Lora Milam

Western Governors University

D208 Data Cleaning

20 February 2022

D208 Performance Assessment Task 2

1 Research Question

Lora Milam Masters Data Analytics (2/14/2022) Program Mentor:d208@wgu.edu

1.1 Question

Due to the concern for penalties of readmission, which fields can be most associated with readmission? In other words, what makes an individual more likely to be readmitted?

1.2 Objectives

Hospitals could benefit from further analysis of these fields and researching further into the correlation between these fields and readmission rates. Future findings could provide valuable information on how to reduce readmission rates.

2 Data Analysis

2.1 Summary of Assumptions

The assumptions of logistic regression are:

- Based on Bernoulli Distribution
- The predictor variable is only a binary nominal range: Yes or No
- It predicts the probability of an outcome instead of the actual outcome itself
- The are no high correlations between predictor variables
- Logarithm of the odds of 1.

2.2 Justification of Tools

I will be utilizing Python's many capabilities to better analyze the database of medical patient records. Python3 is the latest iteration of the programming language Python, as provided within Jupyter Notebooks. Python is a high level, general purpose language that utilizes a variety of packages to tailor data.

Even though there are other methods that can be used to address this problem, I find Python3 and Jupyter Notebooks to be a convenient and intuitive way to visualize and draw conclusions from databases.

2.3 Appropriate Technique

Logistic regression analysis is an appropriate technique for analyzing the research question because the dependent variable is binomial. This analysis will determine the likelihood of a patient being readmitted, based on the independent variables. Determining which explanatory variables that have the most precedent on whether a patient will be readmitted will allow hospitals and medical personnel to focus time and resources more on the appropriate areas.

3 Data Preparation

3.1 Data Preparation Goals

The approach for this analysis includes:

1. Read in dataset: read_csv()
2. Better understand input data by evaluating data structure
3. Name dataset “medical_df” and subdatasets as “df”
4. Evaluate possible misspellings, ambiguous variable names, and absent data
5. Utilize histograms to locate outliers correlated to statistical significance, whether they hide this or create it.
6. Use imputation to replace missing data with meaningful information related to central tendency; mean,median,mode. Possibly remove outliers that are several deviations above the mean

The data set is 10,000 raw medical patient records. The target variable is whether or not, within a month of release, each customer has been readmitted to the hospital. The title of the column is “ReAdmis”.

The predictor variables that are provided in the dataset may have a correlation with the probability of the patient being readmitted due to previous ailments or problems from past admissions. These predictor variables include patient medical conditions (high blood pressure, stroke, obesity, arthritis, diabetes, etc.), patient information (service while hospitalized, days in hospital, type of initial admission, etc.), patient demographics (gender,age, job, education level, etc.). These predictor variables can be seen in the table below:

Variable	Description
CaseOrder:	- integer index - correlated to original order of raw data
Customer_id:	- character string object - unique to patient
Interaction:	- character string object - unique to patient transactions, procedures and admissions
UID:	- character string object - unique to the transactions, procedures and admissions of a patient
City:	- character string object - the city of residence of the patient
State:	- character string object - the state of residence of the patient
County:	- character string object - the county of residence of the patient
Zip:	- integer - the zip code corresponding to the residence of the patient's
Lat:	- continuous numeric (floating numeric) - GPS coordinates indicating the latitude corresponding to the patient's residence
Lng:	- continuous numeric (floating numeric) - GPS coordinates indicating the longitude corresponding to the patient's residence

Population:	- integer - the population that is within a mile radius of patient's resident

Area:	- nominal categorical
	- character string object
	- the area type corresponding to the patient's residence
	- based on unofficial census data
	- the unique values are ['Emergency Admission', 'Elective Admission', 'Observation Admission']
	`In [12]: df['Area'].unique()`
	`Out[12]: array(['Suburban', 'Urban', 'Rural'], dtype=object)`

Timezone:	- nominal categorical
	- character string object
	- the time zone corresponding to the patient's residence
	- the unique values are ['America/Chicago', 'America/New_York', 'America/Los_Angeles', 'America/Indiana/Indianapolis', 'America/Detroit', 'America/Denver', 'America/Nome', 'America/Anchorage', 'America/Phoenix', 'America/Boise', 'America/Puerto_Rico', 'America/Yakutat', 'Pacific/Honolulu', 'America/Menominee', 'America/Kentucky/Louisville', 'America/Indiana/Vincennes', 'America/Toronto', 'America/Indiana/Marengo', 'America/Indiana/Winamac', 'America/Indiana/Tell_City', 'America/Sitka', 'America/Indiana/Knox', 'America/North_Dakota/New_Salem', 'America/Indiana/Vevay', 'America/Adak', 'America/North_Dakota/Beulah']
	`In [13]: df['Timezone'].unique()`
	`Out[13]: array(['America/Chicago', 'America/New_York', 'America/Los_Angeles', 'America/Indiana/Indianapolis', 'America/Detroit', 'America/Denver', 'America/Nome', 'America/Anchorage', 'America/Phoenix', 'America/Boise', 'America/Puerto_Rico', 'America/Yakutat', 'Pacific/Honolulu', 'America/Menominee', 'America/Kentucky/Louisville', 'America/Indiana/Vincennes', 'America/Toronto', 'America/Indiana/Marengo', 'America/Indiana/Winamac', 'America/Indiana/Tell_City', 'America/Sitka', 'America/Indiana/Knox', 'America/North_Dakota/New_Salem', 'America/Indiana/Vevay', 'America/Adak', 'America/North_Dakota/Beulah'], dtype=object)`

Job:	- nominal categorical

	- character string object - the occupation of the patient or insurance holder
Children:	- integer - the amount of children within patient's household
Age:	- integer - the patient's age
Income:	- numeric value - the patient's or insurance holder's annual income

Marital:	- nominal categorical
	- character string object
	- the marital status of the patient or insurance holder
	- the unique values are ['Divorced', 'Married', 'Widowed', 'Never Married', 'Separated']
	`In [17]: df['Marital'].unique()`
	`Out[17]: array(['Divorced', 'Married', 'Widowed', 'Never Married', 'Separated'], dtype=object)`

Gender:	- nominal categorical
	- character string object
	- the gender of the patient
	- the unique values are ['Male', 'Female', 'Prefer not to answer']
	`In [18]: df['Gender'].unique()`
	`Out[18]: array(['Male', 'Female', 'Prefer not to answer'], dtype=object)`

ReAdmis	- binary categorical - character string object - whether or not, within a month of release, each customer has been readmitted to the hospital

	- **target variable**
VitD_levels:	- continuous numeric (floating numeric) - value of the vitamin D levels of the patient - measured in ng/mL
Doc_visits:	- integer - the number of times during the initial hospitalization that the primary physician visited the patient
Full_meals_eaten:	- integer - number of full meals eaten Note: It counts as zero if the patient only eats a partial meal
VitD_supp:	- integer - the number of times that vitamin D supplements were administered to patient
Soft_drink:	- binary categorical - character string object - whether or not a patient on a daily basis drinks three or more sodas - the unique values are [Yes, No]
Initial_admin:	- nominal categorical - character string object - the reason why the patient was initially admitted into the hospital In [19]: `df['Initial_admin'].unique()` Out[19]: `array(['Emergency Admission', 'Elective Admission', 'Observation Admission'], dtype=object)`
HighBlood:	- binary categorical - character string object - whether or not the patient has high blood pressure

	- the unique values are [Yes, No]
Stroke:	- binary categorical - character string object - whether or not the patient has had a stroke - the unique values are [Yes, No]
Complication_risk:	- ordinal categorical - character string object - the level of complication risk - the unique values are [High, Medium, Low]
Overweight:	- binary categorical - integer - whether or not the patient is overweight, as determined by their BMI elements: age, gender, and height - the unique values are [1,0]
Arthritis:	- binary categorical - character string object - whether or not the patient has arthritis - the unique values are [Yes, No]
Diabetes:	- binary categorical - character string object - whether or not the patient has diabetes - the unique values are [Yes, No]
Hyperlipidemia:	- binary categorical

