|  |
| --- |
| D208 |
| Logistic Regression Modeling |
| Task 2 |

|  |
| --- |
| Shantel Johnson  3-30-2023 |

Contents

[Section A 3](#_Toc131083687)

[Part 1: Research Question 3](#_Toc131083688)

[This information could prove useful because by identifying the factors that drive readmission, hospitals can work towards lowering readmission rates and avoiding penalties from the US Centers for Medicare and Medicaid Services (CMS). 3](#_Toc131083689)

[Part 2: Goals 3](#_Toc131083690)

[Section B 3](#_Toc131083691)

[Part 1: Assumptions of Logistic Regression 3](#_Toc131083692)

[Part 2: Python for Data Analysis 3](#_Toc131083693)

[Part 3: Justification of Methods 4](#_Toc131083694)

[Section C 4](#_Toc131083695)

[Part 1: Data Cleaning 4](#_Toc131083696)

[Duplication 4](#_Toc131083697)

[Missing Values 4](#_Toc131083698)

[Outliers 4](#_Toc131083699)

[Part 2: Data Descriptions 4](#_Toc131083700)

[Part 3: Visualization 4](#_Toc131083701)

[Part 4: Data Transformation 5](#_Toc131083702)

[Ordinal Encoding 5](#_Toc131083703)

[Nominal Encoding 5](#_Toc131083704)

[Part 5: Output 5](#_Toc131083705)

[Section D 5](#_Toc131083706)

[Part 1: Initial Model 5](#_Toc131083707)

[Part 2: Feature Selection 5](#_Toc131083708)

[Part 3: Reduced Model 6](#_Toc131083709)

[Section E 6](#_Toc131083710)

[Part 1: Model Comparison 6](#_Toc131083711)

[Part 2: Output 6](#_Toc131083712)

[Part 3: Code 6](#_Toc131083713)

[Section F 6](#_Toc131083714)

[Part 1: Results 6](#_Toc131083715)

[Regression Equation 6](#_Toc131083716)

[Interpretation 7](#_Toc131083717)

[Statistical Significance 8](#_Toc131083718)

[Practical Significance 8](#_Toc131083719)

[Limitations 8](#_Toc131083720)

[Part 2: Next Steps 8](#_Toc131083721)

[Section G 8](#_Toc131083722)

[Part 1: Data Preparation Code 8](#_Toc131083723)

[Data Cleaning 12](#_Toc131083724)

[Data Descriptions 15](#_Toc131083725)

[Visualizations 17](#_Toc131083726)

[Data Transformation 46](#_Toc131083727)

[Part 2: Logistic Regression Modeling Code 50](#_Toc131083728)

[Logistic Regression Model 1: All Features 50](#_Toc131083729)

[**Evaluation** 53](#_Toc131083730)

[Logistic Regression Model 2: Reduced Features 55](#_Toc131083731)

[Section H 59](#_Toc131083732)

[Section I 59](#_Toc131083733)

[Part 1: Web Sources 59](#_Toc131083734)

[Part 2: References 59](#_Toc131083735)

# Section A

## Part 1: Research Question

The dataset for this project comprises observations on patient demographics, medical conditions, and hospital readmission. As the project's data analyst, my responsibility is to answer the research question: What factors cause a patient to be readmitted to the hospital within 30 days of release?

## This information could prove useful because by identifying the factors that drive readmission, hospitals can work towards lowering readmission rates and avoiding penalties from the US Centers for Medicare and Medicaid Services (CMS).

## Part 2: Goals

This project aims to identify factors (i.e., variables) in the dataset that contribute to patient readmission and establish the correlation between them. To achieve these goals, the dataset requires preparation for statistical analysis using logistic regression.

# Section B

## Part 1: Assumptions of Logistic Regression

Four assumptions to be aware of when using a linear regression model are:

* Binary response: Logistic regression assumes that the dependent variable can only take on two possible outcomes (Bobbitt, 2020).
* Independent observations: Observations in the dataset must be independent of one another (Bobbitt, 2020).
* Multicollinearity: Multicollinearity can lead to dubious regression coefficients. As such, predictor variables should be removed from the model until multicollinearity is gone (Bruce, Bruce, & Gedeck, 2019).
* Outliers: Outliers can distort the results of regression modeling. For this reason, predictor variables should have normal distributions (Bruce, Bruce, & Gedeck, 2019).

## Part 2: Python for Data Analysis

For logistic regression in particular, Python is useful because 1) of the availability of packages that are used for data preprocessing and exploratory data analysis (pandas and scipy) and 2) of the availability of packages that are used for building logistic regression models (scikit-learn and statsmodels).

|  |  |
| --- | --- |
| **Package/Library** | **Purpose** |
| pandas | Data manipulation & exploratory data analysis |
| scipy | Exploratory data analysis |
| scikit-learn | Feature engineering & logistic regression modeling |
| statsmodels | Logistic regression modeling |
| matplotlib | Visualization |
| seaborn | Visualization |

## Part 3: Justification of Methods

Like linear regression, logistic regression quantifies the nature of the relationship between one or more predictor variables and the target variable. The key difference between linear and logistic regression, however, is that the outcome for logistic regression is binary (Bruce, Bruce, & Gedeck, 2019). As defined in Section A, Part 1 the research question seeks to understand what factors (the predictor variables) influence whether a patient is readmitted (the target variable). Based on this, we can see that logistic regression is an appropriate technique for analyzing the research question.

# Section C

## Part 1: Data Cleaning

Data cleaning was performed to identify and remediate the common data quality issues of duplication, missing values, and outliers. Refer to Section G, Part 1 to view the associated data cleaning code.

### Duplication

The pandas.DataFrame.duplicated() function was applied on the dataset to return duplicated rows. No issues were detected after using this method.

### Missing Values

The pandas.DataFrame.isnull.sum() function was applied on the dataset to return the number of missing values found in each variable. No issues were detected after using this method.

### Outliers

The scipy.stats.zscore() function was applied on the dataset to return the number of outliers found in each numeric variable. The following data issues were identified:

|  |  |
| --- | --- |
| **Variable** | **Number of Outliers** |
| Population | 218 |
| Children | 202 |
| Income | 143 |
| VitD\_levels | 24 |
| Doc\_visits | 8 |
| Full\_meals\_eaten | 33 |
| vitD\_supp | 70 |

As noted in Section B, Part 1, outliers impact the results of regression modeling. For this reason, records containing outliers were dropped from the dataset using the pandas.DataFrame.drop() function.

## Part 2: Data Descriptions

Summary statistics were calculated for the dependent and all independent variables. Refer to Section G, Part 1 for more information.

## Part 3: Visualization

Distributions of the dependent and independent variables were depicted using univariate and bivariate plots. To view these visualizations, refer to Section G, Part 1.

## Part 4: Data Transformation

Data transformation was performed on categorical variables in preparation for linear regression modeling (refer to Section G, Part 1).

### Ordinal Encoding

Statistical models should be presented with a numeric encoding of ordered categories that represent linear ordering (Kuhn & Johnson, 2019). If a categorical variable consisted strictly of binary values (e.g., yes/no, true false), then these variables were transformed using the pandas.DataFrame.replace() function. Otherwise, they were transformed into numeric values using the sklearn.preprocessing.OrdinalEncoder() function.

### Nominal Encoding

According to Kuhn and Johnson, the most basic approach for re-expressing categorical variables as numeric data is to create C-1 indicator variables, where C represents the possible values of the predictor. (2019). Categorical variables of the nominal type were transformed into numeric values using the pandas.get\_dummies() function.

## Part 5: Output

To view the output of the data preparation tasks performed in Section C, refer to the attached file, medical\_prepared\_data2.csv.

# Section D

## Part 1: Initial Model

An initial logistic regression model was constructed using the independent variables identified in Section C, Part 2. Refer to Section G, Part 2 to view the initial model.

## Part 2: Feature Selection

Feature selection is important in predictive modeling because the predictive performance of the model often decreases as the number of uninformative predictors increases. In addition, removing predictors can reduce the cost of data acquisition and can increase throughput of the software used to make predictions (Kuhn & Johnson, 2019). For this reason, the number of predictor variables was reduced using feature selection.

The sklearn.feature\_selection.SequentialFeatureSelection() function was used to reduce the number of predictor variables included in the model. The sequential feature selector (SFS) uses forward selection to form a feature subset in a greedy fashion; the features are chosen based on the cross-validation score of an estimator, which in the case of logistic regression, is accuracy (Pedregosa, et. al, 2011). The SFS falls under the wrapper methodology for feature selection. According to Kuhn and Johnson, wrapper methods use iterative search procedures that repeatedly supply predictor subsets to the model (2019).

