In [1]:
```
# import packages that will be used for the logistics regression analysis
import pylab
import seaborn as sb
sb.set(style="white")
sb.set(style="whitegrid", color_codes=True)
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import classification_report
from sklearn import metrics
import matplotlib.pyplot as plt
plt.rc("font", size=14)
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from IPython.core.display import HTML
from IPython.display import display
import pandas as pd
from pandas import Series, DataFrame
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE

# import data set that will be used for the logistics regression analysis
pd.set_option('display.max_columns', None)
df = pd.read_csv (r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_clean.csv')

# rename the item columns accordingly
df.rename(columns={'Item1':'Timely_admis','Item2':'Timely_treat',
 'Item3':'Timely_visits','Item4':'Reliability',
 'Item5':'Options','Item6':'Hrs_treat',
 'Item7':'Courteous','Item8':'Active_listen'},inplace=True)
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
```

```
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Timely_admis        10000 non-null  int64
 43  Timely_treat        10000 non-null  int64
 44  Timely_visits       10000 non-null  int64
 45  Reliability         10000 non-null  int64
 46  Options             10000 non-null  int64
 47  Hrs_treat           10000 non-null  int64
 48  Courteous           10000 non-null  int64
 49  Active_listen       10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [2]:

```
# drop all the demographic columns we don't need for this logistics regression analysis
df.drop(['City','State','County','Area','Zip','Lat','Lng','Population','TimeZone','Additional_charges','TotalCharge','I
# verify that all the columns were dropped before proceeding
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 34 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Children            10000 non-null  int64  
 1   Age                 10000 non-null  int64  
 2   Income              10000 non-null  float64
 3   Marital             10000 non-null  object 
 4   Gender              10000 non-null  object 
 5   ReAdmis             10000 non-null  object 
 6   VitD_levels         10000 non-null  float64
 7   Doc_visits          10000 non-null  int64  
 8   Full_meals_eaten    10000 non-null  int64  
 9   vitD_supp           10000 non-null  int64  
 10  Soft_drink          10000 non-null  object 
 11  Initial_admin       10000 non-null  object 
 12  HighBlood           10000 non-null  object 
 13  Stroke              10000 non-null  object 
 14  Complication_risk   10000 non-null  object 
 15  Overweight          10000 non-null  object 
 16  Arthritis           10000 non-null  object 
 17  Diabetes            10000 non-null  object 
 18  Hyperlipidemia      10000 non-null  object 
 19  BackPain            10000 non-null  object 
 20  Anxiety             10000 non-null  object 
 21  Allergic_rhinitis   10000 non-null  object 
 22  Reflux_esophagitis  10000 non-null  object 
 23  Asthma              10000 non-null  object 
 24  Services            10000 non-null  object 
 25  Initial_days        10000 non-null  float64
 26  Timely_admis        10000 non-null  int64  
 27  Timely_treat        10000 non-null  int64  
 28  Timely_visits       10000 non-null  int64  
 29  Reliability         10000 non-null  int64  
 30  Options             10000 non-null  int64  
 31  Hrs_treat           10000 non-null  int64  
 32  Courteous           10000 non-null  int64  
 33  Active_listen       10000 non-null  int64  
dtypes: float64(3), int64(13), object(18)
memory usage: 2.6+ MB
```

In [3]:
```
#check if there is any duplicate data entries present in columns
df[df.duplicated()]
```

Out[3]:

Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_admin	HighBlood

In [4]:

```
# check if there are any duplicated columns in the data set - if there are none then the output should be False
df.columns.duplicated().any()
```

Out[4]: False

In [5]:

```
# check if there are any duplicated rows in the data set - if there are none then the output should be False
df.duplicated().any()
```

Out[5]: False

In [6]:

```
# convert categorical yes/no values to numeric 1/0 values
df = df.replace(to_replace = ['Yes','No'],value = [1,0])
df
```

Out[6]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_admin	H
0	1	53	86575.93	Divorced	Male	0	19.141466	6	0	0	0	Emergency Admission	
1	3	51	46805.99	Married	Female	0	18.940352	4	2	1	0	Emergency Admission	
2	3	53	14370.14	Widowed	Female	0	18.057507	4	1	0	0	Elective Admission	
3	0	78	39741.49	Married	Male	0	16.576858	4	1	0	0	Elective Admission	
4	1	22	1209.56	Widowed	Female	0	17.439069	5	0	2	1	Elective Admission	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	2	25	45967.61	Widowed	Male	0	16.980860	4	2	1	0	Emergency Admission	
9996	4	87	14983.02	Widowed	Male	1	18.177020	5	0	0	0	Elective Admission	
9997	3	45	65917.81	Separated	Female	1	17.129070	4	2	0	1	Elective Admission	
9998	3	43	29702.32	Divorced	Male	1	19.910430	5	2	1	0	Emergency Admission	
9999	8	70	62682.63	Separated	Female	1	18.388620	5	0	1	0	Observation Admission	

10000 rows × 34 columns

In [7]:

```
# convert the non-married Marital status values to "Married/Not Married", then convert "Married/Not Married" to "1/0"
#this will make the Marital variable easier to work with during regression analysis
df['Marital'] = df['Marital'].replace(['Divorced','Widowed','Separated','Never Married'],'Not Married')
df['Marital'] = df['Marital'].replace(['Married','Not Married'],[1,0])
df
```

Out[7]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_admin	Higl
0	1	53	86575.93	0	Male	0	19.141466	6	0	0	0	Emergency Admission	
1	3	51	46805.99	1	Female	0	18.940352	4	2	1	0	Emergency Admission	
2	3	53	14370.14	0	Female	0	18.057507	4	1	0	0	Elective Admission	
3	0	78	39741.49	1	Male	0	16.576858	4	1	0	0	Elective Admission	
4	1	22	1209.56	0	Female	0	17.439069	5	0	2	1	Elective Admission	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	2	25	45967.61	0	Male	0	16.980860	4	2	1	0	Emergency Admission	
9996	4	87	14983.02	0	Male	1	18.177020	5	0	0	0	Elective Admission	
9997	3	45	65917.81	0	Female	1	17.129070	4	2	0	1	Elective Admission	
9998	3	43	29702.32	0	Male	1	19.910430	5	2	1	0	Emergency Admission	
9999	8	70	62682.63	0	Female	1	18.388620	5	0	1	0	Observation Admission	

10000 rows × 34 columns

In [8]:

```
# Showcase the unique values for the Services variable
df['Gender'].unique()
```

Out[8]:

```
array(['Male', 'Female', 'Nonbinary'], dtype=object)
```

In [9]:

```
#convert the non-Female gender values to "Female/non-female", then convert "Female/non-female" to "1/0"
df['Gender'] = df['Gender'].replace(['Male','Nonbinary'],'non-female')
df['Gender'] = df['Gender'].replace(['Female','non-female'],[1,0])
df
```

Out[9]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_admin	Higl
0	1	53	86575.93	0	0	0	19.141466	6	0	0	0	Emergency Admission	
1	3	51	46805.99	1	1	0	18.940352	4	2	1	0	Emergency Admission	
2	3	53	14370.14	0	1	0	18.057507	4	1	0	0	Elective Admission	
3	0	78	39741.49	1	0	0	16.576858	4	1	0	0	Elective Admission	
4	1	22	1209.56	0	1	0	17.439069	5	0	2	1	Elective Admission	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	2	25	45967.61	0	0	0	16.980860	4	2	1	0	Emergency Admission	
9996	4	87	14983.02	0	0	1	18.177020	5	0	0	0	Elective Admission	
9997	3	45	65917.81	0	1	1	17.129070	4	2	0	1	Elective Admission	
9998	3	43	29702.32	0	0	1	19.910430	5	2	1	0	Emergency Admission	
9999	8	70	62682.63	0	1	1	18.388620	5	0	1	0	Observation Admission	

10000 rows × 34 columns

In [10]:
```
# Showcase the unique values for the Services variable
df['Services'].unique()
```

Out[10]:
```
array(['Blood Work', 'Intravenous', 'CT Scan', 'MRI'], dtype=object)
```

In [11]:
```
# Drop the services variable since these values cannot be condensed
df.drop(['Services'],axis = 1,inplace=True)
```

In [12]:
```
# Showcase the unique values for the Complication_risk variable
df['Complication_risk'].unique()
```

Out[12]:
```
array(['Medium', 'High', 'Low'], dtype=object)
```

In [13]:
```
# Drop the services variable since these values cannot be condensed
df.drop(['Complication_risk'],axis = 1,inplace=True)
```

In [14]:
```
# Showcase the unique values for the Initial_admin variable
df['Initial_admin'].unique()
```

Out[14]:
```
array(['Emergency Admission', 'Elective Admission',
       'Observation Admission'], dtype=object)
```

In [15]:
```
# convert the non-emergency admission status values to "Emergency Admission/non-Emergency Admission", then convert "Eme
#this will make the Marital variable easier to work with during regression analysis
df['Initial_admin'] = df['Initial_admin'].replace(['Elective Admission','Observation Admission'],'non-Emergency Admissi
df['Initial_admin'] = df['Initial_admin'].replace(['Emergency Admission','non-Emergency Admission'],[1,0])
df
```

Out[15]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_admin	Higl
0	1	53	86575.93	0	0	0	19.141466	6	0	0	0	1	
1	3	51	46805.99	1	1	0	18.940352	4	2	1	0	1	
2	3	53	14370.14	0	1	0	18.057507	4	1	0	0	0	
3	0	78	39741.49	1	0	0	16.576858	4	1	0	0	0	
4	1	22	1209.56	0	1	0	17.439069	5	0	2	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	2	25	45967.61	0	0	0	16.980860	4	2	1	0	1	
9996	4	87	14983.02	0	0	1	18.177020	5	0	0	0	0	
9997	3	45	65917.81	0	1	1	17.129070	4	2	0	1	0	
9998	3	43	29702.32	0	0	1	19.910430	5	2	1	0	1	
9999	8	70	62682.63	0	1	1	18.388620	5	0	1	0	0	

10000 rows × 32 columns

In [16]:

