Predicting Hospital Readmission Through Logistic Regression

Brandon Scholer

Department of Information Technology, Western Governors University

D208: Predictive Modeling

Dr. Straw

October 27, 2021

Given a patient's medical conditions, medical information, and demographic information is it possible to predict whether that patient will be re-admitted to the hospital within one month? The goal of this analysis is to see if this data holds any statistically significant information that can help as predictors, or if there is any relation between these features and the outcome of the model's predictability.

Logistic regression is the obvious choice to analyze this data, as the variable that is being predicted, ReAdmis, is a binary variable that is in the form of 'yes' or 'no'. Because the observations need to be independent of each other, any overlapping variables should be eliminated. Multicollinearity between the independent variables should be minimal, so highly correlated observations should also be eliminated. With 10,000 observations in the data set, the sample size is sufficiently large for this type of regression. Python was the best choice for this analysis because "comparing with other languages like R, Go, and Rust, Python is much faster and more scalable" while also having a huge collection of free libraries to accomplish tasks from modeling and visualizing the data to looking at the variable correlation (Zhidkov, 2020).

To prepare the data, three important scenarios needed to be addressed. They are as follows, any variables containing information that didn't add anything to the model needed to be removed; variables that didn't have intuitive names needed to be renamed; and variables that had data types that weren't conducive to the analysis needed to be converted. CaseOrder, Customer_id, Interaction, and UID were all identification fields; not data that contributed to the analysis and were therefore dropped. City, County, Zip, State, TimeZone, Lat, Lng, and Job were all were redundant variables or contained many levels which would have needed a further analysis for clustering which was outside the scope of the current analysis, which resulted in them also being dropped. Several variables were named 'Item' followed by the numbers one

through seven. These were renamed to be more descriptive of the data that each item contained. Once this was done, each variable that contained a string of ‘yes’ or ‘no’ was converted to Boolean for easier analysis. Lastly, each categorical variable was converted to data type ‘category’ for simpler processing. The Python code used to complete these steps can be seen in Figure 1.

In Figure 2 the summary statistics for all variables, including the target variable, are shown. The categorical variables show a range of unique categories varying from only three categories in Area, Gender, Initial_admin, and Complication_risk; to five in Marital, with Services having four. Interestingly Initial_admin, Gender, and Services all have a single category that holds half of the observations. For discrete variables, Population holds the most drastic range, with a mean of 10,000; but a min of 0 and a max of just over 122,000. Ages ranged from 18 to 89, with a median age of 53. Of the continuous variables, Income had the largest range with a minimum of $154 and a maximum of $207,249 and a mean of $40,490.50. The mean for TotalCharge was $5312.17 (which is the daily charge to the patient). While the mean for Additional_charges was $12,934.53. The binary values held more ‘noes’ than ‘yeses’ in every variable by roughly 10% or more except in the case of Overweight, in which the opposite was true.

The univariate visualization utilized for the binary explanatory variables was a bar chart, as seen in Figure 3. The binary visualizations are as expected given the summary statistics of those variables; showing that the false values outnumber the true values, sometimes by large margins, in every variable except the Overweight variable in which true outpaces false to a similar extent. Figure 4 shows the relationship between the different categories of the categorical variables. Area and Marriage status variables are nearly evenly split across each of their

categories. Male and Female are nearly even as well, with Nonbinary being considerably less than either of the others within the Gender variable. Emergency Admission makes up roughly half of the total of the Initial Admission category, with the rest evenly split between Elective Admission and Observation Admission. Both Complication Risk and Services gradually step down in count per category (Medium, High, Low in Complication Risk and Blood Work, Intravenous, CT Scan, MRI in Services).

Box plots for the discrete predictor variables, as seen in Figure 5, mostly look as expected given the summary statistics of those variables, except for Population, Children, Full_meals_eaten, and vitD_supp. All four of these variables have interquartile ranges that sit at the bottom of the chart, none of which extend into the center of the chart. Population's entire interquartile range falls under 20,000 while sporadic values increase up to 120,000. Figure 6 shows the histograms for the continuous predictor variables in the cleaned data set. Both the Income and Additional Charges charts appear to have a right skew to them. The Vitamin D Levels chart has a normal distribution, while the Total Charge and Initial Days charts have an inverted bell curve distribution.

The bivariate visualizations for the predictor variables Population, Area, Children, Age, Income, Marital, Gender, and vitD_levels are found in Figure 7. In each case, ReAdmis false leads true. Population starts high and falls off quickly. Area, Marital, and Gender all see a pretty even split among the categories, with ReAdmis following the same pattern between false and true. Age is fairly dispursed quite sporadically, with ReAdmis following a similar trend. Income has a left skew, with no other noticeable pattern in ReAdmis. With regards to VitD_levels, there is a standard distribution among both false and true ReAdmis, with true’s peak being lower than falses’. Figure 8 shows the bivariate visualization for Doc_visits, Full_meals_eaten, vitD_supp,

Soft_drink, Inititial_admin, HighBlood, Stroke, Complication_risk, and Overweight. The bar charts show that the distribution is fairly similar to the univariate analysis on these same variables and that ReAdmis follows the same trend that it did for the last set of bivariate visuals in Figure 7. The only difference here is the same difference from before where Overweight's true values overtake its false values, but ReAdmis still follows the same pattern as the other variables. Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services in Figure 9 all follow the same patterns seen previously. While Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment in Figure 10 all exhibit the same patterns from earlier visualizations; Initial_days and TotalCharge show a direct separation in ReAdmis from the false on the low end to true on the high end in the range of both variables. This shows that there may be quasi-complete separation within the problem. Figure 11 shows the bivariate analysis of the last two variables, Courteous and Active_listening. These two variables follow the same trends seen in the rest of the variables in the data set. Along with quasi-complete separation, it is also important to check for multicollinearity. Using variance inflation factor (VIF) to check for multicollinearity, as seen in Figure 12, four variables have a VIF that is greater than ten. Ten was chosen as the cutoff limit for VIF, so these four variables are removed from the data set.

This left thirty-eight variables remaining to be utilized for the initial logistic regression model. Before being applied to the model, dummy variables were created for the categorical variables left in the data set to alleviate any issues that they could have caused in the model. The summary of the initial model can be seen in Figure 13, which shows that several of the variables have high p-values, indicating that they are not statistically significant. Figure 14 shows the accuracy of the model, sitting at 63.83%, as well as the confusion matrix, presented a heat map.

The heat map shows that the model has a high negative predictive value, but a low positive predictive value. To improve on this model, a backward elimination technique was used with a focus on Akaike Information Criterion (AIC), as seen in Figure 15. The code starts with a model containing all of the features included in the initial model. The feature with the highest p-value is then selected and removed from a new model. This new model's AIC is then compared with the previous model's AIC. If the new model had a lower AIC the new model without the eliminated feature moved forward, if the previous model had the lower AIC then the feature that was removed is added back into the model and marked so that it won't be selected again based on its p-value. This process is then repeated until the lowest AIC has been reached. A new model is then produced using only the remaining nine independent variables. As seen by the summary statistics of the model in Figure 16, half of these variables are statistically significant at a significance level of .05, while they all contribute to a lower AIC than the original model. The confusion matrix, however, as seen by the heat map in Figure 17 only increases the accuracy of the model to 64.8%; only a one percent increase.

