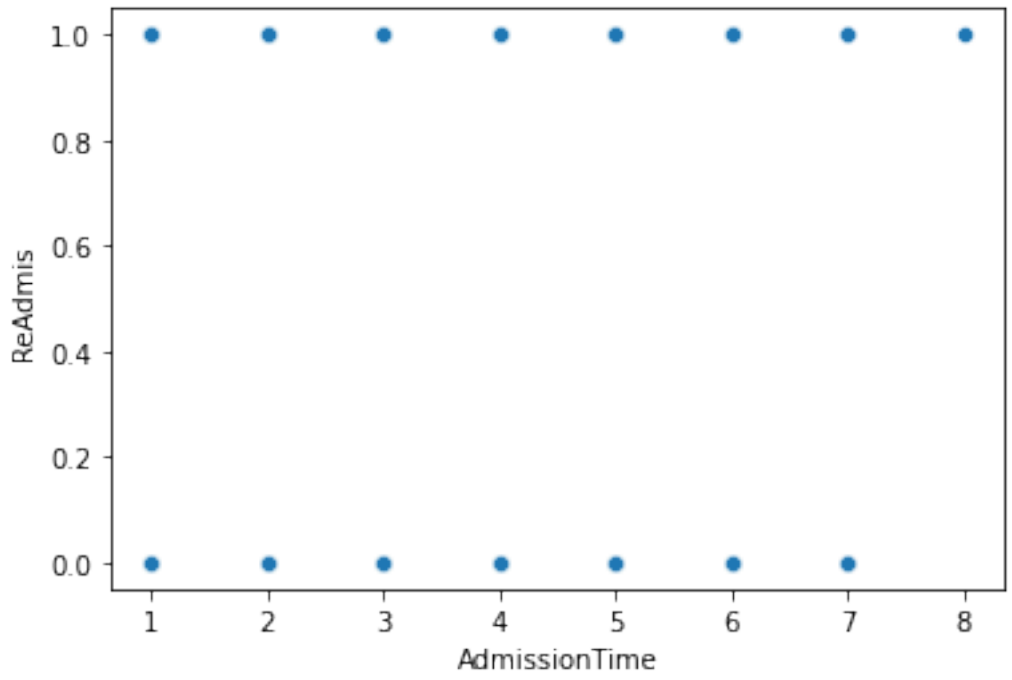
[164]:
```
sns.scatterplot(x = 'Asthma', y ='ReAdmis', data = cmd)
```

[164]: <AxesSubplot:xlabel='Asthma', ylabel='ReAdmis'>



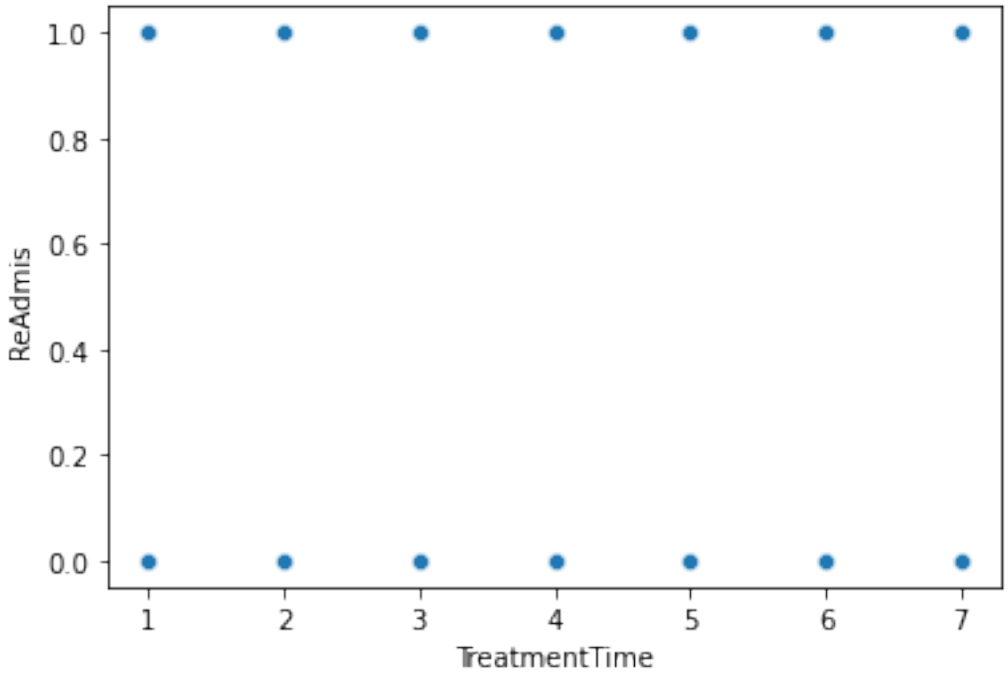
[165]:
```
sns.scatterplot(x = 'AdmissionTime', y ='ReAdmis', data = cmd)
```