```
# describe the dataframe and showcase summary statistics of the variables
df.describe()
```

Out[16]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	2.097200	53.511700	40490.495160	0.202300	0.501800	0.366900	17.964262	5.012200	1.001400
std	2.163659	20.638538	28521.153293	0.401735	0.500022	0.481983	2.017231	1.045734	1.008117
min	0.000000	18.000000	154.080000	0.000000	0.000000	0.000000	9.806483	1.000000	0.000000
25%	0.000000	36.000000	19598.775000	0.000000	0.000000	0.000000	16.626439	4.000000	0.000000
50%	1.000000	53.000000	33768.420000	0.000000	1.000000	0.000000	17.951122	5.000000	1.000000
75%	3.000000	71.000000	54296.402500	0.000000	1.000000	1.000000	19.347963	6.000000	2.000000
max	10.000000	89.000000	207249.100000	1.000000	1.000000	1.000000	26.394449	9.000000	7.000000

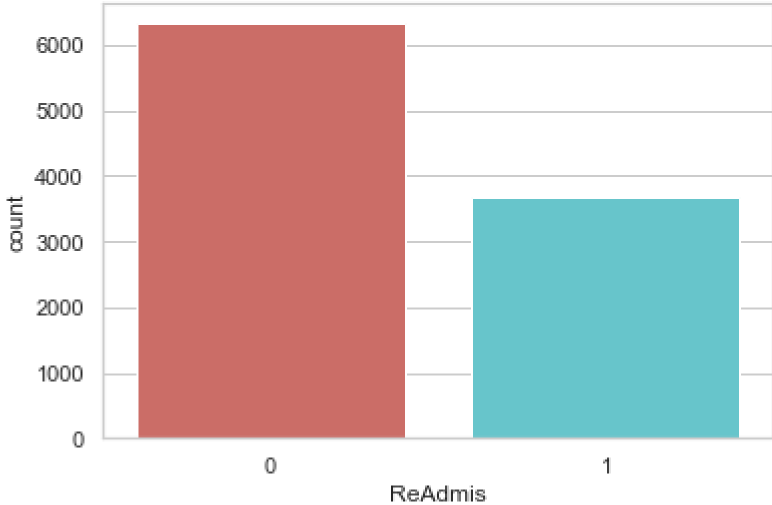
In [17]:

```
# now that all the modifications have been made, export the prepared dataset
df.to_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d208\2medical_clean-PREPAREDTASK2_12-24-2022.csv')
```

In [18]:

```
# Start to visualize the data, including univariate and bivariate analyses
# begin by visualizing the target variable, ReAdmis
print(df['ReAdmis'].value_counts())
sb.countplot(x='ReAdmis', data=df, palette='hls')
plt.show()
```

```
0    6331
1    3669
Name: ReAdmis, dtype: int64
```

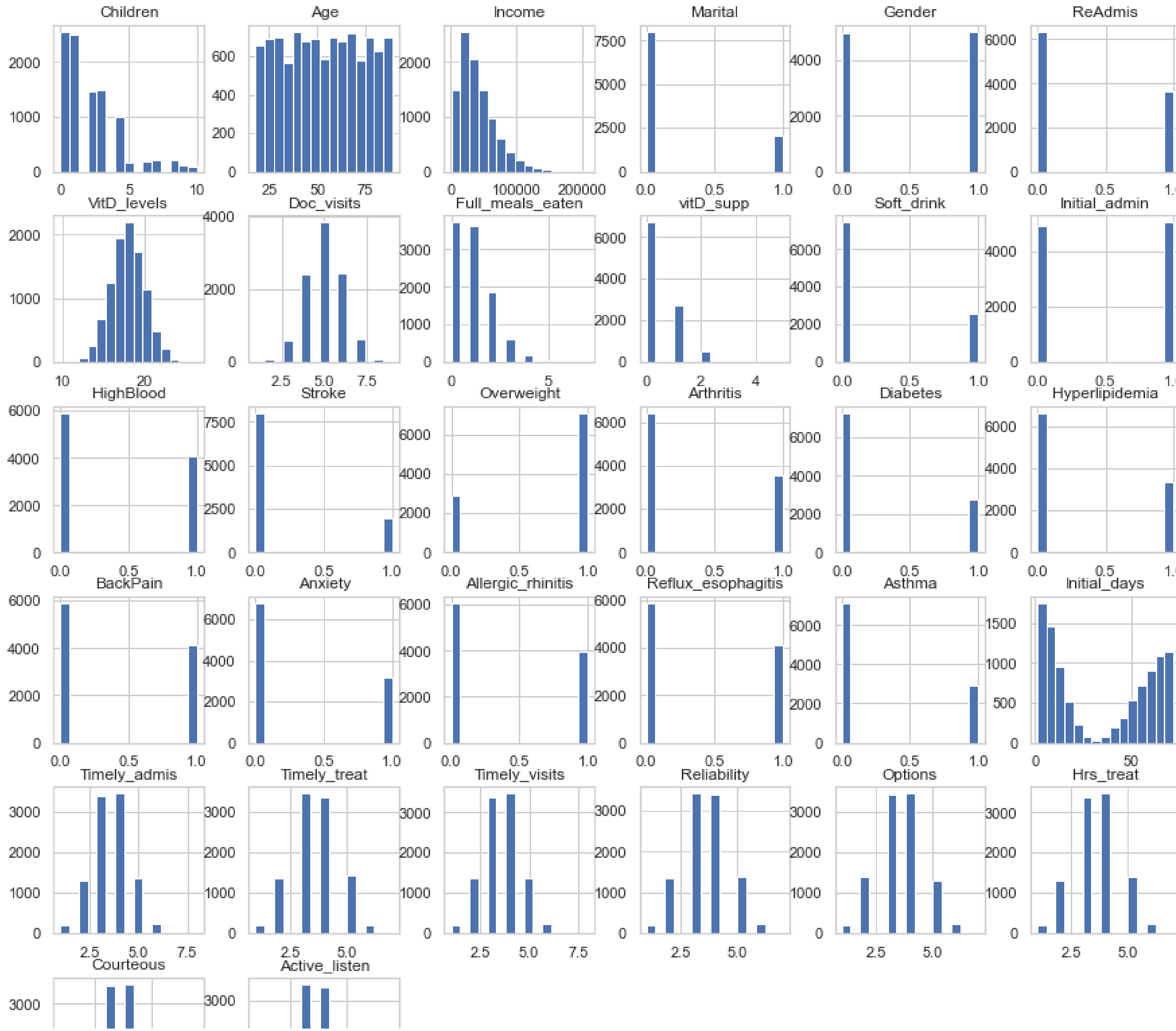


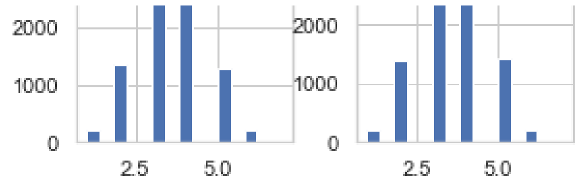
In [19]:
```
# identify the columns for variables
Variables = df.select_dtypes(include = "number").columns
print (Variables)
```