The logistic regression equation created by the reduced model based on the coefficients seen in Figure 17 is:

$$\begin{aligned}\text{logit}(p) = {} & .000001(Population) + .023227(Children) - .000002(Income) \\ & - .013058(VitD\ levels) - .125818(Asthma) - .037520(Timely\ admission) \\ & - .038623(Active\ listening) \\ & + .023996(Initial\ admin\ Emergency\ Admission) \\ & + .166215(Services\ CT\ Scan) - .047152(Services\ Intravenous)\end{aligned}$$

Based on this equation, assuming that all variables other than the variable currently being assessed remain fixed if Population increases by one the odds of being readmitted to the hospital

within thirty days increases by one-millionth of a percent. If Children increases by one, the odds increase by 2.3%; Income decreases odds by two-millionth of a percent; VitD_levels decreases odds by 1.3%; Asthma decreases odds by 13.4%; Timely_admission decreases odds by 3.8%; Active_listening decreases odds by 3.9%; Initial_admin_Emergency_Admission increases odds by 2.4%; Services_CT_Scan increases odds by 18%; and Services_Intravenous decreases odds by 4.7% (Jankovic, 2021).

With a pseudo-R-squared of 0.002370, the fit of the model is not statistically significant. The variables are not a good predictor of the response variable. Given the number of false positives and the low accuracy of the predictions, the practicality of the model is also low. The model does not improve the ability to predict the outcome of the response variable any better than just making a guess. The recommended course of action to potentially improve the model would be to gather more data in other areas or to try clustering the variables that led to quasi-complete separation to get away from the multicollinearity issue.

References

Jankovic, D. (2021, September 15). *A Simple Interpretation of Logistic Regression Coefficients.* Retrieved from towards data science: https://towardsdatascience.com/a-simple-interpretation-of-logistic-regression-coefficients-e3a40a62e8cf

Zhidkov, R. (2020, January 13). *Why Python is Essential for Data Analysis.* Retrieved from RTInisghts: https://www.rtinsights.com/why-python-is-essential-for-data-analysis/

Figure 1

Data Preparation Phase

```
# Read in the .csv file that contains the data
med_rec = pd.read_csv("C:/Users/clown/OneDrive/Documents/MS Data Analytics/Predictive Modeling/Data/medical_clean.csv")
```

```
# Check for missing values
med_rec.isnull().sum().sum()
```

```
0
```

```
# Drop columns that contain individual indentification that isn't necessary to the model
# and rename vague column names to be more descriptive.
med_rec = med_rec.drop(['CaseOrder','Customer_id','Interaction','UID', 'City','County','Zip','State', 'TimeZone','Lat','Lng','Job'], axis=1)
med_rec = med_rec.rename({'Item1':'Timely_admission','Item2':'Timely_treatment','Item3':'Timely_visits','Item4':'Reliability',
              'Item5':'Options','Item6':'Hours_treatment','Item7':'Courteous','Item8':'Active_listening'}, axis='columns')
```

```
# Convert No/Yes to 0/1 in binary variables
boo = ['ReAdmis','Soft_drink','HighBlood','Stroke','Overweight','Arthritis','Diabetes','Hyperlipidemia',
       'BackPain','Anxiety','Allergic_rhinitis','Reflux_esophagitis','Asthma']
med_rec[boo] = med_rec[boo].replace({'No':False, 'Yes':True})
```

```
# Convert data types to category as appropriate to make analysis easier
cat = ['Area','Marital','Gender','Initial_admin','Complication_risk','Services']
med_rec[cat] = med_rec[cat].astype('category')
```

Note. This figure shows the Python Code in JupyterLab that imports the original CSV file, checks for missing values, then removes unnecessary data, and organizes the data so that it is simpler to work with before the prepared data is out into another CSV file.

Figure 2

Summary Statistics for Predictors by Type

In [8]:
```
# Summary statistics for catigorical variables
med_rec.describe(include=['category'])
```

Out[8]:

	Area	Marital	Gender	Initial_admin	Complication_risk	Services
count	10000	10000	10000	10000	10000	10000
unique	3	5	3	3	3	4
top	Rural	Widowed	Female	Emergency Admission	Medium	Blood Work
freq	3369	2045	5018	5060	4517	5265

In [21]:
```
# Summary statistics for discrete and
# ordinal variables of the data type int64
med_rec.describe(include=['int64'])
```

Out[21]:

	Population	Children	Age	Doc_visits	Full_meals_eaten	vitD_supp	Timely_admission	Timely_treatment	Timely_visits	Reliability	Options	Hours_treatment	Courteous	Active_listening
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	9965.253800	2.097200	53.511700	5.012200	1.001400	0.398900	3.518800	3.506700	3.511100	3.515100	3.496900	3.522500	3.494000	3.509700
std	14824.758614	2.163659	20.638538	1.045734	1.008117	0.628505	1.031966	1.034825	1.032755	1.036282	1.030192	1.032376	1.021405	1.042312
min	0.000000	0.000000	18.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	694.750000	0.000000	36.000000	4.000000	0.000000	0.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
50%	2769.000000	1.000000	53.000000	5.000000	1.000000	0.000000	4.000000	3.000000	4.000000	4.000000	3.000000	4.000000	3.000000	3.000000
75%	13945.000000	3.000000	71.000000	6.000000	2.000000	1.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000
max	122814.000000	10.000000	89.000000	9.000000	7.000000	5.000000	8.000000	7.000000	8.000000	7.000000	7.000000	7.000000	7.000000	7.000000

In [10]:
```
# Summary statistics for the continous variables
med_rec.describe(include=['float64'])
```

Out[10]:

	Income	VitD_levels	Initial_days	TotalCharge	Additional_charges
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	40490.495160	17.964262	34.455299	5312.172769	12934.528587
std	28521.153293	2.017231	26.309341	2180.393838	6542.601544
min	154.080000	9.806483	1.001981	1938.312067	3125.703000
25%	19598.775000	16.626439	7.896215	3179.374015	7986.487755
50%	33768.420000	17.951122	35.836244	5213.952000	11573.977735
75%	54296.402500	19.347963	61.161020	7459.699750	15626.490000
max	207249.100000	26.394449	71.981490	9180.728000	30566.070000

In [11]:
```
# Summary statistics for binary variables
# to include the response variable
med_rec.describe(include=['bool'])
```