	- character string object - whether or not the patient has hyperlipidemia - the unique values are [Yes, No]
BackPain:	- binary categorical - character string object - whether or not the patient has chronic backpain - the unique values are [Yes, No]
Anxiety:	- binary categorical - integer - whether or not the patient has an anxiety disorder - the unique values are [1,0]
Allergic_rhinitis:	- binary categorical - character string object - whether or not the patient has allergic rhinitis - the unique values are [Yes, No]
Reflux_esophagitis:	- binary categorical - character string object - whether or not the patient has reflux esophagitis - the unique values are [Yes, No]
Asthma:	- binary categorical - character string object - whether or not the patient has asthma - the unique values are [Yes, No]

Services:	- nominal categorical - character string object - the primary service the patient received while hospitalized - the unique values are ['Blood Work', 'Intravenous', 'CT Scan', 'MRI'] In [20]: `df['Services'].unique()` Out[20]: `array(['Blood Work', 'Intravenous', 'CT Scan', 'MRI'], dtype=object)`
Initial_days:	- numeric value - number of days,during the initial visit, the patient stayed in the hospital
TotalCharge:	- numeric value - patient's average daily charges,during the initial visit, for typical (not specialized) treatments and services
Additional_charges:	- numeric value - patient's average charges,during the initial visit, for additional treatments and services
Item1:	- integer value - from most important to least important, the level of importance of timely admission - the unique values are [1:8]
Item2:	- integer value - from most important to least important, the level of importance of timely treatment - the unique values are [1:8]
Item3:	- integer value - from most important to least important, the level of importance of timely visits - the unique values are [1:8]

Item4:	- integer value - from most important to least important, the level of importance of reliability - the unique values are [1:8]
Item5:	- integer value - from most important to least important, the level of importance of options
Item6:	- integer value - from most important to least important, the level of importance of hours of treatment - the unique values are [1:8]
Item7:	- integer value - from most important to least important, the level of importance of courteous staff - the unique values are [1:8]
Item8:	- integer value - from doctor from most important to least important, the level of importance of evidence of active listening - the unique values are [1:8]

3.2 Summary Statistics

For this analysis, the less relevant variables were removed ('CaseOrder', 'Customer_id', 'Interaction', 'UID', ‘City’, ‘State’, ‘County’, ‘Zip’, ‘Lat’, ‘Lng’, ‘Population’, ‘Area’, ‘Timezone’, 'Job', 'Marital'). Binomial “Yes”/”No” or “Male”/”Female” variables were imputed to a corresponding numeric variable, 1/0. The resulting data frame will consist of 35 columns, including the target variable. The dataset had been preemptively cleaned, resulting in no NULL, NAs or missing values.

Measures of central tendency through histograms and boxplots showed normal distributions for “Initial_days”, “TotalCharge” and “Doc_visits”. The cleaned dataset no longer had any unjustifiable outliers, meaning that any remaining outliers were pertinent to the analysis. The histograms for “ReAdmis”, “TotalCharge”, and “Inital_days” all had a binomial distribution. In

the scatter plots between the dependent variable and the independent variables, there was a direct linear relationship.

3.3 Data Preparation Steps

To properly analyze the data given, first the data will be cleaned by using Python3 and data cleaning methods. Python3 is the latest iteration of the programming language Python, as provided within Jupyter Notebooks. The plan for cleaning the data will be done through the following steps:

1. Import the dataset to a Python dataframe
2. Relabel survey columns to something more identifiable
3. Describe dataframe, structure, and data types
4. Summary statistics
5. Remove any redundant, irrelevant, or misleading fields
6. Impute missing data with measures of central tendency (mean, median, mode) or remove outliers that are several standard deviations above the mean.
7. Convert character object values to numeric values/separate into separate variables using dummy variables where applicable (Himamsh)
8. Univariate and bivariate visualizations
9. Place ‘ReAdmis’ at the end of the dataframe
10. Prepared dataset will be exported as “medical_prepared.csv”
11. No data will be deleted or changed unless it is an obvious mistake. Note: Adjusting data too much will skew the model towards a conclusion that may not generalize due to the data being overly-manipulated.

In [1]:
```
# Standard imports
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Statistics packages
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats

# Scikit-learn
import sklearn
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report

# Import chisquare from SciPy.stats
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
```

In [2]:
```
# Load dataset into Pandas dataframe
medical_df = pd.read_csv('medical_clean.csv')
```

In [3]:
```
# Rename survey columns to more identifiable names
medical_df.rename(columns =
    {'Item1':'Survey_TimelyAdmin',
     'Item2':'Survey_TimelyTreatment',
     'Item3':'Survey_TimelyVisits',
     'Item4':'Survey_Reliability',
     'Item5':'Survey_Options',
     'Item6':'Survey_HoursTreatment',
     'Item7':'Survey_CourteousStaff',
     'Item8':'Survey_ActiveListening'}, inplace=True)

# Display updated Medical dataframe
medical_df
```

Out[3]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	TotalCharge
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	-86.72508	...	3726.702860
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	-85.22907	...	4193.190458
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	-96.63772	...	2434.234222
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	-93.51479	...	2127.830423
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	-76.88958	...	2113.073274
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	B863060	a25b594d-0328-486f-a9b9-0567eb0f9723	39184dc28cc038871912ccc4500049e5	Norlina	NC	Warren	27563	36.42886	-78.23716	...	6850.942000
9996	9997	P712040	70711574-f7b1-4a17-b15f-48c54564b70f	3cd124ccd43147404292e883bf9ec55c	Milmay	NJ	Atlantic	8340	39.43609	-74.87302	...	7741.690000
9997	9998	R778890	1d79569d-8e0f-4180-a207-d67ee4527d26	41b770aeee97a5b9e7f69c906a8119d7	Southside	TN	Montgomery	37171	36.36655	-87.29988	...	8276.481000
9998	9999	E344109	f5a68e69-2a60-409b-a92f-ac0847b27db0	2bb491ef5b1beb1fed758cc6885c167a	Quinn	SD	Pennington	57775	44.10354	-102.01590	...	7644.483000
9999	10000	I569847	bc482c02-f8c9-4423-99de-3db5e62a18d5	95663a202338000abdf7e09311c2a8a1	Coraopolis	PA	Allegheny	15108	40.49998	-80.19959	...	7887.553000

10000 rows × 50 columns

In [4]:
```
# List dataframe columns
df = medical_df.columns
print(df)
```

```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Survey_TimelyAdmin',
       'Survey_TimelyTreatment', 'Survey_TimelyVisits', 'Survey_Reliability',
       'Survey_Options', 'Survey_HoursTreatment', 'Survey_CourteousStaff',
       'Survey_ActiveListening'],
      dtype='object')
```

In [5]:
```
# Number of records and columns of dataframe
medical_df.shape
```

Out[5]:
```
(10000, 50)
```

In [6]:
```
# Describe Medical dataset stats
medical_df.describe()
```

Out[6]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	Doc_visits	...	Tot
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	1000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	53.511700	40490.495160	17.964262	5.012200	...	53
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	20.638538	28521.153293	2.017231	1.045734	...	21
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	18.000000	154.080000	9.806483	1.000000	...	19
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	36.000000	19598.775000	16.626439	4.000000	...	31
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33768.420000	17.951122	5.000000	...	52
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54296.402500	19.347963	6.000000	...	74
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.100000	26.394449	9.000000	...	91

8 rows × 23 columns

In [7]:
```
# Remove less relevant fields from stats description
medical_df = medical_df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
                                      'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
                                      'Marital'])
medical_df.describe()
medical_df.shape
```