```
Index(['Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke', 'Overweight',
       'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Initial_days',
       'Timely_admis', 'Timely_treat', 'Timely_visits', 'Reliability',
       'Options', 'Hrs_treat', 'Courteous', 'Active_listen'],
      dtype='object')
```

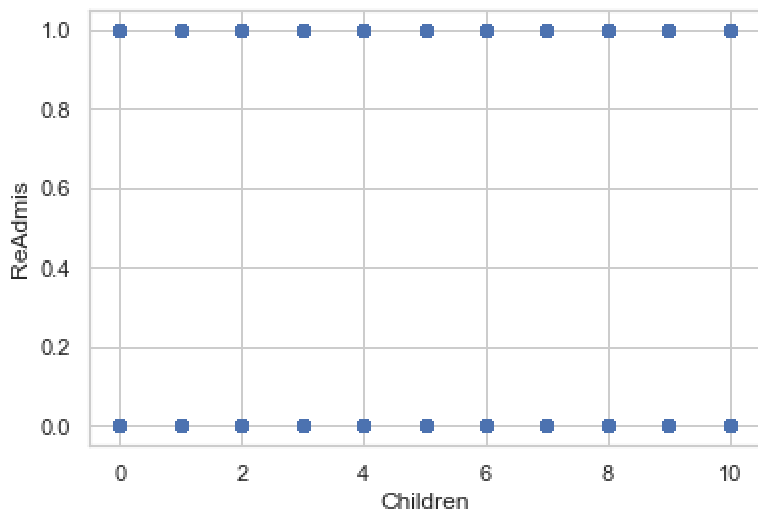
In [20]:
```
# create histogram plots of the identified predictor variables
fig = plt.figure(figsize=(15, 15))
ax = df[Variables].hist(bins = 15, figsize=(15,15))
plt.title('Variables')
fig.tight_layout(h_pad=5, w_pad=5)
plt.show()
```

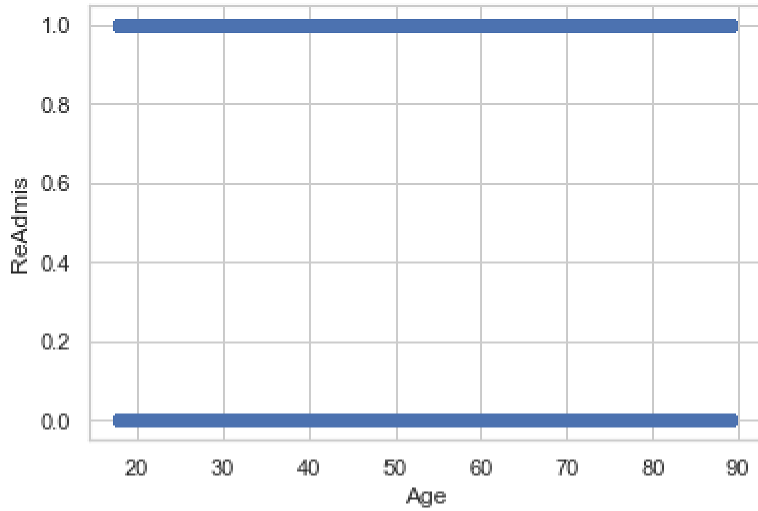
```
<Figure size 1080x1080 with 0 Axes>
```

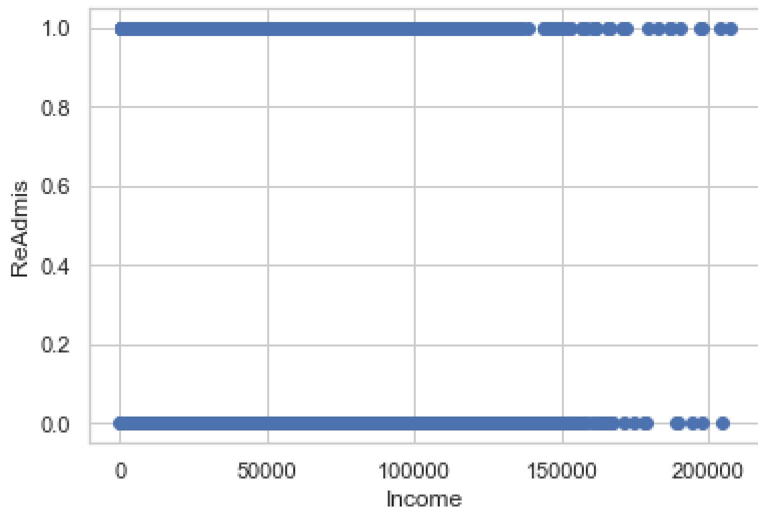

Children
Age
Income
Marital
Gender
ReAdmis
VitD_levels
Doc_visits
Full_meals_eaten
vitD_supp
Soft_drink
Initial_admin
HighBlood
Stroke
Overweight
Arthritis
Diabetes
Hyperlipidemia
BackPain
Anxiety
Allergic_rhinitis
Reflux_esophagitis
Asthma
Initial_days
Timely_admis
Timely_treat
Timely_visits
Reliability
Options
Hrs_treat
Courteous
Active_listen

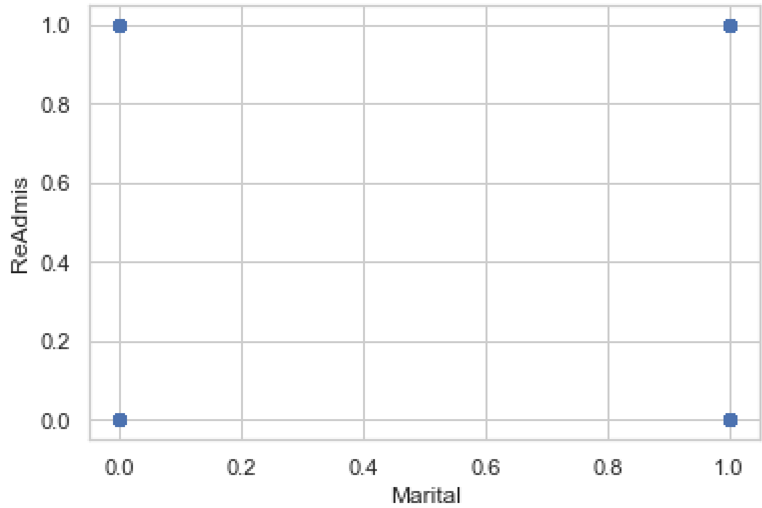


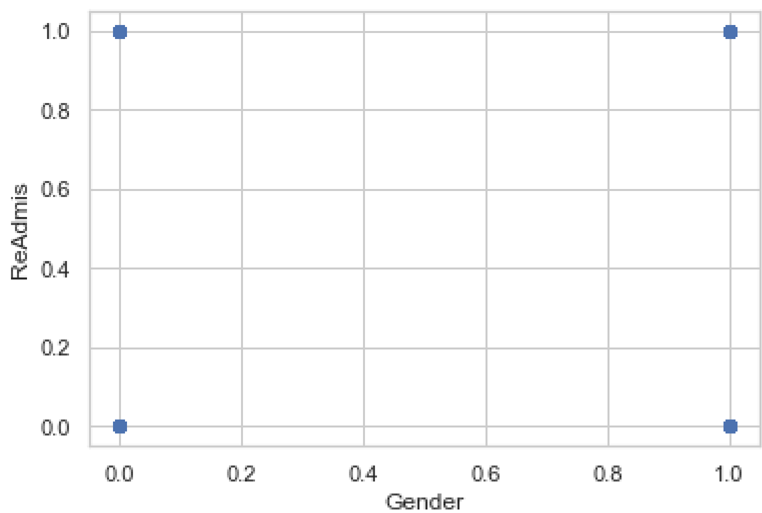
In [21]:
```
#selecting the target varialbe and showcasing the bivariate statistics

X=df.drop('ReAdmis',inplace=False, axis=1)
y=df['ReAdmis']