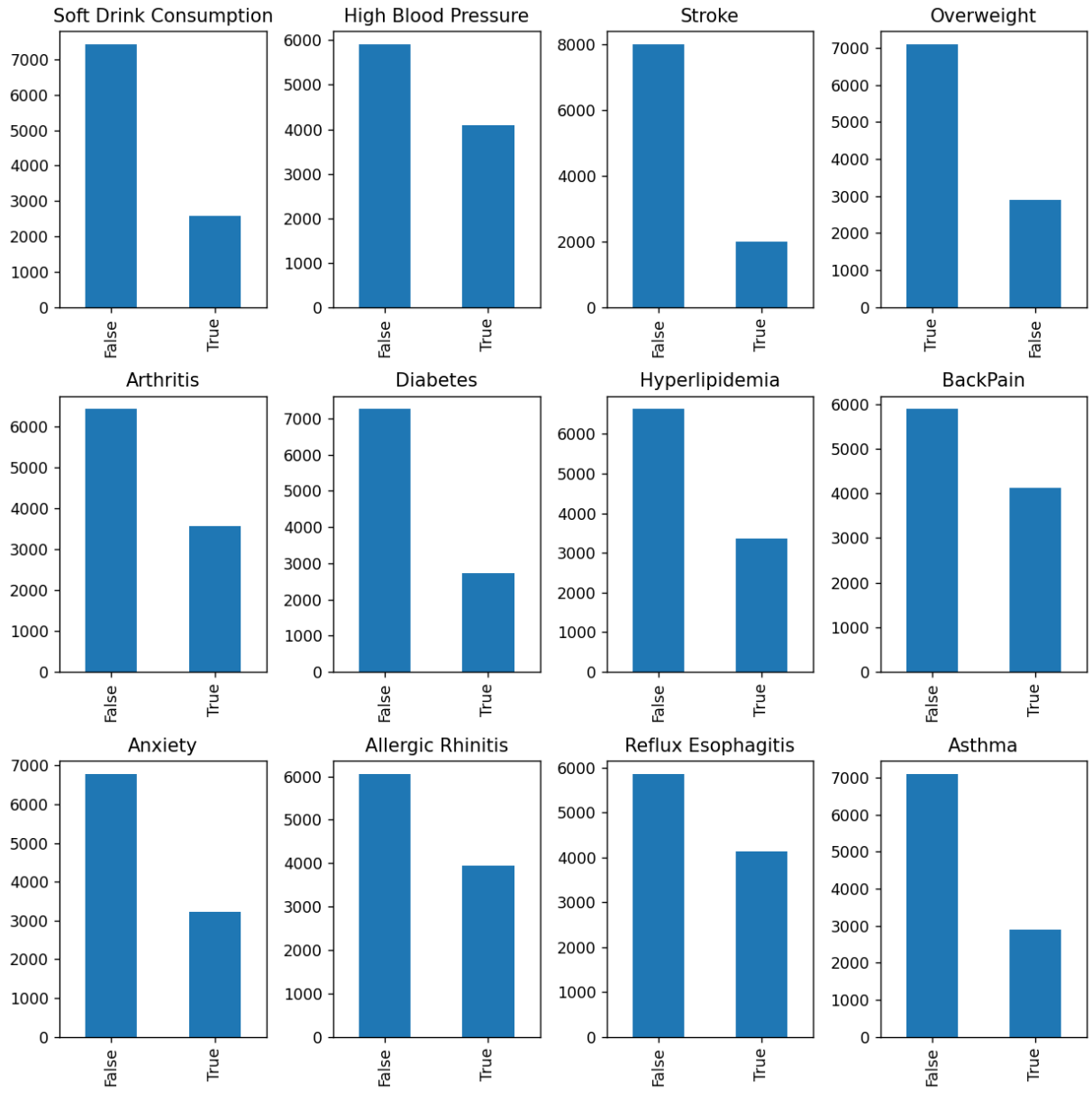
Out[11]:

	ReAdmis	Soft_drink	HighBlood	Stroke	Overweight	Arthritis	Diabetes	Hyperlipidemia	BackPain	Anxiety	Allergic_rhinitis	Reflux_esophagitis	Asthma
count	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
unique	2	2	2	2	2	2	2	2	2	2	2	2	2
top	False	False	False	False	True	False	False	False	False	False	False	False	False
freq	6331	7425	5910	8007	7094	6426	7262	6628	5886	6785	6059	5865	7107

Note. This visual includes the summary statistics for all the variables contained in the data set (to include the dependent variable ReAdmis), sorted by variable type as follows: categorical, discrete, continuous, and binary.

Figure 3

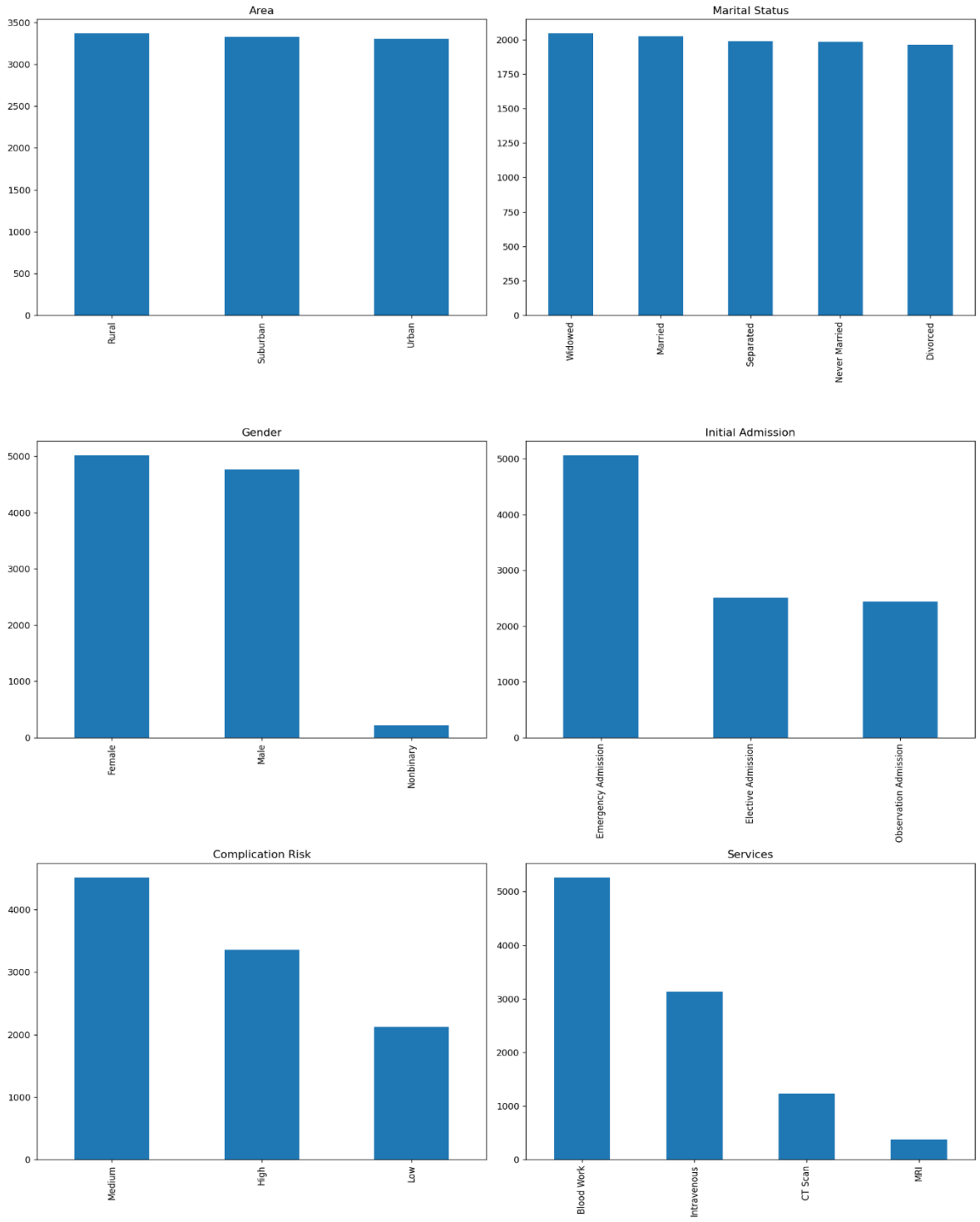
Univariate Visualizations of Binary Predictors



Note. This visualization shows the univariate bar charts of the binary predictor variables.