[165]: <AxesSubplot:xlabel='AdmissionTime', ylabel='ReAdmis'>



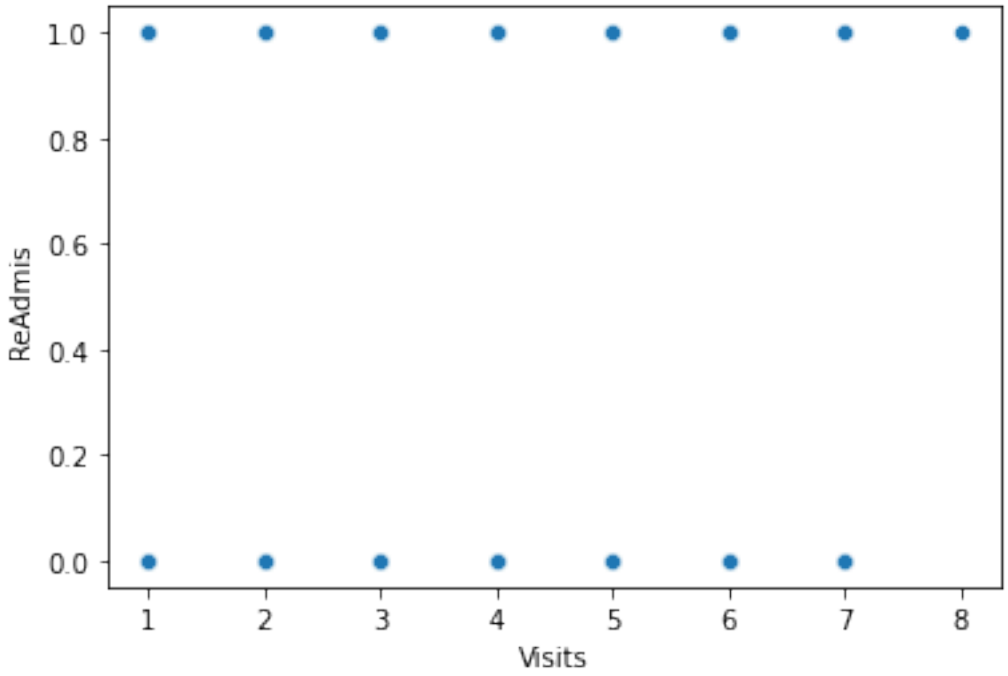
[166]:
```
sns.scatterplot(x = 'TreatmentTime', y ='ReAdmis', data = cmd)
```