for column in X.columns:
    plt.scatter(X[column],y)
    plt.xlabel(column)
    plt.ylabel('ReAdmis')
    plt.show()
```

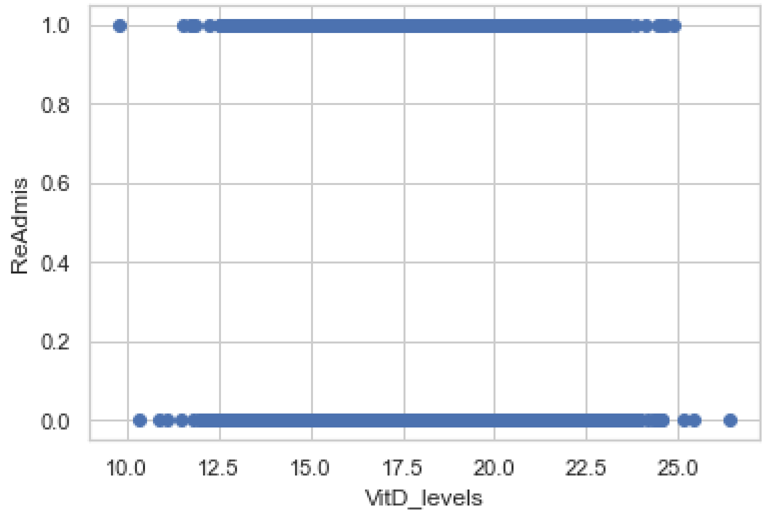



ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
20
30
40
50
60
70
80
90
Age

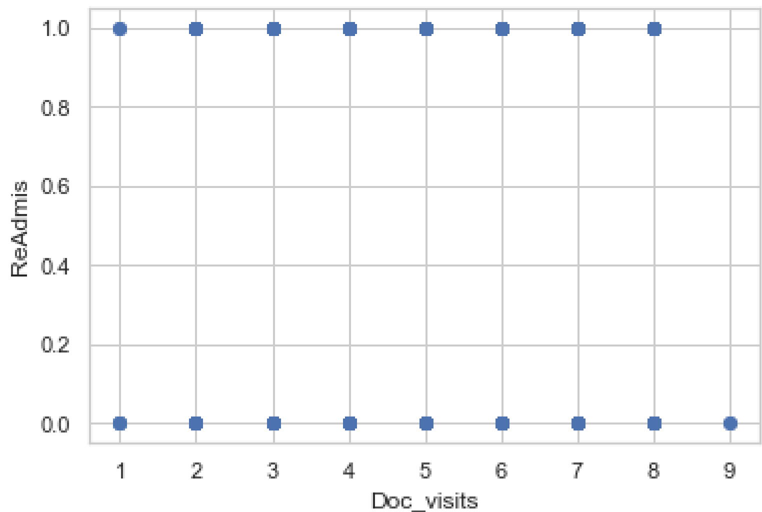

ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
Income


ReAdmis
Marital
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

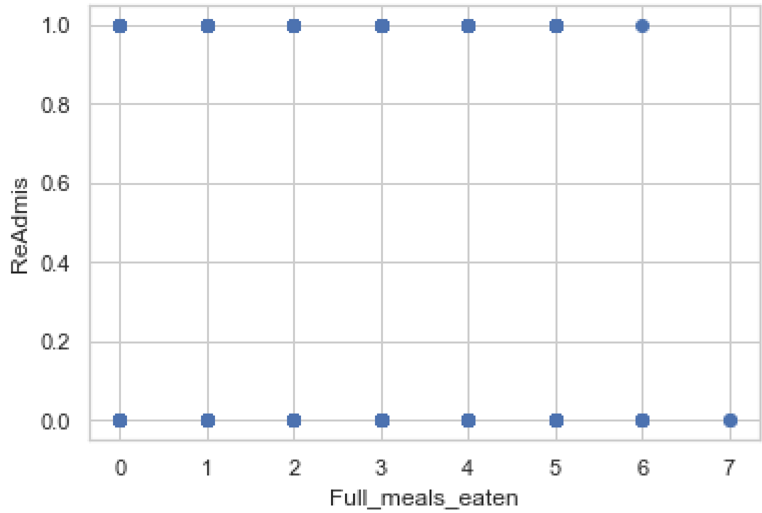

ReAdmis
Gender
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

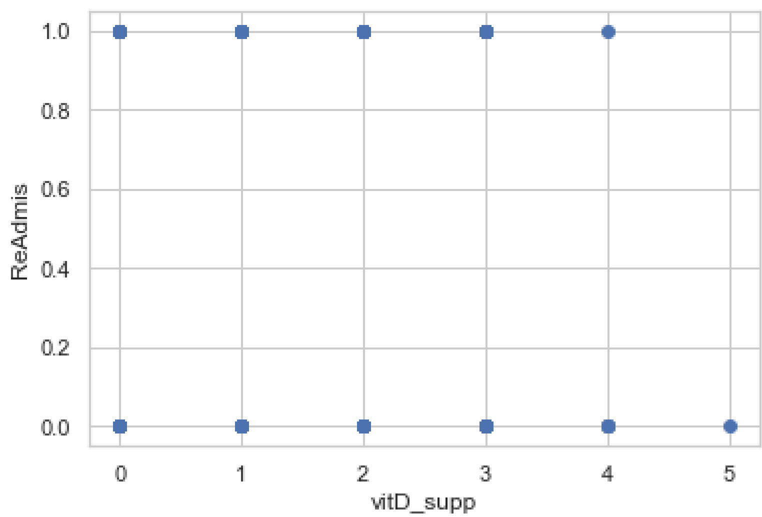


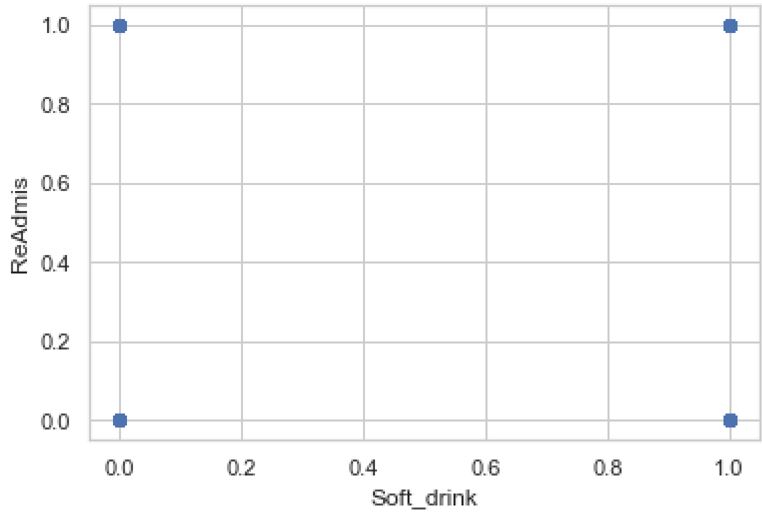
ReAdmis
VitD_levels

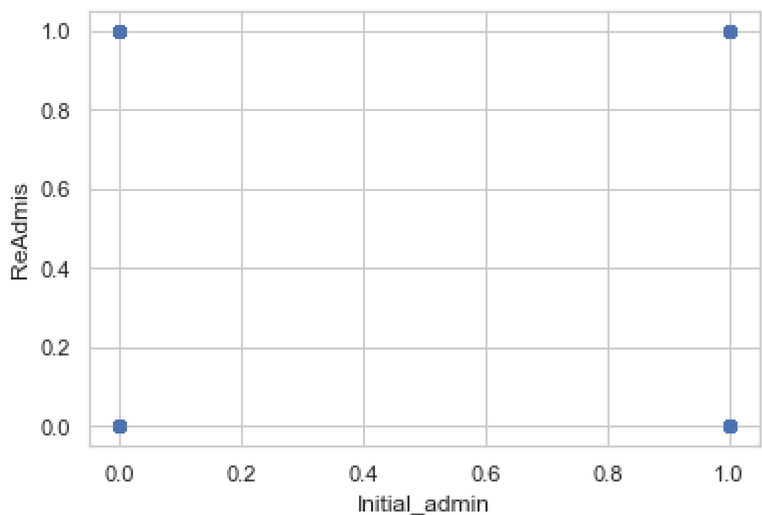


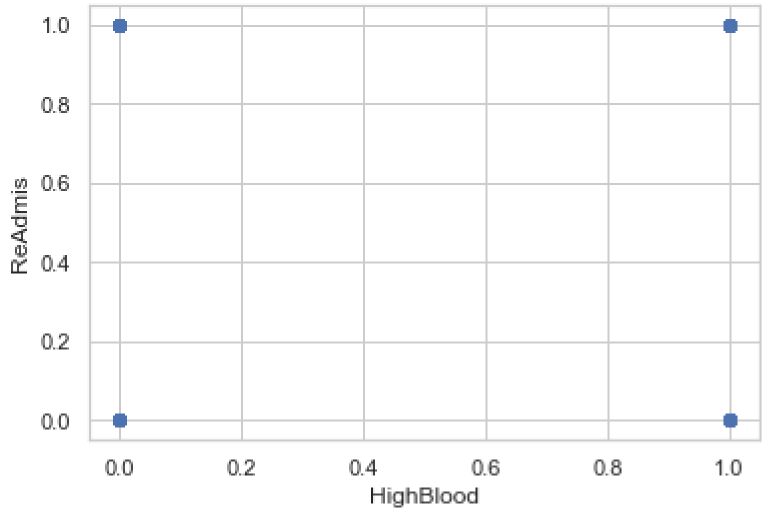
ReAdmis
Doc_visits

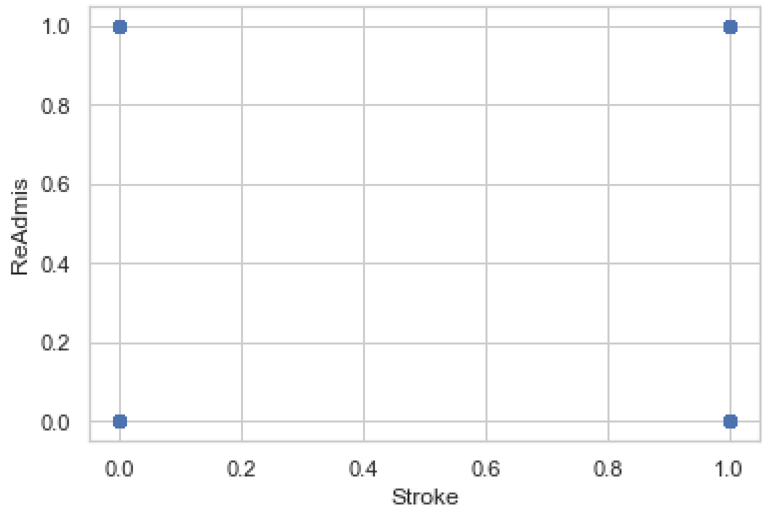

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
0
1
2
3
4
5
6
7
Full_meals_eaten

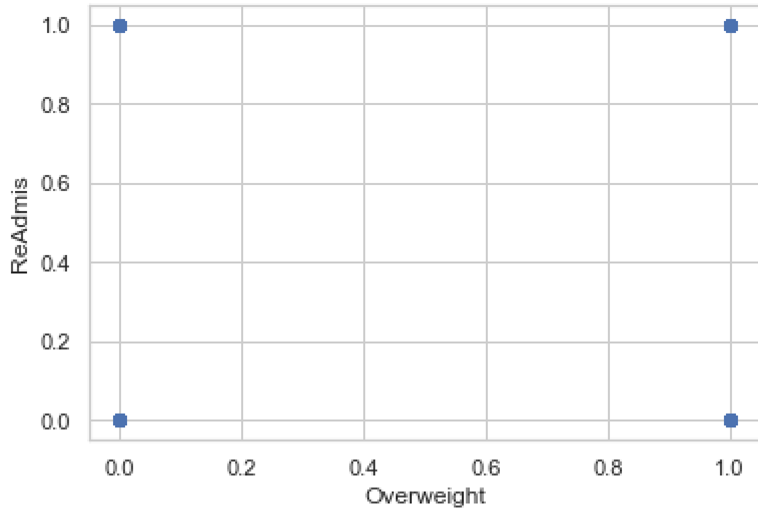

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
0
1
2
3
4
5
vitD_supp

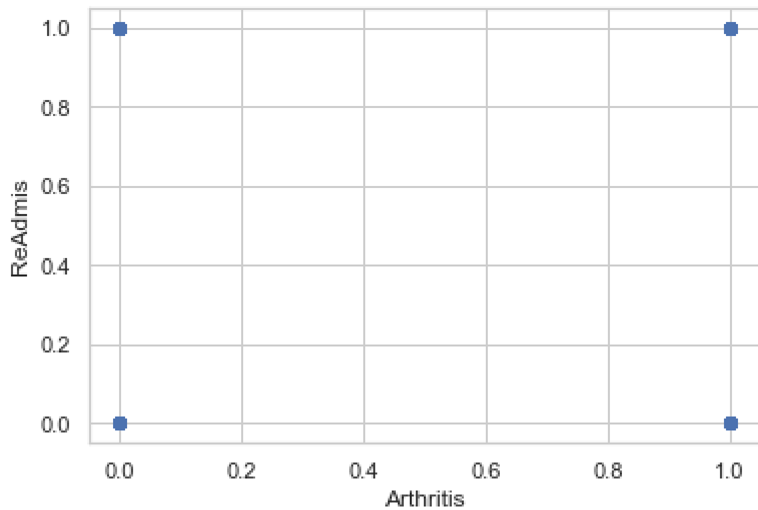

ReAdmis
Soft_drink
0.0
0.2
0.4
0.6
0.8
1.0

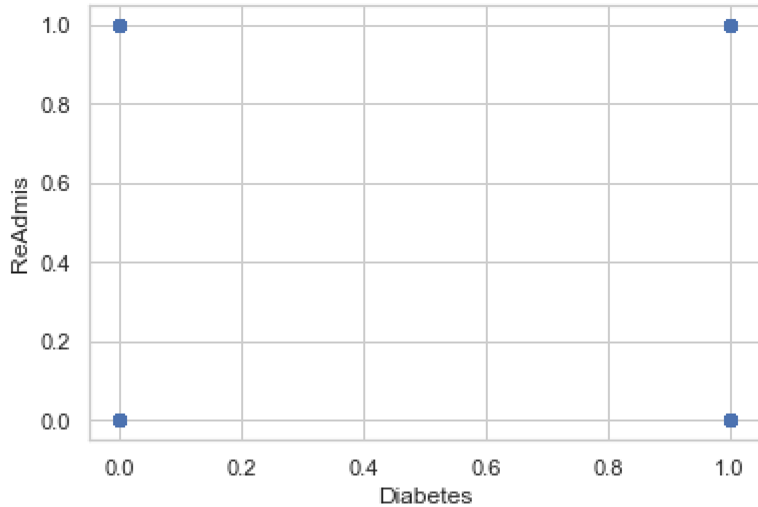

ReAdmis
Initial_admin
0.0
0.2
0.4
0.6
0.8
1.0


ReAdmis
HighBlood
0.0
0.2
0.4
0.6
0.8
1.0

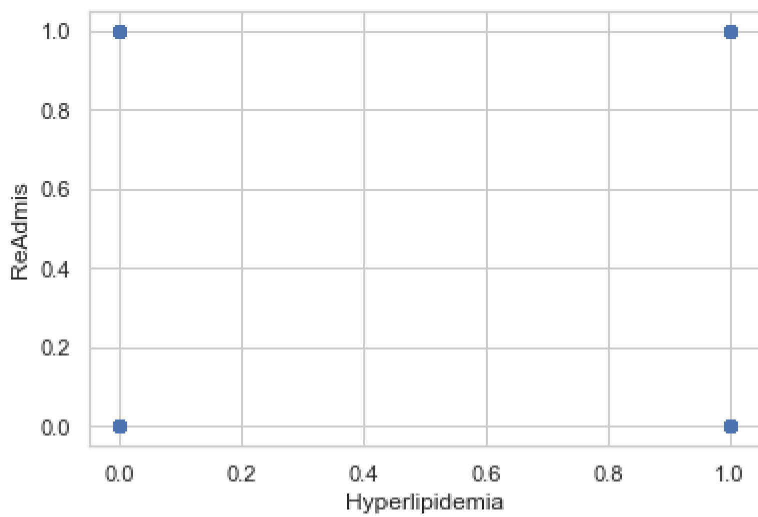

ReAdmis
Stroke
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
0.0 0.2 0.4 0.6 0.8 1.0
Overweight

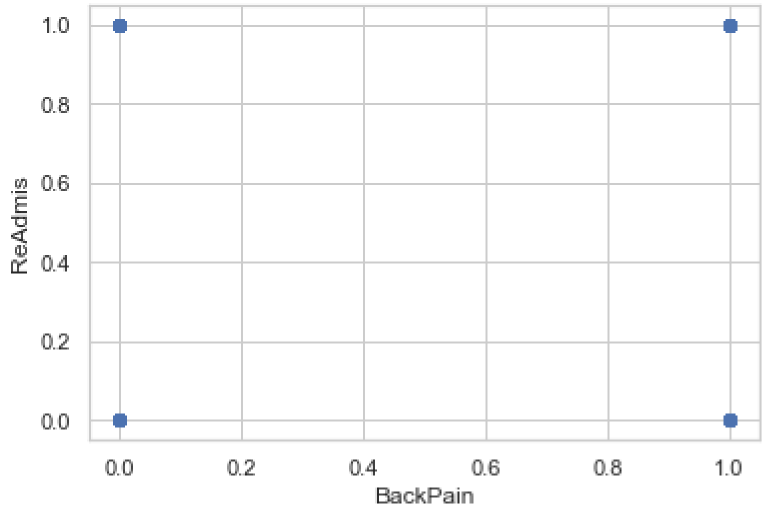

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
0.0 0.2 0.4 0.6 0.8 1.0
Arthritis

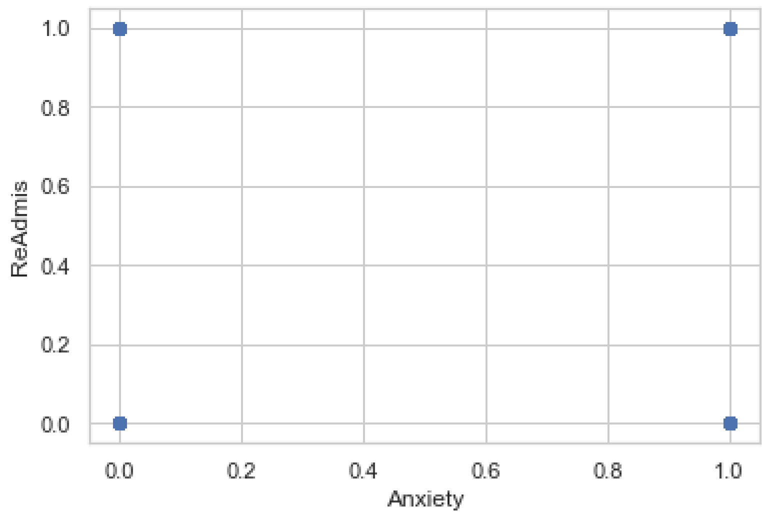


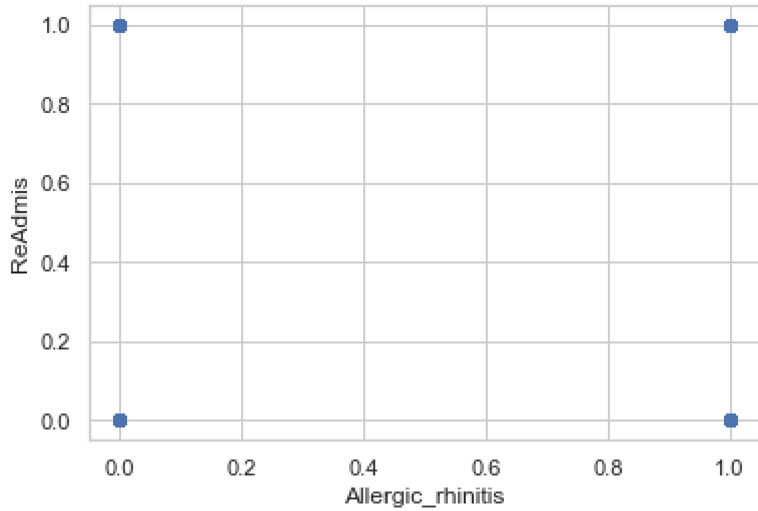
ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Diabetes

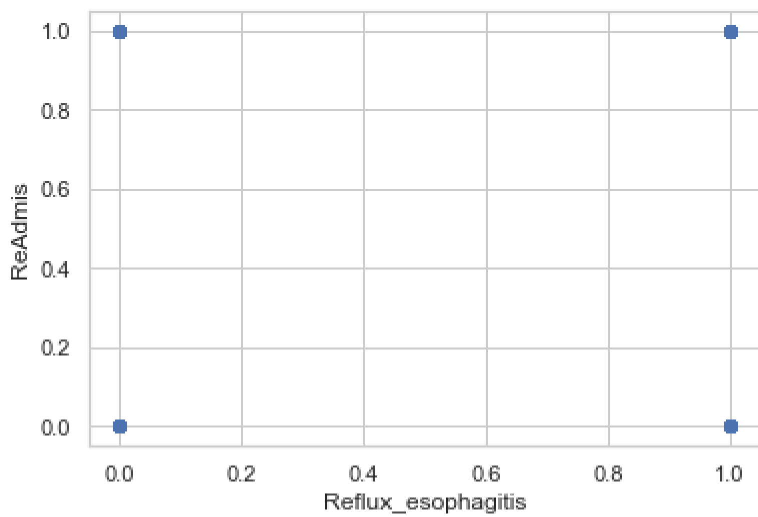


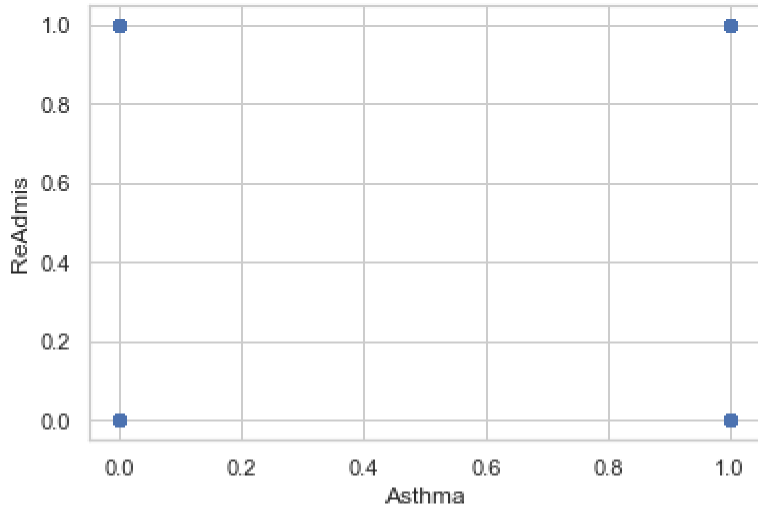
ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Hyperlipidemia

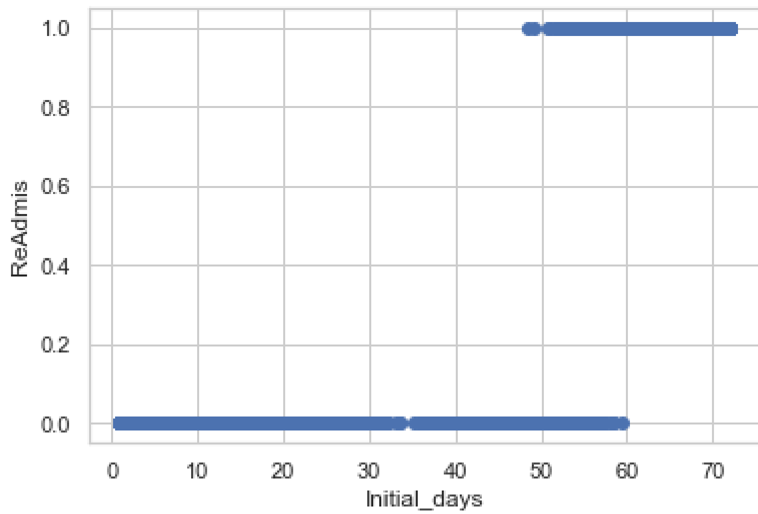

ReAdmis
BackPain
0.0
0.2
0.4
0.6
0.8
1.0

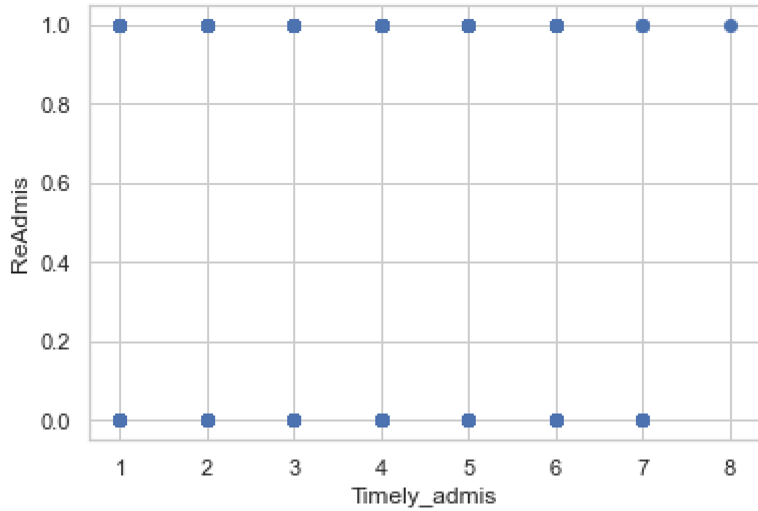

ReAdmis
Anxiety
0.0
0.2
0.4
0.6
0.8
1.0

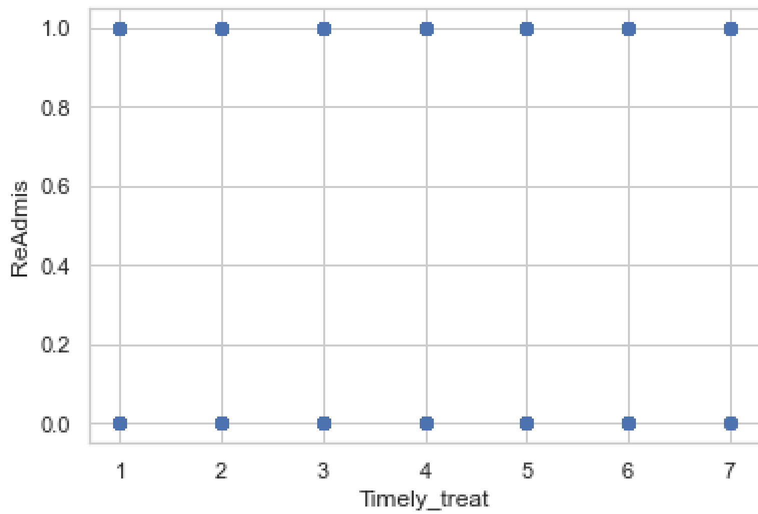

ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Allergic_rhinitis