Figure 4

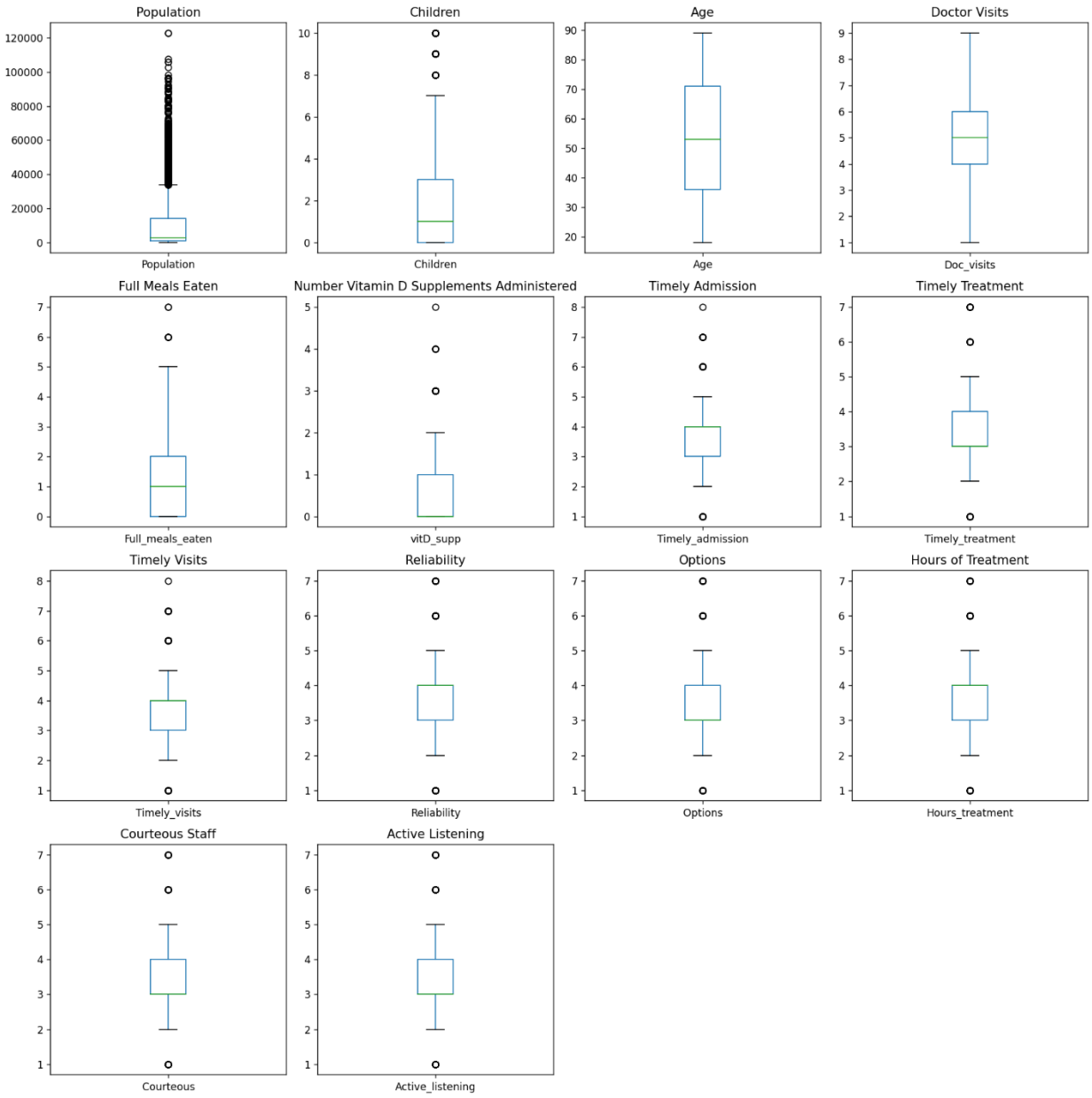
Univariate Visualizations of Categorical Predictors



Note. This visual is of the univariate bar charts for the categorical predictor variables.

Figure 5

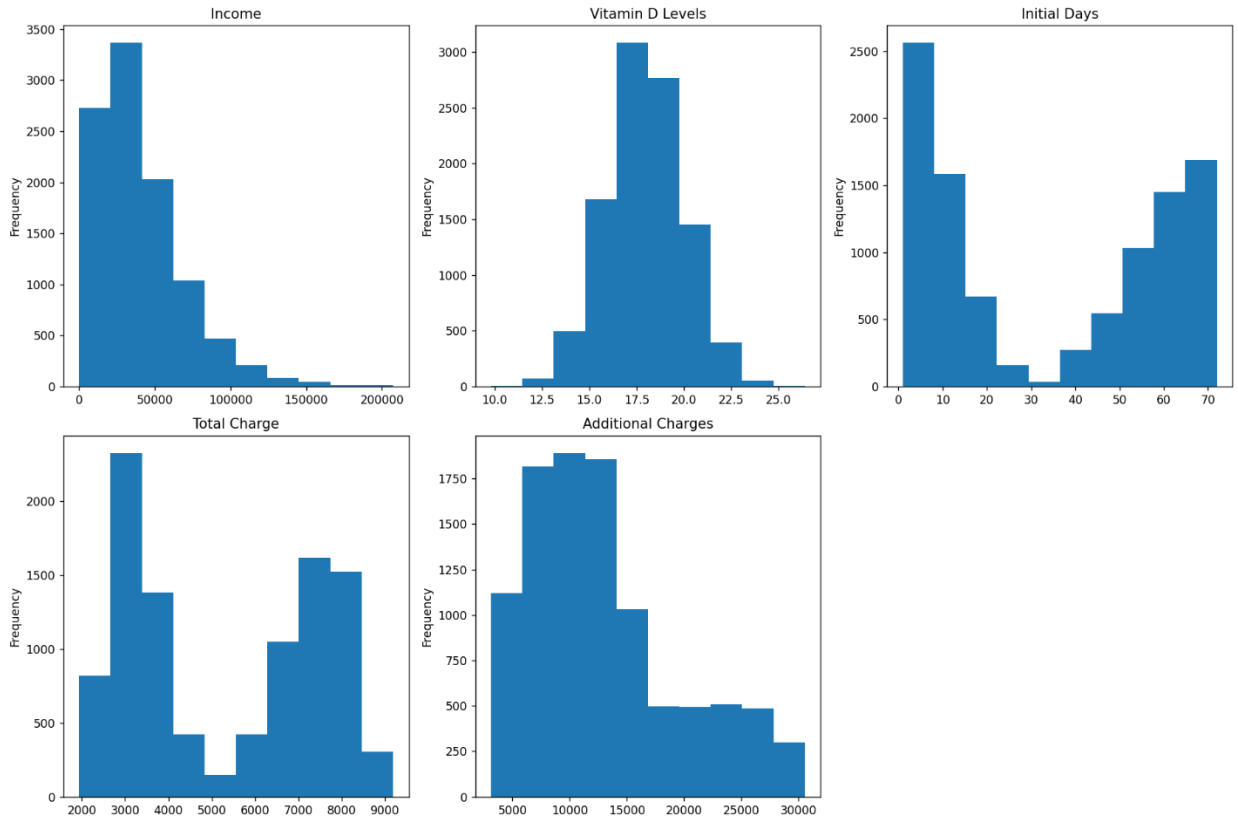
Univariate Visualizations of Discrete Predictors



Note. The univariate visualization of the discrete predictor variables in the form of box plots by variable.