## Part 3: Reduced Model

A reduced logistic regression model was constructed using a subset of the independent variables. Refer to Section G, Part 2 to view the reduced model.

# Section E

## Part 1: Model Comparison

The following tables compare the initial and reduced linear regression models:

|  |  |  |
| --- | --- | --- |
| **Model** | **Predictor Variables** | **Accuracy** |
| Initial | 33 | 0.986 |
| Reduced | 8 | 0.981 |

|  |  |  |
| --- | --- | --- |
| **Confusion Matrix:**  **Initial Model** | | |
|  | **y (0)** | **y (1)** |
| **y\_pred (0)** | 5843 | 67 |
| **y\_pred (1)** | 62 | 3359 |

|  |  |  |
| --- | --- | --- |
| **Confusion Matrix:**  **Reduced Model** | | |
|  | **y (0)** | **y (1)** |
| **y\_pred (0)** | 5824 | 86 |
| **y\_pred (1)** | 87 | 3334 |

While the initial model includes significantly more predictor variables than the reduced model, both models have approximately the same accuracy score. This means that the 33 independent variables in the initial model predict the dependent variable with the same accuracy as the 4 independent variables in the reduced model. In addition, the confusion matrices for the initial and reduced models are near identical for predicted and actual values. For more information, refer to Section G, Part2.

## Part 2: Output

To view the output of the analysis, refer to Part 1 of this section.

## Part 3: Code

To review the executable code used to build the initial and reduced linear regression models, refer to the attached file, ‘D208 Task2 Model.csv’.

# Section F

## Part 1: Results

### Regression Equation

The regression equation for the reduced logistic regression model is given below:

### Interpretation

For each predictor variable, the coefficient sign indicates whether the probability of ReAdmis=1 increases or decreases as the predictor variable increases. Stroke, Complication\_risk, Initial\_days, and Emergency\_Admission all have positive coefficients, meaning that as these variables increase, the probability that ReAdmis=1 increases. The coefficient values represent the estimated change in the log-odds of the target variable given a one-unit increase in the predictor variable (Larose & Larose, 2019). The change in log-odds is specified for each variable below:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Odds** | **Meaning** |
| Stroke | exp(1.3984) = 4.05 | A patient is 4.05 times more likely to be readmitted if they have had a stroke. |
| Complication\_risk | exp(0.6286) = 1.87 | As their complication risk increases, a patient is 1.87 times more likely to be readmitted. |
| Initial\_days | exp(1.126) = 3.08 | For every day of their initial stay in the hospital, a patient is 3.08 times more likely to be readmitted. |
| Emergency\_Admission | exp(1.6313) = 5.11 | A patient is 5.11 times more likely to be readmitted if they had an emergency admission. |

For example, to predict whether a patient who 1) did not have a stroke, 2) with a medium complication risk, 3) that spent three days in the hospital initially, and 4) had an emergency admission was readmitted, plug the variables into the above regression equation:

Based on this result, the patient will not be readmitted to the hospital.

### Statistical Significance

According to the summary output (refer to Section G, Part 2), the predictor variables included in the reduced model are significant as they all have p-values less than 0.05.

### Practical Significance

The practical significance of this model lies in its ability to identify and quantify the factors that drive hospital readmission (stroke, complication risk, initial length of stay, and emergency admission). By using this model, healthcare providers can proactively identify patients at risk of readmission and implement targeted intervention to prevent it.

### Limitations

The following are limitations associated with the analysis:

* The dataset does not include the feature “reason for hospital admission”. Including this factor as a predictor variable in the model may improve its accuracy score, leading to better predictions regarding a patient's likelihood of readmission.
* The feature "Complication\_risk" is assessed by the primary patient assessment and has values of "low", "medium", and "high". However, it is unclear whether this assessment is administered by a medical professional or self-administered. As a result, it is possible that the values of "Complication\_risk" are not standardized, which could potentially skew the results of the model.

## Part 2: Next Steps

I would first recommend adding “reason for hospital admission” as a feature in the dataset. From there, SFS can be re-performed to determine whether “reason for hospital admission” improves the model. Next, I recommend investigating the administration of the primary patient assessment to better understand the quality of data included in the “Complication\_risk” factor. Once these steps are complete, the data engineering team can work on scaling the model for commercial use.

# Section G

## Part 1: Data Preparation Code

In [1]:

# read data into pandas DataFrame

# import pandas for data manipulation

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

import pandas as pd

df = pd.read\_csv('medical dataset/medical\_clean.csv')

In [2]:

# reduce df to relevant features

df = df[['Population',

'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'

]]

In [3]:

# get first 3 rows of data

df.head(3).transpose()

Out[3]:

|  | **0** | **1** | **2** |
| --- | --- | --- | --- |
| **Population** | 2951 | 11303 | 17125 |
| **Children** | 1 | 3 | 3 |
| **Age** | 53 | 51 | 53 |
| **Income** | 86575.93 | 46805.99 | 14370.14 |
| **Marital** | Divorced | Married | Widowed |
| **Gender** | Male | Female | Female |
| **ReAdmis** | No | No | No |
| **VitD\_levels** | 19.141466 | 18.940352 | 18.057507 |
| **Doc\_visits** | 6 | 4 | 4 |
| **Full\_meals\_eaten** | 0 | 2 | 1 |
| **vitD\_supp** | 0 | 1 | 0 |
| **Soft\_drink** | No | No | No |
| **Initial\_admin** | Emergency Admission | Emergency Admission | Elective Admission |
| **HighBlood** | Yes | Yes | Yes |
| **Stroke** | No | No | No |
| **Complication\_risk** | Medium | High | Medium |
| **Overweight** | No | Yes | Yes |
| **Arthritis** | Yes | No | No |
| **Diabetes** | Yes | No | Yes |
| **Hyperlipidemia** | No | No | No |
| **BackPain** | Yes | No | No |
| **Anxiety** | Yes | No | No |
| **Allergic\_rhinitis** | Yes | No | No |
| **Reflux\_esophagitis** | No | Yes | No |
| **Asthma** | Yes | No | No |
| **Services** | Blood Work | Intravenous | Blood Work |
| **Initial\_days** | 10.58577 | 15.129562 | 4.772177 |

In [4]:

# get information on df

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 27 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Population 10000 non-null int64

1 Children 10000 non-null int64

2 Age 10000 non-null int64

3 Income 10000 non-null float64

4 Marital 10000 non-null object

5 Gender 10000 non-null object

6 ReAdmis 10000 non-null object

7 VitD\_levels 10000 non-null float64

8 Doc\_visits 10000 non-null int64

9 Full\_meals\_eaten 10000 non-null int64

10 vitD\_supp 10000 non-null int64

11 Soft\_drink 10000 non-null object

12 Initial\_admin 10000 non-null object

13 HighBlood 10000 non-null object

14 Stroke 10000 non-null object

15 Complication\_risk 10000 non-null object

16 Overweight 10000 non-null object

17 Arthritis 10000 non-null object

18 Diabetes 10000 non-null object

19 Hyperlipidemia 10000 non-null object

20 BackPain 10000 non-null object

21 Anxiety 10000 non-null object

22 Allergic\_rhinitis 10000 non-null object

23 Reflux\_esophagitis 10000 non-null object

24 Asthma 10000 non-null object

25 Services 10000 non-null object

26 Initial\_days 10000 non-null float64

dtypes: float64(3), int64(6), object(18)

memory usage: 2.1+ MB

In [5]:

# get number of unique values for each feature

df.nunique()

Out[5]:

Population 5951

Children 11

Age 72

Income 9993

Marital 5

Gender 3

ReAdmis 2

VitD\_levels 9976

Doc\_visits 9

Full\_meals\_eaten 8

vitD\_supp 6

Soft\_drink 2

Initial\_admin 3

HighBlood 2

Stroke 2

Complication\_risk 3

Overweight 2

Arthritis 2

Diabetes 2

Hyperlipidemia 2

BackPain 2

Anxiety 2

Allergic\_rhinitis 2

Reflux\_esophagitis 2

Asthma 2

Services 4

Initial\_days 9997

dtype: int64

### Data Cleaning[¶](" \l "Data-Cleaning)

#### Issue Identification[¶](#Issue-Identification)

##### Duplicates[¶](#Duplicates)

In [6]:

# check duplication across entire row

df.duplicated().sum()

Out[6]:

0

##### Missing Values[¶](#Missing-Values)

In [7]:

# check for missing values

df.isnull().sum()

Out[7]:

Population 0

Children 0

Age 0

Income 0

Marital 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

dtype: int64

##### Outliers[¶](#Outliers)

In [8]:

# create new DataFrame with numeric features

df\_num = df.select\_dtypes(include='number')

df\_num.head()

Out[8]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Initial\_days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2951 | 1 | 53 | 86575.93 | 19.141466 | 6 | 0 | 0 | 10.585770 |
| **1** | 11303 | 3 | 51 | 46805.99 | 18.940352 | 4 | 2 | 1 | 15.129562 |
| **2** | 17125 | 3 | 53 | 14370.14 | 18.057507 | 4 | 1 | 0 | 4.772177 |
| **3** | 2162 | 0 | 78 | 39741.49 | 16.576858 | 4 | 1 | 0 | 1.714879 |
| **4** | 5287 | 1 | 22 | 1209.56 | 17.439069 | 5 | 0 | 2 | 1.254807 |

In [9]:

# import scipy to calculate z-scores

# Title: scipy/scipy: Scipy

# Author: Gommers, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7655153

from scipy import stats

# replace values in df\_num with z-scores

df\_num = stats.zscore(df\_num)

df\_num.head()

Out[9]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Initial\_days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -0.473168 | -0.507129 | -0.024795 | 1.615914 | 0.583603 | 0.944647 | -0.993387 | -0.634713 | -0.907310 |
| **1** | 0.090242 | 0.417277 | -0.121706 | 0.221443 | 0.483901 | -0.967981 | 0.990609 | 0.956445 | -0.734595 |
| **2** | 0.482983 | 0.417277 | -0.024795 | -0.915870 | 0.046227 | -0.967981 | -0.001389 | -0.634713 | -1.128292 |
| **3** | -0.526393 | -0.969332 | 1.186592 | -0.026263 | -0.687811 | -0.967981 | -0.001389 | -0.634713 | -1.244503 |
| **4** | -0.315586 | -0.507129 | -1.526914 | -1.377325 | -0.260366 | -0.011667 | -0.993387 | 2.547602 | -1.261991 |

In [10]:

# return number of outliers found in each column of df\_num

df\_num[df\_num.abs() >= 3].count()

Out[10]:

Population 218

Children 202

Age 0

Income 143

VitD\_levels 24

Doc\_visits 8

Full\_meals\_eaten 33

vitD\_supp 70

Initial\_days 0

dtype: int64

#### Issue Handling

##### Outliers[¶](#Outliers)

In [11]:

# drop records containing outliers

for col in df\_num.columns:

df.drop(df\_num.loc[df\_num[col].abs() >= 3].index, inplace=True, errors='ignore')

df.shape

Out[11]:

(9331, 27)

### Data Descriptions[¶](" \l "Data-Descriptions)

In [12]:

# get summary statistics for numeric features

df.describe()

Out[12]:

|  | **Population** | **Children** | **Age** | **Income** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Initial\_days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 | 9331.000000 |
| **mean** | 8701.586754 | 1.942343 | 53.510342 | 38895.729734 | 17.961006 | 5.014146 | 0.989176 | 0.377880 | 34.369673 |
| **std** | 12063.299403 | 1.898241 | 20.623584 | 25378.151394 | 1.987202 | 1.039526 | 0.978105 | 0.585608 | 26.328806 |
| **min** | 0.000000 | 0.000000 | 18.000000 | 154.080000 | 12.038470 | 2.000000 | 0.000000 | 0.000000 | 1.001981 |
| **25%** | 683.000000 | 0.000000 | 36.000000 | 19414.450000 | 16.639135 | 4.000000 | 0.000000 | 0.000000 | 7.849501 |
| **50%** | 2640.000000 | 1.000000 | 53.000000 | 33261.940000 | 17.944850 | 5.000000 | 1.000000 | 0.000000 | 30.539196 |
| **75%** | 12463.000000 | 3.000000 | 71.000000 | 53213.445000 | 19.331745 | 6.000000 | 2.000000 | 1.000000 | 61.164190 |
| **max** | 54413.000000 | 8.000000 | 89.000000 | 125977.370000 | 23.941426 | 8.000000 | 4.000000 | 2.000000 | 71.981490 |

In [13]:

# get summary statistics for categorical features

df.describe(exclude='number')

Out[13]:

|  | **Marital** | **Gender** | **ReAdmis** | **Soft\_drink** | **Initial\_admin** | **HighBlood** | **Stroke** | **Complication\_risk** | **Overweight** | **Arthritis** | **Diabetes** | **Hyperlipidemia** | **BackPain** | **Anxiety** | **Allergic\_rhinitis** | **Reflux\_esophagitis** | **Asthma** | **Services** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 | 9331 |
| **unique** | 5 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 |
| **top** | Widowed | Female | No | No | Emergency Admission | No | No | Medium | Yes | No | No | No | No | No | No | No | No | Blood Work |
| **freq** | 1921 | 4677 | 5910 | 6923 | 4724 | 5525 | 7466 | 4224 | 6629 | 5992 | 6774 | 6195 | 5515 | 6315 | 5654 | 5514 | 6623 | 4917 |

### Visualizations[¶](" \l "Visualizations)

#### Univariate Plots[¶](#Univariate-Plots)

In [14]:

# import matplotlib and seaborn for visualizations

# Title: matplotlib/matplotlib

# Author: Caswell, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7697899

# Title: seaborn: statistical data visualization

# Author: Michael Waskom

# Date: 2021

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.4645478

import matplotlib.pyplot as plt

import seaborn as sns

In [15]:

# create new DataFrames numerical and categorical data

numerical = df.select\_dtypes(include='number')

categorical = df.select\_dtypes(exclude='number')

In [16]:

# create all barcharts

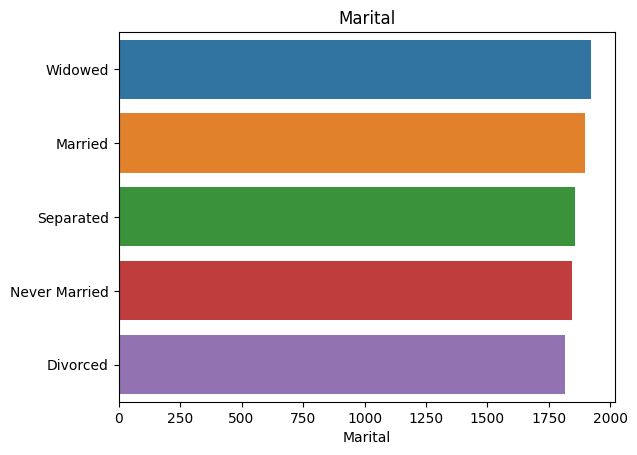
for col in categorical.columns:

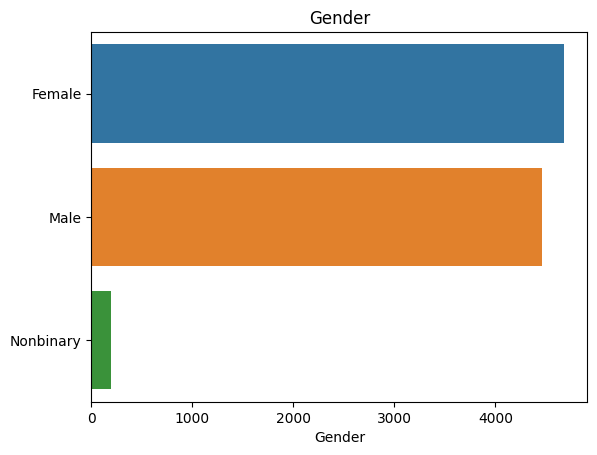
plt.figure()

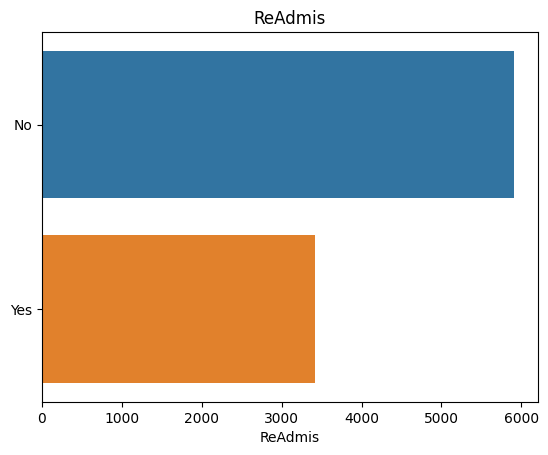
sns.barplot(data=categorical, x=categorical[col].value\_counts(), y=categorical[col].value\_counts().index)