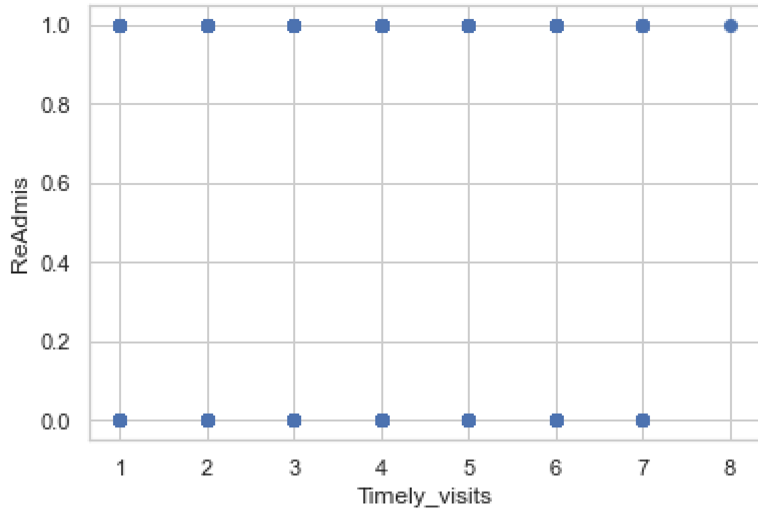

ReAdmis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Reflux_esophagitis

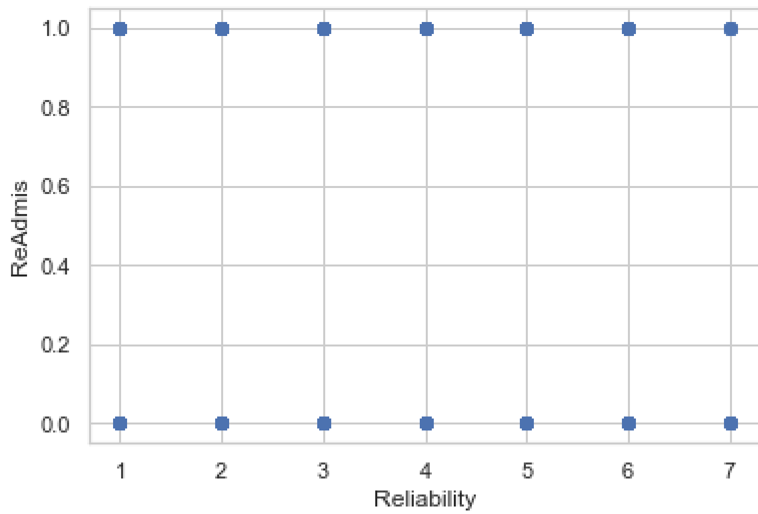

ReAdmis
Asthma
1.0
0.8
0.6
0.4
0.2
0.0

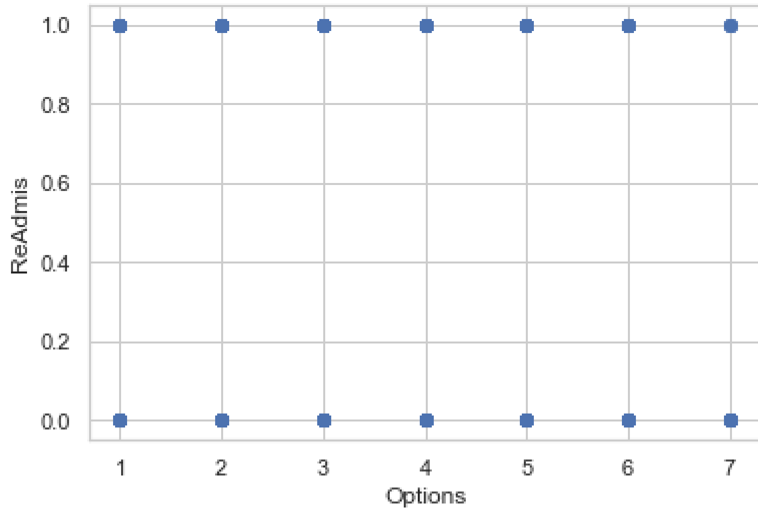

ReAdmis
Initial_days
0
10
20
30
40
50
60
70

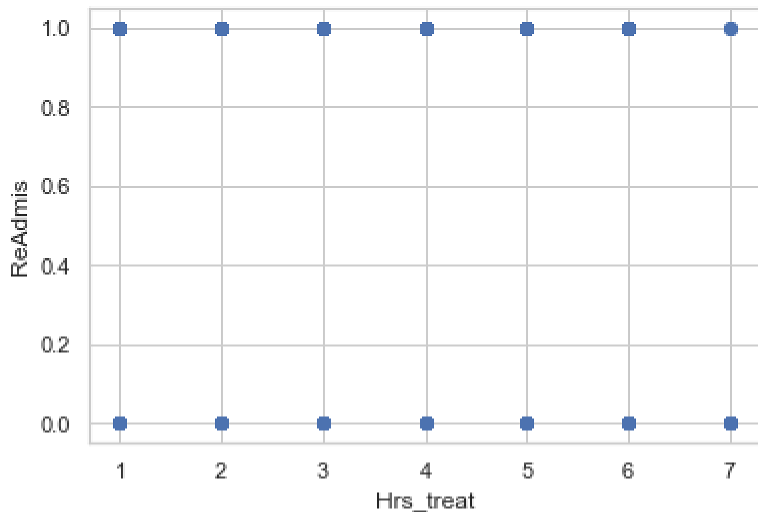

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
8
Timely_admis

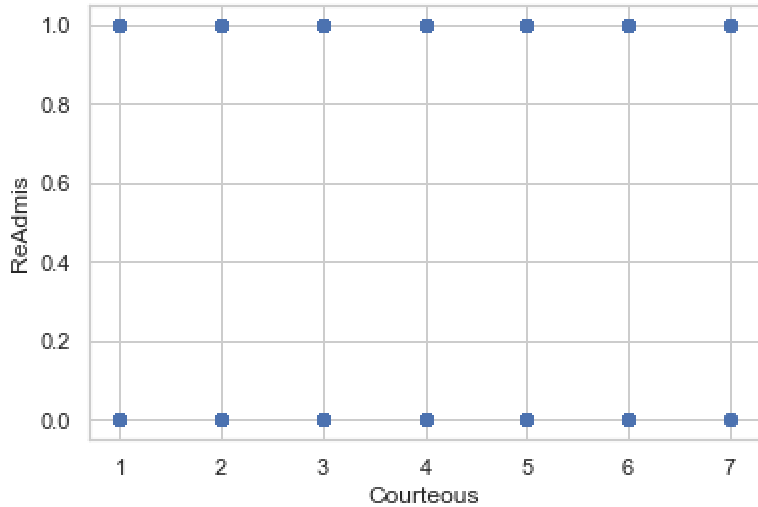

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
Timely_treat

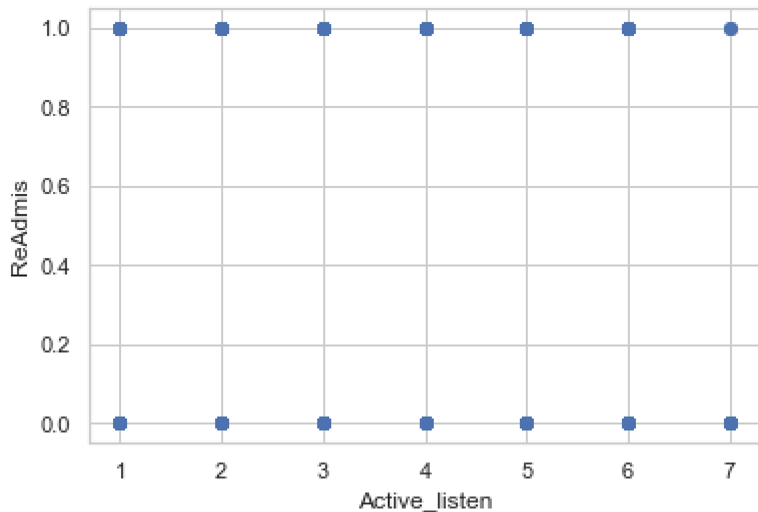

1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
8
Timely_visits


1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
Reliability


1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
Options


1.0
0.8
0.6
0.4
0.2
0.0
ReAdmis
1
2
3
4
5
6
7
Hrs_treat





In [22]:
```
# create the initial logistics model
log_reg_results = sm.Logit(df["ReAdmis"], df[['Children', 'Age', 'Income', 'Marital', 'Gender', 'VitD_levels', 'Doc_vis
print(log_reg_results.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.107566
         Iterations 11
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9969
Method:                           MLE   Df Model:                           30
Date:                Sat, 24 Dec 2022   Pseudo R-squ.:                  0.8363
Time:                        23:42:39   Log-Likelihood:                -1075.7
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
======================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------
Children               0.0050      0.025      0.200      0.841      -0.044       0.054
Age                   -0.0139      0.003     -5.094      0.000      -0.019      -0.009
Income             -6.555e-06   1.92e-06     -3.422      0.001   -1.03e-05    -2.8e-06
Marital               -0.0696      0.141     -0.495      0.620      -0.345       0.206
Gender                -0.3728      0.112     -3.338      0.001      -0.592      -0.154
VitD_levels           -0.4593      0.026    -17.936      0.000      -0.509      -0.409