Figure 6

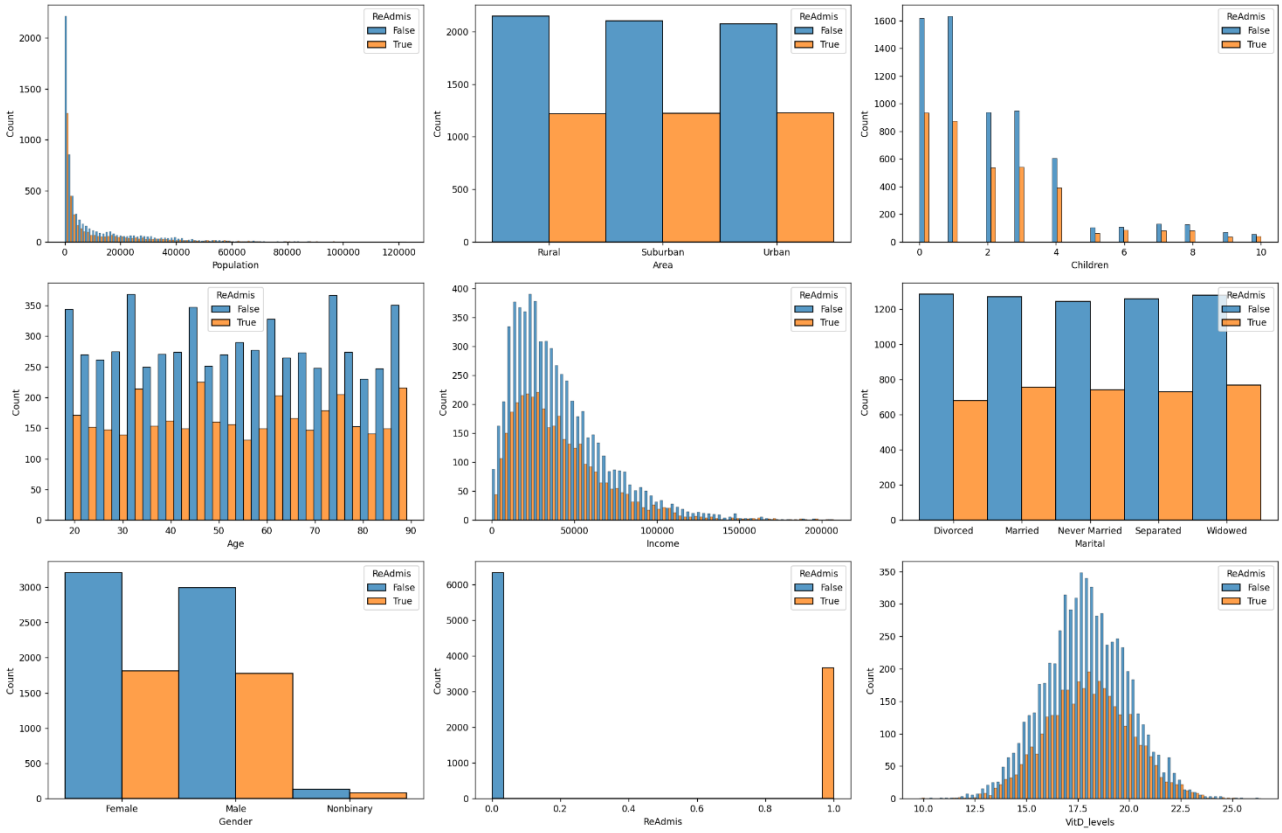
Univariate Visualizations of Continuous Predictor Variables



Note. A visualization containing histograms for each of the continuous predictor variables in the cleaned data set.

Figure 7

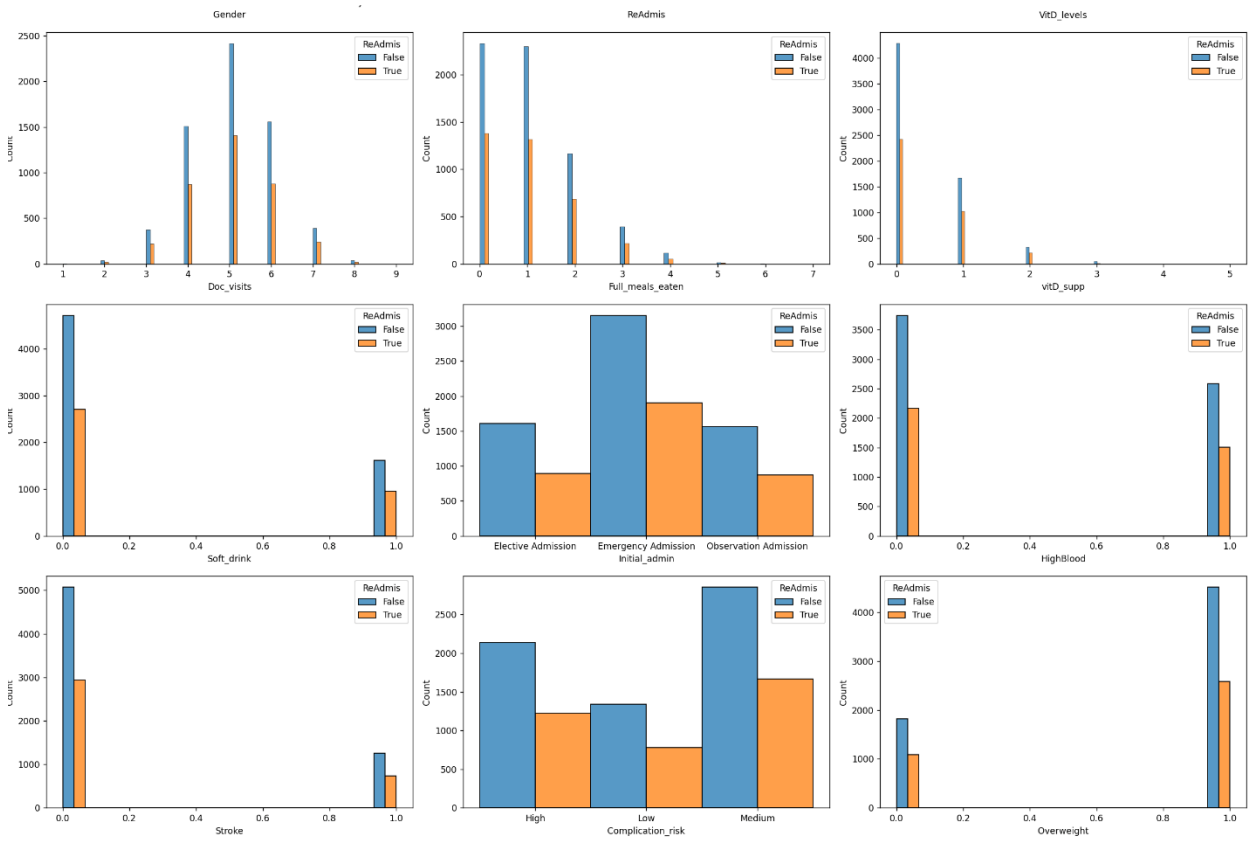
Bivariate Visualization of Population, Area, Children, Age, Income, Marital, Gender, ReAdmis, and vitD_levels



Note. Bar charts of the predictor variables Population, Area, Children, Age, Income, Marital, Gender, and vitD_levels split on the response variable ReAdmis.