Out[7]:
```
(10000, 35)
```

In [8]:
```
# Count of missing values by column
data_nulls = medical_df.isnull().sum()
print(data_nulls)
```

```
Children                0
Age                     0
Income                  0
Gender                  0
ReAdmis                 0
VitD_levels             0
Doc_visits              0
Full_meals_eaten        0
vitD_supp               0
Soft_drink              0
Initial_admin           0
HighBlood               0
Stroke                  0
Complication_risk       0
Overweight              0
Arthritis               0
Diabetes                0
Hyperlipidemia          0
BackPain                0
Anxiety                 0
Allergic_rhinitis       0
Reflux_esophagitis      0
Asthma                  0
Services                0
Initial_days            0
TotalCharge             0
Additional_charges      0
Survey_TimelyAdmin      0
Survey_TimelyTreatment  0
Survey_TimelyVisits     0
Survey_Reliability      0
Survey_Options          0
Survey_HoursTreatment   0
Survey_CourteousStaff   0
Survey_ActiveListening  0
dtype: int64
```

In [9]:
```
# Impute qualitive data fields by creating binary dummy columns then drop pre-existing column
# Exclude redundant values: ex. gender is categorized as male or female, the value 'prefer not to answer' can be identi
dmy = pd.get_dummies(medical_df['Gender'])
dmy = dmy.iloc[:,:-1]
medical_df = pd.concat([medical_df,dmy],axis=1)
medical_df = medical_df.drop(columns = 'Gender')

dmy = pd.get_dummies(medical_df['Initial_admin'])
medical_df = pd.concat([medical_df,dmy],axis=1)
medical_df = medical_df.drop(columns = 'Initial_admin')

dmy = pd.get_dummies(medical_df['Complication_risk'])
medical_df = pd.concat([medical_df,dmy],axis=1)
medical_df = medical_df.drop(columns = 'Complication_risk')

dmy = pd.get_dummies(medical_df['Services'])
medical_df = pd.concat([medical_df,dmy],axis=1)
medical_df = medical_df.drop(columns = 'Services')

medical_df.describe()
```

Out[9]:

	Children	Age	Income	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Initial_days	TotalCharge	Additional_charges
count	10000.000000	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	17.964262	5.012200	1.001400	0.398900	34.455299	5312.172769	12934.528587
std	2.163659	20.638538	28521.153293	2.017231	1.045734	1.008117	0.628505	26.309341	2180.393838	6542.601544
min	0.000000	18.000000	154.080000	9.806483	1.000000	0.000000	0.000000	1.001981	1938.312067	3125.703000
25%	0.000000	36.000000	19598.775000	16.626439	4.000000	0.000000	0.000000	7.896215	3179.374015	7986.487755
50%	1.000000	53.000000	33768.420000	17.951122	5.000000	1.000000	0.000000	35.836244	5213.952000	11573.977735
75%	3.000000	71.000000	54296.402500	19.347963	6.000000	2.000000	1.000000	61.161020	7459.699750	15626.490000
max	10.000000	89.000000	207249.100000	26.394449	9.000000	7.000000	5.000000	71.981490	9180.728000	30566.070000

8 rows × 27 columns

In [10]:
```
df = medical_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'ReAdmis', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Initial_days', 'TotalCharge',
       'Additional_charges', 'Survey_TimelyAdmin', 'Survey_TimelyTreatment',
       'Survey_TimelyVisits', 'Survey_Reliability', 'Survey_Options',
       'Survey_HoursTreatment', 'Survey_CourteousStaff',
       'Survey_ActiveListening', 'Female', 'Male', 'Elective Admission',
       'Emergency Admission', 'Observation Admission', 'Blood Work', 'CT Scan',
       'Intravenous', 'MRI'],
      dtype='object')
```

In [11]:
```
# Impute binomial categoricals as numerics
data = medical_df['ReAdmis']
readmis_dict = {
    "No":0,"Yes":1
}
for k,v in readmis_dict.items():
    data = data.replace(k,v)

medical_df['ReAdmis'] = data

data = medical_df['Soft_drink']
for k,v in readmis_dict.items():
    data = data.replace(k,v)

medical_df['Soft_drink'] = data

data = medical_df['HighBlood']
for k,v in readmis_dict.items():
    data = data.replace(k,v)

medical_df['HighBlood'] = data

data = medical_df['Stroke']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Stroke'] = data

data = medical_df['Arthritis']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Arthritis'] = data

data = medical_df['Diabetes']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Diabetes'] = data

data = medical_df['Hyperlipidemia']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Hyperlipidemia'] = data

data = medical_df['BackPain']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['BackPain'] = data

data = medical_df['Allergic_rhinitis']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Allergic_rhinitis'] = data

data = medical_df['Reflux_esophagitis']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Reflux_esophagitis'] = data

data = medical_df['Asthma']
for k,v in readmis_dict.items():
    data = data.replace(k,v)
medical_df['Asthma'] = data

df = medical_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'ReAdmis', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Initial_days', 'TotalCharge',
       'Additional_charges', 'Survey_TimelyAdmin', 'Survey_TimelyTreatment',
       'Survey_TimelyVisits', 'Survey_Reliability', 'Survey_Options',
       'Survey_HoursTreatment', 'Survey_CourteousStaff',
       'Survey_ActiveListening', 'Female', 'Male', 'Elective Admission',
       'Emergency Admission', 'Observation Admission', 'Blood Work', 'CT Scan',
       'Intravenous', 'MRI'],
      dtype='object')
```

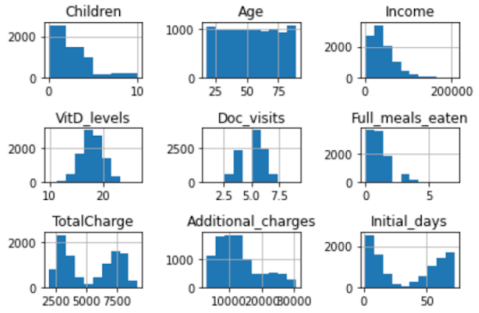
In [12]:
```
# Move ReAdmis to the end of the dataframe
ReAdmis = medical_df[['ReAdmis']]
medical_df.pop("ReAdmis")
medical_df['ReAdmis'] = ReAdmis

df = medical_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Initial_days', 'TotalCharge',
       'Additional_charges', 'Survey_TimelyAdmin', 'Survey_TimelyTreatment',
       'Survey_TimelyVisits', 'Survey_Reliability', 'Survey_Options',
       'Survey_HoursTreatment', 'Survey_CourteousStaff',
       'Survey_ActiveListening', 'Female', 'Male', 'Elective Admission',
       'Emergency Admission', 'Observation Admission', 'Blood Work', 'CT Scan',
       'Intravenous', 'MRI', 'ReAdmis'],
      dtype='object')
```

3.4 Visualizations

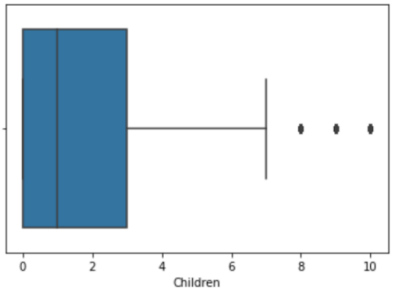
3.4.1 Univariate Statistics

In [13]:
```
# Histogram of continuous variables
medical_df[['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
            'Full_meals_eaten', 'TotalCharge', 'Additional_charges', 'Initial_days']].hist()
plt.savefig('medical_pyplot.jpg')
plt.tight_layout()
```



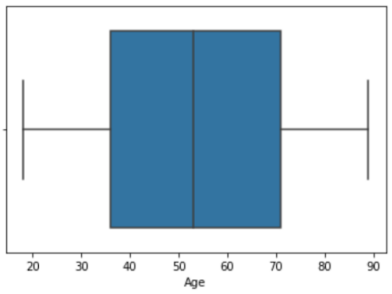
In [14]:
```
# Seaborn boxplots for continuous variables
sns.boxplot('Children', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



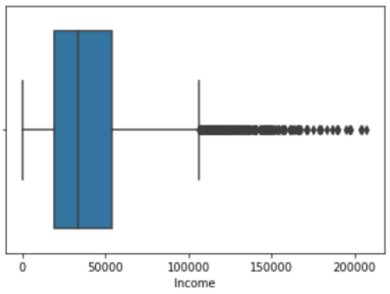
In [15]:
```
sns.boxplot('Age', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