Doc_visits            -0.4609      0.053     -8.763      0.000      -0.564      -0.358
Full_meals_eaten      -0.0743      0.056     -1.338      0.181      -0.183       0.035
vitD_supp             -0.1947      0.086     -2.254      0.024      -0.364      -0.025
Soft_drink            -0.0081      0.128     -0.064      0.949      -0.258       0.242
Initial_admin          0.4679      0.111      4.197      0.000       0.249       0.686
HighBlood              0.1121      0.113      0.990      0.322      -0.110       0.334
Stroke                 0.2222      0.140      1.584      0.113      -0.053       0.497
Overweight            -0.4160      0.122     -3.416      0.001      -0.655      -0.177
Arthritis             -0.4974      0.115     -4.320      0.000      -0.723      -0.272
Diabetes              -0.1160      0.123     -0.944      0.345      -0.357       0.125
Hyperlipidemia        -0.0781      0.118     -0.664      0.507      -0.308       0.152
BackPain              -0.0150      0.112     -0.134      0.894      -0.235       0.205
Anxiety               -0.3797      0.119     -3.198      0.001      -0.612      -0.147
Allergic_rhinitis     -0.3806      0.113     -3.358      0.001      -0.603      -0.158
Reflux_esophagitis    -0.3800      0.113     -3.348      0.001      -0.602      -0.158
Asthma                -0.4210      0.123     -3.432      0.001      -0.661      -0.181
Initial_days           0.4063      0.013     31.118      0.000       0.381       0.432
Timely_admis           0.1473      0.081      1.828      0.068      -0.011       0.305
Timely_treat           0.0719      0.074      0.976      0.329      -0.072       0.216
Timely_visits         -0.3015      0.068     -4.402      0.000      -0.436      -0.167
Reliability           -0.5018      0.060     -8.365      0.000      -0.619      -0.384
Options               -0.9190      0.065    -14.239      0.000      -1.045      -0.792
Hrs_treat             -0.3222      0.068     -4.747      0.000      -0.455      -0.189
Courteous             -0.3539      0.061     -5.823      0.000      -0.473      -0.235
```