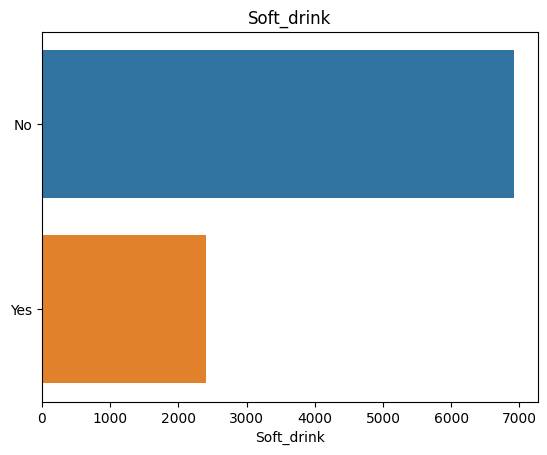
plt.title(col)

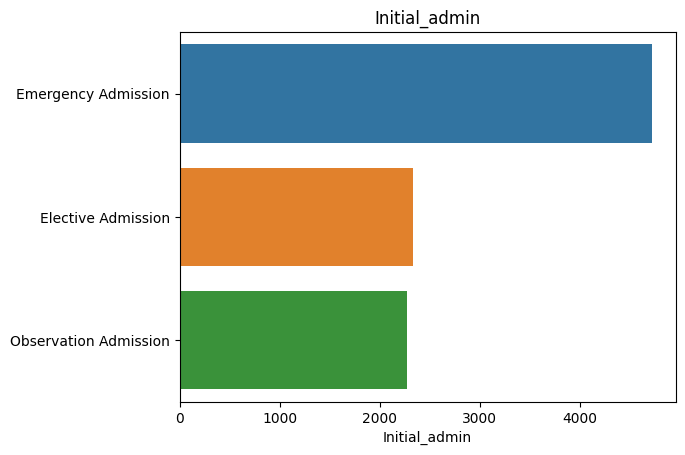
plt.show()