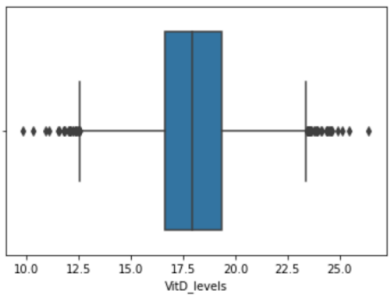
In [16]:
```
sns.boxplot('Income', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



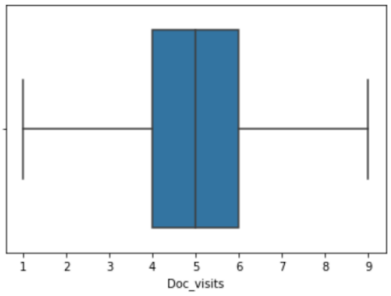
In [17]:
```
sns.boxplot('VitD_levels', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



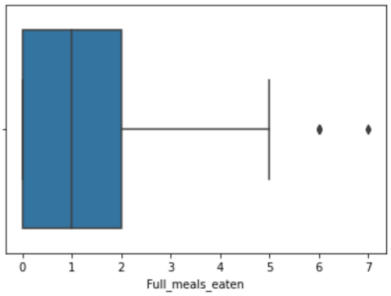
In [18]:
```
sns.boxplot('Doc_visits', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



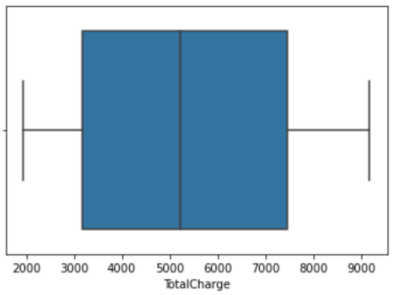
In [19]:
```
sns.boxplot('Full_meals_eaten', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



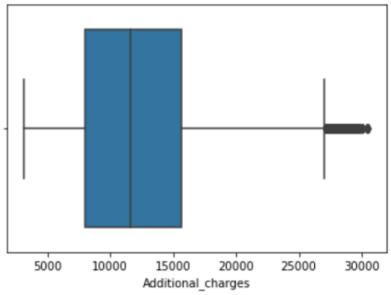
In [20]:
```
sns.boxplot('TotalCharge', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



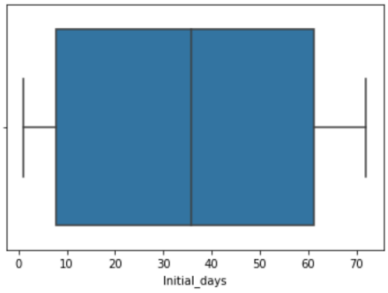
In [21]:
```
sns.boxplot('Additional_charges', data = medical_df)
plt.show()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [22]:
```
sns.boxplot('Initial_days', data = medical_df)
plt.show()
```

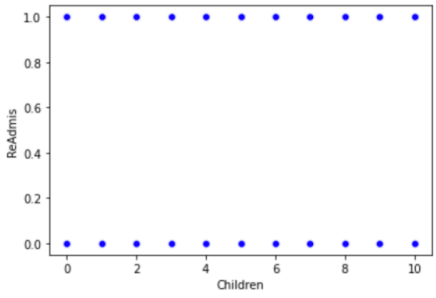
```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



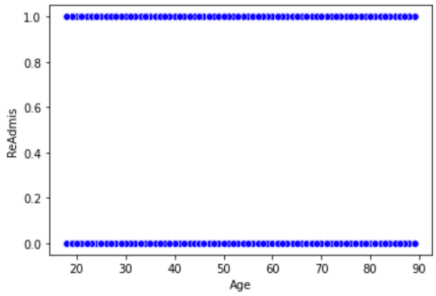
In [23]:
```
# The remaining outliers are justifiable and do not need to be removed
```

3.4.2 Bivariate Statistics

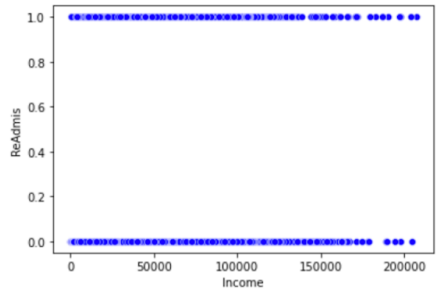
In [24]:
```
# Scatterplots to show relationships between target and independant variables
sns.scatterplot(x=medical_df['Children'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



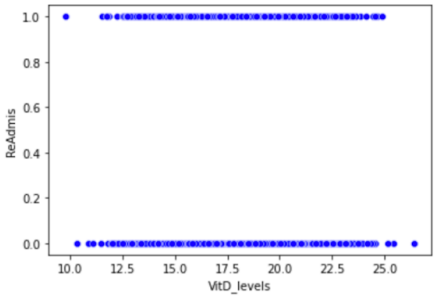
In [25]:
```
sns.scatterplot(x=medical_df['Age'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



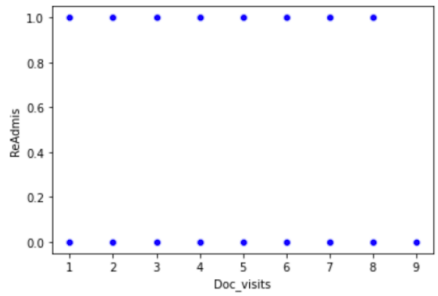
In [26]:
```
sns.scatterplot(x=medical_df['Income'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



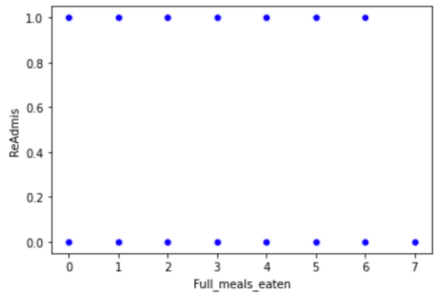
In [27]:
```
sns.scatterplot(x=medical_df['VitD_levels'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



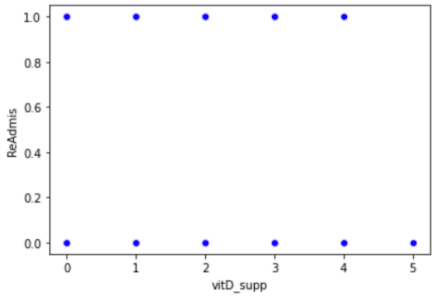
In [28]:
```
sns.scatterplot(x=medical_df['Doc_visits'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



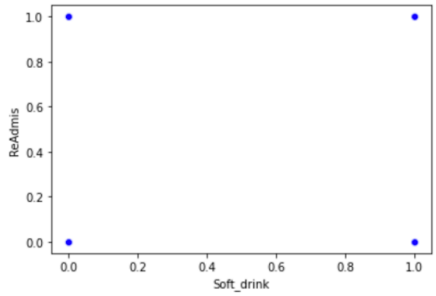
In [29]:
```
sns.scatterplot(x=medical_df['Full_meals_eaten'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



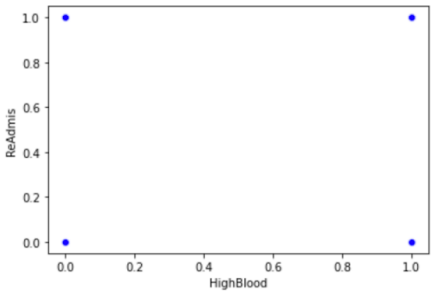
In [30]:
```
sns.scatterplot(x=medical_df['vitD_supp'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