Figure 8

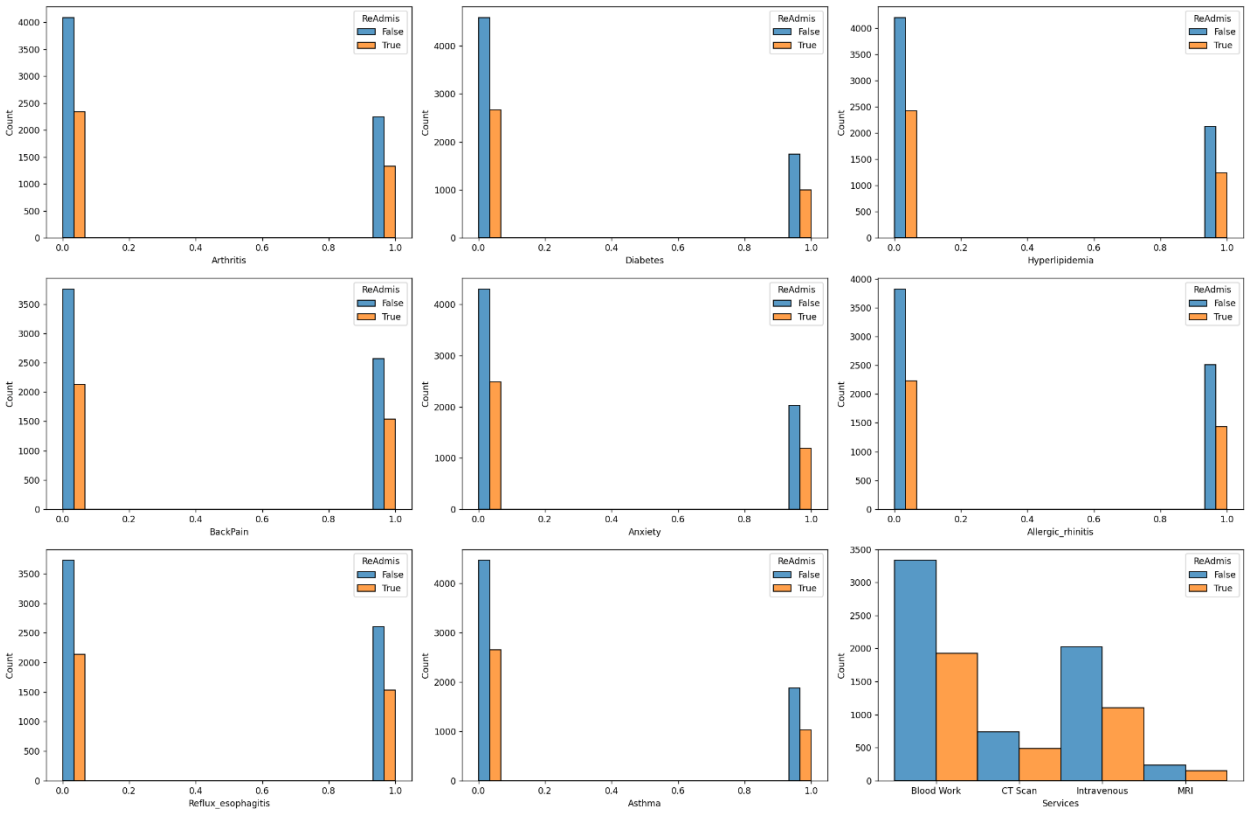
Bivariate Visualization of Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, and Overweight



Note. Bar charts of the predictor variables Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, and Complication_risk, and Overweight split on the response variable ReAdmis.

Figure 9

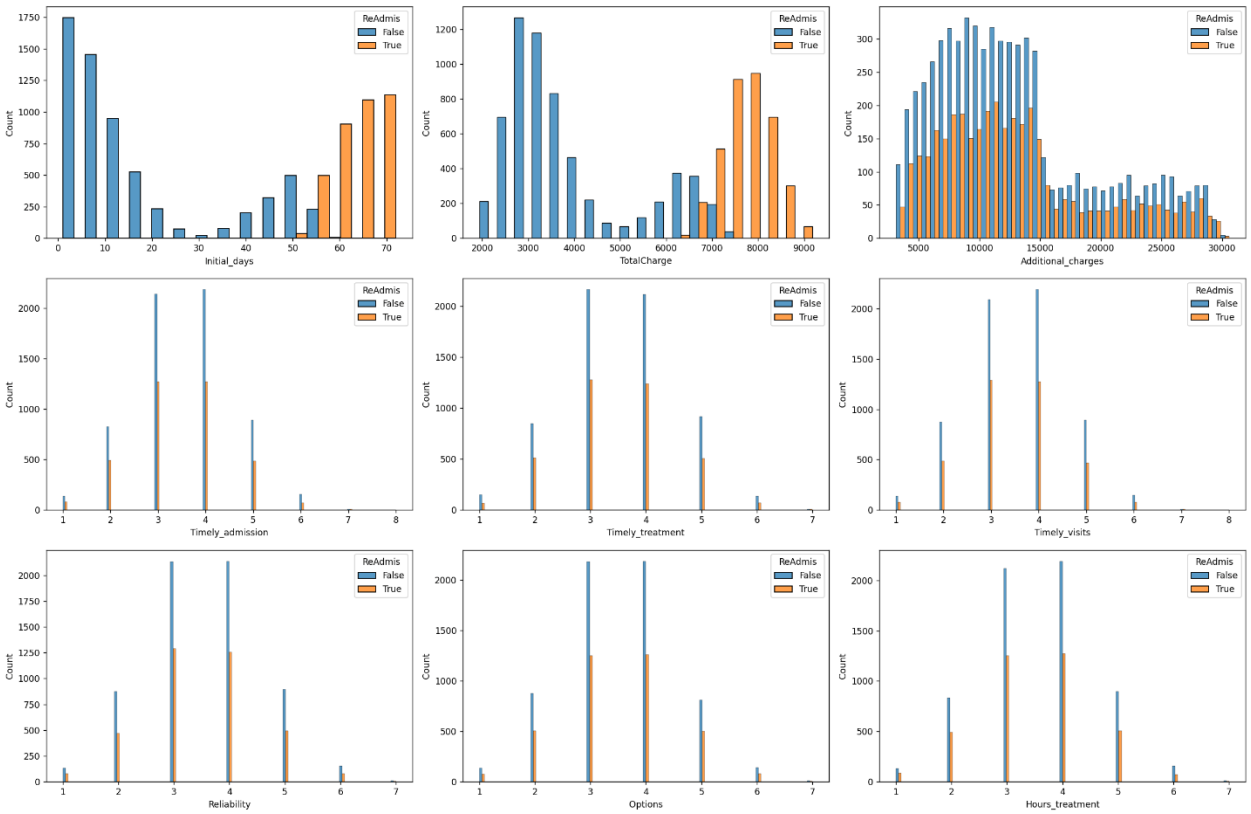
Bivariate Visualization of Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services



Note. Bar charts of the predictor variables Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services split on the response variable ReAdmis.