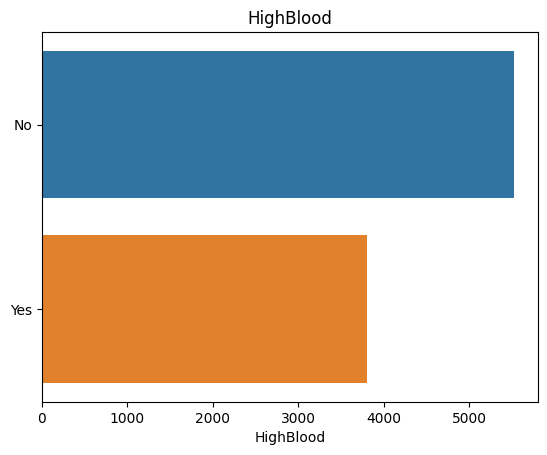


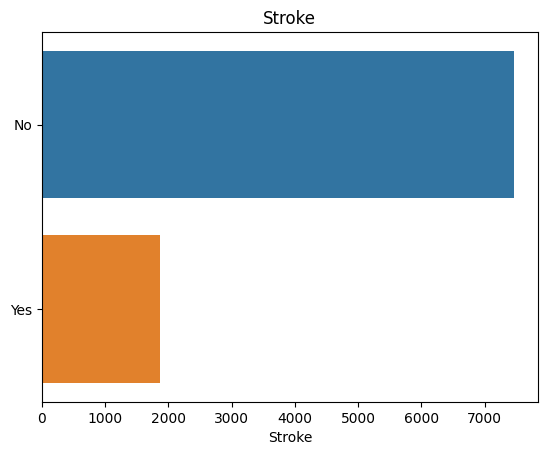


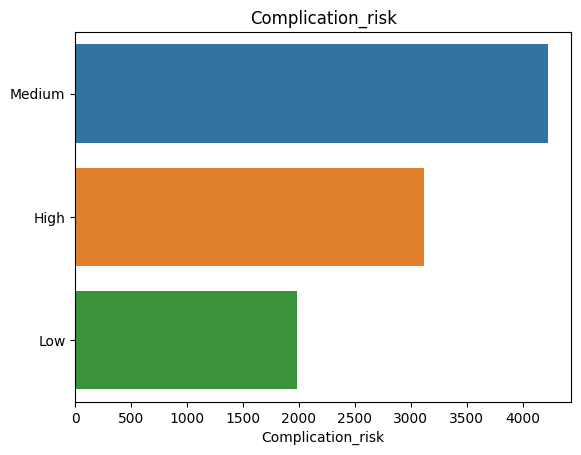


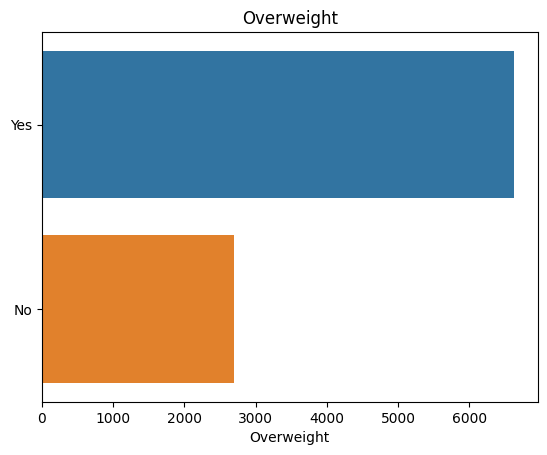


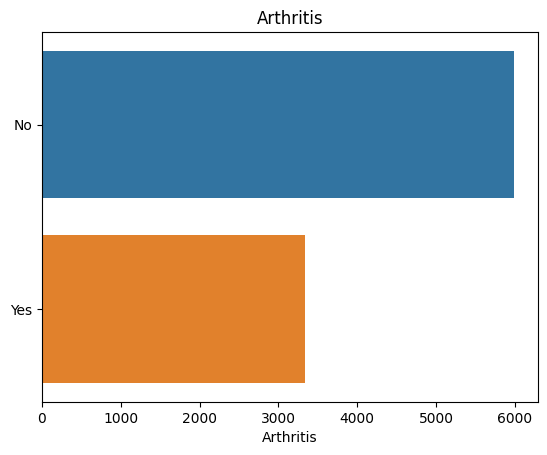


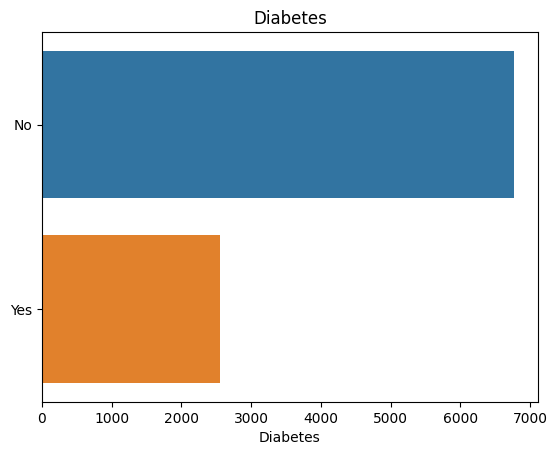


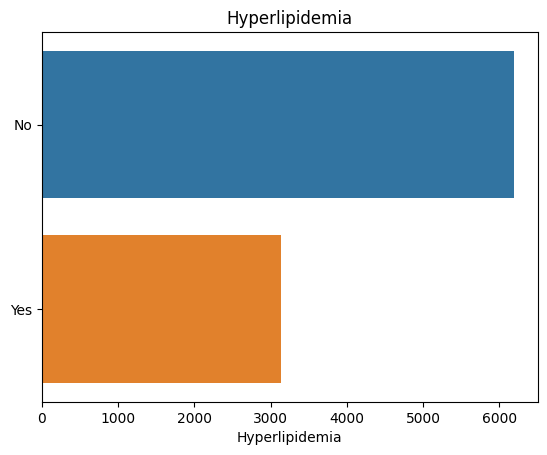


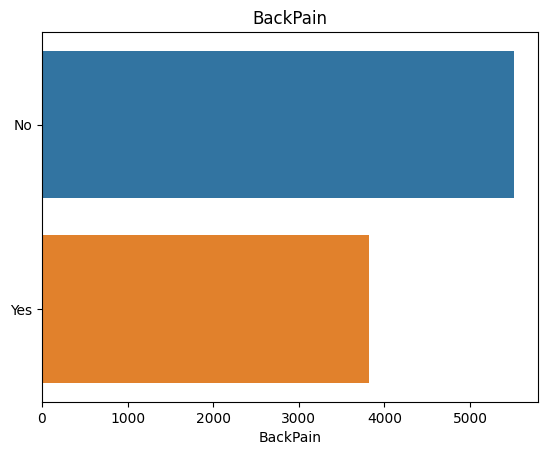


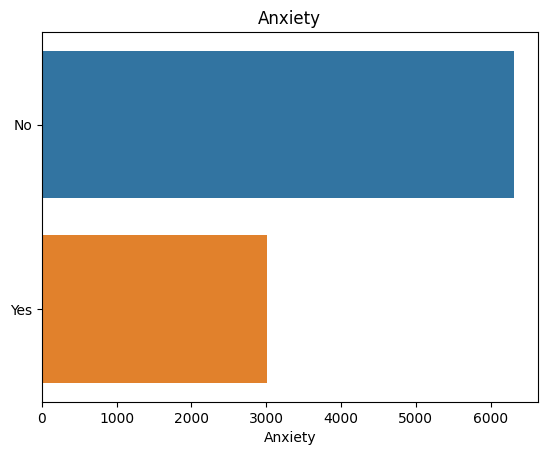


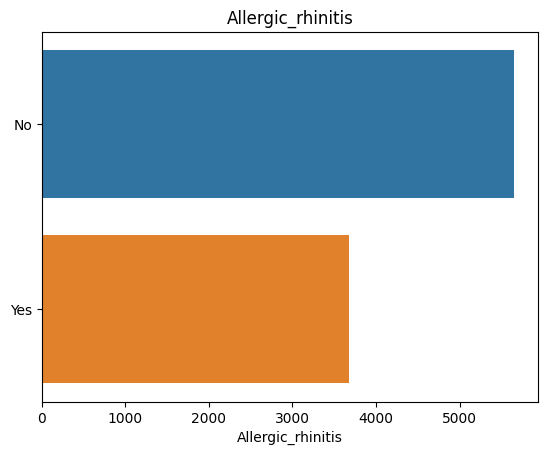


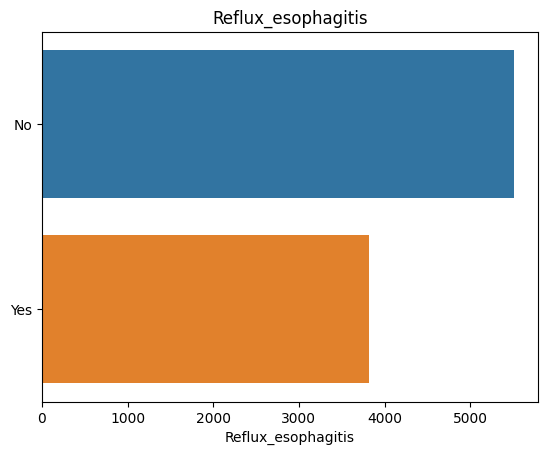


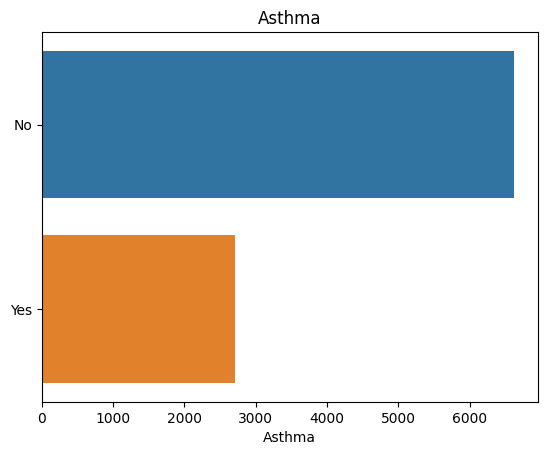


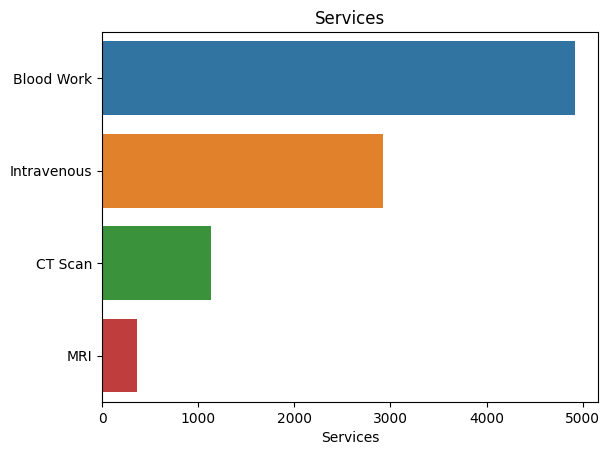












In [17]:

# create all histograms

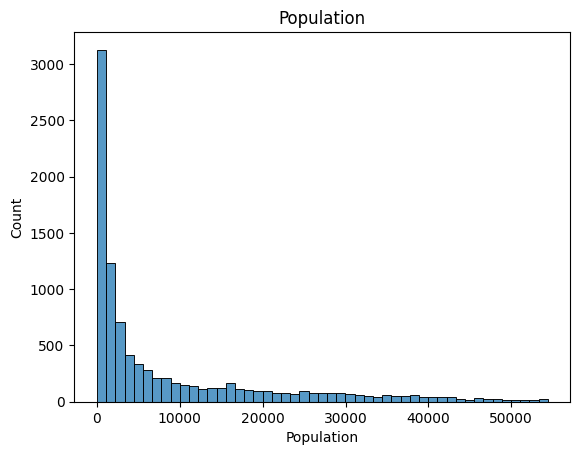
for col in numerical.columns:

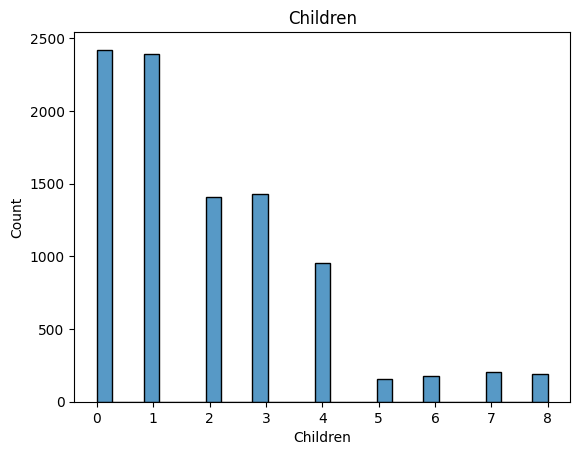
plt.figure()

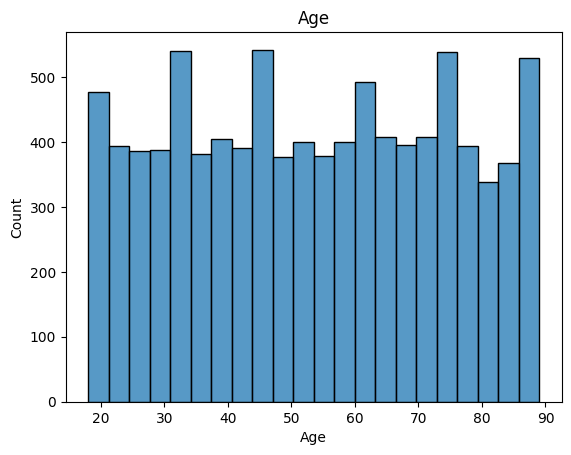
sns.histplot(data=numerical[col])

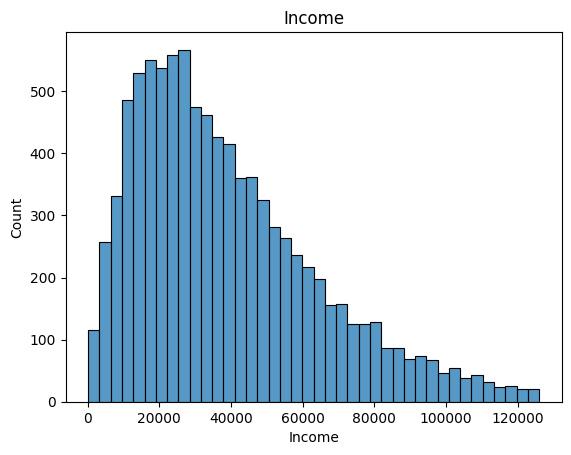
plt.title(col)