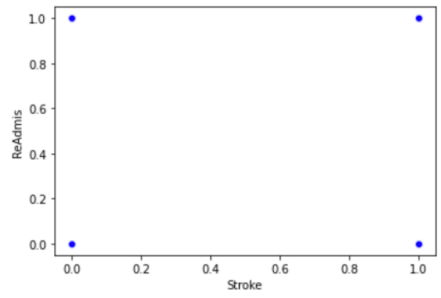
In [31]:
```
sns.scatterplot(x=medical_df['Soft_drink'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



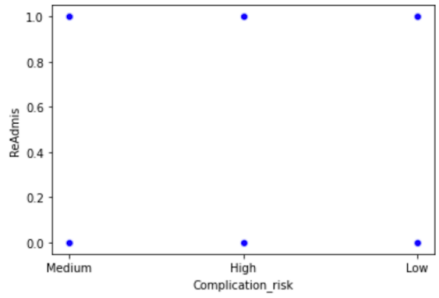
In [32]:
```
sns.scatterplot(x=medical_df['HighBlood'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



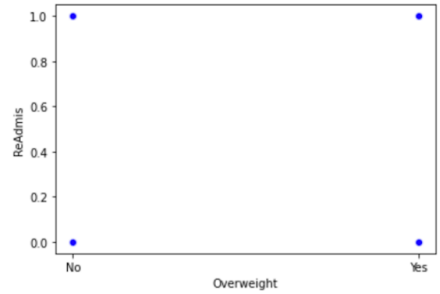
In [33]:
```
sns.scatterplot(x=medical_df['Stroke'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



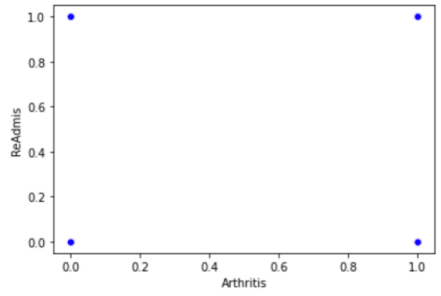
In [34]:
```
sns.scatterplot(x=medical_df['Complication_risk'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



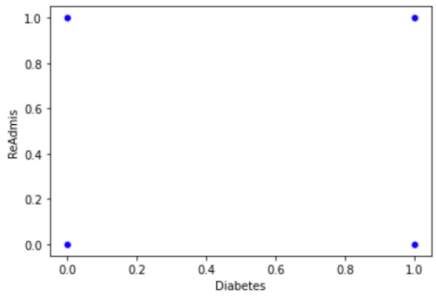
In [35]:
```
sns.scatterplot(x=medical_df['Overweight'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



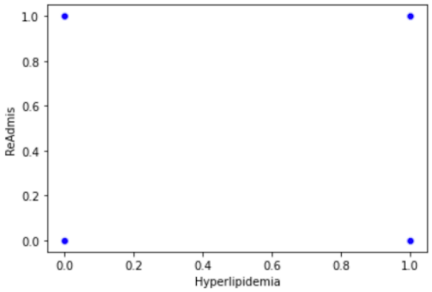
In [36]:
```
sns.scatterplot(x=medical_df['Arthritis'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



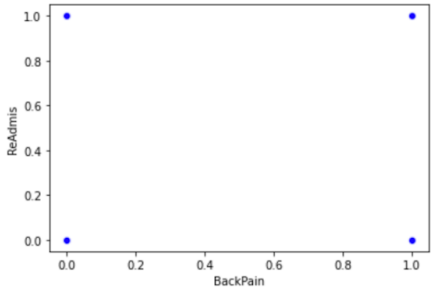
In [37]:
```
sns.scatterplot(x=medical_df['Diabetes'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



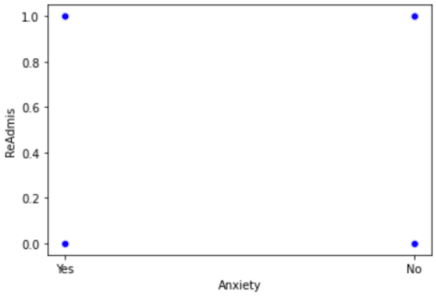
In [38]:
```
sns.scatterplot(x=medical_df['Hyperlipidemia'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



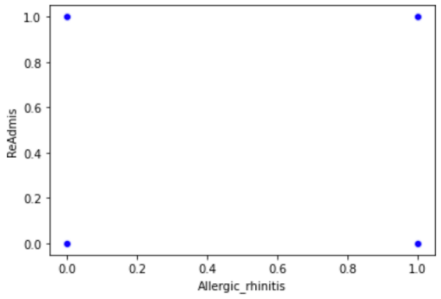
In [39]:
```
sns.scatterplot(x=medical_df['BackPain'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



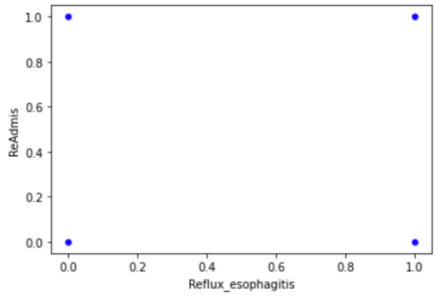
In [40]:
```
sns.scatterplot(x=medical_df['Anxiety'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



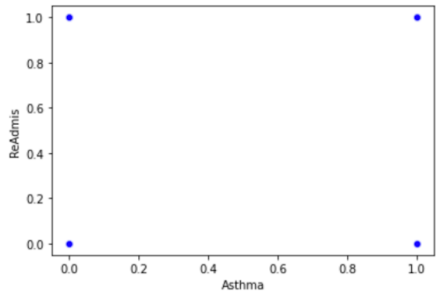
In [41]:
```
sns.scatterplot(x=medical_df['Allergic_rhinitis'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



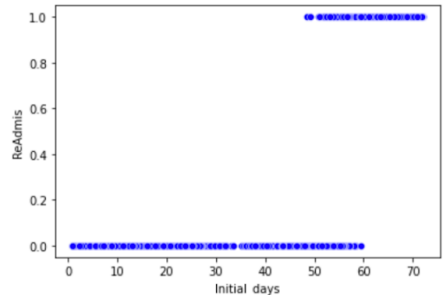
In [42]:
```
sns.scatterplot(x=medical_df['Reflux_esophagitis'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



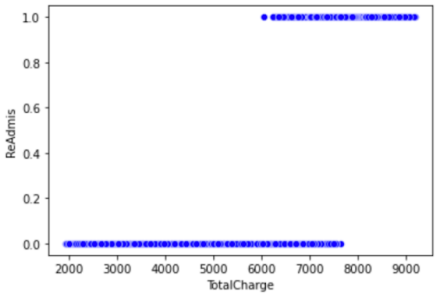
In [43]:
```
sns.scatterplot(x=medical_df['Asthma'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



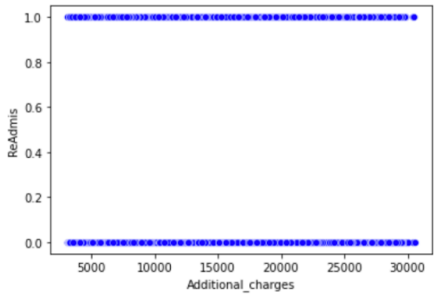
In [44]:
```
sns.scatterplot(x=medical_df['Initial_days'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



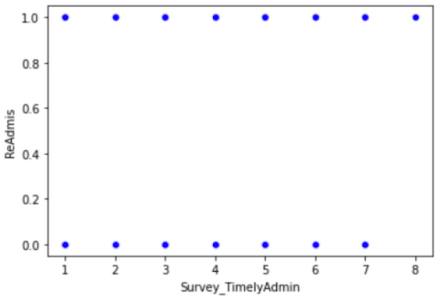
In [45]:
```
sns.scatterplot(x=medical_df['TotalCharge'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



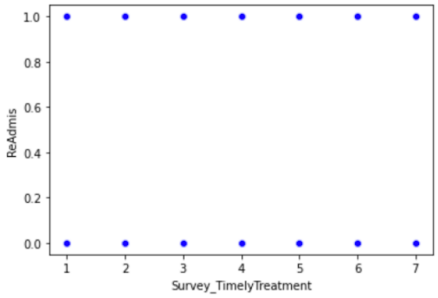
In [46]:
```
sns.scatterplot(x=medical_df['Additional_charges'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