Figure 10

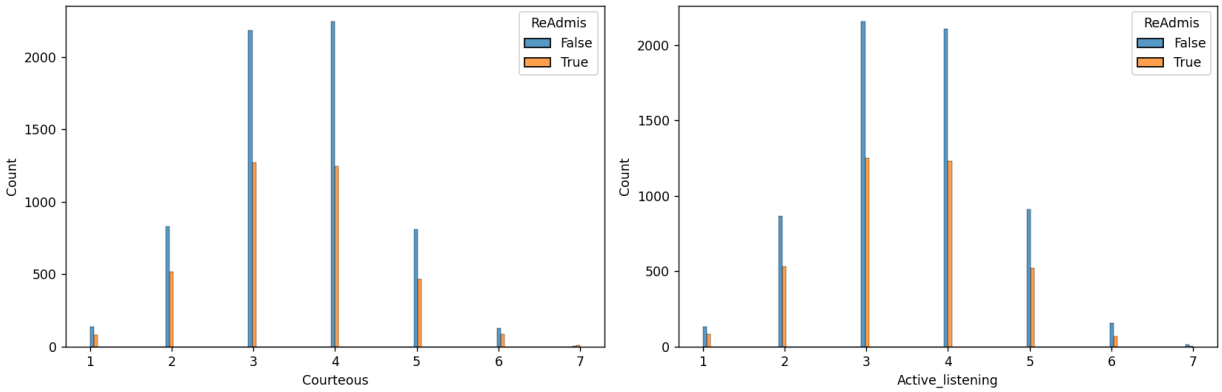
Bivariate Visualization of Initial_days, TotalCharge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment



Note. Bar charts of the predictor variables Initial_days, TotalCharge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment split on the response variable ReAdmis.

Figure 11

Bivariate Visualization of Courteous and Active_listening



Note. Bar charts of the predictor variables Courteous and Active_listening split on the response variable ReAdmis.

Figure 12

Generate Dummy Variables and Check Variance Inflation Factor

```
# Create Dummy Variables for all categorical independent variables
med_rec_dummies=pd.get_dummies(med_rec, drop_first=True)
```

```
# Split the data into dependent and independent variables
dependent = med_rec_dummies.ReAdmis
independent = med_rec_dummies.drop(columns=['ReAdmis'])
```

```
# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = independent.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(independent.astype(float).values, i)
                          for i in range(len(independent.columns))]
```

```
# Creates a list of independent variables that have excessive
# VIF numbers, then drops them from the data
feature_list = []
for feature, VIF in zip(vif_data.feature, vif_data.VIF):
    if VIF > 10:
        print("Variable: ",feature," VIF: ",VIF)
        feature_list.append(feature)
for item in feature_list:
    if item in independent:
        independent = independent.drop(columns=item)
```

```
Variable:  HighBlood  VIF:  13.951439225480703
Variable:  Initial_days  VIF:  543.2590884150784
Variable:  TotalCharge  VIF:  1394.8419163056499
Variable:  Additional_charges  VIF:  16.321394084671535
```

Note. Dummy variables are created for the different levels of the categorical variables. The data set is then split between the independent and dependent variables. A DataFrame is then created to hold the variance inflation factor data(VIF). The generated list is then combed over, looking for a VIF of over 10, which is subsequently dropped from the independent data set.

Figure 13

Initial Logistic Regression Model

```
# Split into train and test data
x_train, x_test, y_train, y_test = train_test_split(independent, dependent, test_size = 0.3)
```

```
# Create the model and print the summary
model = sm.Logit(y_train, x_train.astype(float))
result = model.fit(disp=0)
print(result.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                 7000
Model:                           Logit   Df Residuals:                     6958
Method:                            MLE   Df Model:                           41
Date:                Wed, 27 Oct 2021   Pseudo R-squ.:                0.004224
Time:                        12:44:55   Log-Likelihood:                -4591.1
converged:                       True   LL-Null:                       -4610.6
Covariance Type:            nonrobust   LLR p-value:                    0.5623
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Population                     1.663e-06   1.67e-06      0.997      0.319   -1.61e-06    4.93e-06
Children                          0.0177      0.011      1.552      0.121      -0.005       0.040
Age                               0.0005      0.001      0.433      0.665      -0.002       0.003
Income                        -1.948e-06   8.69e-07     -2.242      0.025   -3.65e-06   -2.45e-07
VitD_levels                      -0.0147      0.010     -1.502      0.133      -0.034       0.004
Doc_visits                       -0.0133      0.023     -0.593      0.553      -0.057       0.031
Full_meals_eaten                 -0.0216      0.025     -0.875      0.382      -0.070       0.027
vitD_supp                         0.0419      0.039      1.065      0.287      -0.035       0.119
Soft_drink                       -0.0124      0.057     -0.218      0.828      -0.124       0.099
Stroke                           -0.0334      0.062     -0.537      0.591      -0.155       0.089

Overweight                       -0.1029      0.054     -1.899      0.058      -0.209       0.003
Arthritis                         0.0237      0.052      0.457      0.648      -0.078       0.125
Diabetes                         -0.0245      0.056     -0.437      0.662      -0.134       0.085
Hyperlipidemia                   -0.0073      0.053     -0.138      0.890      -0.111       0.096
BackPain                          0.0322      0.051      0.637      0.524      -0.067       0.132
Anxiety                          -0.0247      0.053     -0.463      0.643      -0.129       0.080
Allergic_rhinitis                 0.0089      0.051      0.176      0.861      -0.091       0.109
Reflux_esophagitis                0.0380      0.050      0.754      0.451      -0.061       0.137
Asthma                           -0.0433      0.055     -0.786      0.432      -0.151       0.065
Timely_admission                 -0.0229      0.036     -0.641      0.521      -0.093       0.047
Timely_treatment                  0.0408      0.033      1.239      0.215      -0.024       0.105
Timely_visits                    -0.0233      0.030     -0.774      0.439      -0.082       0.036
Reliability                      -0.0017      0.026     -0.067      0.947      -0.052       0.048
Options                           0.0077      0.025      0.305      0.760      -0.042       0.057
Hours_treatment                  -0.0573      0.029     -1.950      0.051      -0.115       0.000
Courteous                         0.0247      0.028      0.893      0.372      -0.030       0.079
Active_listening                 -0.0454      0.026     -1.751      0.080      -0.096       0.005
Area_Suburban                     0.0031      0.061      0.052      0.959      -0.116       0.122
Area_Urban                        0.0479      0.061      0.787      0.431      -0.071       0.167
Marital_Married                   0.0993      0.078      1.265      0.206      -0.055       0.253
Marital_Never Married             0.1023      0.079      1.297      0.195      -0.052       0.257
Marital_Separated                 0.0817      0.079      1.031      0.302      -0.074       0.237
Marital_Widowed                   0.1239      0.078      1.588      0.112      -0.029       0.277
Gender_Male                       0.0312      0.050      0.620      0.535      -0.067       0.130
Gender_Nonbinary                  0.0849      0.172      0.494      0.621      -0.252       0.421
Initial_admin_Emergency Admission 0.0414      0.061      0.682      0.495      -0.078       0.160
Initial_admin_Observation Admission -0.0015   0.071     -0.021      0.983      -0.141       0.138
Complication_risk_Low             0.0325      0.069      0.472      0.637      -0.102       0.167
Complication_risk_Medium          0.0217      0.056      0.385      0.701      -0.089       0.132
Services_CT Scan                  0.0988      0.078      1.273      0.203      -0.053       0.251

Services_Intravenous             -0.0841      0.056     -1.490      0.136      -0.195       0.027
Services_MRI                      0.1280      0.132      0.968      0.333      -0.131       0.387
================================================================================================
```

Note. This is the original logistic regression model with the included summary.