[166]: <AxesSubplot:xlabel='TreatmentTime', ylabel='ReAdmis'>



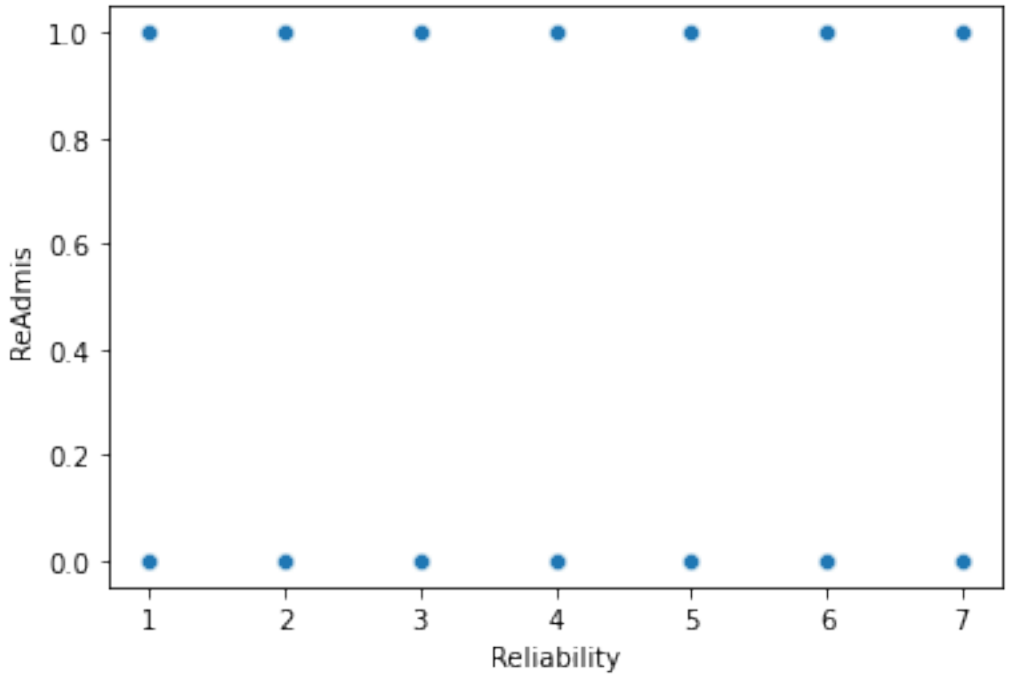
[167]:
```
sns.scatterplot(x = 'Visits', y ='ReAdmis', data = cmd)
```

[167]: <AxesSubplot:xlabel='Visits', ylabel='ReAdmis'>



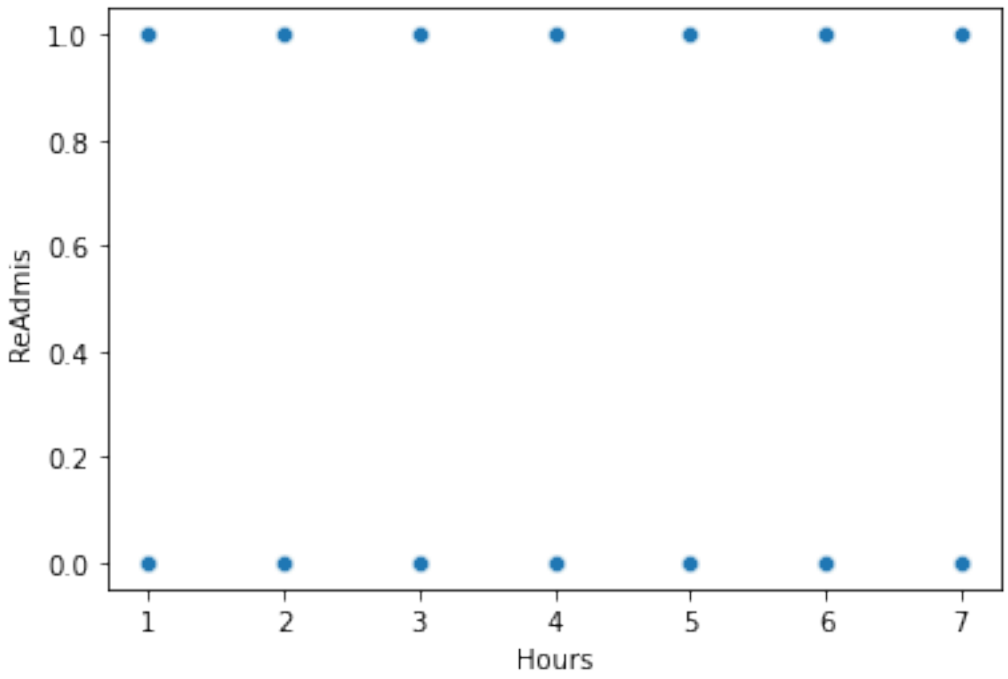
[168]:
```
sns.scatterplot(x = 'Reliability', y ='ReAdmis', data = cmd)
```