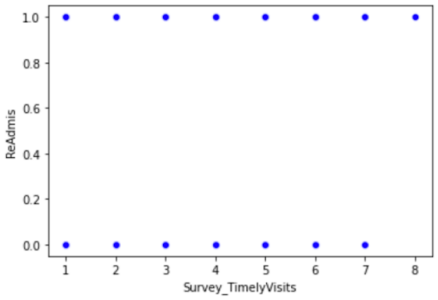
In [47]:
```
sns.scatterplot(x=medical_df['Survey_TimelyAdmin'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



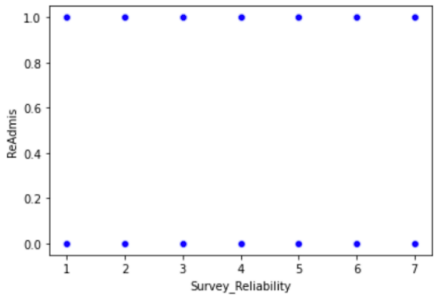
In [48]:
```
sns.scatterplot(x=medical_df['Survey_TimelyTreatment'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



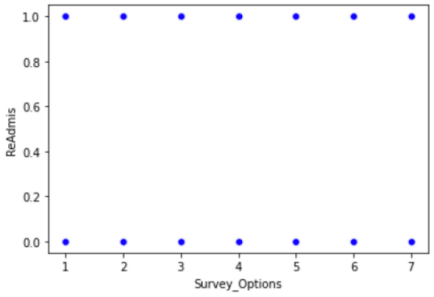
In [49]:
```
sns.scatterplot(x=medical_df['Survey_TimelyVisits'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



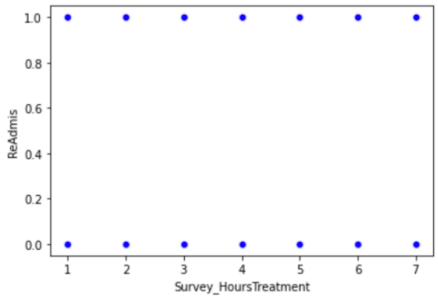
In [50]:
```
sns.scatterplot(x=medical_df['Survey_Reliability'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



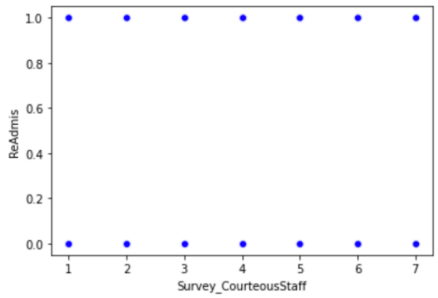
In [51]:
```
sns.scatterplot(x=medical_df['Survey_Options'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



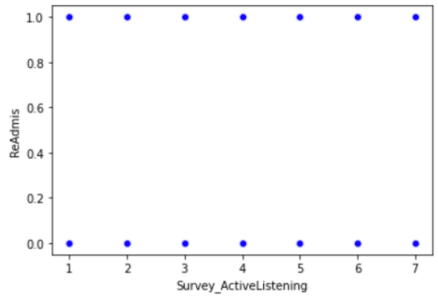
In [52]:
```
sns.scatterplot(x=medical_df['Survey_HoursTreatment'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



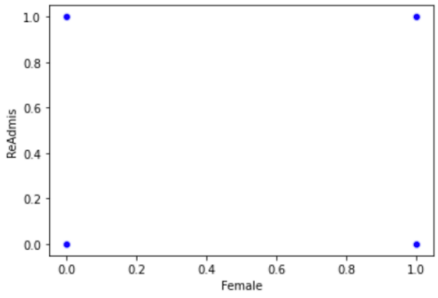
In [53]:
```
sns.scatterplot(x=medical_df['Survey_CourteousStaff'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



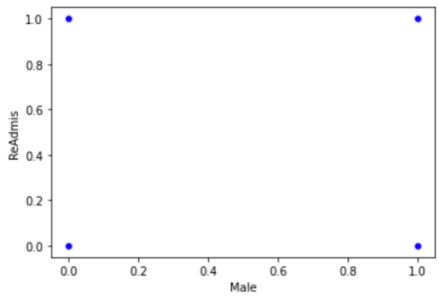
In [54]:
```
sns.scatterplot(x=medical_df['Survey_ActiveListening'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



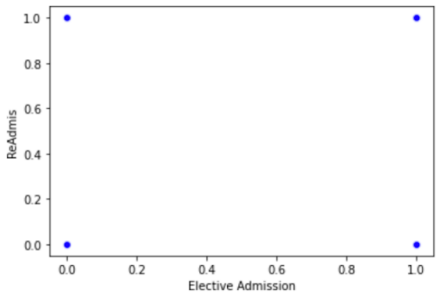
In [55]:
```
sns.scatterplot(x=medical_df['Female'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



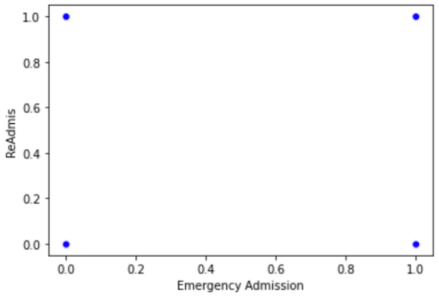
In [56]:
```
sns.scatterplot(x=medical_df['Male'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



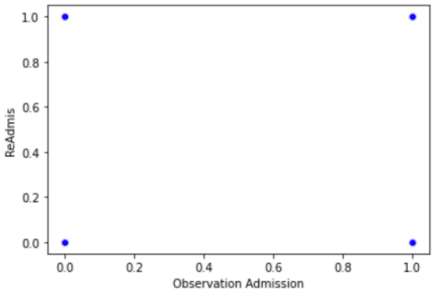
In [57]:
```
sns.scatterplot(x=medical_df['Elective Admission'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



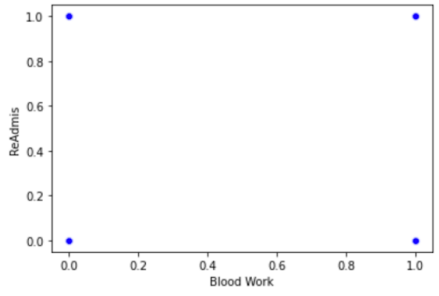
In [58]:
```
sns.scatterplot(x=medical_df['Emergency Admission'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



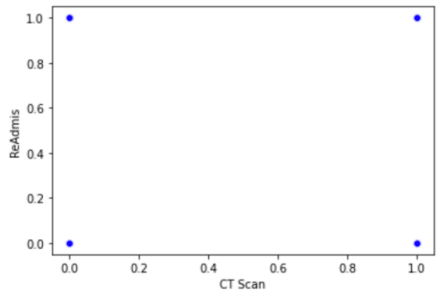
In [59]:
```
sns.scatterplot(x=medical_df['Observation Admission'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



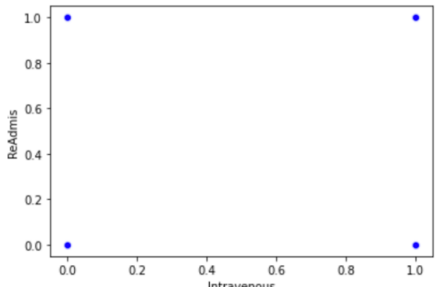
In [60]:
```
sns.scatterplot(x=medical_df['Blood Work'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



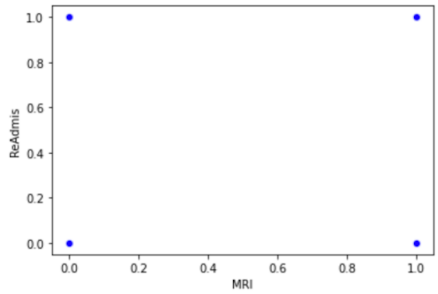
In [61]:
```
sns.scatterplot(x=medical_df['CT Scan'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



In [62]:
```
sns.scatterplot(x=medical_df['Intravenous'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