```
Active_listen         -0.3633      0.058     -6.211      0.000      -0.478      -0.249
=====================================================================================

Possibly complete quasi-separation: A fraction 0.51 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [23]:

```
# create the correlation matrix
matrix_df = pd.read_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d208\2medical_clean-PREPAREDTASK2_12-24-2022.csv')

matrix_df = matrix_df[['Children', 'Age', 'Income', 'Marital', 'Gender', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten

X = matrix_df.iloc[:, 1:-1].values
y = matrix_df.iloc[:,-1].values
```

In [24]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [25]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
C:\Users\fahim\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:444: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[25]:

```
▾ LogisticRegression
LogisticRegression(random_state=0)
```

In [26]:

```
y_pred = classifier.predict(X_test)
```

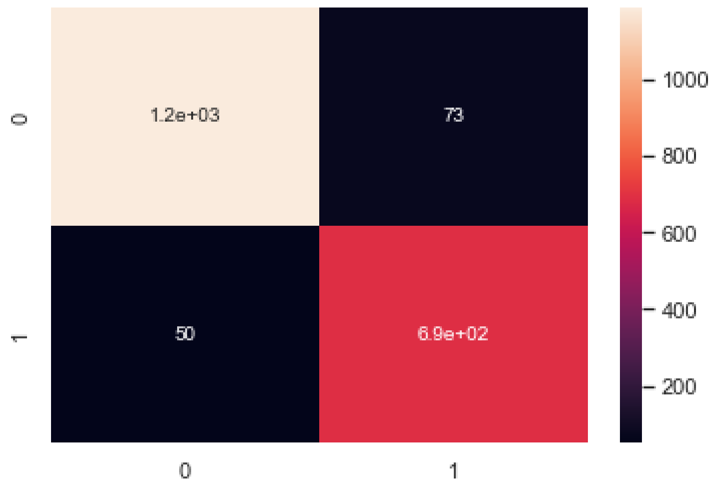
In [27]:

```
#now create the confusion matrix for the initial model
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test, y_pred)
print(matrix)
```

```
[[1189   73]
 [  50  688]]
```

In [28]:
```
y_predict_test = classifier.predict(X_test)
new_matrix = confusion_matrix(y_test, y_predict_test)
sb.heatmap(new_matrix, annot=True)
```

Out[28]: <AxesSubplot:>



In [29]:
```
#retrieve the classificaiton report for the initial model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1262
           1       0.90      0.93      0.92       738

    accuracy                           0.94      2000
   macro avg       0.93      0.94      0.93      2000
weighted avg       0.94      0.94      0.94      2000
```

In [30]:
```
#Create the reduced model with the variables that had a P value below .05 statistical significance level
log_reg_results2 = sm.Logit(df["ReAdmis"], df[['Age', 'Income', 'Gender', 'VitD_levels', 'Doc_visits', 'vitD_supp','Ini
print(log_reg_results2.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.107964
         Iterations 11
                           Logit Regression Results
==============================================================================
Dep. Variable:               ReAdmis   No. Observations:                10000
Model:                         Logit   Df Residuals:                     9979
Method:                          MLE   Df Model:                           20
Date:               Sat, 24 Dec 2022   Pseudo R-squ.:                  0.8357
Time:                       23:42:40   Log-Likelihood:                -1079.6
converged:                      True   LL-Null:                       -6572.9
Covariance Type:           nonrobust   LLR p-value:                     0.000
======================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------
Age                   -0.0138      0.003     -5.099      0.000      -0.019      -0.009
Income             -6.401e-06   1.91e-06     -3.350      0.001   -1.01e-05   -2.66e-06
Gender                -0.3735      0.111     -3.356      0.001      -0.592      -0.155
VitD_levels           -0.4562      0.025    -18.004      0.000      -0.506      -0.406
Doc_visits            -0.4580      0.052     -8.803      0.000      -0.560      -0.356
vitD_supp             -0.1902      0.086     -2.206      0.027      -0.359      -0.021
Initial_admin          0.4636      0.111      4.176      0.000       0.246       0.681
Overweight            -0.4015      0.121     -3.315      0.001      -0.639      -0.164
Arthritis             -0.5081      0.115     -4.430      0.000      -0.733      -0.283
Anxiety               -0.3776      0.118     -3.199      0.001      -0.609      -0.146
Allergic_rhinitis     -0.3773      0.113     -3.346      0.001      -0.598      -0.156
Reflux_esophagitis    -0.3738      0.113     -3.309      0.001      -0.595      -0.152
Asthma                -0.4248      0.122     -3.472      0.001      -0.665      -0.185
Initial_days           0.4039      0.013     31.204      0.000       0.379       0.429
Timely_admis           0.1828      0.072      2.529      0.011       0.041       0.324
Timely_visits         -0.2852      0.066     -4.299      0.000      -0.415      -0.155
Reliability           -0.4999      0.060     -8.393      0.000      -0.617      -0.383
Options               -0.9164      0.064    -14.346      0.000      -1.042      -0.791
Hrs_treat             -0.3167      0.067     -4.711      0.000      -0.448      -0.185
Courteous             -0.3487      0.060     -5.781      0.000      -0.467      -0.230
Active_listen         -0.3621      0.058     -6.230      0.000      -0.476      -0.248
======================================================================================

Possibly complete quasi-separation: A fraction 0.50 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [31]:

```
# create the correlation matrix for the reduced model
matrix_df = pd.read_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d208\2medical_clean-PREPAREDTASK2_12-24-2022.csv')
```

```
matrix_df = matrix_df[['Age', 'Income', 'Gender', 'VitD_levels', 'Doc_visits', 'vitD_supp','Initial_admin', 'Overweight

X = matrix_df.iloc[:, 1:-1].values
y = matrix_df.iloc[:,-1].values
```

In [32]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [33]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
C:\Users\fahim\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:444: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[33]:

```
▾ LogisticRegression
LogisticRegression(random_state=0)
```

In [34]:

```
y_pred = classifier.predict(X_test)
```

In [35]:

```
#now create the confusion matrix for the reduced model
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test, y_pred)
print(matrix)
```

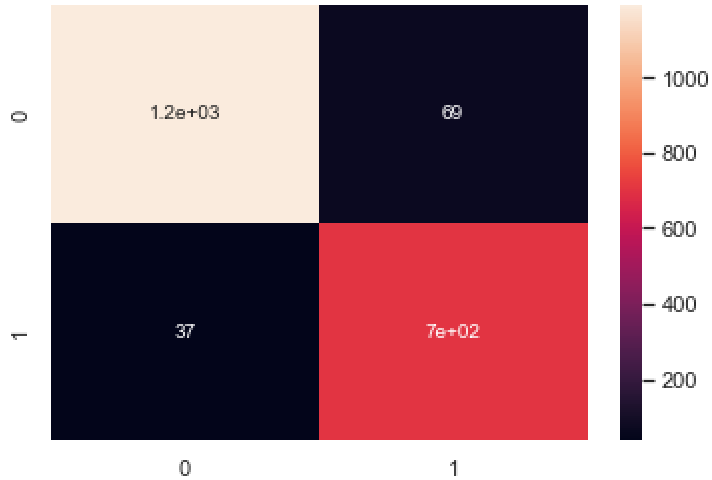
```
[[1193   69]
 [  37  701]]
```

In [36]:

```
y_predict_test = classifier.predict(X_test)
new_matrix = confusion_matrix(y_test, y_predict_test)
sb.heatmap(new_matrix, annot=True)
```

Out[36]:

```
<AxesSubplot:>
```



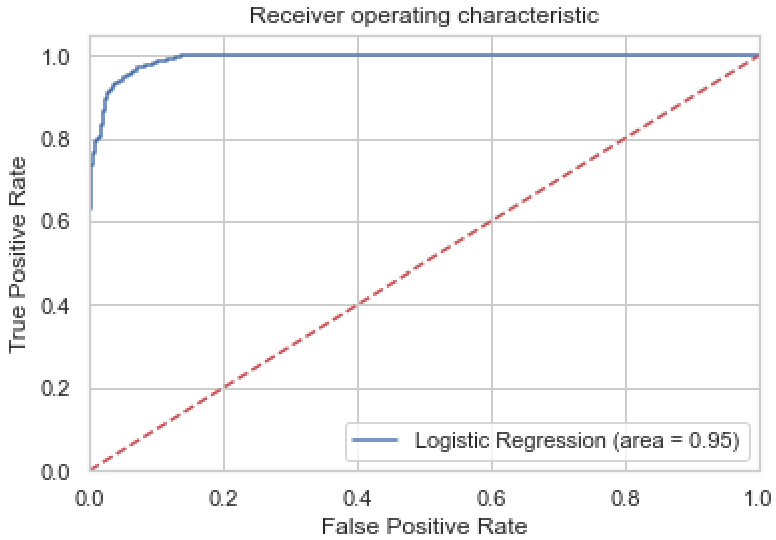
In [37]:

```
#retrieve the classificaiton report for the reduced model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      1262
           1       0.91      0.95      0.93       738

    accuracy                           0.95      2000
   macro avg       0.94      0.95      0.94      2000
weighted avg       0.95      0.95      0.95      2000
```

In [38]:

```
# plot ROC Curve
logit_roc_auc = roc_auc_score(y_test, classifier.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, classifier.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



In [39]:
```
# create an equation of the regression
print('Logit: {:.2f}'.format(logit_roc_auc))
equation = log_reg_results2.summary().tables[1]
print('Estimate [{}] as L = '.format(log_reg_results2.summary().tables[0][1][1]))
equation = pd.DataFrame(equation)
for i in equation.itertuples():
    print(' {:+.3f} x ( {} ) '.format(i[0],i[1]))
```

```
Logit: 0.95
Estimate [Logit] as L =
 +0.000 x (  )
 +1.000 x ( Age )
 +2.000 x ( Income )
 +3.000 x ( Gender )
 +4.000 x ( VitD_levels )
 +5.000 x ( Doc_visits )
 +6.000 x ( vitD_supp )
 +7.000 x ( Initial_admin )
 +8.000 x ( Overweight )
 +9.000 x ( Arthritis )
 +10.000 x ( Anxiety )
 +11.000 x ( Allergic_rhinitis )
 +12.000 x ( Reflux_esophagitis )
 +13.000 x ( Asthma )
 +14.000 x ( Initial_days )
 +15.000 x ( Timely_admis )
 +16.000 x ( Timely_visits )
 +17.000 x ( Reliability )
 +18.000 x ( Options )
 +19.000 x ( Hrs_treat )
 +20.000 x ( Courteous )
 +21.000 x ( Active_listen )
```

In [40]:

```
print(equation)
```

```
                     0          1        2        3      4         5  \
0                         coef  std err        z  P>|z|    [0.025
1                  Age   -0.0138    0.003   -5.099  0.000    -0.019
2               Income -6.401e-06 1.91e-06  -3.350  0.001 -1.01e-05
3               Gender   -0.3735    0.111   -3.356  0.001    -0.592
4          VitD_levels   -0.4562    0.025  -18.004  0.000    -0.506
5           Doc_visits   -0.4580    0.052   -8.803  0.000    -0.560
6            vitD_supp   -0.1902    0.086   -2.206  0.027    -0.359
7        Initial_admin    0.4636    0.111    4.176  0.000     0.246
8           Overweight   -0.4015    0.121   -3.315  0.001    -0.639
9            Arthritis   -0.5081    0.115   -4.430  0.000    -0.733
10             Anxiety   -0.3776    0.118   -3.199  0.001    -0.609
11   Allergic_rhinitis   -0.3773    0.113   -3.346  0.001    -0.598
12  Reflux_esophagitis   -0.3738    0.113   -3.309  0.001    -0.595
13              Asthma   -0.4248    0.122   -3.472  0.001    -0.665
14        Initial_days    0.4039    0.013   31.204  0.000     0.379
15        Timely_admis    0.1828    0.072    2.529  0.011     0.041
16       Timely_visits   -0.2852    0.066   -4.299  0.000    -0.415
17         Reliability   -0.4999    0.060   -8.393  0.000    -0.617
18             Options   -0.9164    0.064  -14.346  0.000    -1.042
19           Hrs_treat   -0.3167    0.067   -4.711  0.000    -0.448
20           Courteous   -0.3487    0.060   -5.781  0.000    -0.467
21       Active_listen   -0.3621    0.058   -6.230  0.000    -0.476

            6
0      0.975]
1      -0.009
2   -2.66e-06
3      -0.155
4      -0.406
5      -0.356
6      -0.021
7       0.681
8      -0.164
9      -0.283
10     -0.146
11     -0.156
12     -0.152
13     -0.185
14      0.429
15      0.324
16     -0.155
17     -0.383
18     -0.791
19     -0.185
```

```
20     -0.230
21     -0.248
```

In [41]:
```
updated_equation = equation.drop(0)
```

In [42]:
```
print(updated_equation)
```

```
                     0          1         2        3      4          5  \
1                  Age    -0.0138     0.003   -5.099  0.000     -0.019
2               Income  -6.401e-06  1.91e-06  -3.350  0.001  -1.01e-05
3               Gender    -0.3735     0.111   -3.356  0.001     -0.592
4           VitD_levels   -0.4562     0.025  -18.004  0.000     -0.506
5           Doc_visits    -0.4580     0.052   -8.803  0.000     -0.560
6            vitD_supp    -0.1902     0.086   -2.206  0.027     -0.359
7        Initial_admin     0.4636     0.111    4.176  0.000      0.246
8           Overweight    -0.4015     0.121   -3.315  0.001     -0.639
9           Arthritis     -0.5081     0.115   -4.430  0.000     -0.733
10             Anxiety    -0.3776     0.118   -3.199  0.001     -0.609
11   Allergic_rhinitis    -0.3773     0.113   -3.346  0.001     -0.598
12  Reflux_esophagitis    -0.3738     0.113   -3.309  0.001     -0.595
13              Asthma    -0.4248     0.122   -3.472  0.001     -0.665
14        Initial_days     0.4039     0.013   31.204  0.000      0.379
15        Timely_admis     0.1828     0.072    2.529  0.011      0.041
16       Timely_visits    -0.2852     0.066   -4.299  0.000     -0.415
17         Reliability    -0.4999     0.060   -8.393  0.000     -0.617
18             Options    -0.9164     0.064  -14.346  0.000     -1.042
19           Hrs_treat    -0.3167     0.067   -4.711  0.000     -0.448
20           Courteous    -0.3487     0.060   -5.781  0.000     -0.467
21       Active_listen    -0.3621     0.058   -6.230  0.000     -0.476

            6
1      -0.009
2   -2.66e-06
3      -0.155
4      -0.406
5      -0.356
6      -0.021
7       0.681
8      -0.164
9      -0.283
10     -0.146
11     -0.156
12     -0.152
13     -0.185
14      0.429
15      0.324
16     -0.155
17     -0.383
18     -0.791
19     -0.185
20     -0.230
21     -0.248
```

In [43]:
```
# create an equation of the logistics regression
print('Logit: {:.2f}'.format(logit_roc_auc))
print('Estimate [{}] as L = '.format(log_reg_results2.summary().tables[0][1][1]))
for i in updated_equation.itertuples():
    print(' {:+.3f} x ( {} ) '.format(float(str(i[2])),i[1]))
```

```
Logit: 0.95
Estimate [Logit] as L = 
 -0.014 x ( Age ) 
 -0.000 x ( Income ) 
 -0.373 x ( Gender ) 
 -0.456 x ( VitD_levels ) 
 -0.458 x ( Doc_visits ) 
 -0.190 x ( vitD_supp ) 
 +0.464 x ( Initial_admin ) 
 -0.402 x ( Overweight ) 
 -0.508 x ( Arthritis ) 
 -0.378 x ( Anxiety ) 
 -0.377 x ( Allergic_rhinitis ) 
 -0.374 x ( Reflux_esophagitis ) 
 -0.425 x ( Asthma ) 
 +0.404 x ( Initial_days ) 
 +0.183 x ( Timely_admis ) 
 -0.285 x ( Timely_visits ) 
 -0.500 x ( Reliability ) 
 -0.916 x ( Options ) 
 -0.317 x ( Hrs_treat ) 
 -0.349 x ( Courteous ) 
 -0.362 x ( Active_listen ) 
```