[168]: <AxesSubplot:xlabel='Reliability', ylabel='ReAdmis'>



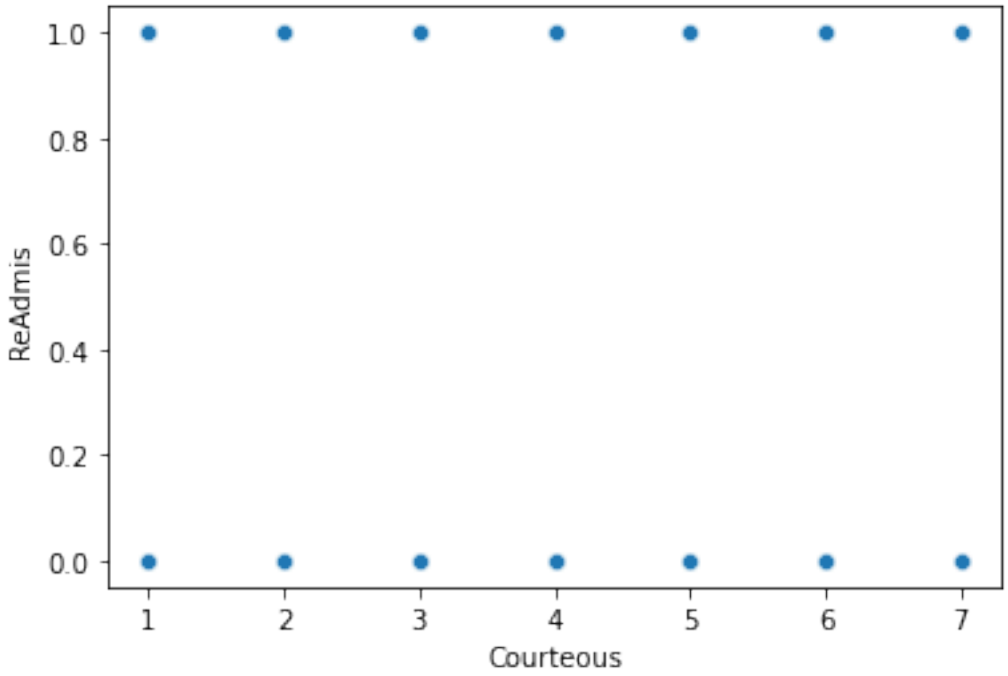
[169]:
```
sns.scatterplot(x = 'Hours', y ='ReAdmis', data = cmd)
```