In [63]:
```
sns.scatterplot(x=medical_df['MRI'], y=medical_df['ReAdmis'], color='blue')
plt.show();
```



3.5 Prepared Dataset

In [64]:
```
# Extract Clean dataset
medical_df.to_csv('medical_prepared_log.csv')

medical_df.columns
```

Out[64]:
```
Index(['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Initial_days', 'TotalCharge', 'Additional_charges',
       'Survey_TimelyAdmin', 'Survey_TimelyTreatment', 'Survey_TimelyVisits',
       'Survey_Reliability', 'Survey_Options', 'Survey_HoursTreatment',
       'Survey_CourteousStaff', 'Survey_ActiveListening', 'Female', 'Male',
       'Elective Admission', 'Emergency Admission', 'Observation Admission',
       'High', 'Low', 'Medium', 'Blood Work', 'CT Scan', 'Intravenous', 'MRI',
       'ReAdmis'],
      dtype='object')
```

This can be found in the attached file named “medical_prepared_log.csv”.

4 Model Comparison

4.1 Initial Multiple Regression Models

In [67]:
```
# Initial estimated regression equation that could be used to predict the probability of ReAdmis, given the only
# non-binary original variables
medical_df['intercept'] = 1
medical_df = pd.get_dummies(medical_df, drop_first = True)
ReAdmis_logit_model = sm.Logit(medical_df['ReAdmis'], medical_df[['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visi
         'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges', 'Survey_TimelyAdmin',
        'Survey_TimelyTreatment', 'Survey_TimelyVisits', 'Survey_Reliability', 'Survey_Options', 'Survey_HoursTreatment
        'Survey_CourteousStaff', 'Survey_ActiveListening', 'intercept']]).fit()
print(ReAdmis_logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.044286
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9981
Method:                           MLE   Df Model:                           18
Date:                Sun, 20 Feb 2022   Pseudo R-squ.:                  0.9326
Time:                        18:49:07   Log-Likelihood:                -442.86
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                     0.0558      0.039      1.425      0.154      -0.021       0.133
Age                         -0.0130      0.006     -2.138      0.033      -0.025      -0.001
Income                  -8.058e-07   3.08e-06     -0.262      0.794   -6.84e-06    5.23e-06
VitD_levels                  0.0328      0.043      0.767      0.443      -0.051       0.117
Doc_visits                  -0.0073      0.081     -0.090      0.928      -0.166       0.152
Full_meals_eaten             0.0067      0.089      0.075      0.940      -0.168       0.182
vitD_supp                   -0.0328      0.136     -0.240      0.810      -0.300       0.235
Initial_days                 0.9238      0.050     18.615      0.000       0.827       1.021
TotalCharge                  0.0019      0.000      7.121      0.000       0.001       0.002
Additional_charges       5.578e-05   1.98e-05      2.823      0.005    1.71e-05    9.45e-05
Survey_TimelyAdmin           0.0077      0.128      0.060      0.952      -0.244       0.259
Survey_TimelyTreatment       0.2462      0.113      2.176      0.030       0.024       0.468
Survey_TimelyVisits         -0.1311      0.108     -1.212      0.225      -0.343       0.081
Survey_Reliability           0.0225      0.096      0.233      0.815      -0.166       0.211
Survey_Options              -0.0888      0.100     -0.887      0.375      -0.285       0.107
Survey_HoursTreatment       -0.0043      0.104     -0.041      0.967      -0.209       0.200
Survey_CourteousStaff        0.0301      0.095      0.315      0.752      -0.157       0.217
Survey_ActiveListening      -0.1779      0.092     -1.944      0.052      -0.357       0.001
intercept                  -63.8903      3.315    -19.275      0.000     -70.387     -57.394
============================================================================================

Possibly complete quasi-separation: A fraction 0.76 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 [66]:
```
medical_df_dummies = medical_df.columns
print(medical_df_dummies)
```

```
Index(['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Initial_days', 'TotalCharge', 'Additional_charges',
       'Survey_TimelyAdmin', 'Survey_TimelyTreatment', 'Survey_TimelyVisits',
       'Survey_Reliability', 'Survey_Options', 'Survey_HoursTreatment',
       'Survey_CourteousStaff', 'Survey_ActiveListening', 'Female', 'Male',
       'Elective Admission', 'Emergency Admission', 'Observation Admission',
       'High', 'Low', 'Medium', 'Blood Work', 'CT Scan', 'Intravenous', 'MRI',
       'ReAdmis', 'intercept'],
      dtype='object')
```

In [69]:
```
# Model including dummy variables
medical_df['intercept'] = 1
medical_df = pd.get_dummies(medical_df, drop_first = True)
ReAdmis_logit_model = sm.Logit(medical_df['ReAdmis'], medical_df[['Children', 'Age', 'Income', 'VitD_levels',
       'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges',
       'Survey_TimelyAdmin', 'Survey_TimelyTreatment', 'Survey_TimelyVisits', 'Survey_Reliability', 'Survey_Options',
       'Survey_HoursTreatment','Survey_CourteousStaff', 'Survey_ActiveListening','Female', 'Male',
       'Elective Admission', 'Emergency Admission', 'Observation Admission', 'Blood Work', 'CT Scan', 'Intravenous',
       'MRI','High', 'Low', 'Medium', 'intercept']]).fit()
print(ReAdmis_logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.038911
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9972
Method:                           MLE   Df Model:                           27
Date:                Sun, 20 Feb 2022   Pseudo R-squ.:                  0.9408
Time:                        18:53:42   Log-Likelihood:                -389.11
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                     0.0701      0.042      1.662      0.097      -0.013       0.153
Age                         -0.0245      0.007     -3.526      0.000      -0.038      -0.011
Income                   -1.08e-06   3.33e-06     -0.324      0.746   -7.61e-06    5.45e-06
VitD_levels                  0.0240      0.046      0.518      0.604      -0.067       0.115
Doc_visits                   0.0101      0.090      0.113      0.910      -0.166       0.186
Full_meals_eaten            -0.0181      0.097     -0.187      0.851      -0.207       0.171
vitD_supp                   -0.0689      0.150     -0.460      0.646      -0.363       0.225
Initial_days                 1.4997      0.112     13.420      0.000       1.281       1.719
TotalCharge                 -0.0032      0.001     -3.299      0.001      -0.005      -0.001
Additional_charges           0.0001   2.38e-05      4.631      0.000    6.36e-05       0.000
Survey_TimelyAdmin           0.0806      0.139      0.580      0.562      -0.192       0.353
Survey_TimelyTreatment       0.2072      0.121      1.708      0.088      -0.031       0.445
Survey_TimelyVisits         -0.1757      0.117     -1.506      0.132      -0.404       0.053
Survey_Reliability           0.0491      0.103      0.477      0.634      -0.153       0.251
Survey_Options              -0.1239      0.106     -1.171      0.242      -0.331       0.084
Survey_HoursTreatment       -0.0245      0.111     -0.220      0.826      -0.242       0.193
Survey_CourteousStaff        0.0729      0.102      0.714      0.475      -0.127       0.273
Survey_ActiveListening      -0.1733      0.097     -1.786      0.074      -0.364       0.017
Female                      -0.4906      0.618     -0.794      0.427      -1.701       0.720
Male                        -0.3090      0.617     -0.501      0.617      -1.518       0.900
Elective Admission         -11.7926        nan        nan        nan         nan         nan
Emergency Admission         -8.0599        nan        nan        nan         nan         nan
Observation Admission      -11.0459        nan        nan        nan         nan         nan
Blood Work                  -8.6950        nan        nan        nan         nan         nan
CT Scan                     -7.1821        nan        nan        nan         nan         nan
Intravenous                 -8.6812        nan        nan        nan         nan         nan
MRI                         -6.3396        nan        nan        nan         nan         nan
High                        -8.8487        nan        nan        nan         nan         nan
Low                        -11.6099        nan        nan        nan         nan         nan
Medium                     -10.4396        nan        nan        nan         nan         nan
intercept                  -30.8982   7.01e+06  -4.41e-06      1.000   -1.37e+07    1.37e+07
============================================================================================