Figure 14

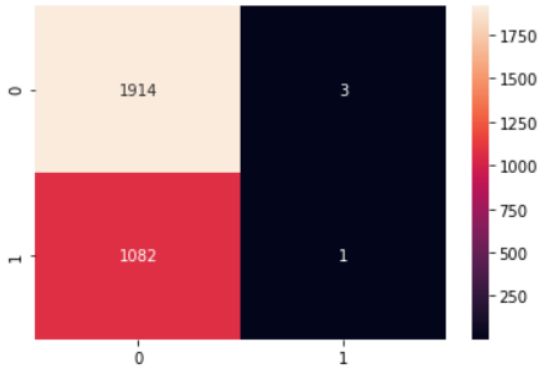
Confusion Matrix of the Original Model

```
# Creates a list of predictions
yhat = result.predict(x_test.astype(float))
prediction = list(map(round, yhat))

# confusion matrix
cm = metrics.confusion_matrix(y_test, prediction)
%matplotlib inline
sns.heatmap(cm, annot=True, fmt='0000')

# accuracy score of the model
print('Test accuracy = ', metrics.accuracy_score(y_test, prediction))
```

```
Test accuracy =  0.6383333333333333
```



Note. The code uses the original model to make predictions based on the set of X test inputs. The predictions made are then compared against the actual values of the Y values. A heat map is then created using the confusion matrix of the true and false negatives and positives.

Figure 15

Feature Selection using Backward Elimination with Akaike Information Criterion

```
# Global variables to perfrom checks
sig_features = []
highest = 'no'
last_highest = 'yes'
aic_check = independent
current_aic = result.aic


# Removes features with the highest p-values, then evaluates the AIC
# in order to determine if the feature helped predictability
elim_result = sm.Logit(dependent, independent.astype(float)).fit(disp=0)
min_aic = elim_result.aic
while highest != last_highest:
    max_pvalue = 0.05
    last_highest = highest

    if min_aic > current_aic and highest != 'no':
        min_aic = current_aic
        aic_check = aic_check.drop(columns=highest)
    elif min_aic < current_aic and highest != 'no':
        sig_features.append(highest)
        independent = aic_check

    for feature, pvalue in zip(independent.columns, elim_result.pvalues):
        if pvalue > max_pvalue and feature not in sig_features:
            max_pvalue = pvalue
            highest = feature
    if highest != last_highest:
        independent = independent.drop(columns=highest)
        elim_result = sm.Logit(dependent, independent.astype(float)).fit(disp=0)
        current_aic = elim_result.aic

independent.columns
```

Note. This code uses backward elimination to remove the features with the highest p-values. It then compares the Akaike Information Criterion (AIC) of the model without the feature to the previous model's AIC, either leaving the feature out or replacing the feature depending on which model has the lower AIC. It then moves on to the next feature, until it has removed the features required to achieve the lowest AIC.

Figure 16

Reduced Logistic Regression Model

```
# Split into train and test data
x_train, x_test, y_train, y_test = train_test_split(independent, dependent, test_size = 0.3)

model = sm.Logit(y_train, x_train.astype(float))
result = model.fit(disp=0)
print(result.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                 7000
Model:                           Logit   Df Residuals:                     6990
Method:                            MLE   Df Model:                            9
Date:                Wed, 27 Oct 2021   Pseudo R-squ.:                0.002370
Time:                        12:52:24   Log-Likelihood:                -4614.3
converged:                       True   LL-Null:                       -4625.3
Covariance Type:            nonrobust   LLR p-value:                  0.009134
=====================================================================================================
                                        coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
Population                          3.269e-06   1.63e-06      2.002      0.045    6.91e-08    6.47e-06
Children                               0.0228      0.011      1.998      0.046       0.000       0.045
Income                             -9.055e-07   8.61e-07     -1.052      0.293   -2.59e-06    7.81e-07
VitD_levels                           -0.0133      0.006     -2.239      0.025      -0.025      -0.002
Asthma                                -0.1205      0.055     -2.191      0.028      -0.228      -0.013
Timely_admission                      -0.0406      0.024     -1.682      0.093      -0.088       0.007
Active_listening                      -0.0483      0.024     -2.024      0.043      -0.095      -0.002
Initial_admin_Emergency Admission      0.0442      0.049      0.894      0.371      -0.053       0.141
Services_CT Scan                       0.1317      0.078      1.695      0.090      -0.021       0.284
Services_Intravenous                  -0.0379      0.055     -0.688      0.491      -0.146       0.070
=====================================================================================================
```

Note. The code and summary for the reduced logistic regression model after the appropriate features have been selected.

Figure 17

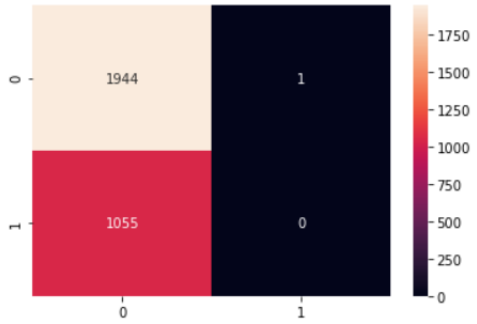
Confusion Matrix of the Reduced Model

```
# Creates a list of predictions
yhat = result.predict(x_test.astype(float))
prediction = list(map(round, yhat))

# confusion matrix
cm = metrics.confusion_matrix(y_true=y_test, y_pred=prediction)
sns.heatmap(cm, annot=True, fmt='0000')

# accuracy score of the model
print('Test accuracy = ', metrics.accuracy_score(y_true=y_test, y_pred=prediction))
```

```
Test accuracy =  0.648
```



Note. The code uses the reduced model to make predictions based on the set of X test inputs. The predictions made are then compared against the actual values of the Y values. A heat map is then created using the confusion matrix of the true and false negatives and positives.

Figure 18

Coefficients of the Variables for the Logistic Regression Function

```
result.params
```

```
Population                              0.000001
Children                                0.023227
Income                                 -0.000002
VitD_levels                            -0.013058
Asthma                                 -0.125818
Timely_admission                       -0.037520
Active_listening                       -0.038623
Initial_admin_Emergency Admission       0.023996
Services_CT Scan                        0.166215
Services_Intravenous                   -0.047152
```

Note. These are the coefficients to the variables in the reduced model for the logistic regression equation.