[169]: <AxesSubplot:xlabel='Hours', ylabel='ReAdmis'>



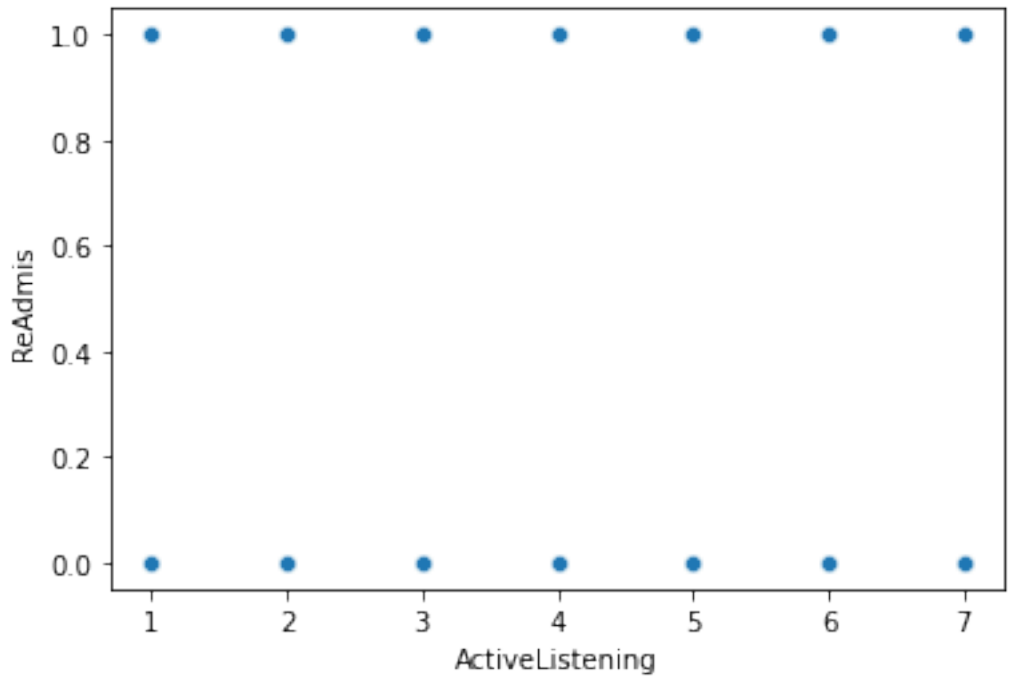
[170]:
```
sns.scatterplot(x = 'Courteous', y ='ReAdmis', data = cmd)
```

[170]:
```
<AxesSubplot:xlabel='Courteous', ylabel='ReAdmis'>
```



[171]:
```
sns.scatterplot(x = 'ActiveListening', y ='ReAdmis', data = cmd)
```

[171]: <AxesSubplot:xlabel='ActiveListening', ylabel='ReAdmis'>



10 Save Data Set

[172]:
```
cmd.to_csv('clean_data_208_T2.csv')
```

11 Initial Model

[173]:
```
# Import statsmodel for logit functionality
import statsmodels.api as sm
#create dependent and independent variables
x = cmd[['Doc_visits','Full_meals_eaten', 'VitD_supp', 'Initial_days',␣
 ↪'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis','Asthma',␣
 ↪'AdmissionTime','TreatmentTime','Visits','Reliability',␣
 ↪'Options','Hours','Courteous','ActiveListening','Soft_drink', 'HighBlood',␣
 ↪'Stroke','Overweight', 'Arthritis', 'Hyperlipidemia']]
y = cmd[['ReAdmis']]
x = add_constant(x)
# Create model and fit line
logReg = sm.Logit(y, x).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.226092
         Iterations 8
```

[174]:
```
# Summary of Model
print(logReg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                           Logit   Df Residuals:                     9976
Method:                            MLE   Df Model:                           23
Date:                 Tue, 04 Jan 2022   Pseudo R-squ.:                  0.6560
Time:                         12:22:03   Log-Likelihood:                -2260.9
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
======
                         coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const                 -7.5955      0.436    -17.435      0.000      -8.449
-6.742
Doc_visits             0.0570      0.037      1.552      0.121      -0.015
0.129
Full_meals_eaten       0.0194      0.038      0.511      0.609      -0.055
0.094
VitD_supp             -0.0843      0.059     -1.419      0.156      -0.201
0.032
Initial_days           0.1518      0.003     46.797      0.000       0.145
0.158
BackPain               0.0075      0.078      0.097      0.923      -0.145
0.160
Anxiety                0.0509      0.085      0.601      0.548      -0.115
```

```
0.217
Allergic_rhinitis     -0.2150      0.078     -2.745      0.006      -0.369
-0.061
Reflux_esophagitis    -0.0824      0.078     -1.058      0.290      -0.235
0.070
Asthma                -0.1004      0.086     -1.174      0.241      -0.268
0.067
AdmissionTime          0.0716      0.055      1.301      0.193      -0.036
0.179
TreatmentTime         -0.0146      0.051     -0.287      0.774      -0.115
0.085
Visits                -0.0114      0.047     -0.241      0.810      -0.104
0.081
Reliability            0.0631      0.042      1.504      0.133      -0.019
0.145
Options                0.0192      0.044      0.433      0.665      -0.068
0.106
Hours                 -0.0428      0.046     -0.931      0.352      -0.133
0.047
Courteous             -0.0394      0.042     -0.928      0.353      -0.123
0.044
ActiveListening       -0.0402      0.041     -0.989      0.323      -0.120
0.039
Soft_drink             0.0631      0.099      0.640      0.522      -0.130
0.257
HighBlood              0.1624      0.078      2.071      0.038       0.009
0.316
Stroke                 0.1606      0.096      1.681      0.093      -0.027
0.348
Overweight             0.0409      0.079      0.515      0.606      -0.115
0.196
Arthritis             -0.1248      0.080     -1.564      0.118      -0.281
0.032
Hyperlipidemia         0.0941      0.082      1.150      0.250      -0.066
0.255
==============================================================================
======
```