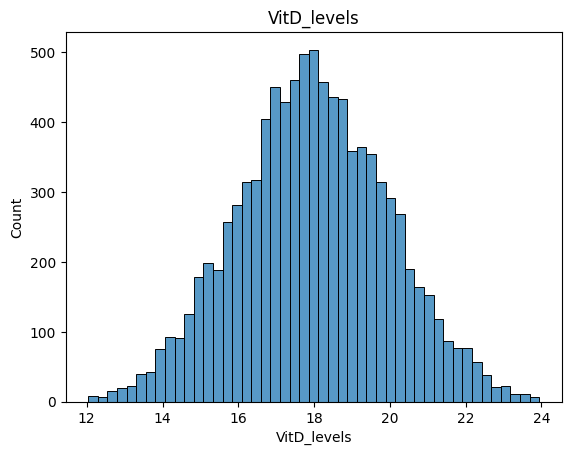
plt.show()

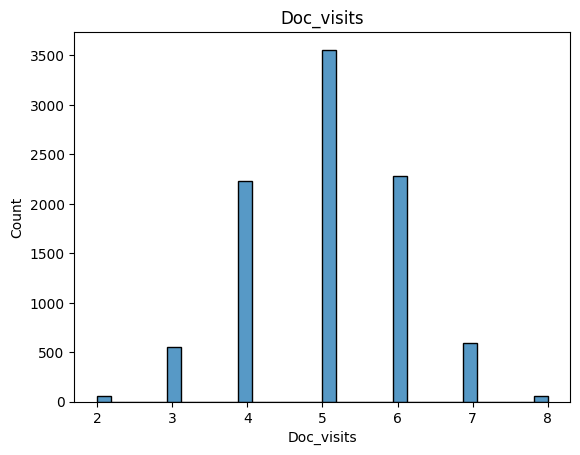


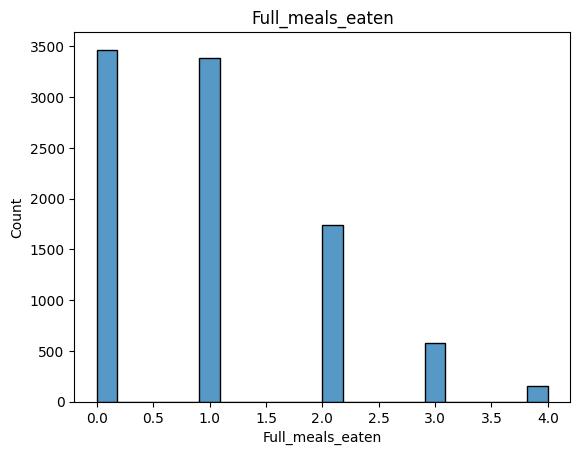












Chart, bar chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

#### Bivariate Plots[¶](#Bivariate-Plots)

In [18]:

# create all catplots

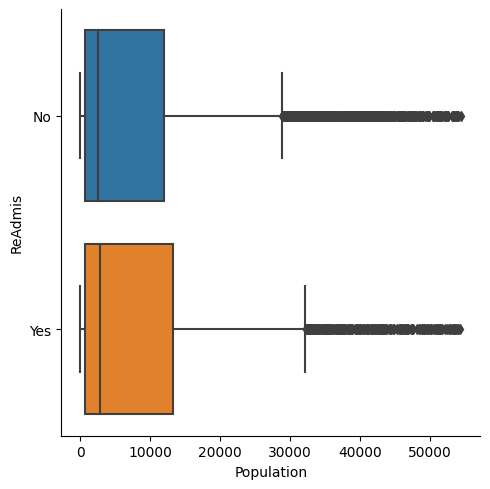
for col in numerical.columns:

plt.figure()

sns.catplot(data=df, x=col, y='ReAdmis', kind='box')

plt.show()

<Figure size 640x480 with 0 Axes>

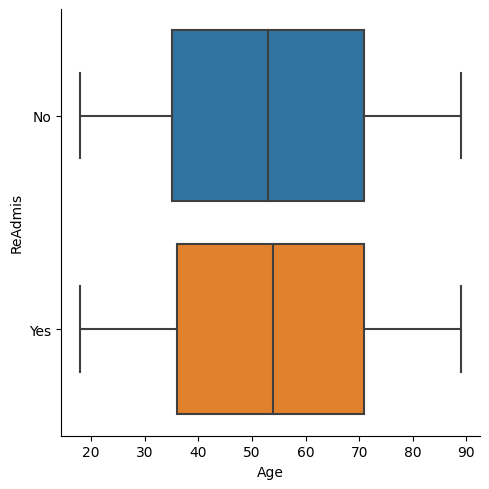


<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>



<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

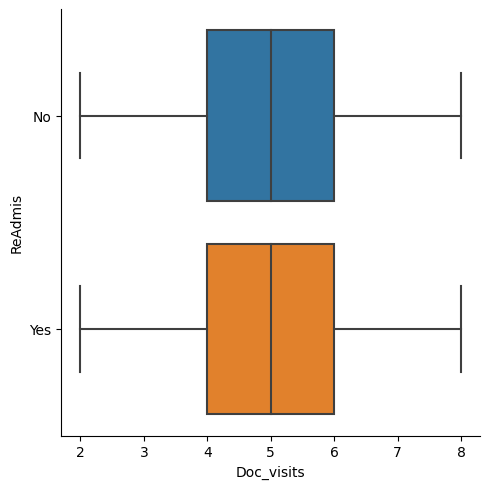
Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>



<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

<Figure size 640x480 with 0 Axes>

Chart, box and whisker chart

Description automatically generated

In [21]:

# drop ReAdmis from categorical DataFrame

categorical.drop(columns='ReAdmis', inplace=True)

In [22]:

# create all multi barcharts

for col in categorical.columns:

data = df[[col, 'ReAdmis']].value\_counts()

plt.figure()

sns.barplot(x=data.index.get\_level\_values(0), y=data.values, hue=data.index.get\_level\_values(1))

plt.show()

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

### Data Transformation[¶](" \l "Data-Transformation)

In [23]:

categorical.nunique()

Out[23]:

Marital 5

Gender 3

Soft\_drink 2

Initial\_admin 3

HighBlood 2

Stroke 2

Complication\_risk 3

Overweight 2

Arthritis 2

Diabetes 2

Hyperlipidemia 2

BackPain 2

Anxiety 2

Allergic\_rhinitis 2

Reflux\_esophagitis 2

Asthma 2

Services 4

dtype: int64

In [24]:

# encode ordinal data

# start with binary values

df.replace(to\_replace='No', value=0, inplace=True)

df.replace(to\_replace='Yes', value=1, inplace=True)

In [25]:

# encode remaining ordinal data

# import scikit-learn to perform label encoding

# Title: scikit-learn/scikit-learn

# Author: Grisel, et al.

# Date: 2022

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.6543413

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder(categories=[['Low', 'Medium', 'High']])

df['Complication\_risk'] = oe.fit\_transform(df[['Complication\_risk']])

In [26]:

# perform dummy variable encoding nominal data

df = pd.get\_dummies(df, columns=['Marital', 'Gender', 'Initial\_admin', 'Services'], drop\_first=True)

df.head(5).transpose()

Out[26]:

|  | **0** | **1** | **2** | **3** | **4** |
| --- | --- | --- | --- | --- | --- |
| **Population** | 2951.000000 | 11303.000000 | 17125.000000 | 2162.000000 | 5287.000000 |
| **Children** | 1.000000 | 3.000000 | 3.000000 | 0.000000 | 1.000000 |
| **Age** | 53.000000 | 51.000000 | 53.000000 | 78.000000 | 22.000000 |
| **Income** | 86575.930000 | 46805.990000 | 14370.140000 | 39741.490000 | 1209.560000 |
| **ReAdmis** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **VitD\_levels** | 19.141466 | 18.940352 | 18.057507 | 16.576858 | 17.439069 |
| **Doc\_visits** | 6.000000 | 4.000000 | 4.000000 | 4.000000 | 5.000000 |
| **Full\_meals\_eaten** | 0.000000 | 2.000000 | 1.000000 | 1.000000 | 0.000000 |
| **vitD\_supp** | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 2.000000 |
| **Soft\_drink** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **HighBlood** | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Stroke** | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Complication\_risk** | 1.000000 | 2.000000 | 1.000000 | 1.000000 | 0.000000 |
| **Overweight** | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Arthritis** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Diabetes** | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| **Hyperlipidemia** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **BackPain** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Anxiety** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Allergic\_rhinitis** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **Reflux\_esophagitis** | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Asthma** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Initial\_days** | 10.585770 | 15.129562 | 4.772177 | 1.714879 | 1.254807 |
| **Marital\_Married** | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Marital\_Never Married** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Marital\_Separated** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Marital\_Widowed** | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 |
| **Gender\_Male** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| **Gender\_Nonbinary** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Initial\_admin\_Emergency Admission** | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Initial\_admin\_Observation Admission** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Services\_CT Scan** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **Services\_Intravenous** | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **Services\_MRI** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |

In [27]:

df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 9331 entries, 0 to 9999

Data columns (total 34 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Population 9331 non-null int64

1 Children 9331 non-null int64

2 Age 9331 non-null int64

3 Income 9331 non-null float64

4 ReAdmis 9331 non-null int64

5 VitD\_levels 9331 non-null float64

6 Doc\_visits 9331 non-null int64

7 Full\_meals\_eaten 9331 non-null int64

8 vitD\_supp 9331 non-null int64

9 Soft\_drink 9331 non-null int64

10 HighBlood 9331 non-null int64

11 Stroke 9331 non-null int64

12 Complication\_risk 9331 non-null float64

13 Overweight 9331 non-null int64

14 Arthritis 9331 non-null int64

15 Diabetes 9331 non-null int64

16 Hyperlipidemia 9331 non-null int64

17 BackPain 9331 non-null int64

18 Anxiety 9331 non-null int64

19 Allergic\_rhinitis 9331 non-null int64

20 Reflux\_esophagitis 9331 non-null int64

21 Asthma 9331 non-null int64

22 Initial\_days 9331 non-null float64

23 Marital\_Married 9331 non-null uint8

24 Marital\_Never Married 9331 non-null uint8

25 Marital\_Separated 9331 non-null uint8

26 Marital\_Widowed 9331 non-null uint8

27 Gender\_Male 9331 non-null uint8

28 Gender\_Nonbinary 9331 non-null uint8

29 Initial\_admin\_Emergency Admission 9331 non-null uint8

30 Initial\_admin\_Observation Admission 9331 non-null uint8

31 Services\_CT Scan 9331 non-null uint8

32 Services\_Intravenous 9331 non-null uint8

33 Services\_MRI 9331 non-null uint8

dtypes: float64(4), int64(19), uint8(11)

memory usage: 1.8 MB

In [28]:

# save prepared data to csv

df.to\_csv('medical\_prepared\_data2.csv', header=True)

In [ ]:

## Part 2: Logistic Regression Modeling Code

### Logistic Regression Model 1: All Features[¶](" \l "Linear-Regression-Model-1:-All-Features)

In [1]:

# read csv into DataFrame

# import pandas for data manipulation

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

import pandas as pd

df = pd.read\_csv('medical\_prepared\_data2.csv', index\_col=0)

In [2]:

# install statsmodels

# !pip install statsmodels

In [3]:

# import statsmodels to perform linear regression

import statsmodels.api as sm

# Title: statsmodels/statsmodels

# Author: Seabold, et al.

# Date: 2017

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.275519

# define the variables

x = df.drop(columns='ReAdmis').assign(const=1)

y = df['ReAdmis']

# perform the regression and fitting the model

model1 = sm.GLM(y, x, family=sm.families.Binomial()).fit()

model1.summary()

Out[3]:

|  |  |  |  |
| --- | --- | --- | --- |
| Generalized Linear Model Regression Results | | | |
| **Dep. Variable:** | ReAdmis | **No. Observations:** | 9331 |
| **Model:** | GLM | **Df Residuals:** | 9297 |
| **Model Family:** | Binomial | **Df Model:** | 33 |
| **Link Function:** | Logit | **Scale:** | 1.0000 |
| **Method:** | IRLS | **Log-Likelihood:** | -308.78 |
| **Date:** | Wed, 29 Mar 2023 | **Deviance:** | 617.56 |
| **Time:** | 14:54:07 | **Pearson chi2:** | 1.32e+03 |
| **No. Iterations:** | 13 | **Pseudo R-squ. (CS):** | 0.7129 |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Population** | 3.309e-06 | 8.96e-06 | 0.369 | 0.712 | -1.42e-05 | 2.09e-05 |
| **Children** | 0.0931 | 0.053 | 1.767 | 0.077 | -0.010 | 0.196 |
| **Age** | 0.0016 | 0.005 | 0.310 | 0.757 | -0.008 | 0.011 |
| **Income** | -7.68e-07 | 3.99e-06 | -0.192 | 0.848 | -8.6e-06 | 7.06e-06 |
| **VitD\_levels** | 0.0328 | 0.052 | 0.633 | 0.527 | -0.069 | 0.134 |
| **Doc\_visits** | -0.0022 | 0.103 | -0.022 | 0.983 | -0.204 | 0.199 |
| **Full\_meals\_eaten** | -0.0173 | 0.108 | -0.160 | 0.873 | -0.230 | 0.195 |
| **vitD\_supp** | 0.0138 | 0.185 | 0.074 | 0.941 | -0.349 | 0.377 |
| **Soft\_drink** | 0.3146 | 0.248 | 1.266 | 0.206 | -0.172 | 0.802 |
| **HighBlood** | 0.9203 | 0.228 | 4.039 | 0.000 | 0.474 | 1.367 |
| **Stroke** | 1.6878 | 0.277 | 6.088 | 0.000 | 1.144 | 2.231 |
| **Complication\_risk** | 0.8709 | 0.152 | 5.717 | 0.000 | 0.572 | 1.169 |
| **Overweight** | -0.3241 | 0.239 | -1.356 | 0.175 | -0.793 | 0.144 |
| **Arthritis** | -1.4939 | 0.237 | -6.290 | 0.000 | -1.959 | -1.028 |
| **Diabetes** | 0.5165 | 0.243 | 2.130 | 0.033 | 0.041 | 0.992 |
| **Hyperlipidemia** | 0.3796 | 0.224 | 1.693 | 0.090 | -0.060 | 0.819 |
| **BackPain** | 0.2032 | 0.214 | 0.948 | 0.343 | -0.217 | 0.623 |
| **Anxiety** | -1.0365 | 0.233 | -4.442 | 0.000 | -1.494 | -0.579 |
| **Allergic\_rhinitis** | -0.4302 | 0.219 | -1.965 | 0.049 | -0.859 | -0.001 |
| **Reflux\_esophagitis** | -0.4245 | 0.220 | -1.931 | 0.054 | -0.856 | 0.006 |
| **Asthma** | -1.3244 | 0.247 | -5.361 | 0.000 | -1.809 | -0.840 |
| **Initial\_days** | 1.4916 | 0.087 | 17.053 | 0.000 | 1.320 | 1.663 |
| **Marital\_Married** | 0.3676 | 0.339 | 1.084 | 0.278 | -0.297 | 1.032 |
| **Marital\_Never Married** | 0.4807 | 0.352 | 1.366 | 0.172 | -0.209 | 1.170 |
| **Marital\_Separated** | 0.0919 | 0.353 | 0.260 | 0.795 | -0.600 | 0.783 |
| **Marital\_Widowed** | 0.2914 | 0.343 | 0.850 | 0.395 | -0.380 | 0.963 |
| **Gender\_Male** | 0.2201 | 0.217 | 1.014 | 0.311 | -0.205 | 0.646 |
| **Gender\_Nonbinary** | 0.1192 | 0.757 | 0.157 | 0.875 | -1.365 | 1.604 |
| **Initial\_admin\_Emergency Admission** | 2.6812 | 0.301 | 8.919 | 0.000 | 2.092 | 3.270 |
| **Initial\_admin\_Observation Admission** | 0.9397 | 0.293 | 3.203 | 0.001 | 0.365 | 1.515 |
| **Services\_CT Scan** | 1.6142 | 0.383 | 4.215 | 0.000 | 0.864 | 2.365 |
| **Services\_Intravenous** | 0.0711 | 0.239 | 0.298 | 0.766 | -0.397 | 0.540 |
| **Services\_MRI** | 2.9961 | 0.525 | 5.706 | 0.000 | 1.967 | 4.025 |
| **const** | -84.4050 | 5.012 | -16.839 | 0.000 | -94.229 | -74.581 |

#### Evaluation

##### Accuracy[¶](#Accuracy)

In [4]:

# import sklearn for metrics

from sklearn import metrics

# generate predictions; round to 0 or 1

y\_pred = round(model1.predict(x))