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

The pseudo R squared value went from .9326 to .9408 when the dummy variables were added to the analysis. This means that less than 1% of the variance is associated with the categorical data points. Since this is the case, the removal of the dummy variables from this analysis is justified
Current multiple regression equation:

y = .0558*Children - .0130*Age - 8.058e-07*Income + .0328*VitD_levels - .0073*Doc_visits + .0067*Full_meals_eaten - .0328*vitD_supp + .9238*Initial_days + .0019*TotalCharge + 5.578e-05*Additional_charges + .0077*Survey_TimelyAdmin + .2462*Survey_TimelyTreatment - .1311*Survey_TimelyVisits + .0225*Survey_Reliability - .0888*Survey_Options - .0043*Survey_HoursTreatment - .0301*Survey_CourteousStaff - .1779*Survey_ActiveListening - 63.8903

4.2 Justification

Based on the MLE model, the pseudo R squared value = 93.26%, which is good for finding the variance of the model. Also, the coefficients of the variables are very low (less than .5), except for the variable Initial_days. Variables Initial_days and TotalCharge have a p-value of 0.000 which means they are statistically significant.

The next step will be to remove predictor variables with p-values greater than .05, which indicates that the variable would not be statistically significant.

The remaining variables will be:

Age
Initial_days
TotalCharge
Additional_charges
Survey_TimelyTreatment
Survey_ActiveListening

4.3 Reduced Multiple Regression Model

In [70]:
```
# Reduced multiple regression model
medical_df['intercept'] = 1
ReAdmis_logit_model_reduced = sm.Logit(medical_df['ReAdmis'], medical_df[['Age','Initial_days', 'TotalCharge',
                                                                          'Additional_charges', 'Survey_TimelyTreatment',
                                                                          'Survey_ActiveListening', 'intercept']]).fit()
print(ReAdmis_logit_model_reduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.044611
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                ReAdmis   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9993
Method:                           MLE   Df Model:                            6
Date:                Sun, 20 Feb 2022   Pseudo R-squ.:                 0.9321
Time:                        21:07:08   Log-Likelihood:                -446.11
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                         -0.0124      0.006     -2.054      0.040      -0.024      -0.001
Initial_days                 0.9177      0.049     18.755      0.000       0.822       1.014
TotalCharge                  0.0019      0.000      7.132      0.000       0.001       0.002
Additional_charges        5.65e-05   1.97e-05      2.870      0.004    1.79e-05    9.51e-05
Survey_TimelyTreatment       0.1986      0.085      2.339      0.019       0.032       0.365
Survey_ActiveListening      -0.1667      0.085     -1.958      0.050      -0.334       0.000
intercept                  -63.3240      3.105    -20.392      0.000     -69.410     -57.238
============================================================================================

Possibly complete quasi-separation: A fraction 0.75 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

After removing the variables, the model still explains the variance at 93.21%.

Current multiple regression equation:

y = -.0124*Age + .9177*Initial_days + .0019*TotalCharge + 5.65e-05*Additional_charges + .1989*Survey_TimelyTreatment - .1667*Survey_ActiveListening- 63.3240

5 Model Analysis

5.1 Models Comparison

The second model still explains 93.21% of variance, as shown by the pseudo R squared. There was a suggested alpha threshold of .05 to keep predictor variables. Notably as ReAdmis = 1, Age and Survey_ActiveListening have negative values. The remaining variables have a direct relationship.

These inverse relationships suggest that as a patient gets older and as the importance of active listening increases, the probability of the patient being readmitted decreases. Comparedly, as the initial days, total charge, additional charges, and the importance of a timely treatment increase so does the probability of a readmission.

In [69]:
```
# Confusion Matrix
# Import the prepared dataset
dataset = pd.read_csv('medical_prepared_log.csv')
x = dataset.iloc[:,1:-1].values
y = dataset.iloc[:, -1].values
```

In [70]:
```
# Split the dataset into the training and test sets
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 [71]:
```
# Training the Logistic Regression model on the training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)
```

Out[71]:
```
LogisticRegression(random_state=0)
```

In [72]:
```
# Predict the test set results
y_predict = classifier.predict(x_test)
```

In [73]:
```
# Make the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict)
print(cm)
```

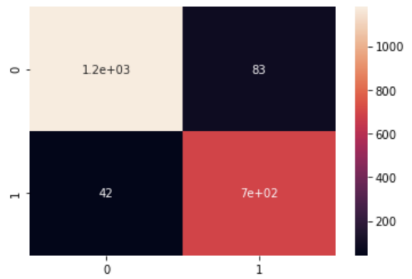
```
[[1179   83]
 [  42  696]]
```

In [74]:
```
# Compute the accuracy with k-Fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 94.89 %
Standard Deviation: 0.87 %
```

In [75]:
```
y_predict_test = classifier.predict(x_test)
cm2 = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm2, annot=True)
```

Out[75]:
```
<AxesSubplot:>
```



```
In [76]: # Classification Report
         from sklearn.metrics import classification_report
         print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      1262
           1       0.89      0.94      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
```

5.2 Output

Output and calculations can be found above.

5.3 Code

All code for analysis can be found above.

6 Summary

6.1 Results of Analysis

- The final logistic regression equation has **6** independent variables:
  y = -.0124*Age + .9177*Initial_days + .0019*TotalCharge + 5.65e-05*Additional_charges + .1989*Survey_TimelyTreatment - .1667*Survey_ActiveListening- 63.3240
- The coefficients suggest that for every 1 unit of:
  - Age - probability of readmission will decrease by .0124
  - Initial_days - probability of readmission will increase by .9177
  - TotalCharge - probability of readmission will increase by .0019
  - Additional_charges - probability of readmission will increase by 5.65e-05
  - Survey_TimelyTreatment - probability of readmission will increase by .1989
  - Survey_ActiveListening - probability of readmission will decrease by .1667
- P-values for Initial_days and TotalCharge, 0.000, indicate that they are more statistically significant than the other predictor variables.

6.2 Limitations of Analysis

Generally, the major limitation of logistic regression is that it assumes that the dependent variable and the independent variables have a linear relationship. It can also be noted that it is inconclusive to whether or not the dependent variable is in fact affecting the independent variables instead. Another limitation that can be found with this analysis is the dataset is small and could benefit from more data collection.

6.3 Recommended Course of Action

From this analysis, there is a significant linear relationship between readmission rates and the initial days a patient is admitted/the total amount charged for the initial visit. It could be assumed that it would be in the best interest of medical personnel to increase productivity of a hospital so as to decrease readmission rates. Intuitively however, there are most likely some unknown variables affecting the relationship, as well as the knowledge that this analysis does not indicate causation. In this scenario, an unknown variable could be a pre-existing illness. A pre-existing illness would increase the days of an initial visit, increase the total amount charged to the patient due to the need of a specialist and most likely need followup appointments. Overall, there would need to be more data collected and further analysis done before any definitive conclusions can be drawn.

7 Supporting Documents

7.1 Video

This can be found within the attached file ‘Panopto Recording’.

7.2 Sources for Third-Party Code

Himamsh, Viveka. “How to Create Dummy Variables in Python with Pandas?” *GeeksforGeeks*, 8 Oct. 2021, https://www.geeksforgeeks.org/how-to-create-dummy-variables-in-python-with-pandas/.

“Python List Pop() Method.” Python List Pop() Method, https://www.w3schools.com/python/ref_list_pop.asp.

7.3 Sources

Gagner, David. (2022). D207 Exploratory Data Analysis . Salt Lake City ; Western Governors University.

Western Governors University. (n.d.). D207 D208 D209 Medical Data Considerations and Dictionary. Salt Lake City.