12 Reduced Model

[175]:
```
from statsmodels.tools import add_constant
# Set variables to identified variables only
x2 = cmd[['Initial_days', 'Allergic_rhinitis', 'HighBlood', 'Stroke']]
y2 = cmd[['ReAdmis']]
x2con = add_constant(x2)
# Create model and fit line
LogRegRed = sm.Logit(y2, x2con).fit()
```

```
print(LogRegRed.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.226987
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9995
Method:                           MLE   Df Model:                            4
Date:                Tue, 04 Jan 2022   Pseudo R-squ.:                  0.6547
Time:                        12:22:03   Log-Likelihood:                -2269.9
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
=====
                         coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----
const                -7.2866      0.175    -41.649      0.000      -7.630
-6.944
Initial_days          0.1510      0.003     46.948      0.000       0.145
0.157
Allergic_rhinitis    -0.2115      0.078     -2.709      0.007      -0.365
-0.058
HighBlood             0.1650      0.078      2.110      0.035       0.012
0.318
Stroke                0.1515      0.095      1.592      0.111      -0.035
0.338
==============================================================================
=====
```

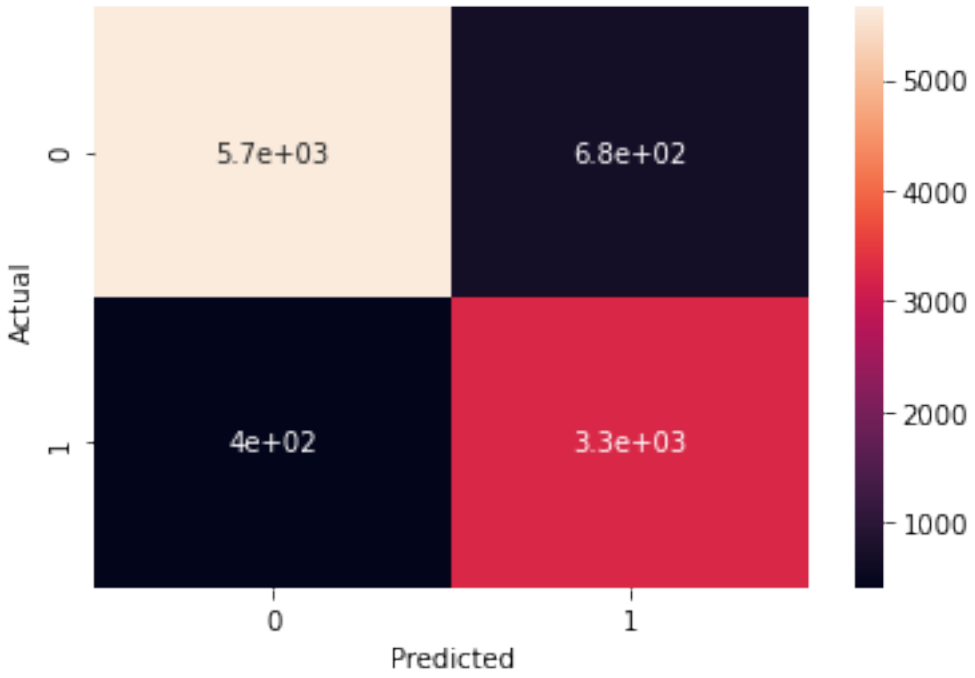
13 Confusion Matrix

[176]:
```
x2 = x2.astype('<U32')
x2.dtypes
```

[176]:
```
Initial_days         object
Allergic_rhinitis    object
HighBlood            object
Stroke               object
dtype: object
```