#calculate accuracy

metrics.accuracy\_score(y, y\_pred)

Out[4]:

0.9861751152073732

##### Confusion Matrix[¶](#Confusion-Matrix)

In [5]:

# import sklearn for metrics

from sklearn import metrics

# generate confusion matrix

metrics.confusion\_matrix(y, y\_pred)

Out[5]:

array([[5843, 67],

[ 62, 3359]], dtype=int64)

##### Multicollinearity[¶](#Multicollinearity)

In [6]:

# calculate VIF to detect multicollinearity

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["feature"] = x.columns

# calculate VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x.values, i)

for i in range(len(x.columns))]

vif\_data

Out[6]:

|  | **feature** | **VIF** |
| --- | --- | --- |
| **0** | Population | 1.002998 |
| **1** | Children | 1.003669 |
| **2** | Age | 1.004163 |
| **3** | Income | 1.003025 |
| **4** | VitD\_levels | 1.004470 |
| **5** | Doc\_visits | 1.003784 |
| **6** | Full\_meals\_eaten | 1.005529 |
| **7** | vitD\_supp | 1.004694 |
| **8** | Soft\_drink | 1.004004 |
| **9** | HighBlood | 1.003840 |
| **10** | Stroke | 1.002053 |
| **11** | Complication\_risk | 1.003421 |
| **12** | Overweight | 1.003341 |
| **13** | Arthritis | 1.004004 |
| **14** | Diabetes | 1.003938 |
| **15** | Hyperlipidemia | 1.003997 |
| **16** | BackPain | 1.006736 |
| **17** | Anxiety | 1.002081 |
| **18** | Allergic\_rhinitis | 1.002981 |
| **19** | Reflux\_esophagitis | 1.003091 |
| **20** | Asthma | 1.002853 |
| **21** | Initial\_days | 1.004603 |
| **22** | Marital\_Married | 1.636364 |
| **23** | Marital\_Never Married | 1.624777 |
| **24** | Marital\_Separated | 1.624693 |
| **25** | Marital\_Widowed | 1.641306 |
| **26** | Gender\_Male | 1.023522 |
| **27** | Gender\_Nonbinary | 1.023451 |
| **28** | Initial\_admin\_Emergency Admission | 1.500451 |
| **29** | Initial\_admin\_Observation Admission | 1.501206 |
| **30** | Services\_CT Scan | 1.084174 |
| **31** | Services\_Intravenous | 1.098667 |
| **32** | Services\_MRI | 1.034698 |
| **33** | const | 137.783788 |

### Logistic Regression Model 2: Reduced Features[¶](" \l "Linear-Regression-Model-2:-Reduced-Feat)

In [7]:

# import packages from scikit-learn for feature reduction

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_selection import SequentialFeatureSelector

# Title: scikit-learn/scikit-learn

# Author: Grisel, et al.

# Date: 2022

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.6543413

# re-define independent variables

x = df.drop(columns='ReAdmis')

# logistic regression estimator to be used in feature selection

reg = LogisticRegression(max\_iter=1000).fit(x, y)

In [8]:

# reduce features using SequentialFeatureSelector

sfs = SequentialFeatureSelector(reg, n\_features\_to\_select=4)

sfs.fit(x, y)

Out[8]:

SequentialFeatureSelector(estimator=LogisticRegression(max\_iter=1000),

n\_features\_to\_select=4)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.   
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

SequentialFeatureSelector

SequentialFeatureSelector(estimator=LogisticRegression(max\_iter=1000),

n\_features\_to\_select=4)

estimator: LogisticRegression

LogisticRegression(max\_iter=1000)

LogisticRegression

LogisticRegression(max\_iter=1000)

In [9]:

# return selected features

sfs.get\_feature\_names\_out()

Out[9]:

array(['Stroke', 'Complication\_risk', 'Initial\_days',

'Initial\_admin\_Emergency Admission'], dtype=object)

In [10]:

# reduce x to selected features

x = df[['Stroke', 'Complication\_risk', 'Initial\_days',

'Initial\_admin\_Emergency Admission']].assign(const=1)

In [11]:

# re-perform linear regression

# perform the regression and fitting the model

model2 = sm.GLM(y, x, family=sm.families.Binomial()).fit()

model2.summary()

Out[11]:

|  |  |  |  |
| --- | --- | --- | --- |
| Generalized Linear Model Regression Results | | | |
| **Dep. Variable:** | ReAdmis | **No. Observations:** | 9331 |
| **Model:** | GLM | **Df Residuals:** | 9326 |
| **Model Family:** | Binomial | **Df Model:** | 4 |
| **Link Function:** | Logit | **Scale:** | 1.0000 |
| **Method:** | IRLS | **Log-Likelihood:** | -396.11 |
| **Date:** | Wed, 29 Mar 2023 | **Deviance:** | 792.22 |
| **Time:** | 14:54:38 | **Pearson chi2:** | 2.01e+03 |
| **No. Iterations:** | 12 | **Pseudo R-squ. (CS):** | 0.7075 |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Stroke** | 1.3984 | 0.240 | 5.832 | 0.000 | 0.928 | 1.868 |
| **Complication\_risk** | 0.6286 | 0.127 | 4.941 | 0.000 | 0.379 | 0.878 |
| **Initial\_days** | 1.1260 | 0.057 | 19.813 | 0.000 | 1.015 | 1.237 |
| **Initial\_admin\_Emergency Admission** | 1.6313 | 0.198 | 8.244 | 0.000 | 1.243 | 2.019 |
| **const** | -62.9604 | 3.186 | -19.764 | 0.000 | -69.204 | -56.717 |

#### Evaluation

##### Accuracy[¶](#Accuracy)

In [12]:

# re-evaluate model

# generate predictions; round to 0 or 1

y\_pred = round(model2.predict(x))

#calculate accuracy

metrics.accuracy\_score(y, y\_pred)

Out[12]:

0.9814596506269424

##### Confusion Matrix[¶](#Confusion-Matrix)

In [13]:

# re-generate confusion matrix

metrics.confusion\_matrix(y, y\_pred)

Out[13]:

array([[5824, 86],

[ 87, 3334]], dtype=int64)

##### Multicollinearity[¶](#Multicollinearity)

In [14]:

# re-create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["feature"] = x.columns

# re-calculate VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x.values, i)

for i in range(len(x.columns))]

vif\_data

Out[14]:

|  | **feature** | **VIF** |
| --- | --- | --- |
| **0** | Stroke | 1.000261 |
| **1** | Complication\_risk | 1.000282 |
| **2** | Initial\_days | 1.000373 |
| **3** | Initial\_admin\_Emergency Admission | 1.000290 |
| **4** | const | 6.454001 |

In [15]:

# import matplotlib to create visualizations

# Title: matplotlib/matplotlib

# Author: Caswell, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7697899

# Title: seaborn: statistical data visualization

# Author: Michael Waskom

# Date: 2021

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.4645478

import matplotlib.pyplot as plt

import seaborn as sns

# TODO

In [ ]:

# Section H

To view a walkthrough demonstration of the code referenced in Section G, refer to the following Panopto link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ebb66c88-8320-4dab-90c4-afd50167e4df>

# Section I

## Part 1: Web Sources

Caswell, et al. (2023, March). doi:10.5281/zenodo.7697899

Gommers, et al. (2023). doi:10.5281/zenodo.7655153

Grisel, et al. (2022). doi:10.5281/zenodo.6543413

Seabold, et al. (2017). doi:10.5281/zenodo.275519

The pandas development team. (2023). doi:10.5281/zenodo.7741580

Waskom, M. (2021). doi:10.5281/zenodo.4645478

## Part 2: References

Bobbitt, Z. (2020, October). *The 6 Assumptions of Logistic Regression (with Examples)*. Retrieved March 2023, from Statology: https://www.statology.org/assumptions-of-logistic-regression/

Bruce, P., Bruce, A., & Gedeck, P. (2019). *Practical Statistics for Data Scientists : 50+ Essential Concepts Using R and Python* (2 ed.). O'Reilly Media, Incorporated. Retrieved March 2023

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection : A Practical Approach for Predictive Models.* CRC Press LLC. Retrieved March 2023

Larose, D. T., & Larose, C. D. (2019). *Data Science Using Python and R.* Hoboken: Wiley. Retrieved March 2023

Pedregosa, e. (2011). *scikit-learn*. Retrieved March 2023, from sklearn.feature\_selection.SequentialFeatureSelector: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SequentialFeatureSelector.html