[177]:
```
# Import necessary libraries
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

```
import seaborn as sn
# create data frame of actual and predicted outcomes
y_actual = cmd['ReAdmis'].astype(int)
y_predicted = LogRegRed.predict(x2con)
y_predicted = y_predicted.round()
y_predicted = y_predicted.astype(int)
data = [y_actual, y_predicted]

headers = ["y_actual", "y_predicted"]

df3 = pd.concat(data, axis=1, keys=headers)

confusion_matrix = pd.crosstab(df3['y_actual'], df3['y_predicted'],␣
↪rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion_matrix, annot=True)
plt.show()
```



14 Part IV: Model Comparison

14.1 D2: Justify Variable Selection

For the reduced model, we will include any variable with a p-value less than .1. This value was selected as it is a common cut off value for statistical significance. For the reduced model we will continue with the following predictor variables: initial days of admission, stroke, allergic rhinitis,

and high blood pressure. We can see that there are 3 variables around pre-existing conditions in this list. ## E1: Discussion of Initial and Reduced Models The initial model had a pseudo-R-squared value of 0.656. This means that we were able to explain 66% of variance through the regression equation that was created. From here, we removed the variables that did not meet the threshold of $p < 0.1$. This resulted in us going from 24 predictor variables down to only 4. The reduced model produced a pseudo-R-squared value of 0.655. While there is no statistically relevant increase or decrease of accuracy from the reduced model, by reducing the number of variables we can reduce the possibility of randomness impacting our reduced model while keeping the R-squared the same. # Part V: Data Summary and Implications ## Equation of Reduced Model The regression equation for the reduced model is: y = -7.286 + (0.151 * initial admission) + (-0.211 * Allergic Rhinitis) + (0.165 * High blood pressure) + (0.151 * Stroke)

14.2 Interpretation of Coefficients and Statistical Significance

For our reduced model we see that allergic rhinitis has a negative coefficient while the other 3 predictors have a positive coefficient. It would appear from the model that initial length of stay is the largest determining factor in whether the patient is readmitted. With it being the only continuous variable, it will drastically overshadow the categorical predictors. The overall model explains nearly 66% of variance in whether a patient is likely to be readmitted.

This model is currently limited by the type and amount of data we have collected. I believe it would be beneficial to refine our survey questions further, possibly reducing the number of options so that we may get a clearer picture of our patient's experience.

14.3 Recommendation

I would recommend that we begin focusing our time on patients who have had extended stays and have either high blood pressure or a previous stroke in their medical records. This model would indicate that we are completely addressing the patient's condition during extended stays, which in turn is resulting in readmission and fines to our hospital. We could assemble a team of subject matter experts to review this report and provide recommendations on possible modification to current procedures for providers, that could reduce the number of fines in the future.

15 Annotations

Assumptions of logistic regression. Statistics Solutions. (2021, August 11). Retrieved December 29, 2021, from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/

Chantal D. Larose, & Daniel T. Larose. (2019). Data Science Using Python and R. Wiley.

Pandas Development Team. (2008). pandas.DataFrame.drop — pandas 1.3.0 documentation. Pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.

Python - seaborn.residplot() method. GeeksforGeeks. (2020, August 17). Retrieved December 26, 2021, from https://www.geeksforgeeks.org/python-seaborn-residplot-method/

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Shin, T. (2021, December 4). Understanding multicollinearity and how to detect it in Python.

Medium. Retrieved December 18, 2021, from https://towardsdatascience.com/everything-you-need-to-know-about-multicollinearity-2f21f082d6dc

[ ]: